Debreceni Egyetem
Természettudományi és Technológiai Kar
Matematikai Intézet

Diplomamunka

Gráfok címkézéseinek vizsgálata

készítette:

Juhász Gyula

témavezető: Dr. Tengely Szabolcs

Debrecen, 2020

Tartalomjegyzék

Köszönetnyilvánítás

Először szeretnék köszönetet mondani a családomnak, akik az egyetemi éveim alatt mindig mellettem voltak, és támogattak a legnehezebb napokon is.
Továbbá, szeretnék köszönetet mondani Dr. Tengely Szabolcs témavezetőmnek, akinek segítsége nélkül nem jöhetett volna létre ezen dolgozat.
Végül, szeretnék köszönetet mondani a Debreceni Egyetem Matematikai Intézetében dolgozó valamennyi oktatónak lelkiismeretes munkájukért.

1. Bevezető

1967-ben Alexander Rosa cikkében [1] találkozhattunk először a graceful címkézés fogalmával, amelyet még β-címkézésként találhatunk meg ebben a cikkben. Egy gráfnak egy címkézése β-címkézés, hogyha létezik olyan injektív f csúcscímkézés, és bijektív g élcímkézés, hogy minden $e = uv \in E$ esetén $|f(u) - f(v)| = g(e)$. Vagyis, egy élnek a címkéje, a rá illeszkedő csúcsok címkéinek abszolút értékben vett különbsége, ahol az élek címkéi mind különbözőek. Solomon W. Golomb már ezt a címkézést, graceful címkézésként említi az 1972-ben megjelent "How to number a graph" cikkében [2].
Azonban az évek során a kutatók számos új definíciót, és tételt alkottak meg, amelyek közül talán E.K.Gnang tétele volt a legjelentősebb. Gnang 2019-ben kiadott publikációjában [3], a fagráfok esetén adott egy általános bizonyítást a graceful címkézés létezésére. Ezen eredmény ellenőrzése, jelen pillanatban is folyamatban van.
Ahogy múltak az évek, újabb gráfcsaládokról látták be, hogy létezik graceful címkézésük, viszont a gráfok sokasága miatt, még mindig vannak olyan gráfcsaládok, amelyekről nem tudni, hogy létezik-e graceful címkézésük. Ilyenkor viszont felmerül a kérdés, hogy egy adott gráfcsalád esetén, létezik-e olyan csúcscímkézés, amelyet ki lehet terjeszteni az egész gráfcsaládra.
Az előző dolgozatomhoz hasonlóan, a diplomamunkám során is speciális gráfoknak az implementálását tűztem ki célul, illetve olyan gráfoknak a címkézését vizsgáltam, amelyekről eddig csak sejteni lehetett, hogy létezik graceful címkézése. Az implementálás részét minden esetben a SageMath programcsomagban [4] végeztem el, viszont néhány algoritmust az alap Java programozási nyelv segítségével próbáltam megalkotni. A vizsgált gráfcsaládok között szerepelnek továbbra is a fagráfok, illetve speciális körgráfok, de szeretnék egy olyan gráfcsaládot is bemutatni, amelynél nem feltétlen követelmény például, hogy összefüggő legyen a gráf. Viszont az implementálások során fontos, hogy a formulák precízen legyenek leírva a cikkekben. Így a cikkek feldolgozása során nem csak az implementálásra kellett figyelnem, hanem a formulák helyességére is. Sok esetben a helyes programeszközök használata után derült ki, hogy néhány cikk esetében elírások, hiányosságok vannak. Így ezen esetekben a formulákat javítani kellett. Például javításra szorultak a G_n gráfok, a medál gráfok, a C_tC_n gráfok, a sárkány gráfok, a párhuzamos húrokkal rendelkező körgráfok, illetve a pók gráfok formulái is.

A dolgozatom írása során viszont egy érdekes gondolatom támadt a gráfok konstrukciója során. Amikor régebben egy mátrixot implementálni szerettem volna, először is megnéztem annak a gráfnak a szomszédsági mátrixát, hogy milyen tulajdonságokkal rendelkezik. A szomszédsági mátrix egy bináris mátrix, amelyről a graceful címkézés miatt feltesszük, hogy a főátlójában csak nullák szerepelnek, hiszen graceful címkézés során csak azoknak a gráfoknak van graceful címkézése, amelyek egyszerű gráfok. Ekkor támadt az az ötletem, hogy mi történne akkor, hogyha én egy gráf szomszédsági mátrixaként a Hadamard-mátrixot venném némi módosítással. Ennek eredményeit is szeretném bemutatni ebben a dolgozatban, amelyet megelőz egy kisebb kódelméleti ismertető.

2. Graceful címkézés keresése

Először is ebben a fejezetben néhány alapvető definíciót, illetve tételt szeretnék ismertetni, amelyek nélkül nehezen értelmezhető maga a graceful címkézés. A tételek és definíciók után, egy rövid eljárást szeretnék bemutatni, amely során bármilyen gráf esetén megállapítható, hogy az adott gráfnak létezik-e graceful címkézése, vagy sem. Viszont ez az eljárás csak kis csúcsszámú gráfok esetén használható, hiszen az eljárást nagyban meghatározza az éleknek, és a csúcsoknak a száma.

2.1. Definíciók és fontosabb állítások

2.1.1. Definíció. Legyen $G = (V, E, \varphi)$ véges gráf. Ekkor a G gráf csúcsainak a címkézésén egy olyan Γ bijektív leképezést értünk, ahol az összes $v \in |V(G)|$ csúcshoz hozzárendelünk különböző nemnegatív egész számokat.
Egy gráf éleinek címkézésén egy olyan Δ bijektív leképezést értünk, ahol az összes $e \in |E(G)|$ élhez hozzárendelünk különböző nemnegatív egész számokat.

2.1.2. Definíció. Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf, ahol $|V(G)| = n$, $|E(G)| = m$. Ha van olyan $f : V(G) \to \{0, \ldots, m\}$ injektív függvény, hogy bármely $e \in E(G)$ esetén az

$$|f(v) - f(w)| \quad (v, w \in V(G))$$

értékek különbözőek, akkor ezt az f függvényt a G gráf *graceful címkézésének* nevezzük. Ekkor az élek címkéinek a halmaza: $\{1, \ldots, m\}$.

2.1.1. Lemma. *Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf, $|V(G)| = m$ és $|E(G)| = n$. Ha a G gráf gracful, akkor $m \leq n + 1$.*

2.1.1. Tétel. *Legyen $G = (V, E, \varphi)$ véges, teljes gráf és $|V(G)| = n$.*
Ha n = 2,3,4, akkor a G gráfnak van graceful címkézése. Ha $n > 4$, akkor a G gráfnak nincs graceful címkézése.

Megjegyzés. Továbbá, érdemes megjegyezni, hogy az évek során egyre több ember vizsgálta a fagráfok graceful címkézéseit. Alexander Rosa "On certain valuations of the vertices of a graph" cikkében [1] már bebizonyította néhány fagráfcsaládról, hogy graceful gráfok, viszont nem tudott olyan tételt megfogalmazni, ami

azt állítja, hogy minden fa graceful. Pár évtizeddel később, viszont a további fagráfcsaládok sikeres vizsgálata során, Ringel és Kotzig sejtésként fogalmazta meg, hogy minden fának van graceful címkézése. Ezt Ringel-Kotzig-Rosa sejtésnek, vagy más néven "A graceful fák sejtésének" nevezték.
Azonban 2019-ben E.K.Gnang, a Purdue Egyetem professzora, "On graceful and harmonious labelings of trees" munkájában [3] adott egy bizonyítást a fagráfok graceful tulajdonságaival kapcsolatosan, véges testek segítségével.

2.1.2. Tétel. *Legyen G véges, egyszerű gráf, $|V(G)| \leq 4$. Ekkor a G gráfnak van graceful címkézése.*

Megjegyzés. Ha a gráfnak pontosan 5 csúcsa van, akkor csak a C_5-nek és a K_5-nek nincs graceful címkézése.

2.1.3. Definíció. Legyen adott egy G egyszerű gráf, ahol $|V(G)| = \{v_1, \ldots, v_m\}$. A G gráfból képzett $M(G)$ egyszerű gráfot, a G gráf Mycielski-gráfjának nevezzük, ha csúcsai $V(G) \cup \{u_1, \ldots, u_m\} \cup \{w\}$, és egy $v_i \in V(G)$ csúcs pontosan azokkal a csúcsokkal szomszédos, amivel G-ben is szomszédos volt. Továbbá, ha v_i szomszédos volt v_j-vel, akkor az Mycielski-gráfban v_i legyen szomszédos u_j-vel, és u_i legyen szomszédos v_j-vel. Végül, w legyen szomszédos az összes u_i csúccsal, ahol $i = 1, \ldots, m$.

Megjegyzés. Belátható, hogy egy nemüres, egyszerű gráfnak és a Mycielski-gráfjának a maximális klikkmérete megegyezik, illetve a gráfnak egy k darab színnel történő csúcsszínezése esetén a Mycielski-gráfjának $k + 1$ darab színnel már van jó csúcsszínezése. Továbbá, ha $2 \leq a \leq b$, akkor létezik olyan egyszerű gráf, amelynek a maximális klikkmérete éppen a, míg a színezéseinek a száma b.

2.2. Backtrack-algoritmus és a graceful címkézés

A backtrack-algoritmus, vagy más néven visszalépéses-megoldáskereső algoritmus az egyik legegyszerűbb algoritmus abból a szempontból, hogyha az adatbázisunk egyszerű. Azaz, ha az adatbázisunk fastruktúrájú, akkor belátható időn belül az algoritmus eredményt adhat. Azonban, ha van az adatszerkezetünkben kör, akkor előfordulhat, hogy az algoritmus soha nem áll le. Így feltehetjük, hogy a backtrack-algoritmus figyel a körökre is. Általában a programozó ilyenkor visszaléptetést használ, azaz feljegyzi a már használt csomópontokat, és ha minden lehetőséget megvizsgált már ebből kiindulva, és nem talált megoldást, akkor azt többet már nem fogja bejárni.
Mivel ez fa-adatstruktúrákra van értelmezve, így nem nehéz olyan algoritmust készíteni, ami bizonyos problémákra keres megoldást gráfokban. Graceful címkézés esetén, az algoritmus nézzen ki a következőképpen:

(i.) Bemenetként megadjuk az algoritmusnak magát a gráfot, az éleknek a listáját, egy kezdeti címkézést, a kezdő csúcsnak a lehető legkisebb értékét, és végül a már kiosztott élcímkéknek egy listáját. A kezdeti élcímkézés pontosan annyi darab (-1)-est tartalmaz, amennyi csúcsa van a gráfunknak. Továbbá, a lehető legkisebb felvehető érték a 0 lesz, mivel a graceful címkézés során bármelyik csúcsnak a címkéje lehet 0. Mivel indításkor a gráf élei nincsenek címkézve, így az élcímkéknek a listája kezdetben üres lesz.

(ii.) Ha az éleknek a listájában nincs (-1)-es elem, akkor találtunk egy lehetséges megoldást. A megoldások között lehetnek azonosak, a szimmetria miatt.

(iii.) Ha egy csúcs nem volt még kiterjesztve ilyen címkével, azaz még nem volt ilyen címkéjű csúcs, és a szomszédos csúcsok által generált él címkéje nem szerepel benne a jelenlegi élcímkék listájában, akkor jegyezzük fel ezeket, és folytassuk tovább a következő csúcsnak az ellenőrzésével. Ha már szerepel az élcímkék listájában a csúcsok által generált élcímke, akkor vessük el ezt a csúcscímkét, és folytassuk tovább a következő csúcscímkével, ameddig csak lehet.

(iv.) Ha egy csúcs volt már kiterjesztve, azaz eddig megfelelt a graceful feltételnek, de nincs olyan szomszédos csúcsa ami már ne lett volna kiterjesztve ebből a csúcsból, akkor nem lehetséges az eddigi csúcscímkékkel a kiterjesztés, így ebből a csúcsból visszalépünk egy már hamarabb vizsgált szomszédos csúcsához, és folytatjuk tovább a csúcsok vizsgálatait.

Az algoritmus a következőképpen néz ki a gyakorlatban. Ez az algoritmus implementálva van Python, illetve Java nyelven is. Viszont mivel a beépített gráfok a Cocalc matematikai csomagban szerepelnek, és ennek alapja a Python nyelv, így én is a Python programnyelvet választottam a további implementációk során.

Graceful-backtrack algoritmus

```
1  def kiterjesztheto(G,csucs,cimkek,l,elcimkek):
2
3          if l in cimkek:
4                  return False
5          seged = list()
6          for szomszed in G[csucs]:
7                  if cimkek[szomszed] != -1:
8                          ertek = abs(l - cimkek[szomszed])
9                          if (ertek in elcimkek) or (ertek in seged):
10                                 return False
11                         seged.append(ertek)
12         return True
13
14 def graceful_backtrack(G, elek, cimkek, index, elcimkek):
15         if cimkek.count(-1) == 0:
16         with open('graceful_cimkek.txt', 'a') as filehandle:
17                 for listaelem in cimkek:
18                         filehandle.writelines("%s " %
19                                 pozicio for pozicio in cimkek)
20                         filehandle.writelines("\n")
21                         break
22         return True
23
24 for l in xrange(0,len(elek)+1):
25         if kiterjesztheto(G, index, cimkek, l, elcimkek):
26                 cimkek[index] = l
27
28                 for szomszed in G[index]:
29                         if cimkek[szomszed] != -1:
30                                 elcimkek.add(
31                 abs(cimkek[index]-cimkek[szomszed]))
32
33                 graceful_backtrack(G,elek,cimkek,index+1,elcimkek)
34
35                 cimkek[index] = -1
36                 for szomszed in G[index]:
37                         if cimkek[szomszed] != -1:
38                 elcimkek.discard(abs(l-cimkek[szomszed]))
39
40         return False
```

Ez az algoritmus jól működik, hiszen vegyünk egy egyszerű fagráfot, legyen ez P_3. Ennek az útgráfnak 3 darab csúcsa van, és 2 darab éle. A program futtatása során a következő eredményeket kapjuk: $[0, 2, 1]$, $[1, 0, 2]$, $[1, 2, 0]$, $[2, 0, 1]$.

Azaz, izomorfiától eltekintve, megkaptuk a P_3 gráf csúcsainak összes lehetséges címkézését.
Azonban, attól függetlenül, hogy az algoritmus véges sok lépés után megadja a gráfnak az összes csúcscímkézését, a probléma nehézségét az adja, hogy vizsgálva például az útgráfokat, nehéz minden osztálybeli gráfra megadni képletet. Továbbá, az sem könnyíti meg a helyzetünket, hogy sokszor még kis gráfok esetén (mondjuk 10 csúcsú gráfok esetén), sokszor órákba, sőt napokba is telhet, mire az algoritmus véget ér. Ennek megoldására általában meg szokták osztani a magokat, és megpróbálják a programot különálló részenként egyszerre végrehajtani, hogy így gyorsabban eredményhez tudjunk jutni. Másik megoldás az, ha videokártyát használunk a gyorsítás során. A CUDA, az NVIDIA grafikus processzorainak általános célú programozására használható környezete, amit általában adatok elemzésénél, és neurális hálózatok esetén használnak.
Viszont a backtrack-algoritmus rekurzív ebben az esetben, és így szinte lehetetlen a szálasítása Python és Java nyelvek alatt. Így az osztályok kevés számú csúcsú gráfjaira kell a graceful algoritmusunkat lefuttatni, és az összes címkézési lehetőségek között kell szabályszerűséget keresnünk, amely nagyobb csúcsú gráfok esetén is helyes graceful címkézést eredményez.
Ezen algoritmust használtam fel az álkerék gráfok páratlan esetének vizsgálata során. Ha az $n = 5$ esetet vizsgáljuk, és erre futtatjuk le a backtrack-algoritmust, akkor az élek száma miatt, viszonylag nagy lesz a futásidő. Viszont, feltételezhetjük, hogy a központi csúcs a 0 címkével rendelkezik. Ebben az esetben, néhány óra alatt megkapjuk az összes lehetséges megoldást, amely 0-val kezdődik. Tovább szűkíthetjük a kört, abban az esetben, hogyha feltesszük, hogy a belső gyűrűjében található csúcsokhoz a legnagyobb címkéket rendeljük, hiszen 0 központi csúcs miatt, szükségünk van ezen legnagyobb címkékre is. Ebben az esetben már csak 10 darab csúcsra kell lefuttatni az algoritmus, és ezekben kell részsorozatokat keresnünk. Így ezen feltételek mellett, már csak 1358 darab megoldást kapunk. Hasonlóan kell eljárni az $n = 7$ esetben is. Ha legeneráljuk az összes lehetséges megoldást ebben az esetben is ezen feltételek mellett, akkor az így kapott megoldások, illetve az $n = 5$ megoldásai alapján kell a sorozatot meghatározni. Ha megtaláltuk ezeket a részsorozatok, akkor a páratlan esetről is be lehet látni, hogy bármely n esetén, ahol n páratlan, létezik graceful címkézése az álkerék gráfnak.
Hasonlóan kell megvizsgálni azokat a gráfokat is, amelyeket a Mycielski-konstrukció során alkotunk meg. Vegyük az C_7 gráfot. Először alkalmazzuk erre a gráfra a backtrack-algoritmust, és gyártsuk le az összes lehetséges csúcscímkézését. Továbbá, alkalmazzuk erre a gráfra a Mycielski-konstrukciót. Az így kapott gráfot adjuk meg a backtrack-algoritmusnak. Az algoritmus véges sok lépés után leáll. Hasonlóan kell eljárni a többi Mycielski-gráf esetén is, és az előző esethez hasonlóan részsorozatokat kell keresgélni a listákban.

Így a backtrack-algoritmus, bár megtalálja az összes lehetséges megoldást véges időn belül, viszont a futási ideje nagyon rossz, illetve a szimmetria miatt rengeteg megoldást fog adni. Így a backtrack-algoritmust célszerűbb arra használni graceful gráfok esetén, hogy belássuk azt, hogy az adott gráf nem graceful gráf.

3. Hadamard-mátrixok és graceful címkézés

3.1. Hadamard-mátrixok

A Hadamard-mátrixokról az első publikációk [5] a 19. században jelentek meg. J.J.Sylvester nevéhez fűződik a mátrixoknak a definiálása, de nevüket J.S.Hadamard-ról kapták, aki a 19. század végén jellemezte ezeket a mátrixokat.

3.1.1. Definíció. Egy $A_{n\times n}$ mátrixot *Hadamard-mátrixnak* nevezünk, ha minden eleme ± 1, illetve sorai páronként ortogonálisak.

3.1.1. Tétel. *Ha $A_{n\times n}$ és minden elemére teljesül, hogy $|a_{ij}| \leq 1$, akkor*

$$|det(A)| \leq n^{\frac{n}{2}}.$$

Bizonyítás. Legyenek az A mátrix sorai rendre $a_1, a_2, \ldots, a_n$. Továbbá azt is tudjuk, hogy a $det(A)$ az A mátrix sorvektorai által kifeszített paralelepipedon térfogata. Mivel

$$|a_1| \cdot |a_2| \cdot \ldots \cdot |a_n|$$

a paralelepipedon éleinek szorzata, így a következő igaz:

$$|det(A)| \leq |a_1| \cdot \ldots \cdot |a_n|.$$

Viszont a tétel alapján

$$|a_i| = (a_{i1}^2 + a_{i2}^2 + \ldots + a_{in}^2)^{\frac{1}{2}} \leq n^{\frac{1}{2}}$$

adódik. Tehát $|det(A)| \leq n^{\frac{n}{2}}$. Egyenlőség pontosan akkor áll fenn, hogyha az A mátrix sorai ortogonálisak. □

3.1.2. Tétel. *Legyen $A_{n\times n}$ mátrix, amelynek minden eleme ± 1. Ekkor a következő állítások ekvivalensek:*

(i.) $|det(A)| = n^{\frac{n}{2}}$,

(ii.) $AA^T = n \cdot I_n$,

(iii.) $A^T A = n \cdot I_n$.

Bizonyítás. $(i.) \to (ii.)$ Mivel a mátrix egy sorvektora $(\pm 1, \pm 1, \ldots, \pm 1)$ alakú, így ennek hossza $\sqrt{n}$ lesz. Így adódik, hogy

$$\begin{pmatrix} \sqrt{n}^2 & 0 & \ldots & 0 \\ 0 & \sqrt{n}^2 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & \sqrt{n}^2 \end{pmatrix} = n \cdot \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix}.$$

$(ii.) \to (iii.)$

$$AA^T = n \cdot I_n$$
$$A^T AA^T = A^T n \cdot I_n$$
$$A^T AA^T A = A^T n \cdot I_n A$$
$$A^T A = n \cdot I_n$$

$(iii.) \to (i.)$ Induljunk ki az

$$A^T A = n \cdot I_n$$

egyenletből. Az egyenlőség a mátrixok determinánsainak vizsgálatakor is fennáll, tehát

$$det(A) \cdot det(A^T) = n^n.$$

Ebből egyből adódik, hogy

$$det(A)^2 = n^n,$$

amelyből gyökvonással adódik, hogy

$$|det(A)| = n^{\frac{n}{2}}.$$

□

3.1.3. Tétel. *Ha létezik $A_{n\times n}$ Hadamard-mátrix, akkor $n = 1$, $n = 2$, $n \equiv 0 \pmod 4$ közül az egyik teljesül.*

Bizonyítás. $n = 1$ és $n = 2$ esetén is léteznek Hadamard-mátrixok.

$$n = 1: \quad \begin{pmatrix} \pm 1 \end{pmatrix},$$

$$n = 2: \quad \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix}.$$

Így feltehető, hogy $n \geq 3$. Azt tudjuk, hogyha egy Hadamard-mátrix tetszőleges sorát megszorozzuk (-1)-gyel, akkor szintén Hadamard-mátrixot fogunk kapni. Így feltehetjük, hogy A legelső sora csak 1-ket tartalmaz. Mivel a mátrix sorai ortogonálisak, így két sor pontosan $\frac{n}{2}$ helyen különbözik, vagy egyezik meg. Legyen $n = 2k$. Ekkor a mátrix következő sorának első k eleme legyen 1, második k

eleme pedig legyen (-1). Tegyük fel, hogy a harmadik sorban s darab 1 van az első k pozíció bármelyik helyén. Ebből adódik, hogy a második k pozícióban csak $(k-s)$ darab 1 szerepelhet. Vegyük a két sor skaláris szorzatát. Így kapjuk, hogy $0 = s-(k-s)-(k-s)+s = s-k+s-k+s+s = 4s-2k$. Amiből adódik, hogy $s = \frac{n}{4}$. □

Ha Hadamard-mátrixokat szeretnénk alkotni, akkor annak legkönnyebb módja Sylvester-eljárását követése. Az eljárás során semmi másra nincs szükségünk csak Hadamard-mátrixokra, illetve egy műveletre, ez a bizonyos Kronecker-szorzat művelet.

3.1.4. Tétel. *Ha $A_{n\times n}$ Hadamard-mátrix, akkor* $\begin{pmatrix} A & A \\ A & -A \end{pmatrix}$ *is Hadamard-mátrix.*

Bizonyítás. Mivel A Hadamard-mátrix, így elemei csak ±1 lehetnek. Így $-A$ elemei is csak ±1 lehetnek. Továbbá, legyen az új mátrixunk $B = \begin{pmatrix} A & A \\ A & -A \end{pmatrix}$. Ekkor B elemei csak ±1 lehetnek az előzőek miatt. Számoljuk ki BB^T-t. Ekkor

$$BB^T = \begin{pmatrix} A & A \\ A & -A \end{pmatrix} \cdot \begin{pmatrix} A & A \\ A & -A \end{pmatrix}^T = \begin{pmatrix} A & A \\ A & -A \end{pmatrix} \cdot \begin{pmatrix} A^T & A^T \\ A^T & -A^T \end{pmatrix} =$$

$$= \begin{pmatrix} 2AA^T & 0 \\ 0 & 2AA^T \end{pmatrix} = 2n \cdot \begin{pmatrix} I_n & 0 \\ 0 & I_n \end{pmatrix}.$$

□

3.1.5. Tétel. *$n = 2^m$ esetén létezik $n \times n$-es Hadamard-mátrix, ahol $m \in \mathbb{Z}^+$.*

Bizonyítás. Az előző tétel véges sokszori alkalmazásával kapjuk az állítást. □

3.2. Egyszerűsített Hadamard-mátrixok

A Hadamard-mátrixok ismertetése, illetve a mátrixok konstrukciója után látható, hogy bár a mátrixok $n \times n$ mátrixok, és minden elem ±1, viszont ha ezt a mátrixot egy az egyben szomszédsági mátrixnak tekintenénk, az így kapott gráf nem lenne egyszerű, hiszen az mátrix első eleme miatt keletkezne hurokél. Továbbá, azt is fontos megjegyezni, hogy az első csúcs a mátrixban szomszédos a mátrix összes csúcsával.

Így ahhoz, hogy számunkra megfelelő szomszédsági mátrixot kapjunk, az alábbi módosításokat kell elvégezni a Hadamard-mátrixokon. Először is vegyünk egy olyan transzformációt, ami a mátrix elemein hat, és a -1 értékű elemekből 0-t készít, míg a többi elemet békén hagyja, magyarul cseréljük ki a mátrix összes -1 elemét 0-ra. Továbbá, hagyjuk el azokat a csúcsokat amikből minden csúcsba megy él. Ez a legelső csúcs, így elhagyjuk az első sort, és az első oszlopot.

Végül vegyünk egy A-val jelölt mátrixot, amely $(n-1) \times (n-1)$ mátrix, és a főátlón kívül csak 0-kat tartalmaz, míg a főátlóban az elemek megegyeznek a Hadamard-mátrixból készített új mátrixunk főátlóbeli elemeivel. Vonjuk ki ezt az A mátrixot a módosított Hadamard-mátrixból. Az így kapott mátrixban a főátlóban már csak 0 elemek fognak szerepelni, míg a többi elem változatlan marad. Az így kapott $(n-1)\times(n-1)$ mátrix szomszédsági mátrixnak tekinthető, amit az $n \times n$ Hadamard-mátrixból képeztünk. Az így kapott mátrixot nevezzük az $n \times n$ Hadamard-mátrix egy *csonkított Hadamard-mátrixának*.

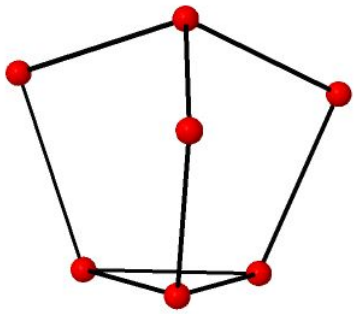
Példa. Tekintsük a 8×8-as Hadamard-mátrixot. Jelöljük ezt H_8-cal.

$$H_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}. \tag{3.2.1}$$

A belőle konstruált csonkított Hadamard-mátrixot jelöljük $\overline{H_7}$-tel, amely a következőképpen néz ki.

$$\overline{H_7} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}. \tag{3.2.2}$$

Ha ezt, mint szomszédsági mátrix tekintjük, akkor az általa generált gráf egyszerű lesz, illetve mivel a csonkított Hadamard-mátrix minden sora azonos súlyú, azaz az 1-sek darabszáma minden sor esetén megegyezik , így az egyszerűsítés után kapott mátrixból képzett gráfnak, csak másodfokú, illetve harmadfokú csúcsai lehetnek. Hiszen, eredetileg egy csúcs kivételével minden csúcsnak a fokszáma 3 volt. Viszont a főátlónak az egyszerűsítése után, csökkenhet bizonyos csúcsoknak a fokszáma.



3.1. ábra. $\overline{H_7}$ irányítatlan gráfja

Tekintsük ezt a gráfot. Erre a gráfra alkalmazzuk az ismert backtrack-graceful algoritmust. Ennek a gráfnak is van graceful címkézése, az összes lehetséges címkézés 216-féle lehet.
Vizsgáljuk meg ezután a következő Hadamard-mátrixból alkotott csonkított Hadamard-mátrixot, amely a $\overline{H_{11}}$ lesz. Hasonlóan az algoritmus ebben az esetben is megoldásokkal tér vissza, viszont ebben az esetben a futási idő exponenciálisan nő. Így egy átlagos számítógép esetében, a futási idő az összes megoldás megtalálásához ebben az esetben, körülbelül 10-12 nap. A futtatás során 10540 darab megoldást talált a programunk, amely a szimmetria miatt tartalmazhat azonos címkézéseket is, csak más sorrendben. Így a $\overline{H_7}$, illetve a $\overline{H_{11}}$ esetében is létezik graceful címkézés. Ezzel a gráfcsaláddal kapcsolatosan a sejtésem az, hogy az így megkonstruált gráfoknak létezik graceful címkézése.

4. Speciális graceful körgráfok implementálása

Ebben a fejezetben speciális körgráfokat szeretnék bemutatni, amelyeket a SageMath programcsomag segítségével implementáltam. Ezek a gráfok mind a körgráfból keletkeztek, viszont mindegyik gráf jellegzetes tulajdonságokkal bír.

4.1. C_n gráfok

4.1.1. Definíció. Legyen $C_n = (V, E, \varphi)$ véges gráf. Legyenek C_n csúcsai rendre $v_1, v_2, \ldots, v_n$. Ha $1 \leq i \leq n-1$, akkor a v_i csúcs legyen szomszédos a v_{i+1} csúccsal. Továbbá, ha $i = n$, akkor v_n legyen szomszédos a v_1 csúccsal. Az így konstruált gráfot n csúcsú körgráfnak nevezzük, amelyet C_n-nel jelölünk.

4.1.1. Tétel. *A C_n gráf pontosan akkor graceful gráf, hogyha $n \equiv 0, 3 \pmod 4$.*

Bizonyítás. Először is vegyük észre, hogy a körgráfok tartalmaznak Euler-vonalat. Viszont, hogyha egy gráf tartalmaz Euler-vonalat, akkor az pontosan akkor lehet graceful, hogyha $n \equiv 0, 3 \pmod 4$. Ebben az esetben, legyen $V(C_n) = \\ = \{v_0, v_2, \ldots, v_{n-1}\}$.

(i.) Ha $n \equiv 0 \pmod 4$, akkor definiáljuk az f csúcscímkézést a következőképpen:
Ha $0 \leq i \leq n-2$, és i páros, akkor

$$f(v_i) = \frac{i}{2}.$$

Ha $1 \leq i \leq \frac{n}{2} - 1$, és i páratlan, akkor

$$f(v_i) = n - \frac{i-1}{2}.$$

Végül, ha $\frac{n}{2} + 1 \leq i \leq n-1$, és i páratlan, akkor

$$f(v_i) = n - \frac{i-1}{2} - 1.$$

(ii.) Ha $n \equiv 3 \pmod 4$, definiáljuk az f csúcscímkézést a következőképpen:
Ha $0 \leq i \leq n-1$, és i páros, akkor

$$f(v_i) = \frac{i}{2}.$$

Ha $1 \leq i \leq \frac{n+1}{2} - 1$, és i páratlan, akkor

$$f(v_i) = n - \frac{i-1}{2}.$$

Végül, ha $\frac{n+1}{2} + 1 \leq i \leq n-2$, és i páratlan, akkor

$$f(v_i) = n - \frac{i-1}{2} - 1.$$

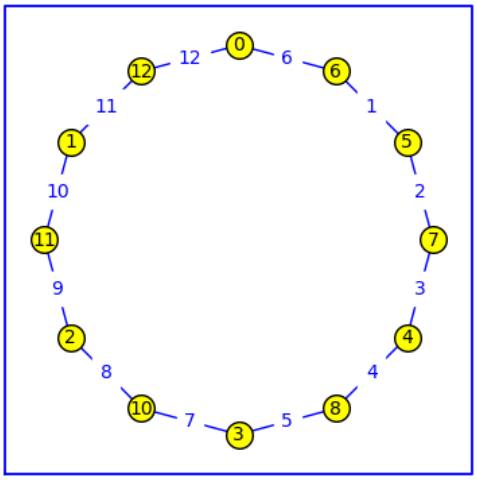
□

Mivel a tétel bizonyítása konstruktív, így könnyen lehet implementálni. Először is a címkéket fogom elkészíteni a gráf csúcsaihoz, azaz az f függvényt fogom definiálni. Ezután felhasználom, hogy a körgráf implementálva van a SageMath programcsomagban, és egyszerűen csak hozzárendelem a címkéket.

```
1  def Labels(n):
2      L=[]
3      if(n%4==1 or n%4==2):
4          return 'A gráf nem graceful gráf!'
5      elif(n%4==0):
6          for i in range(1,n+1):
7              if(i%2!=0):
8                  L.insert(i,(i-1)/2)
9              else:
10                 if(i<=n/2):
11                     L.insert(i,n+1-i/2)
12                 else:
13                     L.insert(i,n-i/2)
14     elif(n%4==3):
15         for i in range(1,n+1):
16             if(i%2==0):
17                 L.insert(i,n+1-i/2)
18             else:
19                 if(i<=(n-1)/2):
20                     L.insert(i,(i-1)/2)
21                 else:
22                     L.insert(i,(i+1)/2)
23     return L
```

```
1  def Cycle_graph(n):
2      labels=Labels(n)
3      G=graphs.CycleGraph(n)
4      G.relabel(labels,inplace=True)
5      for u,v,l in G.edges(): G.set_edge_label(u,v,abs(u-v))
```

```
6       return G.graphplot(edge_labels=True,edge_color='blue',
7               vertex_color='yellow',vertex_size=200,graph_border=True).show()
```



4.1. ábra. C_{12} gráf

4.2. $B_{n,n}$ gráfok

4.2.1. Definíció. Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf. V csúcsai legyenek rendre $v_0, v_1, \ldots, v_n, u_0, u_1, \ldots, u_n$. Tekintsük a v_0 csúcsot, és kössük össze éllel az összes csúccsal. Ezután tekintsük az u_0 csúcsot, és kössük össze éllel az összes csúccsal, kivéve v_0-val, hiszen a gráf egyszerű. Az így konstruált G gráfot $B_{n,n}$-nel jelöljük, és sapka gráfnak nevezzük.

Megjegyzés. A sapka nevet a gráf jellegzetes kinézete alapján kapta. Továbbá erről a gráfról Samir Vaidya és N. H. Shah 2013-as cikkében [6] olvashattunk először.

4.2.1. Tétel. *A $B_{n,n}$ gráf egy graceful gráf.*

Bizonyítás. Tekintsük először a $B_{n,n}$ gráfot. Ennek a csúcsai rendre $v_0, v_1, \ldots, v_n$, $u_0, u_1, \ldots, u_n$. Továbbá, v_0 és u_0 jelölje a két kitüntetett csúcsot. Így ennek a gráfnak összesen $2n + 2$ darab csúcsa van, azaz $|V(B_{n,n})| = 2n + 2$, illetve $4n + 1$ darab éle, azaz $|E(B_{n,n})| = 4n + 1$. Tekintsük a következő csúcscímkézést $f : V(G) \to \{0, 1, \ldots, 4n + 1\}$ esetén.
Legyen

$$f(v_0) = 0, \qquad \textit{illetve} \qquad f(u_0) = 4n + 1.$$

Ha $1 \leq i \leq n$, akkor

$$f(v_i) = i, \qquad \textit{illetve} \qquad f(u_i) = f(v_n) + i.$$

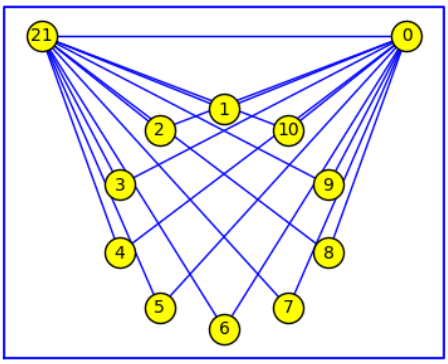
Ez az f függvény viszont a graceful tulajdonság miatt, definiál egy f^* bijektív függvényt, amely az élekhez rendel címkéket, vagy számokat.
Azaz $f^* : E(G) \to \{1, 2, \ldots, 4n+1\}$. Mivel az f^* egy bijektív függvény, értékkészlete $\{1, 2, \ldots, 4n+1\}$, így f graceful címkézése $B_{n,n}$-nek, azaz a $B_{n,n}$ gráf graceful gráf. □

A SageMath programcsomag programozási nyelve a Python programnyelv. Viszont ebben a csomagban már az alapvető függvények implementálásra kerültek, így azokat a függvényeket már nem kell nekünk megírnunk, hanem importálás segítségével nyugodtan felhasználhatjuk ezeket.
Az implementálás során egy egyszerű függvényt definiáltam, amely bemenetként a $B_{n,n}$ gráf n értékét várja. A függvény lefutása után egy listát fog visszaadni, amiben szerepelni fognak a csúcsoknak a címkéi. A címkéknek a konstrukciója a bizonyításban leírt módon fog zajlani.

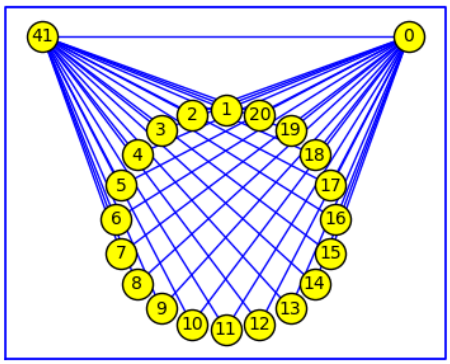
```
1  def Labels(n):
2      L=[]
3      L.insert(0,0)
4      L.insert(1,4*n+1)
5      szamlalo=2
6      for i in range(1,n+1):
7          L.insert(szamlalo,i)
8          szamlalo=szamlalo+1
9      temp=len(L)
10     for i in range(1,n+1):
11         L.insert(szamlalo,L[temp-1]+i)
12         szamlalo=szamlalo+1
13     return L
```

Bár a SageMath-ban vannak beépített gráfok, viszont a csúcsok hozzárendelésekor nekünk figyelnünk kell az egyértelmű hozzárendelésre. Sajnos a beépített gráfoknál az indexelés miatt gyakran a listánk elemeit nem lehet hozzárendelni a csúcsokhoz. Így egy függvényt definiálunk, amely először is felépíti a kívánt gráfot. Ennek alapértelmezetten a csúcsok címkéi 0-tól $(2n-1)$-ig tartanak.
A hozzárendelés után a vizuális részét is nekünk kell megtervezni, mivel a gráf elkészítése után a SageMath nem ugyanúgy fogja szemléltetni a gráfokat a futtatások után. Továbbá, előfordulhat, hogy néhány csúcsot ugyanarra a koordinátára fog helyezni a beépített függvény. Viszont lehetőségünk van koordináták segítségével megadni, hogy hogyan szeretnénk ábrázolni a gráfunkat. Ehhez csak ki kell számolni az X, illetve Y koordinátákat. Jelen esetben a két kitüntetett csúcson kívül minden csúcsot kör struktúrában jelenítettem meg, míg a két kimaradó csúcsot a körön kívülre helyeztem, az átláthatóság miatt.

Az n bemenetre csak azt követeljük meg, hogy pozitív egész szám legyen, hiszen már $n = 0$ esetén is létezik gráf. Ez a gráf ebben az esetben a K_2 gráf lesz, azaz a 2 csúcsú teljes gráf, amiről tudjuk, hogy graceful gráf.



4.2. ábra. $B_{5,5}$ gráf



4.3. ábra. $B_{10,10}$ gráf

```
1  def Cap_graph(n):
2      if n>0:
3          labels=Labels(n)
4          G=Graph()
5          G.add_vertex(0)
6          G.add_vertex(2*n+1)
7          G.add_edges([(0,2*n+1)])
8          for i in range(1,2*n+1):
9              G.add_edges([(0,i)])
10         G.relabel(labels,inplace=True)
11         labels.pop(0)
12         labels.pop(0)
13         for i in labels:
14             G.add_edges([(4*n+1,i)])
15         pos_dict={}
16         j=0
17         for i in labels:
18             x = 3*float(2*cos(pi/2 + ((2*pi)/(2*n))*j))
19             y = 3*float(2*sin(pi/2 + ((2*pi)/(2*n))*j))
20             j=j+1
21             pos_dict[i] = [x,y]
22         pos_dict[0]=[10,10]
23         pos_dict[4*n+1]=[-10,10]
24         for u,v,l in G.edges(): G.set_edge_label(u,v,abs(u-v))
25         G.graphplot(pos=pos_dict,edge_color='blue',vertex_color='yellow',
26                 vertex_size=300).show()
27     else:
```

```
28          print ("Ilyen paraméterrel a gráf nem graceful gráf!")
```

4.3. G_n gráfok

4.3.1. Definíció. Tekintsük a $C_n = (V, E, \varphi)$ körgráfot, ahol $V = \{v_1, v_2, \ldots, v_n\}$. Vegyünk egy $v \notin V$ csúcsot, és válasszunk egy $v_i \in C_n$ $(i = 1, \ldots, n)$ tetszőleges csúcsot. Legyen v szomszédos azokkal a csúcsokkal, amellyel a v_i csúcs is szomszédos volt. Jelöljük az így adódó új gráfot G_n-nel.

Megjegyzés. Hasonlóan, mint az előző gráfcsaládnál itt is Samir Vaidya definiálta ezeket a gráfokat. A cikkben [7] azonban a konstruktív bizonyítás során az $n \equiv 0 \pmod 4$ eset hibásan szerepel. Ha az ő általa leírt konstrukciót használnám, akkor $n = 8$-ra még helyes lenne a címkézés, viszont nagyobb n esetén már az éleknek a számozása nem lenne bijektív, így nem lehet graceful a gráf. A hiba az

$$f(v_i) = (n+2) - \left(\frac{i-1}{2}\right),$$

ami helyett a helyes formula a következő lenne:

$$f(v_i) = (n+2) - \left(\frac{i-1}{2}\right) - 1.$$

Továbbá, ugyanebben az esetben, hogyha az $i = \frac{n+4}{2} + 1$, akkor a megadott képlet helytelen eredményt adhat. A helyes címke a v_i csúcs esetén a következő lenne:

$$f(v_i) = \frac{n+i-1}{2}.$$

4.3.1. Tétel. *Ha $3 \leq n$, akkor a G_n gráfok graceful gráfok.*

Bizonyítás. Legyenek $v_1, v_2, \ldots, v_n$ a C_n gráf csúcsai, és készítsük el a fent leírtaknak megfelelően a G gráfot. Tegyük fel, hogy a tetszőlegesen kiválasztott csúcs a v_1 volt, és az új csúcsot jelöljük v-vel. Legyen $f : V \to \{0, 1, \ldots, n+1\}$, és definiáljuk a következőképpen:

(i.) Az $n \equiv 0 \pmod 4$ eset, ahol $n \neq 4$.
Ebben az esetben a következőképpen definiáljuk az f csúcscímkézést:

$$f(v) = 0, \qquad \textit{illetve} \qquad f(v_1) = \frac{n}{2} + 1.$$

Továbbá, ha $2 \leq i \leq \frac{n}{2}$, és i páros:

$$f(v_i) = (n+2) - \left(\frac{i-2}{2}\right).$$

Ellenkező esetben:

$$f(v_i) = \left(\frac{i-1}{2}\right).$$

Ha $i = \frac{n+4}{2} + 1$:

$$f(v_i) = \frac{n+i-1}{2}.$$

Ha $\frac{n}{2} + 3 \leq i \leq n$, és i páratlan:

$$f(v_i) = (n+2) - \left(\frac{i-1}{2}\right) - 1.$$

Ellenkező esetben:

$$f(v_i) = \frac{i-2}{2}.$$

(ii.) Az $n \equiv 1 \pmod 4$ eset.
Ebben az esetben a következőképpen definiáljuk az f csúcscímkézést:

$$f(v) = 0, \qquad \textit{illetve} \qquad f(v_1) = \frac{n+1}{2}.$$

Továbbá, ha $2 \leq i \leq \frac{n+1}{2}$, és i páros:

$$f(v_i) = (n+2) - \left(\frac{i-2}{2}\right).$$

Ellenkező esetben:

$$f(v_i) = \left(\frac{i-1}{2}\right).$$

Ha $\frac{n+3}{2} \leq i \leq n$, és i páratlan:

$$f(v_i) = (n+2) - \left(\frac{i-1}{2}\right).$$

Ellenkező esetben:

$$f(v_i) = \frac{i}{2}.$$

(iii.) Az $n \equiv 2 \pmod 4$ eset.
Ebben az esetben a következőképpen definiáljuk az f csúcscímkézést:

$$f(v) = 0, \qquad \textit{illetve} \qquad f(v_1) = \frac{n}{2} + 3.$$

Továbbá, ha $2 \leq i \leq \frac{n+4}{2}$, és i páros:

$$f(v_i) = (n+2) - \left(\frac{i-2}{2}\right).$$

Ellenkező esetben:

$$f(v_i) = \left(\frac{i-1}{2}\right).$$

Ha $\frac{n+8}{2} \leq i \leq n$, és i páratlan:

$$f(v_i) = (n+2) - \left(\frac{i-3}{2}\right).$$

Ellenkező esetben:

$$f(v_i) = \frac{i+2}{2}.$$

(iv.) Az $n \equiv 3 \pmod 4$ eset.
Ebben az esetben a következőképpen definiáljuk az f csúcscímkézést:

$$f(v) = 0, \qquad \textit{illetve} \qquad f(v_1) = \frac{n+1}{2}.$$

Továbbá, ha $2 \leq i \leq \frac{n+1}{2}$, és i páros:

$$f(v_i) = (n+2) - \left(\frac{i-2}{2}\right).$$

Ellenkező esetben:

$$f(v_i) = \left(\frac{i-1}{2}\right).$$

Ha $\frac{n+3}{2} \leq i \leq n$, és i páratlan:

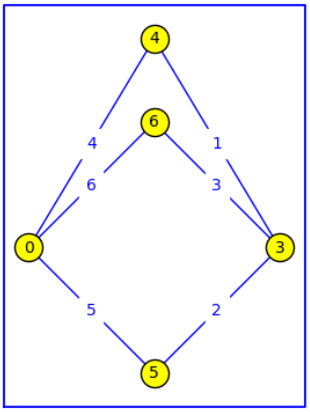
$$f(v_i) = (n+2) - \left(\frac{i-1}{2}\right).$$

Ellenkező esetben:

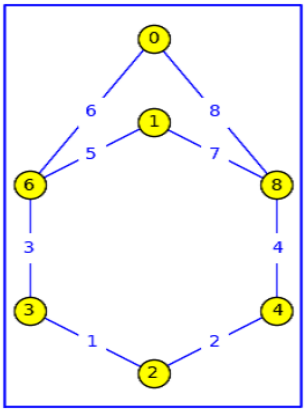
$$f(v_i) = \frac{i}{2}.$$

Az így definiált csúcscímkézés egy f^* élcímkézést indukált, ahol az f^* bijektív, így az f csúcscímkézés egyben graceful címkézés is, tehát a G_n gráf graceful gráf. □

Megjegyzés. A bizonyításban nem esett szó az $n = 4$, és az $n = 6$ esetekről. Ezzel a konstrukcióval bár nem címkézhető jól a gráf, úgy hogy graceful tulajdonságú legyen, viszont megadható olyan csúcscímkézés, ahol teljesül a graceful tulajdonság.



4.4. ábra. G_4 gráf



4.5. ábra. G_6 gráf

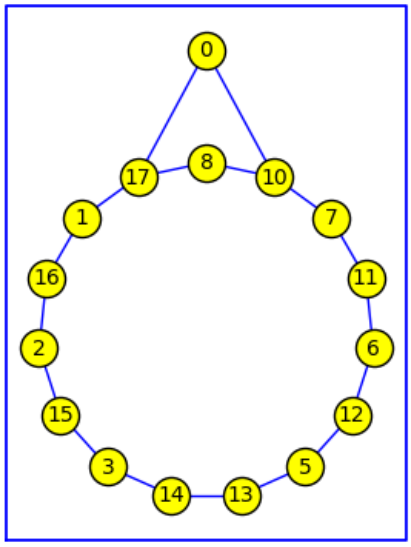
Hasonlóan, mint az előző gráfcsalád esetében, itt is a címkézés implementálásával kell kezdeni, ami a következőképpen néz ki Python programnyelven:

```
1  def Labels(n):
2      L=[]
3      if n==4:
4          L=[4,6,0,5,3]
5      elif n==6:
6          L=[0,1,6,3,2,4,8]
7      if n % 4 == 0 and n!=4 :
8          L.insert(0,0)
9          L.insert(1,n/2+1)
10         for i in range(2,n+1):
11             if 2<=i<=n/2+2:
12                 if i % 2==0:
13                     L.insert(i,(n+2)-(i-2)/2)
14                 else:
15                     L.insert(i,(i-1)/2)
16             elif i==(n+4)/2+1:
17                 L.insert(i,(n+i-1)/2)
18             else:
19                 if i % 2==0:
20                     L.insert(i,(i-2)/2)
21                 else:
22                     L.insert(i,(n+2)-(i-1)/2-1)
23     elif n%4==1 :
24         L.insert(0,0)
25         L.insert(1,(n+1)/2)
26         for i in range(2,n+1):
27             if 2<=i<=(n+1)/2:
```

```
                if i%2==0:
                    L.insert(i,(n+2)-(i-2)/2)
                else:
                    L.insert(i,(i-1)/2)
            else:
                if i%2==0:
                    L.insert(i,i/2)
                else:
                    L.insert(i,(n+2)-(i-1)/2)
    elif n%4==2 and n!=6:
        L.insert(0,0)
        L.insert(1,n/2+3)
        for i in range(2,n+1):
            if 2<=i<=(n+4)/2:
                if i%2==0 :
                    L.insert(i,(n+2)-(i-2)/2)
                else:
                    L.insert(i,(i-1)/2)
            if i==(n+4)/2+1:
                L.insert(i,i/2)
            if (n+8)/2<=i<=n:
                if i%2==0:
                    L.insert(i,(i+2)/2)
                else:
                    L.insert(i,(n+2)-(i-3)/2)
    elif n%4==3:
        L.insert(0,0)
        L.insert(1,(n+1)/2)
        for i in range(2,n+1):
            if 2<=i<=(n+1)/2:
                if i % 2==0:
                    L.insert(i,(n+2)-(i-2)/2)
                else:
                    L.insert(i,(i-1)/2)
            else:
                if i%2==0:
                    L.insert(i,i/2)
                else:
                    L.insert(i,(n+2)-(i-1)/2)

    return L
```

Hasonlóan, mint az előző gráfcsalád esetében először itt is a kört kell elsőnek megkonstruálni. Miután megvan a szokásos rendezést használva, illetve a címkéket felhasználva megkapjuk a G_n gráfot.

```sage
def G_n(n):
    if n>=3:
        labels=Labels(n)
        G=Graph()
        G.add_vertex()
        for i in range(1,n+1):
            if i!=n:
                G.add_edges([(i,i+1)])
            else:
                G.add_edges([(i,1)])
        G.add_edges([(0,2),(0,n)])
        G.relabel(labels,inplace=True)
        pos_dict={}
        labels_0=labels.pop(0)
        j=0
        for i in labels:
            x = 3*float(2*cos(pi/2 + ((2*pi)/(n))*j))
            y = 3*float(2*sin(pi/2 + ((2*pi)/(n))*j))
            j=j+1
            pos_dict[i] = [x,y]
        pos_dict[labels_0]=[0,10]
        for u,v,l in G.edges(): G.set_edge_label(u,v,abs(u-v))
        G.graphplot(pos=pos_dict,edge_labels=True,
                    edge_color='blue',vertex_color='yellow',
                    vertex_size=300,graph_border=True).show()
```



4.6. ábra. G_{15} gráf

4.4. Medál gráfok

Medál gráfokkal (angol nevén Pendant Graphs), már a szakdolgozatomban is foglalkoztam, ahol különböző eseteket mutattam be, amelyek függtek a legkisebb számú címke elhelyezésétől. A most bemutatott formulát 2000-ben Watson ismertette dolgozatában [8].

4.4.1. Definíció. Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf. V csúcsai legyenek rendre $v_0, v_1, \ldots, v_{n-1}, u_0, u_1, \ldots, u_{n-1}$. Ha $0 \leq i \leq n-2$, akkor a v_i csúcs legyen szomszédos a v_{i+1} csúccsal,illetve ha $i = n-1$, akkor v_{n-1} legyen szomszédos a v_0 csúccsal. Továbbá, minden $0 \leq i \leq n-1$ esetén a v_i legyen szomszédos az u_i-vel. Az így konstruált gráfot medál gráfnak nevezzük. Jelöljük M_n-nel.

4.4.1. Tétel. *Az M_n gráfok graceful gráfok.*

Bizonyítás. A bizonyítás konstruktív módon fog történni. Legyenek M_n csúcsai rendre $v_1, v_2, \ldots, v_n, u_1, u_2, \ldots, u_n$.

(i.) Az $n \equiv 0 \pmod 4$ eset.
Ebben az esetben a következőképpen definiáljuk az f csúcscímkézést:

$$f(v_i) = \begin{cases} i-1 & , \; ha \; i = 1, 3, \ldots, \frac{n}{2} - 1 \\ i & , \; ha \; i = \frac{n}{2} + 1, \frac{n}{3} + 3, \ldots, n-1 \\ 2n+1-i & , \; ha \; i = 2, 4, \ldots, n, \end{cases}$$

illetve

$$f(u_i) = \begin{cases} 2n - f(v_i) & , \; ha \; i = 1, 2, \ldots, \frac{n}{2} \\ 2n+1-f(v_i) & , \; ha \; i = \frac{n}{2}+1, \frac{n}{2}+2, \ldots, n. \end{cases}$$

(ii.) Az $n \equiv 1 \pmod 4$ eset.
Ebben az esetben a következőképpen definiáljuk az f csúcscímkézést:

$$f(v_i) = \begin{cases} i-1 & , \; ha \; i = 1, 3, \ldots, n, \; ahol \; i \neq \frac{n+1}{2} \\ n-3 & , \; ha \; i = \frac{n+1}{2} \\ 2n+1-i & , \; ha \; i = 2, 4, \ldots, \frac{n-1}{2} \\ 2n-i & , \; ha \; i = \frac{n+3}{2}, \frac{n+7}{2}, \ldots, n-1, \end{cases}$$

illetve

$$f(u_i) = \begin{cases} 2n - f(v_i) & , \; ha \; i = 1, 2, \ldots, \frac{n-3}{2} \\ 2n+1-f(v_i) & , \; ha \; i = \frac{n-1}{2} \\ 2n-2-f(v_i) & , \; ha \; i = \frac{n+1}{2} \\ 2n-1-f(v_i) & , \; ha \; i = \frac{n+3}{2}, \frac{n+5}{2}, \ldots, n. \end{cases}$$

(iii.) Az $n \equiv 2 \pmod 4$ eset.
Ebben az esetben a következőképpen definiáljuk az f csúcscímkézést:

$$f(v_i) = \begin{cases} i-1 & , \; ha \; i = 1,3,\ldots,n, \; ahol \; i \neq \frac{n}{2} \\ \frac{n}{2}-2 & , \; ha \; i = \frac{n}{2} \\ 2n+1-i & , \; ha \; i = 2,4,\ldots,\frac{n}{2}-1 \\ 2n-i & , \; ha \; i = \frac{n}{2}+1, \frac{n}{2}+3,\ldots,n, \end{cases}$$

illetve

$$f(u_i) = \begin{cases} 2n-f(v_i) & , \; ha \; i = 1,2,\ldots,\frac{n}{2} \\ 2n+1-f(v_i) & , \; ha \; i = \frac{n}{2}-1 \\ 2n-2-f(v_i) & , \; ha \; i = \frac{n}{2} \\ 2n-1-f(v_i) & , \; ha \; i = \frac{n}{2}+1, \frac{n}{2}+2,\ldots,n. \end{cases}$$

(iv.) Az $n \equiv 3 \pmod 4$ eset.
Ebben az esetben a következőképpen definiáljuk az f csúcscímkézést:

$$f(v_i) = \begin{cases} i-1 & , \; ha \; i = 1,3,\ldots,\frac{n-1}{2} \\ i & , \; ha \; i = \frac{n+3}{2}, \frac{n+7}{2},\ldots,n \\ 2n+1-i & , \; ha \; i = 2,4,\ldots,n-1, \end{cases}$$

illetve

$$f(u_i) = \begin{cases} 2n-f(v_i) & , \; ha \; i = 1,2,\ldots,\frac{n+1}{2} \\ 2n+1-f(v_i) & , \; ha \; i = \frac{n+3}{2}, \frac{n+5}{2},\ldots,n. \end{cases}$$

Az így meghatározott f csúcscímkézés egy f^* élcímkézést indukál, ami bijektív, így teljesül a graceful tulajdonság, tehát M_n graceful gráf. □

Megjegyzés. A felhasznált irodalomban a bizonyítás során itt is elírás történt, mégpedig $n \equiv 3 \pmod 4$ esetben, ahol a helyes érték $2n-i$, és nem $2n-1$. Továbbá, $n \equiv 1 \pmod 4$ esetben, a szerző $\frac{n-3}{2}$ értéket adott meg, viszont a helyes érték $n-3$.

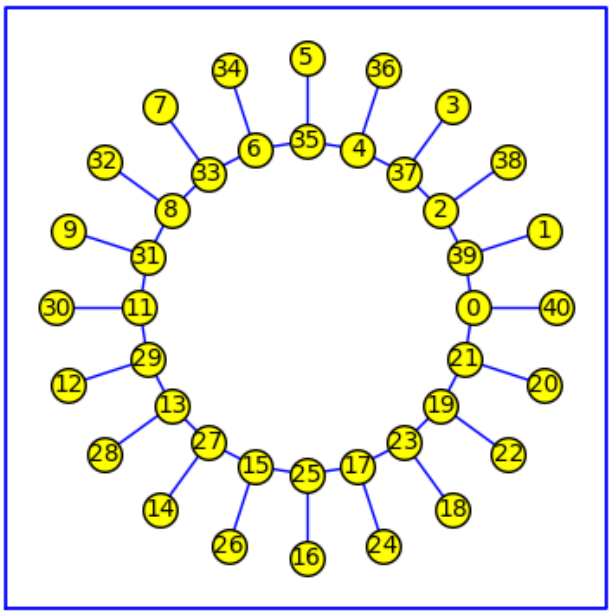
```
1  def Labels(n):
2      L=[]
3      if n % 4==0:
4          for i in range (1,n+1):
5              if i<=(n/2)-1 and i%2!=0 :
6                  L.insert(i-1,i-1)
7              elif i<=n-1 and i%2!=0 and i>n/2:
8                  L.insert(i-1,i)
9              else:
10                 L.insert(i-1,2*n+1-i)
11         for j in range (1,n+1):
```

```
            if j<=n/2:
                L.insert(n+j-1,2*n-L[j-1])
            else:
                L.insert(n+j-1,2*n+1-L[j-1])
    if n % 4==1:
        for i in range(1,n+1):
            if i!=(n+1)/2 and i%2!=0:
                L.insert(i-1,i-1)
            elif i==(n+1)/2:
                L.insert(i-1,(n-3)/2)
            elif i<=(n-1)/2 and i%2==0:
                L.insert(i-1,2*n+1-i)
            else:
                L.insert(i-1,2*n-i)
        for j in range(1,n+1):
            if j<=(n-3)/2:
                L.insert(n+j-1,2*n-L[j-1])
            elif j==(n-1)/2:
                L.insert(n+j-1,2*n+1-L[j-1])
            elif j==(n+1)/2:
                L.insert(n+j-1,2*n-2-L[j-1])
            else:
                L.insert(n+j-1,2*n-1-L[j-1])
    if n % 4==2:
        for i in range(1,n+1):
            if i!=n/2 and i%2!=0:
                L.insert(i-1,i-1)
            elif i==n/2:
                L.insert(i-1,n/2-2)
            elif i<=n/2-1 and i%2==0:
                L.insert(i-1,2*n+1-i)
            else:
                L.insert(i-1,2*n-i)
        for j in range(1,n+1):
            if j<=n/2-2:
                L.insert(n+j-1,2*n-L[j-1])
            elif j==n/2-1:
                L.insert(n+j-1,2*n+1-L[j-1])
            elif j==n/2:
                L.insert(n+j-1,2*n-2-L[j-1])
            else:
                L.insert(n+j-1,2*n-1-L[j-1])
    if n % 4==3:
        for i in range(1,n+1):
            if i<=(n-1)/2 and i%2!=0:
                L.insert(i-1,i-1)
```

```
58              elif i<=n-1 and i%2==0:
59                  L.insert(i-1,2*n+1-i)
60              else:
61                  L.insert(i-1,i)
62          for j in range(1,n+1):
63              if j<=(n+1)/2:
64                  L.insert(n+j-1,2*n-L[j-1])
65              else:
66                  L.insert(n+j-1,2*n+1-L[j-1])
67      return L
```

A gráf megalkotása:

```
1  def Graph(n):
2      G=graphs.CycleGraph(n)
3      for i in range(0,n):
4          G.add_vertex()
5          G.add_edge((i,n+i))
6      return G
```



4.7. ábra. M_{20} gráf

A gráf ábrázolása:

```
def draw(G, L):
    temp=n
    R1=20
    R2=30
    pos_dict = {}
    for i in range(len(L)/2):
        x=R1*float(cos(((2*pi)/(len(L)/2))*i))
        y=R1*float(sin(((2*pi)/(len(L)/2))*i))
        pos_dict[L[i]] = [x,y]
    for i in range(len(L)/2,len(L)):
        x=R2*float(cos(((2*pi)/(len(L)/2))*i))
        y=R2*float(sin(((2*pi)/(len(L)/2))*i))
        pos_dict[L[i]] = [x,y]
    G.relabel( [ L[i] for i in range(G.order()) ], inplace=True)
    for u,v,l in G.edges():
        G.set_edge_label(u,v,abs(u-v))
        pl = G.graphplot(pos=pos_dict,edge_color='blue',
                vertex_color='yellow',vertex_size=200,graph_border=True)

    pl.show(figsize=[4,7])
```

4.5. Álkerék gráfok

A szakdolgozatomban különböző speciális kerék gráfokat vizsgáltam, amiről belátható volt, hogy létezik graceful címkézése ezeknek a gráfoknak. Most viszont nem új kerekekkel fogom bővíteni a gráfot, hanem épp szűkíteni fogom azt. A most bemutatott gráfokat G. Sethuraman és K. Sankar definiálták 2015-ben kiadott cikkükben [9].

4.5.1. Definíció. Legyen $SW_n = (V, E, \varphi)$ egy véges, egyszerű gráf. Induljunk ki a $W(n)$ kerékgráfból. Jelöljük el a középponti csúcsot u_0-val, a többi csúcs pedig legyen rendre $u_1, u_2, \ldots, u_n$. Vegyük azokat az éleket, amelyekre u_0 illeszkedik, és osszuk fel azokat. Továbbá, a maradék éleket is osszuk fel hasonlóan. Azaz $V = \{u_0, u_1, u_2, \ldots, u_n, u_{n+1} \ldots, u_{2n}, w_1, w_2 \ldots, w_n\}$ adódik, ahol w_i jelöli a maradék élek felosztása során keletkezett újabb csúcsokat.$(i = 1, \ldots, n)$ Az így megalkotott SW_n gráfot, álkerék gráfnak nevezzük, ahol n a kiinduló kerék gráfnak a csúcsszáma.

4.5.1. Tétel. *Ha $4 \leq n$, és páros, akkor az SW_n gráf graceful gráf.*

Bizonyítás. Az SW_n gráf csúcsainak száma $3n+1$, míg éleinek száma $4n$. Jelöljük m-mel az éleknek a számát. Továbbá, legyen $f : V(SW_n) \to \{0, 1, \ldots, m\}$ csúcs-

címkézés, és definiáljuk a a következőképpen:

$$f(u_0) = 0,$$

$$f(u_i) = \begin{cases} m+1-i & , ha\ 1 \leq i \leq n \\ 2i-1-2n & , ha\ n+1 \leq i \leq 2n. \end{cases}$$

Továbbá,

$$f(w_i) = \begin{cases} 6i-2 & , ha\ 1 \leq i \leq \frac{n}{2} \\ 4n+1-2i & , ha\ \frac{n}{2}+1 \leq i \leq n. \end{cases}$$

Az így megkonstruált f függvény egy f^* bijektív függvényt indukál, amellyel teljesül a graceful tulajdonság, így SW_n graceful gráf. □

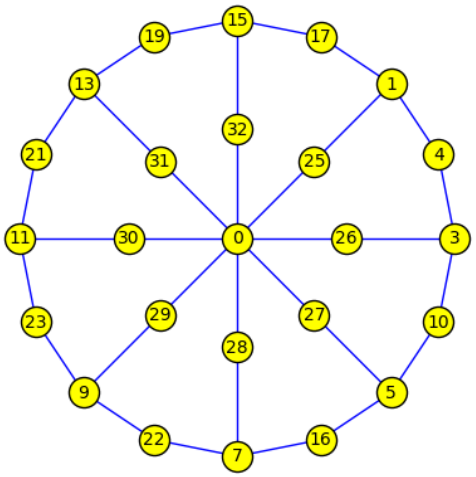
A cikkben csak a páros esetet mutatták be a szerzők, a páratlan esettel nem foglalkoztak, viszont leírták, hogy az a sejtésük, hogy páratlan esetben is ezek a gráfok graceful gráfok.
Megvizsgáltam a backtrack-algoritmussal az $n = 5$, illetve $n = 7$. A backtrack-algoritmus segítségével pillanatok alatt megoldáshoz juthatunk. $n = 5$ esetén legyen a központi csúcsunk v, a belső csúcsok legyenek $v_1, \ldots, v_5$, míg a külső gyűrűn lévő csúcsok legyenek $v_6, v_7, \ldots, v_{15}$. A csúcsok címkéit reprezentáljuk egy tömbben, azaz $[f(v), f(v_1), f(v_2), \ldots, f(v_{14}), f(v_{15})]$.
Ebben az esetben egy lehetséges graceful címkézés a következőképpen néz ki: $[0, 20, 12, 17, 16, 15, 1, 2, 4, 7, 3, 10, 5, 18, 9, 19]$.
Hasonlóan $n = 7$ esetén tömb reprezentációban, a graceful címkézés a következőképpen néz ki: $[0, 16, 27, 24, 26, 22, 25, 28, 1, 2, 4, 7, 3, 8, 14, 5, 12, 20, 6, 23, 10, 21]$.
Továbbá, az implementálás első lépése még mindig a címkéknek a legyártása lesz. Viszont a gondot épp a csúcscímkézés definiálása okozza, hiszen a listába az i változó szerint kerülnek bele a címkék. A pozíció megadásánál, viszont nem lehet ezeket egyszerűen csak ábrázolni. Részlistákat kell létrehozni, amely segítségével először a belső gyűrűt fogjuk megalkotni, majd a külső gyűrűt. Először a központi csúcsnak adjuk a címkét, amely minden esetben 0 címke fog lenni. Majd felcímkézzük a belső gyűrűt az egyik részlista címkéivel, illetve hasonlóan a külső gyűrűt a másik részlista címkéivel.

```
1  def Labels(n):
2      L=[]
3      L.insert(3*n+2,0)
4      for i in range(1,2*n+1):
5          if 1<=i<=n:
6              L.insert(i,4*n+1-i)
7          elif n+1<=i<=2*n:
8              L.insert(i,2*i-1-2*n)
9      for j in range(1,n+1):
10         if 1<=j<=n/2:
```

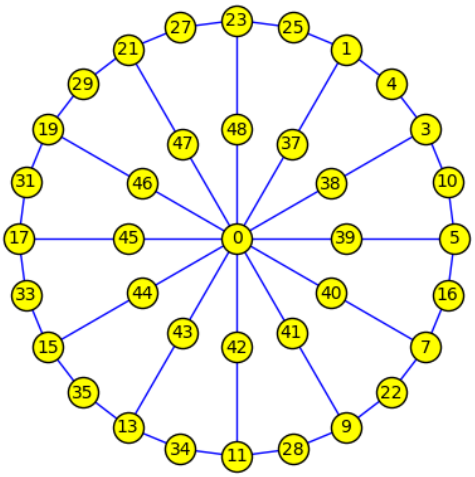
```
            L.insert(j+2*n+1,6*j-2)
        elif (n/2)+1<=j<=n:
            L.insert(j+2*n+1,4*n+1-2*j)
    return L
```

```
def graph(n):
    labels=Labels(n)
    L=Labels(n)
    temp_0=L[n+1:2*n+1]
    temp_1=L[2*n+1:3*n+1]
    Z=L[1:n+1]
    T=L[n+1:3*n+1]
    K=[]
    j=len(K)
    for i in range(0,n):
        K.insert(j,temp_0[i])
        j=j+1
        K.insert(j,temp_1[i])
        j=j+1
    t=K.pop(len(K)-1)
    K.reverse()
    K.insert(len(K),t)
    G=Graph()
    G.add_vertex(0)
    for i in range(1,n+1):
        G.add_vertex(i)
    for i in range(n+1,3*n+1):
        G.add_vertex(i)
    G.relabel(labels,inplace=True)
    for i in range(0,len(K)):
        if i!=len(K)-1:
            G.add_edges([(K[i],K[i+1])])
        else:
            G.add_edges([(K[i],K[0])])
    j=0
    szamlalo=0
    for i in range(0,len(K),2):
        G.add_edges([(K[i],Z[szamlalo])])
        szamlalo=szamlalo+1
        j=j+1

    pos_dict={}
    j=0
    for i in Z:
        x = 25*float(2*cos(pi/2 + ((pi)/(len(Z)/2))*j))
```

```
        y = 25*float(2*sin(pi/2 + ((pi)/(len(Z)/2))*j))
        j=j+1
        pos_dict[i] = [x,y]

    j=0
    for i in K:
        x = 50*float(2*cos(pi/2 + ((pi)/(len(K)/2))*j))
        y = 50*float(2*sin(pi/2 + ((pi)/(len(K)/2))*j))
        j=j+1
        pos_dict[i] = [x,y]
    pos_dict[0]=[0,0]

    for i in range(4*n,3*n,-1):
        G.add_edges([(0,i)])
    G.graphplot(pos=pos_dict,edge_color='blue',vertex_color='yellow',
        vertex_size=300).show()
def sub_wheel(n):
    if n%2==0:
        graph(n)
    else:
        return "Páratlan esetre ez a konstrukció nem hajtható végre!"
```



4.8. ábra. SW_8 gráf



4.9. ábra. SW_{12} gráf

4.6. Sín gráfok

4.6.1. Definíció. Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf. Tekintsük a P_k útgráfot. Az útgráf minden csúcsán végezzünk zárt fül ragasztást ugyanolyan m hosszú fülekkel. Ezen fülek nem tartalmaznak közös csúcsot. Az így kapott $[P_k, C_m]$ gráfot sín gráfnak nevezzük, és jelöljük $S_{k,m}$-mel.

Megjegyzés. Ezeket a gráfokat 2018-ban két indiai kutató S. Venkatesh és N.Amirthavahini mutatta be először cikkükben [10].

4.6.1. Tétel. *Az $S_{k,m}$ gráf pontosan akkor graceful, hogyha* $m \equiv 0 \pmod 4$.

Bizonyítás. Vizsgáljuk meg az $S_{k,m}$ gráfot. Erre a gráfra a következőek igazak: $p := |V(S_{k,m})| = mk$, és $q := |E(S_{k,m})| = mk + k - 1$. Továbbá, a gráf csúcsai legyenek $i = 1, 2, \ldots, k$ esetén $v_{i,1}, v_{i,2} \ldots, v_{i,m}$. Definiáljuk az f csúcscímkézést a következőképpen:

$$f(v_{1,1}) = q, \qquad \textit{illetve} \qquad f(v_{1,2}) = 0.$$

Ha j páratlan, akkor

$$f(v_{1,j}) = \begin{cases} q - (\frac{j-1}{2}) & , \textit{ ha } 3 \leq j \leq \frac{m}{2} \\ q - (\frac{j+1}{2}) & , \textit{ ha } \frac{m}{2} < j \leq m. \end{cases}$$

Ha j páros, és $2 \leq j \leq m$, akkor

$$f(v_{1,j}) = \frac{j}{2} - 1.$$

Ha $2 \leq i \leq k$, és i páros, akkor

$$f(v_{i,1}) = f(v_{i-1,m-1}) - 1,$$

$$f(v_{i,2}) = f(v_{i-1,m}) + 1,$$

$$f(v_{i,j}) = f(v_{i,1}) - \left(\frac{j-1}{2}\right), \textit{ ha } 3 \leq j \leq m-1 \textit{ és } j \textit{ páratlan},$$

illetve

$$f(v_{i,j}) = \begin{cases} f(v_{i,2}) + \frac{j}{2} - 1 & , \textit{ ha } 4 \leq j \leq \frac{m}{2} \textit{ és } j \textit{ páros} \\ f(v_{i,2}) + \frac{j}{2} & , \textit{ ha } \frac{m}{2} \leq j \leq m \textit{ és } j \textit{ páros}. \end{cases}$$

Ha $3 \leq i \leq k$, és i páratlan, akkor

$$f(v_{i,1}) = f(v_{i-1,m-1}) - 1,$$

$$f(v_{i,2}) = f(v_{i-1,m}) + 1,$$

$$f(v_{i,j}) = f(v_{i,2}) + \left(\frac{j-2}{2}\right), \textit{ ha } 4 \leq j \leq m \textit{ és } j \textit{ páros},$$

illetve

$$f(v_{i,j}) = \begin{cases} f(v_{i,1}) + \frac{j-1}{2} - 1 & , \textit{ ha } 3 \leq j \leq \frac{m}{2} \textit{ és } j \textit{ páratlan} \\ f(v_{i,1}) + \frac{j+1}{2} & , \textit{ ha } \frac{m}{2} \leq j \leq m \textit{ és } j \textit{ páratlan}. \end{cases}$$

Az így konstruált f függvény egy f^* bijektív élcímkézést indukál, amiből következik, hogy teljesül a graceful tulajdonság, így $S_{k,m}$ graceful gráf. □

A címkézés implementálása a fentebb látható módon történik.

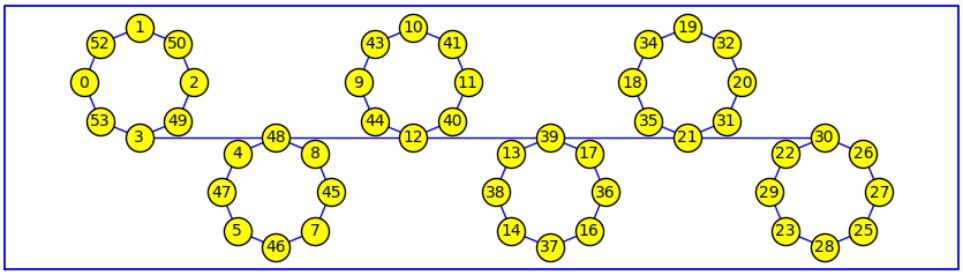
```
def Labels(k,m):
    L=[]
    q=m*k+k-1
    L.insert(0,q)
    szamlalo=1
    for j in range(2,m+1):
        if j%2!=0 and 3<=j<=m/2:
            L.insert(szamlalo,q-(j-1)/2)
            szamlalo=szamlalo+1
        elif j%2!=0 and m/2<j<=m:
            L.insert(szamlalo,q-(j+1)/2)
            szamlalo=szamlalo+1
        else:
            L.insert(szamlalo,j/2-1)
            szamlalo=szamlalo+1
    for i in range(2,k+1):
        if i%2==0:
            L.insert(szamlalo,L[szamlalo-1]+1)
            szamlalo=szamlalo+1
            temp_0=L[szamlalo-3]-1
            temp_1=L[szamlalo-1]
            for j in range(3,m+1):
                if j%2!=0:
                    L.insert(szamlalo,temp_0-(j-1)/2)
                    szamlalo=szamlalo+1
                elif j%2==0 and 4<=j<=m/2:
                    L.insert(szamlalo,temp_1+j/2-1)
                    szamlalo=szamlalo+1
                else:
                    L.insert(szamlalo,temp_1+j/2)
                    szamlalo=szamlalo+1
            L.insert(szamlalo,temp_0)
            szamlalo=szamlalo+1
        else:
            L.insert(szamlalo,L[szamlalo-3]-1)
            szamlalo=szamlalo+1
            L.insert(szamlalo,L[szamlalo-3]+1)
            szamlalo=szamlalo+1
            temp_0=L[szamlalo-2]
            temp_1=L[szamlalo-1]
            for j in range(3,m+1):
                if j%2!=0 and 3<=j<=m/2:
                    L.insert(szamlalo,temp_0-(j-1)/2)
                    szamlalo=szamlalo+1
                elif j%2!=0 and m/2<j<=m:
                    L.insert(szamlalo,temp_0-(j+1)/2)
```

```
                    szamlalo=szamlalo+1
                else:
                    L.insert(szamlalo,temp_1+(j-2)/2)
                    szamlalo=szamlalo+1
    return L
```

A gráf konstrukciója során először az útgráfot készítjük el, majd zárt füleket ragasztunk rá. A pozicionálás eltolásokkal, és tükrözésekkel valósul meg, így a végső gráfunk ábrázolása hasonlít az informatikában használt sín/busz topológiára.

```
def bus_graph(k,m):
    if m%4==0:
        G=Graph()
        labels=Labels(k,m)
        temp=0
        while(temp!=k):
            for i in range(0+temp*m,m+temp*m):
                if i!=m-1+temp*m:
                    G.add_edges([(i,i+1)])
                else:
                    G.add_edges([(i,0+temp*m)])
            temp=temp+1
        for i in range(0,k-1):
            G.add_edges([((m-1)+i*m,2*m-1+i*m)])
        G.relabel(labels,inplace=True)
        pos_dict={}
        S=[]
        szamlalo=0
        for i in range(0,k):
            S.insert(szamlalo,labels[(m-1)+i*m])
            szamlalo=szamlalo+1
        pos_dict={}
        x=0
        for i in S:
            x=x+100
            y=0
            pos_dict[i] = [x,y]
        j=1
        r=20
        temp=0
        fix=0
        valto=1
        while(temp!=k*m):
            if valto%2==0:
                for i in range(0+temp,m+temp):
                    x = r*float(2*cos(pi/2 + ((2*pi)/(m))*j))+fix
```

```
                    y = r*float(2*sin(pi/2 + ((2*pi)/(m))*j))
                    j=j+1
                    pos_dict[labels[i]] = [x,y]
                fix=fix+100
                temp=temp+m
                valto=valto+1
            else:
                for i in range(0+temp,m+temp):
                    x = r*float(2*cos(pi/2 + ((2*pi)/(m))*j))+fix
                    y = r*float(2*sin(pi/2 + ((2*pi)/(m))*j))-80
                    j=j+1
                    pos_dict[labels[i]] = [x,-y]
                fix=fix+100
                temp=temp+m
                valto=valto+1
        e_edge=[]
        szamlalo=0
        for u,v,l in G.edges():
            e_edge.insert(szamlalo,abs(u-v))
            szamlalo=szamlalo+1
            G.set_edge_label(u,v,abs(u-v))
        e_edge.sort()
        print("Élek_címkéi: ")
        print(e_edge)

        for u,v,l in G.edges(): G.set_edge_label(u,v,abs(u-v))
        G.graphplot(pos=pos_dict,edge_color='blue',
        vertex_color='yellow',vertex_size=300).show(figsize=[10,20])
    else:
        print ("Sajnos nem felel meg az m érték a feltételeknek!")
```



4.10. ábra. $S_{6,8}$ gráf

4.7. $C_t(C_n)$ gráfok

4.7.1. Definíció. Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf. Tekintsük a C_{n_0} körgráfot. A körgráf minden csúcsán végezzünk zárt fül ragasztást ugyanolyan n hosszú fülekkel. Ezen fülek nem tartalmaznak közös csúcsot. Az így kapott $[C_{n_0}, C_n]$ gráfot körlánc gráfnak nevezzük.

4.7.1. Tétel. *A $[C_{n_0}, C_n]$ gráf, pontosan akkor graceful gráf, ha $n_0, n \equiv 0 \pmod 4$.*

Bizonyítás. Legyenek a C_{n_0} gráf csúcsai rendre $v_1, v_2, \ldots, v_{n_0}$, míg C_n csúcsai rendre $v_{i,1}, v_{i,2}, \ldots, v_{i,n}$, ahol i jelöli, hogy melyik körben található az adott csúcs. $p = n_0 + n_0(n-1)$ csúcsok száma, és $q = n_0 + n_0 n$ az élek száma. Definiáljuk az f csúcscímkét a következőképpen, ahol $f : V([C_{n_0}, C_n] \to \{1, 2, \ldots, q\}$:

$$f(v_{1,1}) = q, \qquad \textit{illetve} \qquad f(v_{1,2}) = 0.$$

$$f(v_{1,j}) = \begin{cases} f(v_{1,1}) + \frac{j-1}{2} & , \textit{ ha } j < \frac{n}{2} \textit{ és } j \textit{ páratlan} \\ f(v_{i,1}) - \frac{j+1}{2} & , \textit{ ha } \frac{n}{2} \leq j \textit{ és } j \textit{ páratlan} \\ \frac{j}{2} - 1 & , \textit{ ha } j \textit{ páros.} \end{cases}$$

Továbbá, ha $3 \leq i \leq n_0$, i páratlan, és $i \neq \frac{n_0}{2} + 1$, akkor

$$f(v_{i,1}) = f(v_{i-1,n-1}) - 1, \qquad \textit{illetve} \qquad f(v_{i,2}) = f(v_{i-1,n}) + 1.$$

Ha $i = \frac{n_0}{2} + 1$, akkor

$$f(v_{i,2}) = f(v_{i-1,n}) + 2.$$

Ha $3 \leq j \leq n$, akkor

$$f(v_{i,j}) = \begin{cases} f(v_{i,1}) + \frac{j-1}{2} & , \textit{ ha } j < \frac{n}{2} \textit{ és } j \textit{ páratlan} \\ f(v_{i,1}) - \frac{j+1}{2} & , \textit{ ha } \frac{n}{2} \leq j \textit{ és } j \textit{ páratlan} \\ f(v_{i,2}) + \frac{j}{2} - 1 & , \textit{ ha } j \textit{ páros.} \end{cases}$$

Továbbá, ha $2 \leq i \leq n_0$, i páros, és $i \neq \frac{n_0}{2} + 1$, akkor

$$f(v_{i,1}) = f(v_{i-1,n-1}) - 1, \qquad \textit{illetve} \qquad f(v_{i,2}) = f(v_{i-1,n}) + 1.$$

Ha $i = \frac{n_0}{2} + 1$, akkor

$$f(v_{i,2}) = f(v_{i-1,n}) + 2.$$

Ha $3 \leq j \leq n$, akkor

$$f(v_{i,j}) = \begin{cases} f(v_{i,1}) + \frac{j-1}{2} & , \textit{ ha } j \textit{ páros} \\ f(v_{i,2}) + \frac{j}{2} - 1 & , \textit{ ha } 4 \leq j \leq \frac{n}{2} \textit{ és } j \textit{ páros} \\ f(v_{i,2}) + \frac{j}{2} & , \textit{ ha } j \textit{ páros és } \frac{n}{2} < j. \end{cases}$$

Az így definiált f csúcscímkézés egy f^* bijektív élcímkézést indukál, amelyből következik, hogy teljesül a graceful tulajdonság, így a gráf graceful gráf. □

A címkézés implementálása a szokásos módon zajlik. Viszont az implementálás során a cikkben szereplő formulát alkalmazva nem sikerült legyártani a címkéket, mivel a cikkben egy elírás történt a bizonyítás páros esetében. Ez csak egy szimpla indexelési hiba, ahol a helyes érték az $f(v_{i,2})$ lenne, és nem az $f(v_{i,1})$.

```
def Labels(n_0,n):
    L=[]
    p=n_0+n_0*(n-1)
    q=n_0+n_0*n
    szamlalo=0
    L.insert(szamlalo,q)
    szamlalo=szamlalo+1
    L.insert(szamlalo,0)
    szamlalo=szamlalo+1
    for j in range(3,n+1):
        if j%2!=0 and j<n/2:
            L.insert(szamlalo,L[0]-(j-1)/2)
            szamlalo=szamlalo+1
        elif j%2!=0 and j>=n/2:
            L.insert(szamlalo,L[0]-(j+1)/2)
            szamlalo=szamlalo+1
        else:
            L.insert(szamlalo,j/2-1)
            szamlalo=szamlalo+1
    for i in range(2,n_0+1):
        if i % 2 ==0:
            L.insert(szamlalo,L[len(L)-2]-1)
            szamlalo=szamlalo+1
            if i!=n_0/2+1:
                L.insert(szamlalo,L[len(L)-2]+1)
                szamlalo=szamlalo+1
            else:
                L.insert(szamlalo,L[len(L)-2]+2)
                szamlalo=szamlalo+1
            temp_1=L[len(L)-2]
            temp_2=L[len(L)-1]
            for j in range(3,n+1):
                if j%2!=0:
                    L.insert(szamlalo,temp_1-(j-1)/2)
                    szamlalo=szamlalo+1
                elif j%2==0 and 4<=j<=n/2:
                    L.insert(szamlalo,temp_2+j/2-1)
                    szamlalo=szamlalo+1
                else:
                    L.insert(szamlalo,temp_2+j/2)
                    szamlalo=szamlalo+1
```

```
        else:
            L.insert(szamlalo,L[len(L)-2]-1)
            szamlalo=szamlalo+1
            if i!=n_0/2+1:
                L.insert(szamlalo,L[len(L)-2]+1)
                szamlalo=szamlalo+1
            else:
                L.insert(szamlalo,L[len(L)-2]+2)
                szamlalo=szamlalo+1
            temp_1=L[len(L)-2]
            temp_2=L[len(L)-1]
            for j in range(3,n+1):
                if j%2!=0 and j<n/2:
                    L.insert(szamlalo,temp_1-(j-1)/2)
                    szamlalo=szamlalo+1
                elif j%2!=0 and j>=n/2:
                    L.insert(szamlalo,temp_1-(j+1)/2)
                    szamlalo=szamlalo+1
                else:
                    L.insert(szamlalo,temp_2+j/2-1)
                    szamlalo=szamlalo+1
    return L
```

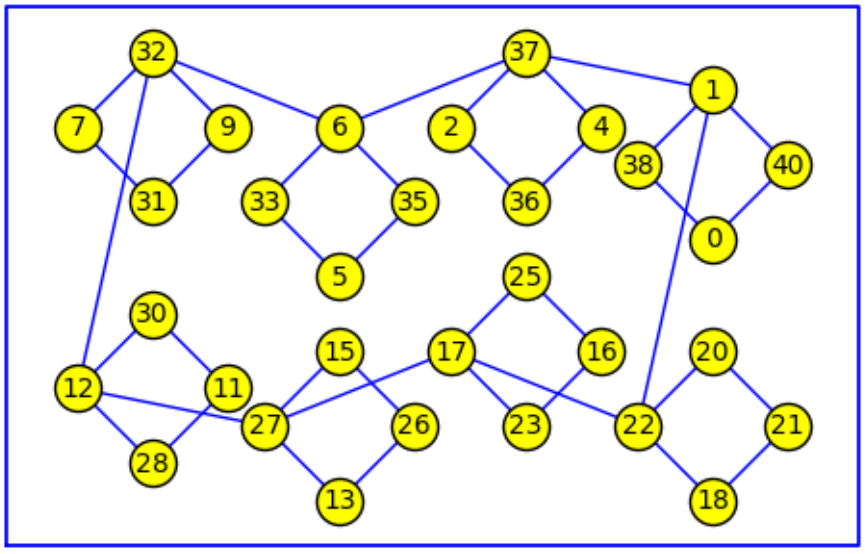
Mivel egy viszonylag sok csúcsot tartalmazó gráfról beszélünk, így a pozicionálás nem egyszerű. Az egyszerű körök pozicionálása a szokásos módon történt, viszont az átláthatóság miatt, jobbnak láttam, hogyha az alap C_{n_0} gráfot nem körnek ábrázolom, hanem bizonyos esetekben cikk-cakkosan.

```
def cycles(n_0,n):
    labels=Labels(n_0,n)
    G=Graph()

    db=0
    temp=0
    while(db<n_0):
        for i in range(0+temp,n+temp):
            if i!=n-1+temp:
                G.add_edges([(i,i+1)])
            else:
                G.add_edges([(i,0+temp)])
        temp=temp+n
        db=db+1


    temp=1
    while(1):
```

```
        G.add_edges([(temp*n-1,temp*n)])
        if (temp*n<len(G.vertices())-n):
            G.add_edges([(temp*n,(temp+2)*n-1)])
            temp=temp+2
        else:
            break

    G.add_edges([(n-1,(n_0-1)*n)])
    G.relabel(labels,inplace=True)

    csucsok=[]
    for i in range(0,n_0):
        csucsok.insert(i,[])

    szamlalo=0
    db=0
    tul=0
    while(tul<len(csucsok)):
        for j in range(0,n):
            csucsok[tul].insert(szamlalo,labels[db])
            szamlalo=szamlalo+1
            db=db+1
        tul=tul+1
    for i in range(0,len(csucsok)):
        if i%2==0:
            csucsok[i].reverse()

    pos_dict={}
    if n_0==4:
        j=0
        for i in csucsok[0]:
            x = float(cos(pi/2 + ((2*pi)/len(csucsok[0]))*j))
            y = float(sin(pi/2 + ((2*pi)/len(csucsok[0]))*j))
            j=j+1
            pos_dict[i] = [x-3,-y+2]
        j=0
        for i in csucsok[1]:
            x = float(cos(pi/2 + ((2*pi)/len(csucsok[1]))*j))
            y = float(sin(pi/2 + ((2*pi)/len(csucsok[1]))*j))
            j=j+1
            pos_dict[i] = [x-3,y-2]
        j=0
        for i in csucsok[2]:
            x = float(cos(pi/2 + ((2*pi)/len(csucsok[2]))*j))
            y = float(sin(pi/2 + ((2*pi)/len(csucsok[2]))*j))
            j=j+1
```

```
65              pos_dict[i] = [x,y-2]
66          j=0
67          for i in csucsok[3]:
68              x = float(cos(pi/2 + ((2*pi)/len(csucsok[3]))*j))
69              y = float(sin(pi/2 + ((2*pi)/len(csucsok[3]))*j))
70              j=j+1
71              pos_dict[i] = [x,-y+2]
72      else:
73          db=0
74          s=0
75          fix_0=(n_0/2-1)*5
76          fix_1=4
77          while(db<n_0/2):
78              for i in csucsok[s]:
79                  x = 2*float(cos(pi + ((2*pi)/len(csucsok[0]))*j))
80                  y = 2*float(sin(pi + ((2*pi)/len(csucsok[0]))*j))
81                  j=j+1
82                  if s==0:
83                      pos_dict[i]=[x+fix_0,y+fix_1-1]
84                  else:
85                      pos_dict[i]=[x+fix_0,y+fix_1]
86              fix_0=fix_0-5
87              if s%2==0:
88                  fix_1=4
89              else:
90                  fix_1=2
91              s=s+1
92              db=db+1
93          db=0
94          fix_0=0
95          fix_1=-4
96          s=n_0/2
97
98          while(db<n_0/2):
99              j=0
100             for i in csucsok[s]:
101                 x = 2*float(cos(pi + ((2*pi)/len(csucsok[s]))*j))
102                 y = 2*float(sin(pi + ((2*pi)/len(csucsok[s]))*j))
103                 j=j+1
104                 if s==n_0/2:
105                     pos_dict[i] = [x+fix_0,y+fix_1+1]
106                 else:
107                     pos_dict[i] = [x+fix_0,y+fix_1]
108             fix_0=fix_0+5
109             if s%2==0:
110                 fix_1=-4
```

```
            else:
                fix_1=-2
            s=s+1
            db=db+1

    if n_0<=8:
        G.graphplot(pos=pos_dict,edge_color='blue',
                    vertex_color='yellow',vertex_size=300,
                            graph_border=True).show(figsize=[5,5])
    elif 8<n_0<=16:
        G.graphplot(pos=pos_dict,edge_color='blue',
                vertex_color='yellow',vertex_size=300,
                        graph_border=True).show(figsize=[10,5])
    else:
         G.graphplot(pos=pos_dict,edge_color='blue',
                 vertex_color='yellow',vertex_size=300,
                         graph_border=True).show(figsize=[15,5])
```



4.11. ábra. C_8C_4 gráf

4.8. Sárkány gráfok

4.8.1. Definíció. Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf. Legyen a G gráf központi csúcsa v. Tegyük fel, hogy a v csúcs pontosan t darab másik csúccsal szomszédos, legyenek ezek rendre $u_1, u_2, \ldots, u_n$. Ragasszunk ezekre a csúcsokra

egy három hosszú zárt fület, ahol egy csúcs pontosan csak az egyik fülön lehet. Az így megkonstruált gráfot sárkány gráfnak nevezzük, és D_n-nel jelöljük.

4.8.1. Tétel. *Bármely $n > 1$ esetén a D_n gráf graceful gráf.*

Bizonyítás. Jelöljük a D_n gráf zárt fül ragasztásával keletkező köreinek csúcsait rendre $v_{i,1}, v_{i,2}, v_{i,3}, v_{i,4}$, ahol i jelöli az i-edik kört, és $1 \leq i \leq n$. Továbbá, jelölje a központi csúcsot v. A gráf csúcsainak számát jelöljük p-vel, ahol $p = 4n + 1$, illetve éleinek számát jelöljük q-val, ahol $q = 5n$. Ekkor az f csúcscímkézést definiáljuk a következőképpen:

(i.) Ha $i = 1$, akkor

$$f(v_{1,j}) = \begin{cases} q - (\frac{j-1}{2}) & , \textit{ ha } j \textit{ páratlan} \\ j - 2 & , \textit{ ha } j \textit{ páros}, \end{cases}$$

illetve

$$f(v_{2,j}) = \begin{cases} q - 4 + (j-1) & , \textit{ ha } j \textit{ páratlan} \\ 4 + (\frac{j-2}{2}) & , \textit{ ha } j \textit{ páros}. \end{cases}$$

(ii.) Ha $3 \leq i \leq n$, és i páratlan, akkor

$$f(v_{i,1}) = q - 5\left(\frac{i-1}{2}\right),$$

$$f(v_{i,2}) = 5\left(\frac{i-1}{2}\right) + 1,$$

$$f(v_{i,3}) = q - 5\left(\frac{i-1}{2}\right) - 2,$$

illetve

$$f(v_{i,4}) = 5\left(\frac{i-1}{2}\right) + 2.$$

(iii.) Ha $4 \leq i \leq n$, és i páros, akkor

$$f(v_{i,1}) = \frac{5i}{2},$$

$$f(v_{i,2}) = q - \frac{5i}{2} + 2,$$

$$f(v_{i,3}) = \frac{5i}{2} - 2,$$

illetve

$$f(v_{i,4}) = q - \frac{5i}{2} + 1.$$

Az így definiált f csúcscímkézés egy f^* élcímkézést indukál, amely bijektív, így teljesül a graceful tulajdonság, azaz a D_n gráf graceful gráf. □

Megjegyzés. A feldolgozott cikk [10] bizonyításában az $f(v_{1,j})$ esetben hiba történt, hiszen a helyes érték a $j-2$ lenne, és nem pedig a $2(j-2)$.

Az implementálás a címkék elkészítésével kezdődik, amely elsőként a központi csúcsnak a címkéjét adja meg, majd négyesével a köröknek a csúcscímkézést. Jelen esetben t-vel jelöljük a zárt füleknek a számát.

```
def Labels(t):
    L=[]
    L.insert(4*t,3)
    for j in range(1,5):
        if j%2!=0:
            L.insert(j,5*t-(j-1)/2)
        else:
            L.insert(j,(j-2))
    for j in range(1,5):
        if j%2!=0:
            L.insert(j+4,5*t-4+(j-1))
        else:
            L.insert(j+4,4+(j-2)/2)
    for i in range(3,t+1):
            if i%2!=0:
                L.insert(i*5-4,5*t-5*(i-1)/2)
                L.insert(i*5-3,5*(i-1)/2+1)
                L.insert(i*5-2,5*t-5*(i-1)/2-2)
                L.insert(i*5-1,5*(i-1)/2+2)
            else:
                L.insert(i*5-4,5*i/2)
                L.insert(i*5-3,5*t-5*i/2+2)
                L.insert(i*5-2,5*i/2-2)
                L.insert(i*5-1,5*t-5*i/2+1)
    return L
```

A pozicionálás rétegenként történik. Először a legyártott címkéket tartalmazó listát részlistákra osztjuk, amelyből épp annyi darab lesz ahány fület ragasztottunk a gráf csúcsaira. A központi csúcsot kezeljük külön esetként, amelyet a végén pozicionálunk. A négy hosszú listákból először vegyük ki az első elemeket és ezeket ábrázoljuk egy kör mentén. Ezután ábrázoljuk a listák harmadik elemeit is ilyen módon, viszont a kör átmérője kisebb legyen. Végül a listák maradék elemeit folytonosan ábrázoljuk egy kör mentén. Az átmérője legyen a két másik kör átmérője között.

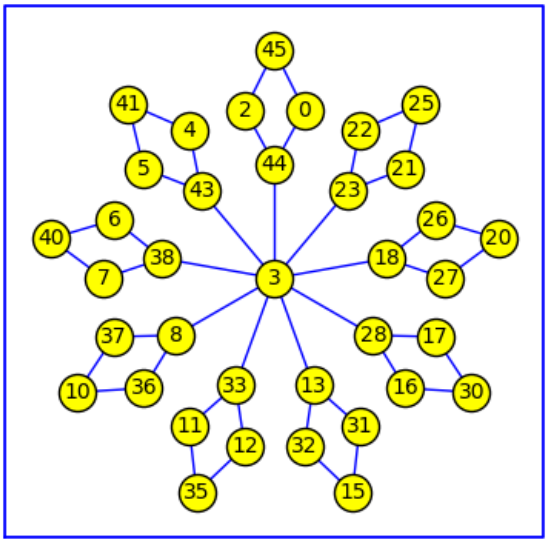
```
def graph(labels,t):
    alap=t
    G=Graph()
```

```
    G.add_vertex(0)
    for i in range(1,5):
        G.add_vertices([i])
    G.add_edges([(1,2),(2,3),(3,4),(4,1),(3,0)])
    temp=4
    while(t-1>0):
        for i in range(1+temp,1+temp):
            G.add_vertices([i*1])
        G.add_edges([(1+temp,2+temp),(2+temp,3+temp),(3+temp,4+temp),
            (4+temp,1+temp),(3+temp,0)])
        temp=temp+4
        t=t-1
    G.relabel(labels,inplace=True)
    pos_dict={}
    F=[]
    db=0
    temp=0
    for i in labels:
        if db%3==0 and db!=0:
            F.insert(temp,i)
            temp=temp+1
            db=0
        else:
            db=db+1
    F_0=[]
    db=0
    temp=0
    for i in labels:
        db=db+1
        if db%4==2:
            F_0.insert(temp,i)
            temp=temp+1
    F_1=[]
    db=0
    temp=0
    for i in labels:
        db=db+1
        if db%4==3:
            F_1.insert(temp,i)
            temp=temp+1
    F_2=[]
    db=0
    temp=0
    for i in labels:
        db=db+1
        if db%4==1 and i!=3:
```

```
            F_2.insert(temp,i)
            temp=temp+1
    j=0
    for i in F:
        x = 10*float(2*cos(pi/2 + ((pi)/(len(F)/2))*j))
        y = 10*float(2*sin(pi/2 + ((pi)/(len(F)/2))*j))
        j=j+1
        pos_dict[i] = [x,y]

    j=0
    for i in F_0:
        x = 20*float(2*cos(pi/2 + ((pi)/(len(F_0)/2))*j))
        y = 20*float(2*sin(pi/2 + ((pi)/(len(F_0)/2))*j))
        j=j+1
        pos_dict[i] = [x,y]
    j=0
    if(alap>11):
        for i in F_1:
            x = 15*float(2*cos(pi/(2.1)+ ((pi)/(len(F_1)/2))*j))
            y = 15*float(2*sin(pi/(2.1)+ ((pi)/(len(F_1)/2))*j))
            j=j+1
            pos_dict[i] = [x,y]
        j=0
        for i in F_2:
            x = 15*float(2*cos(pi/(1.9) + ((pi)/(len(F_2)/2))*j))
            y = 15*float(2*sin(pi/(1.9) + ((pi)/(len(F_2)/2))*j))
            j=j+1
            pos_dict[i] = [x,y]
        pos_dict[3]=[0,0]
        G.graphplot(pos=pos_dict,edge_color='blue',
                vertex_color='yellow',vertex_size=200,graph_border=True).show()
    else:
        for i in F_1:
            x = 15*float(2*cos(pi/(2.25)+ ((pi)/(len(F_1)/2))*j))
            y = 15*float(2*sin(pi/(2.25)+ ((pi)/(len(F_1)/2))*j))
            j=j+1
            pos_dict[i] = [x,y]
        j=0
        for i in F_2:
            x = 15*float(2*cos(pi/(1.8) + ((pi)/(len(F_2)/2))*j))
            y = 15*float(2*sin(pi/(1.8) + ((pi)/(len(F_2)/2))*j))
            j=j+1
            pos_dict[i] = [x,y]
        pos_dict[3]=[0,0]
        G.graphplot(pos=pos_dict,edge_color='blue',
                vertex_color='yellow',vertex_size=300,graph_border=True).show()
```

Végül meghívjuk a függvényeket, és kezeljük azokat az eseteket, amelyek nem valósíthatók meg ezzel a konstrukcióval.

```
def dragon_graphs(t):
    if t<2:
        return "A t paraméter 1-nél nagyobb kell, hogy legyen!"
    labels=Labels(t)
    graph(labels,t)
```



4.12. ábra. D_9 gráf

4.9. Körgráfok párhuzamos húrokkal

Az alábbi speciális gráf esetében az a furcsa eset állt elő, hogy alapértelmezetten van egy formula a csúcscímkézés előállítására, viszont ez nem annyira szép. Viszont, ha a gráfot alkotó körgráfot nem körgráfnak, hanem két útgráfnak tekintjük, és egy közös csúccsal összekötjük ezeket, akkor a páratlan esetben sokkal szebb képlet megadható, amely nem tartalmaz törtrészeket.

Megjegyzés. A cikkek, amiket felhasználtam az egy A. Elumalai, és Sethuraman Guruswamy 2012-es cikk [11], illetve egy 2017-es A. Solairaju, S. Subbulakshmi, és R. Kokila cikk [12]. Az első cikkben az $n > 6$ esetre adnak bizonyítást, míg a második cikkben csak a páratlan esetre adnak bizonyítást, viszont a cikk szerzői nem vették figyelembe, hogy ez egy speciális körgráf. Továbbá, a cikkben elírások is történtek, hiszen ők egy v_i csúcs esetén, az $f(v_i) = q - \left(\frac{i-1}{2}\right)$ értéket adták meg, ahol a helyes érték az $f(v_i) = q - \left(\frac{i+1}{2}\right)$ lenne, ahol i páratlan. Továbbá, ők

megkülönböztetnek a cikkükben bizonyos v_i csúcsokat, azonban ez nem szükséges, az előző javítás miatt. Ha az első hibát nem javítanánk, még akkor sem lenne jó a formula ezekkel a megkülönböztetett esetekben sem.

4.9.1. Definíció. Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf. Legyenek a csúcsai rendre $v_0, v_1, \ldots, v_{n-1}$, és vezessen él v_i-ből v_{i+1}, ahol $0 \leq i \leq n-2$, és v_{n-1}-ből vezessen él v_0. Továbbá, ha n páros, akkor legyen v_j szomszédos v_{n-j}-vel, ahol $1 \leq j \leq \frac{n}{2} - 1$. Ha n páratlan, akkor legyen v_j szomszédos v_{n-j}-vel, ahol $1 \leq j \leq \frac{n-1}{2} - 1$. Az így megkonstruált gráfot párhuzamos húrokkal rendelkező körgráfnak nevezzük. Jelöljük ezt CWP_n-vel.

4.9.1. Tétel. *Ha $n > 6$, és n egész, akkor CWP_n graceful gráf.*

Bizonyítás. Legyenek a gráf csúcsai rendre $v_0, v_1, \ldots, v_{n-1}$. Jelöljük a gráf éleinek számát q-vel, ahol $q = \frac{3n-s}{2}$, és $s = 3$, ha n páratlan, míg $s = 2$, ha n páros. Definiáljuk az f csúcscímkézést a következőképpen:

(i.) Ha n páros, akkor

$$f(v_0) = 0,$$

$$f(v_1) = 1,$$

$$f(v_{n-1}) = \frac{3n-2}{2},$$

$$f(v_{2i}) = \frac{3n-6i+2}{2},\ ha\ 1 \leq i \leq \left\lfloor \frac{n-4}{4} \right\rfloor,$$

$$f(v_{n-(2i+1)}) = \frac{3n-6i}{2},\ ha\ 1 \leq i \leq \left\lfloor \frac{n-4}{4} \right\rfloor,$$

illetve

$$f(v_{n-2i}) = 3i,\ ha\ 1 \leq i \leq \left\lceil \frac{n-4}{4} \right\rceil.$$

Továbbá, ha $1 \leq i \leq \delta$, akkor $n \equiv 0 \pmod 4$ esetén, legyen $\delta = \left\lfloor \frac{n-4}{4} \right\rfloor$-1. Ha $1 \leq i \leq \delta$, akkor $n \equiv 2 \pmod 4$ esetén, legyen $\delta = \left\lfloor \frac{n-4}{4} \right\rfloor$, és

$$f(v_{2i+1}) = 3i + 2.$$

Továbbá,

$$f(v_{\frac{n}{2}-1}) = \begin{cases} \frac{3n}{4} & , ha \quad n \equiv 0 \pmod 4 \\ \frac{3(n+2)}{4} & , ha \quad n \equiv 2 \pmod 4, \end{cases}$$

illetve

$$f(v_{\frac{n}{2}}) = \begin{cases} \frac{3n-8}{4} & , ha \quad n \equiv 0 \pmod 4 \\ \frac{3(n+4)+2}{4} & , ha \quad n \equiv 2 \pmod 4. \end{cases}$$

Az így definiált f csúcscímkézés egy f^* bijektív élcímkézést indukál, amelyből következik, hogy teljesül a graceful tulajdonság, azaz a gráf graceful gráf, ha n páros.

(ii.) Ha n páratlan, akkor legyen $s = \frac{n-1}{2}$, és $q = 3s$. Ekkor

$$f(v_0) = 0,$$

$$f(v_{n-1}) = q,$$

illetve

$$f(v_i) = \begin{cases} q - \frac{i+1}{2} & , \textit{ ha } i \textit{ páratlan} \\ \frac{i}{2} & , \textit{ ha } i \textit{ páros.} \end{cases}$$

Ebben az esetben külön címkézzük el a v_0 csúcsot, amelyből nem vezet ki párhuzamos húr, illetve a v_{n-1} csúcsot is. A többi csúcs alternálva címkéződik el, mint az útgráf esetén. Ez a csúcscímkézés is egy f^* élcímkézést indukál, ami szintén bijektív, amiből következik, hogy a gráf graceful gráf páratlan esetben is.

□

Mivel a bizonyításban két esetet különböztettünk meg, így a gráf címkézésének implementálása során is két függvényt fogok definiálni ennek megfelelően.
Ha n páratlan:

```
1  def Labels_0(n):
2      L=[]
3      L.insert(0,0)
4      q=3*n
5      szamlalo=1
6      for i in range(1,2*n):
7          if i%2!=0:
8              L.insert(szamlalo,q-(i+1)/2)
9              szamlalo=szamlalo+1
10         else:
11             L.insert(szamlalo,i/2)
12             szamlalo=szamlalo+1
13     L.insert(szamlalo,q)
14     return L
```

Ha n páros:

```
1  def Labels_1(n):
2      s=0
3      L=[]
4      L.insert(0,0)
5      L.insert(1,1)
6      szamlalo=2
7      for i in range(2,n):
```

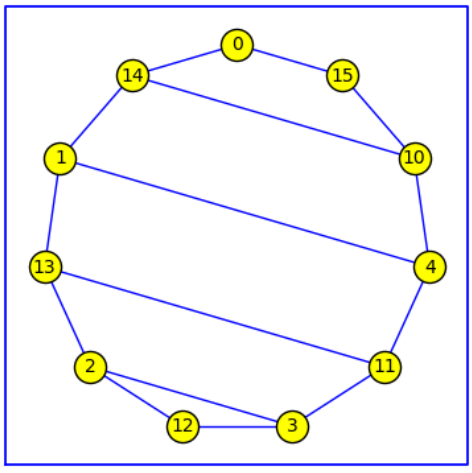
```
        if i==n-1:
            s=(3*n-2)/2
        for j in range(1,ceil((n-4)/4)+1):
            if i==2*j and j!=ceil((n-4)/4)+1:
                s=(3*n-6*j+2)/2
                break
            elif i==n-(2*j+1) and j!=ceil((n-4)/4)+1:
                s=(3*n-6*j)/2
                break
            elif i==n-2*j  :
                s=3*j
                break
            elif i==2*j+1 and j!=ceil((n-4)/4):
                if n%4==0 and j!=floor((n-4)/4):
                    s=3*j+2
                    break
                elif n%4==2 and j!=floor((n-4)/4)+1:
                    s=3*j+2
                    break
            elif i==(n/2-1):
                if n%4==0 :
                    s=3*n/4
                    break
                elif n%4==2 :
                    s=3*(n+2)/4
                    break
            elif i==(n/2):
                if n%4==0 :
                    s=(3*n-8)/4
                    break
                elif n%4==2 :
                    s=(3*(n+4)+2)/4
                    break
            if i==3:
                print(i,s)
        L.insert(szamlalo,s)
        szamlalo=szamlalo+1
    return L
```

A gráf készítése során csak az $n > 6$ feltételre kell ügyelni, a pozicionálás a szokásos ábrázolási mód szerint történik.

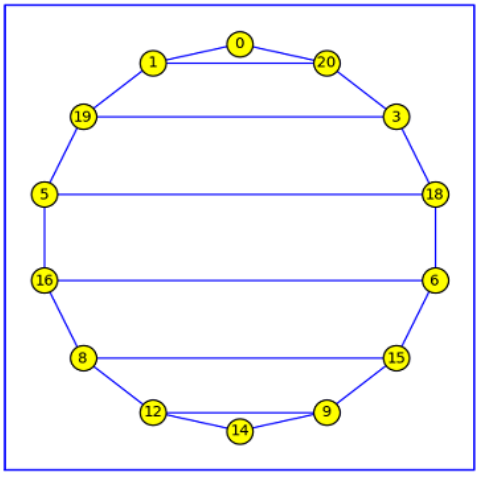
```
def cycle_graph_with_parallels(n):
    if n%2!=0:
        if n>6:
            n=(n-1)/2
```

```
            labels=Labels_0(n)
            G=Graph()
            for i in range(0,2*n+1):
                if i!=2*n:
                    G.add_edges([(i,i+1)])
            for i in range(0,n):
                G.add_edges([(i,2*n-i)])
            G.relabel(labels,inplace=True)
            pos_dict={}
            j=0
            for i in labels:
                x = float(2*cos(pi/2 + ((pi)/(len(labels)/2))*j))
                y = float(2*sin(pi/2 + ((pi)/(len(labels)/2))*j))
                j=j+1
                pos_dict[i] = [x,y]

            for u,v,l in G.edges():
                G.set_edge_label(u,v,abs(u-v))
            if n<=5:
                G.graphplot(pos=pos_dict,edge_color='blue',
                            vertex_color='yellow',vertex_size=300,
                                    graph_border=True).show()
            elif 5<n<=10:
                G.graphplot(pos=pos_dict,edge_color='blue',
                            vertex_color='yellow',vertex_size=300,
                                    graph_border=True).show(figsize=[10,5])
            else:
                G.graphplot(pos=pos_dict,edge_color='blue',
                            vertex_color='yellow',vertex_size=300,
                                    graph_border=True).show(figsize=[15,5])
        else:
            print("A konstrukció miatt 6<n szükséges.")
    else:
        if n>6 :
            labels=Labels_1(n)
            G=Graph()
            for i in range(0,n):
                if i!=n-1:
                    G.add_edges([(i,i+1)])
                else:
                    G.add_edges([(0,i)])
            koz=(n/2)-1
            db=1
            while(db<=koz):
                G.add_edges([(0+db,n-db)])
                db=db+1
```

```
51              G.relabel(labels,inplace=True)
52              pos_dict={}
53              j=0
54              for i in labels:
55                  x = float(2*cos(pi/2 + ((pi)/(len(labels)/2))*j))
56                  y = float(2*sin(pi/2 + ((pi)/(len(labels)/2))*j))
57                  j=j+1
58                  pos_dict[i] = [x,y]
59              for u,v,l in G.edges():
60                  G.set_edge_label(u,v,abs(u-v))
61              if n<=9:
62                  G.graphplot(pos=pos_dict,edge_color='blue',
63                              vertex_color='yellow',vertex_size=300,
64                                      graph_border=True).show()
65              elif 9<n<=20:
66                  G.graphplot(pos=pos_dict,edge_color='blue',
67                              vertex_color='yellow',vertex_size=300,
68                                      graph_border=True).show(figsize=[10,5])
69              else:
70                  G.graphplot(pos=pos_dict,edge_color='blue',
71                              vertex_color='yellow',vertex_size=300,
72                                      graph_border=True).show(figsize=[12,12])
73          else:
74              print("A konstrukció miatt 6<n szükséges.")
```



4.13. ábra. CWP_{11} gráf



4.14. ábra. CWP_{14} gráf

4.10. Körgráfok cikk-cakk húrokkal

Mint a neve is sugallja, ezeknek a gráfoknak is az alapja a C_n gráfok. A különbség csak az, hogy ezekben a gráfokban egyes csúcsokból nem csak a hozzá legközelebb álló legnagyobb, és legkisebb sorszámú csúcsba megy él, hanem megengedünk bizonyos éleket a körgráfon belül is.

4.10.1. Definíció. Egy $G = (V, E, \varphi)$ egyszerű gráfot cikk-cakk húros körgráfnak nevezünk, ha a $|V(G)| \geq 8$, és $V(G) = \{v_0, \ldots, v_{n-1}\}$ esetén a körgráf szomszédos csúcsaiból kiinduló két darab élen kívül az alábbi feltételek alapján illeszkedik új él a csúcsokra:

(i.) Ha $n \equiv 0 \pmod 4$, akkor $\frac{n-2}{2}$ és $\frac{n+2}{2}$ csúcsok között illeszkedjen új él.

(ii.) Ha $n \equiv 1 \pmod 4$, akkor $\frac{n-3}{2}$ és $\frac{n+3}{2}$ csúcsok között illeszkedjen új éj.

(iii.) Ha $n \equiv 2 \pmod 4$, akkor $\frac{n+4}{2}$ és $\frac{n}{2}$ csúcsok között illeszkedjen új él.

(iv.) Ha $n \equiv 3 \pmod 4$, akkor $\frac{n+5}{2}$ és $\frac{n-1}{2}$ csúcsok között illeszkedjen új éj.

Jelöljük ezt a gráfot CWZ_n-nel.

Ezeket a gráfokat A. Elumalai és A.A. Ephremnath indiai kutatók definiálták egyik 2013-as cikkükben [13]. Azonban már hamarabb más kutatókat is foglalkoztatott speciális körgráfoknak a létezése. 1985-ben, például Koh és Yap cikkükben [14] definiálták elsőként a P_k húrral rendelkező körgráfokat, ahol a körgráf csúcsaira az k csúcsú útgráf végpontjai illeszkedtek. Továbbá, a nevükhöz fűződik a gracefulság bebizonyítása az egy darab húrral rendelkező körgráfok esetében is.

4.10.1. Tétel. *Ha $n \geq 8$, akkor létezik graceful címkézése az n csúcsú cikk-cakk gráfnak.*

Bizonyítás. Legyen az n csúcsú cikk-cakk körgráf csúcsai rendre $v_0, v_1, \ldots, v_{n-1}$. Legyen $M = |E(CWZ_n)|$. Definiáljuk az f csúcscímkézést a következőképpen:

(i.) Ha $n \equiv 0 \pmod 4$, akkor $M = \frac{3n-2}{2}$, ahol $n = 4k$ $(k \geq 2)$ alakú. Ekkor

$$f(v_0) = 0,$$

$$f(v_{2i}) = i, \text{ ahol } 1 \leq i \leq \frac{n-4}{4},$$

$$f(v_{\frac{n+4i-4}{2}}) = \frac{n+4i}{4}, \text{ ahol } 1 \leq i \leq \frac{n}{4},$$

illetve

$$f(v_{2i-1}) = \frac{3n-2i}{2}, \text{ ahol } 1 \leq i \leq \frac{n}{2}.$$

(ii.) Ha $n \equiv 1 \pmod 4$, akkor $M = \frac{3n-3}{2}$, ahol $n = 4k+1$ $(k \geq 2)$ alakú. Ekkor

$$f(v_0) = 0,$$

$$f(v_{\frac{n-1}{2}}) = \frac{3n+1}{4},$$

$$f(v_{2i}) = i, \ ahol \ 1 \leq i \leq \frac{n-5}{4},$$

$$f(v_{\frac{(n+4i-3)}{2}}) = \frac{n+4i-5}{4}, \ ahol \ 1 \leq i \leq \frac{n-1}{4},$$

$$f(v_{2i-1}) = \frac{3n-2i-1}{2}, \ ahol \ 1 \leq i \leq \frac{n-1}{4},$$

illetve

$$f(v_{\frac{(n+4i-1)}{2}}) = \frac{5n-4i-1}{4}, \ ahol \ 1 \leq i \leq \frac{n-1}{4}.$$

(iii.) Ha $n \equiv 2 \pmod 4$, és $k = 2$, ahol $n = 4k+2$ alakú, akkor vegyük a következő csúcscímkézést. Legyen $C = \{v_0, \ldots, v_9\}$ és $D = \{8, 14, 9, 13, 1, 11, 4, 2, 3, 0\}$. Ekkor rendeljük egymáshoz az azonos indexű halmazbeli elemeket, és jelentse azt a (v_i, j), hogy a címkézés az i-edik csúcshoz a j-edik címkét rendeli $(i = 0, \ldots, 9, \ j = 1, \ldots, 14)$.
Végül definiáljuk a $k \geq 3$ esetén a címkéket, ahol $M = \frac{3n-2}{2}$. Ekkor

$$f(v_{n-1}) = 0,$$

$$f(v_{\frac{n+6}{2}}) = \frac{3n-2}{4},$$

$$f(v_{\frac{n}{2}}) = \frac{5n-6}{4},$$

$$f(v_{2i+2}) = i, \ ahol \ 1 \leq i \leq \frac{n-6}{4},$$

$$f(v_{\frac{(n-2i+6)}{2}}) = \frac{n+8i-10}{4}, \ ahol \ 1 \leq i \leq 2,$$

$$f(v_{\frac{(n+4i+6)}{2}}) = \frac{n+4i+6}{4}, \ ahol \ 1 \leq i \leq \frac{n-10}{4},$$

$$f(v_{n-2i-1}) = \frac{n+2i+2}{2}, \ ahol \ 1 \leq i \leq \frac{n-10}{4},$$

$$f(v_{2i-1}) = \frac{3n-2i}{2}, \ ahol \ 1 \leq i \leq \frac{n-2}{4},$$

illetve

$$f(v_{4-2i}) = n-3i+4, \ ahol \ 1 \leq i \leq 2.$$

(iv.) Ha $n \equiv 3 \pmod 4$, akkor $M = \frac{3n-3}{2}$, ahol $n = 4k+3$ $(k \geq 2)$ alakú. Ekkor

$$f(v_0) = \frac{3n-3}{2},$$

$$f(v_1) = 0,$$

$$f(v_2) = n-2,$$

$$f(v_{2i+1}) = i, \ ahol \ 1 \leq i \leq \frac{n+1}{4},$$

$$f(v_{\frac{(n-4i+5)}{2}}) = \frac{n+4i+5}{4}, \ ahol \ 1 \leq i \leq \frac{n-3}{4},$$

$$f(v_{n-2i}) = \frac{n+2i+1}{2}, \ ahol \ 1 \leq i \leq \frac{n-7}{4},$$

illetve

$$f(v_{n-2i+1}) = \frac{3n-2i-3}{2}, \ ahol \ 1 \leq i \leq \frac{n-3}{4}.$$

Az így megadott f csúcscímkézés mindegyik esetben egy f^* bijektív élcímkézést indukál, amelyből következik, hogy teljesül a graceful tulajdonság, azaz a gráf graceful gráf. □

```
1  def Labels(n):
2      L=[]
3      if n>=8:
4          if n % 4==0:
5              for i in range(0,n):
6                  L.insert(0,0)
7              L[0]=0
8              for i in range(0,n):
9                  if 1<=i and i<=(n-4)/4:
10                     L[2*i]=i
11                 if 1<=i and i<=n/4:
12                     L[(n+4*i-4)/2]=(n+4*i)/4
13                 if 1<=i and i<=n/2:
14                     L[2*i-1]=(3*n-2*i)/2
15         elif n % 4==1:
16             for i in range(0,n):
17                 L.insert(0,0)
18             L[0]=0
19             L[(n-1)/2]=(3*n+1)/4
20             for i in range(0,n):
21                 if 1<=i<=(n-5)/4:
22                     L[2*i]=i
23                 if 1<=i<=(n-1)/4:
24                     L[(n+4*i-3)/2]=(n+4*i-5)/4
```

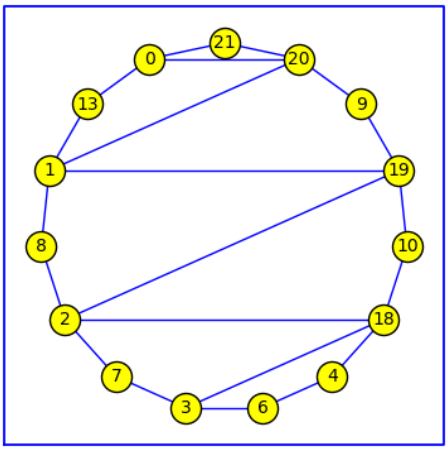
```
                    L[2*i-1]=(3*n-2*i-1)/2
                    L[(n+4*i-1)/2]=(5*n-4*i-1)/4
        elif n % 4==2 and n==10:
            L=[8,14,9,13,1,11,4,2,3,0]
        elif n % 4==2 and n!=10:
            for i in range(0,n):
                L.insert(0,0)
            L[n-1]=0
            L[(n+6)/2]=(3*n-2)/4
            L[n/2]=(5*n-6)/4
            for i in range(0,n):
                if 1<=i<=(n-6)/4:
                    L[2*i+2]=i
                if 1<=i<=2:
                    L[(n-2*i+6)/2]=(n+8*i-10)/4
                    L[4-2*i]=n-3*i+4
                if 1<=i<=(n-10)/4:
                    L[(n+4*i+6)/2]=(n+4*i+6)/4
                    L[n-2*i-1]=(n+2*i+2)/2
                if 1<=i<=(n-2)/4:
                    L[2*i-1]=(3*n-2*i)/2
        elif n % 4==3:
            for i in range(0,n):
                L.insert(0,0)
            L[0]=(3*n-3)/2
            L[1]=0
            L[2]=n-2
            for i in range(0,n):
                if 1<=i<=(n+1)/4:
                    L[2*i+1]=i
                if 1<=i<=(n-3)/4:
                    L[(n-4*i+5)/2]=(n+4*i+5)/4
                if 1<=i<=(n-7)/4:
                    L[n-2*i]=(n+2*i+1)/2
                if 1<=i<=(n-3)/4:
                    L[n-2*i+1]=(3*n-2*i-3)/2
    return L
```

A gráf elkészítése definíció alapján történik az α és β értékekkel.

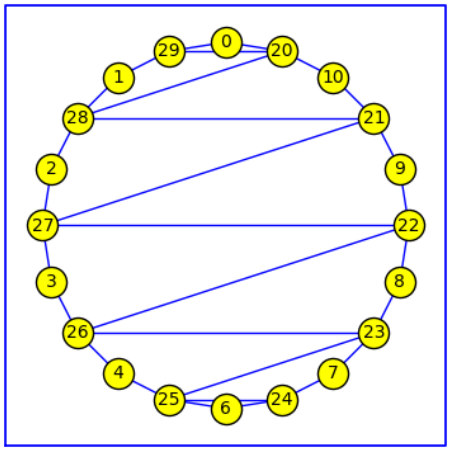
```
def ZigZag(n):
    if n>=8:
        labels=Labels(n)
        G=Graph()
        if n%4==0:
            alpha=(n-2)/2
```

```
            beta=(n+2)/2
        elif n%4==1:
            alpha=(n-3)/2
            beta=(n+3)/2
        elif n%4==2:
            alpha=(n+4)/2
            beta=n/2
        elif n%4==3:
            alpha=(n+5)/2
            beta=(n-1)/2
        G.add_vertex(0)
        if n%4==0:
            for i in range(1,alpha+1,2):
                G.add_edges([(i,n-i)])
            for i in range(1,alpha,2):
                G.add_edges([(n-i,i+2)])
            for i in range(0,n):
                if i!=n-1:
                    G.add_edges([(i,i+1)])
                else:
                    G.add_edges([(0,n-1)])
        elif n%4==1:
            for i in range(1,alpha+1,2):
                G.add_edges([(i,n-i)])
            for i in range(1,alpha,2):
                G.add_edges([(n-i,i+2)])
            for i in range(0,n):
                if i!=n-1:
                    G.add_edges([(i,i+1)])
                else:
                    G.add_edges([(0,n-1)])
        elif n%4==2:
            for i in range(1,beta,2):
                G.add_edges([(i,n-i)])
            for i in range(1,beta,2):
                G.add_edges([(n-i,i+2)])
            for i in range(0,n):
                if i!=n-1:
                    G.add_edges([(i,i+1)])
                else:
                    G.add_edges([(0,n-1)])
        elif n%4==3:
            for i in range(1,beta,2):
                G.add_edges([(i,n-i)])
            for i in range(1,beta,2):
                G.add_edges([(n-i,i+2)])
```

```
            for i in range(0,n):
                if i!=n-1:
                    G.add_edges([(i,i+1)])
                else:
                    G.add_edges([(0,n-1)])
        G.relabel(labels,inplace=True)
        pos_dict = {}
        j=0
        for i in labels:
            x = float(2*cos(pi/2 + ((pi)/(len(labels)/2))*j))
            y = float(2*sin(pi/2 + ((pi)/(len(labels)/2))*j))
            j=j+1
            pos_dict[i] = [x,y]
        for u,v,l in G.edges():
            G.set_edge_label(u,v,abs(u-v))
        G.graphplot(pos=pos_dict,edge_color='blue',
            vertex_color='yellow',vertex_size=300).show()
    else:
        print ("A konstrukció miatt 7<n szükséges.")
```



4.15. ábra. ZWP_{15} gráf



4.16. ábra. ZWP_{20} gráf

5. Speciális graceful fagráfok, és álrács gráfok implementálása

A speciális körgráfok implementálása után néhány fagráfcsalád implementálását tűztem ki célul. Az előző dolgozatomban a caterpillar gráf általános címkézését mutattam be, amivel különböző méretű caterpillar gráfokat lehetett létrehozni. Ebben a részben az egyik legegyszerűbb fagráfot, a csillag gráfot szeretném elsőnek bemutatni. A csillag gráfok után egy kis változtatással eljuthatunk a pók gráfok egyik fajtájához, végül pedig egy speciális, több komponensből álló gráfot szeretnék bemutatni.

5.1. Csillag gráfok

5.1.1. Definíció. Egy fa gráfot csillag gráfnak nevezünk, ha $n+1$ darab csúcsa van, és az $n+1$ darab csúcsból, pontosan n darab levél csúcs.

Megjegyzés. A csillag gráf megegyezik a $K_{1,n}$ teljes páros gráffal. Továbbá, a csillag gráfok is egyszerű caterpillar gráfok.

5.1.1. Tétel. *A csillag gráfok graceful gráfok.*

Bizonyítás. Legyen a gráf központi csúcsa v, a többi csúcsa rendre $v_1, v_2, \ldots, v_n$. Mivel n darab levele van a gráfnak, és ezek közt semelyik kettő nem szomszédos, viszont mindegyik szomszédos a v csúccsal, így az $1, 2, \ldots, n$ címkék közül a nem közbülső csúcsot tetszőlegesen el lehet címkézni. Míg a közbülső csúcshoz rendeljük hozzá a 0 csúcscímkét. Az így konstruált csúcscímkézés egy f^* élcímkézést indukál, és mivel f^* bijektív, így teljesül a graceful tulajdonság. Tehát a csillag gráfok graceful gráfok. □

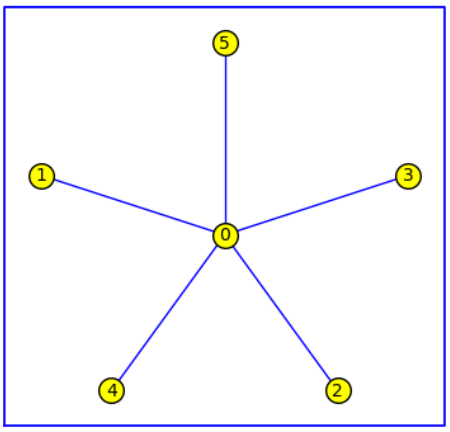
Az implementálás során a címkéket úgy készítettem el, hogy alternáló sorozatot alkossanak.

```
1 def Labels(n):
2     L=[]
3     j=0
4     for i in range(0,n+1):
```

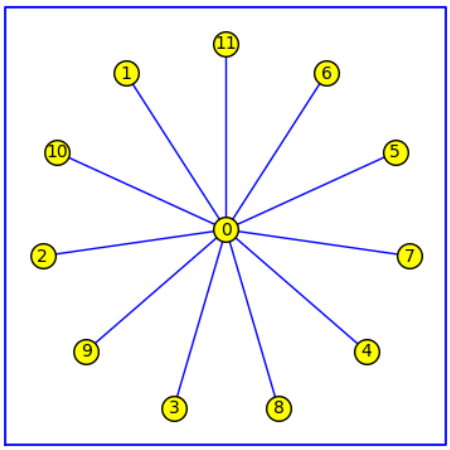
```
5           if(i%2!=0):
6               L.insert(i,n)
7               n=n-1
8           elif(i%2==0):
9               L.insert(i,j)
10              j=j+1
11      return L
```

```
def graph(labels,n):
    G=graphs.StarGraph(n)
    G.relabel(labels,inplace=True)
    return G.graphplot(edge_color='blue',vertex_color='yellow',
            vertex_size=200,graph_border=True).show()
```

```
def Star_graphs(n):
    labels=Labels(n)
    graph(labels,n)
```



5.1. ábra. S_5 gráf



5.2. ábra. S_{11} gráf

5.2. Pók gráfok

5.2.1. Definíció. Egy S gráfot pók gráfnak nevezünk, ha fa gráf, és pontosan egy olyan csúcsa van, amelyiknek fokszáma nagyon, mint kettő. Ha létezik ilyen csúcs, akkor ezt a csúcsot a gráf elágazási pontjának nevezzük. Továbbá, a pók gráfban az elágazási pontból induló különböző hosszúságú útjait nevezzük a pók gráf lábainak.

5.2.1. Tétel. *Legyen $S = (V, E, \varphi)$ véges pók gráf l darab lábbal, amelyeknek hossza m, vagy $m+1$ lehet, ahol $m \geq 1$. Ekkor az S gráf graceful gráf.*

Bizonyítás. Tegyük fel, hogy $l \geq 3$, mivel $l \leq 2$ esetén a pók gráfok egyben út gráfok is, azokról viszont tudjuk, hogy graceful gráfok.
Az f csúcscímkézést vizsgáljuk meg az alábbi esetekben.

(i.) Tegyük fel, hogy l páratlan.
Ebben az esetben, legyen $l = l_0 + l_1$, ahol l_0 jelenti az m hosszú lábaknak a számát, míg l_1 jelenti az $m+1$ hosszú lábaknak a számát. Ekkor a csúcsok száma $n = l_0 m + l_1 + 1$. Jelölje v az elágazási csúcsot, míg a gráf többi csúcsa esetén, jelölje $v_{i,j}$ azt a csúcsot, amelyik az i-edik lában van, és j távolságra van az elágazási csúcstól.
Definiáljuk az f csúcscímkézést a következőképpen:

$$f(v) = 1.$$

Ha i és j mindkettő páratlan számok, akkor

$$f(v_{i,j}) = n - \frac{i-1}{2} - \frac{(j-1)l}{2}.$$

Ha i és j mindkettő páros számok, akkor

$$f(v_{i,j}) = n - \frac{l-1}{2} - \frac{i}{2} - \frac{(j-2)l}{2}.$$

Ha i páros, míg j páratlan szám, akkor

$$f(v_{i,j}) = \frac{i}{2} + \frac{(j-1)l}{2}.$$

Ha i páratlan, míg j páros szám, akkor

$$f(v_{i,j}) = \frac{l-1}{2} + \frac{i+1}{2} + \frac{(j-2)l}{2}.$$

Az így meghatározott f csúcscímkézés egy f^* élcímkézést indukál, amely bijektív, így teljesül a graceful tulajdonság, így a gráf graceful gráf.

(ii.) Tegyük fel, hogy l páros.
Továbbá, tegyük fel azt is, hogy ebben az esetben van m hosszú láb. Töröljük a lábon lévő összes csúcsot, kivéve a központi csúcsot. Így kaptunk egy $l-1$ darab, páratlan lábú pók gráfot. Használjuk erre az első esetben leírt konstrukciót. Jelöljük el ezt a gráfot T_0-val. Továbbá, $n' := |V(T_0)|$. Definiáljuk az f_0 függvényt, úgy hogy $f_0(u) = n' - f(u)$, bármely $u \in V(T_0)$ esetén. Az így keletkezett gráfot jelöljük T_1-gyel. Adjunk hozzá a T_1 gráfhoz egy w_1 csúcsot, amely legyen szomszédos a v központi csúccsal. Definiáljuk az f_1

függvényt, úgy hogy $f_1(w_1) = 0$, és $f_1(u) = f_0(u)+1$, bármely $u \in V(T_0)$ esetén. Továbbá, definiáljuk az f_2 függvényt, úgy hogy $f_2(u) = n'+1-f_1(u)$, bármely $u \in V(T_0)$ esetén, és legyen $f_2(w_1) = n'+1$.
A következő lépésben adjunk hozzá a gráfhoz egy w_2 csúcsot, amely legyen szomszédos w_1-gyel. Majd alkalmazzuk rá a fent definiált függvényeket ugyanúgy sorban, mint w_1 esetén. Majd folytassuk ezt a konstrukciót w_m-ig.
Mivel az eredeti csúcscímkézés összes csúcsát folyamatosan ugyanúgy változtattuk, és mindig olyan új csúcscímkét definiáltunk, amely csúcscímke még nem volt, így minden csúcs csúcscímkéje különböző. Mivel minden csúcscímke különböző, és a csúcscímkézés egy f^* bijektív élcímkézést indukál, ezért teljesül a graceful tulajdonság, így a gráf graceful gráf.

□

A gráf implementálása során figyelembe kell venni bizonyos eseteket is, aminél a konstruktív bizonyítás problémába kerülhet. A gráf csúcscímkézése során három paramétert kell megadnunk az l_0-t, az l_1-et és az m-et. Ezek alapján probléma azokban az esetekben fordulhat elő, amikor valamelyik l_i érték 0, ahol $i \in \{0, 1\}$. Ebben az esetben viszont egy csillag gráf adódna, amelyről tudjuk, hogy graceful gráf.
Továbbá, meg kell említeni, hogy az itt szereplő bizonyítás először Bahls cikkében [15] jelent meg, viszont később Elina Robeva ezt felhasználta cikkében [16], ahol már helytelenül szerepelnek a bizonyításban leírt formulák. Három hibát követett el a cikk szerzője. Két esetben tévesen adott hozzá az értékekhez 1-et, míg a harmadik hibája az volt, hogy elfelejtett kivonni $\frac{i}{2}$-t, a bizonyítás i és j is páros esetében.

```
1  def Labels(l_0,l_1,m):
2      l=l_0+l_1
3      if l_0!=0:
4          n=l*m+l_1
5      else:
6          n=l*(m+1)
7      L=[]
8      L.insert(0,0)
9      szamlalo=1
10     db=0
11     if l%2!=0:
12         for i in range(1,l+1):
13             if db<l_1:
14                 for j in range(1,m+2):
15                     if i%2!=0 and j%2!=0:
16                         L.insert(szamlalo,n-(i-1)/2-(j-1)/2*l)
```

```
                        szamlalo=szamlalo+1
                    elif i%2==0 and j%2==0:
                        L.insert(szamlalo,n-(l-1)/2-i/2-(j-2)/2*l)
                        szamlalo=szamlalo+1
                    elif i%2==0 and j%2!=0:
                        L.insert(szamlalo,i/2+(j-1)*l/2)
                        szamlalo=szamlalo+1
                    else:
                        L.insert(szamlalo,(l-1)/2+(i+1)/2+(j-2)*l/2)
                        szamlalo=szamlalo+1
                db=db+1
            else:
                for j in range(1,m+1):
                    if i%2!=0 and j%2!=0:
                        L.insert(szamlalo,n-(i-1)/2-(j-1)/2*l)
                        szamlalo=szamlalo+1
                    elif i%2==0 and j%2==0:
                        L.insert(szamlalo,n-(l-1)/2-i/2-(j-2)/2*l)
                        szamlalo=szamlalo+1
                    elif i%2==0 and j%2!=0:
                        L.insert(szamlalo,i/2+(j-1)*l/2)
                        szamlalo=szamlalo+1
                    else:
                        L.insert(szamlalo,(l-1)/2+(i+1)/2+(j-2)*l/2)
                        szamlalo=szamlalo+1
                db=db+1
    else:
        F=[]
        if l_0!=0:
            n_0=n-m
        else:
            n_0=n-m-1
        if l_0>0:
            L=Labels(l_0-1,l_1,m)
        else:
            L=Labels(l_0,l_1-1,m)
        db=0
        F_0=[]
        szamlalo=0
        for i in range(0,len(L)):
            F.insert(szamlalo,n_0-L[i])
            szamlalo=szamlalo+1
        L.clear()
        if l_0!=0:
            while(db<=m):
                szamlalo=0
```

```
63                  for i in range(0,len(F)):
64                      F_0.insert(szamlalo,F[i]+1)
65                      szamlalo=szamlalo+1
66                  F_0.insert(szamlalo,0)
67                  F.clear()
68                  for i in range(0,len(F_0)):
69                      F.insert(szamlalo,n_0-F_0[i]+1)
70                      szamlalo=szamlalo+1
71                  db=db+1
72                  n_0=n_0+1
73                  F_0.clear()
74              L.clear()
75          else:
76              while(db<=m):
77                  szamlalo=0
78                  for i in range(0,len(F)):
79                      F_0.insert(szamlalo,F[i]+1)
80                      szamlalo=szamlalo+1
81                  F_0.insert(szamlalo,0)
82                  F.clear()
83                  for i in range(0,len(F_0)):
84                      F.insert(szamlalo,n_0-F_0[i]+1)
85                      szamlalo=szamlalo+1
86                  db=db+1
87                  n_0=n_0+1
88                  F_0.clear()
89              L.clear()
90          for i in range(0,len(F)):
91              L.insert(i,F[i])
92      return L
```

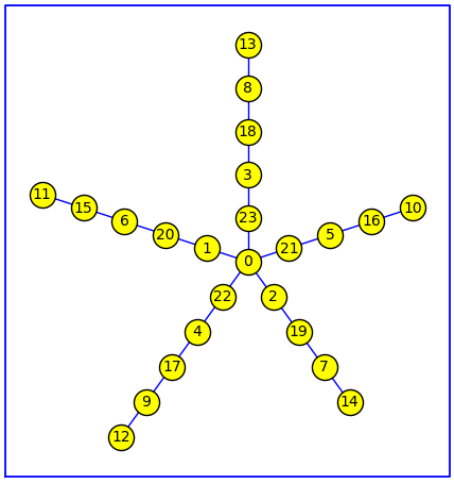
A pozicionálás úgy történik, hogy először a központi csúcs helyét adjuk meg, majd mindegyik részlistából kiolvasom az első elemet. Ezeket egy közös körön ábrázolom. Ezt m hosszú lábak esetén meg tudom valósítani. Probléma abban az esetben léphet fel, amikor elfogynak az m hosszú lábak. Ilyenkor továbbra is a részlistákról olvasom be az értékeket, viszont ha egy olyan listáról szeretne beolvasni értéket, amelynek az adott pozíciójában nincs már érték, akkor a (-1) értéket fogom beolvasni. Ebben az esetben a kör ábrázolásakor nem fog megjelenni új csúcs, viszont a helyére nem fog kerülni másik csúcs, így elkerülöm azt, hogy más helyre kerüljenek az azonos listából származó csúcsok. Végül már csak az átmérőket kell azonosan növelni, és így a gráf szépen ábrázolhatóvá válik.

```
1   def spider(l_0,l_1,m):
2       labels=Labels(l_0,l_1,m)
3       if (l_0+l_1)%2==0 and l_0!=0:
```

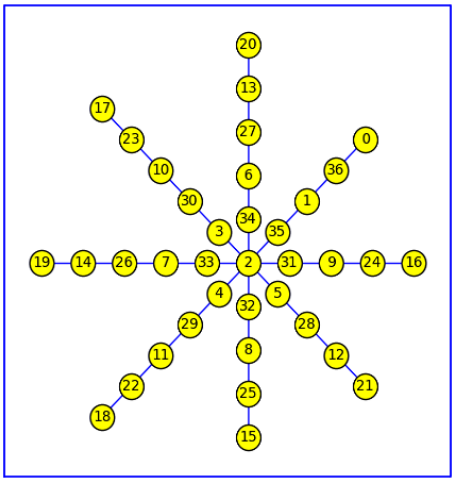
```
4           del labels[-1]
5       G=Graph()
6       db=0
7       temp=0
8       G.add_vertex(0)
9       while(db<l_1):
10          for i in range(1+temp,m+1+temp):
11              G.add_edges([(i,i+1)])
12          db=db+1
13          temp=temp+m+1
14      temp=0
15      for i in range(1,l_1+1):
16          G.add_edges([(0,1+temp)])
17          temp=temp+m+1
18      db=0
19      temp=0
20      s=l_1*(m+1)+1
21      n=l_0*m+l_1*m
22      while(db<l_0):
23          for i in range(s+temp,s+m-1+temp):
24              G.add_edges([(i,i+1)])
25          db=db+1
26          temp=temp+m
27      temp=0
28      for i in range(1,l_0+1):
29          G.add_edges([(0,s+temp)])
30          temp=temp+m
31      G.relabel(labels,inplace=True)
32
33      S=[]
34      for i in range(1,(l_0+l_1)*m+l_1+1):
35          S.insert(i,i)
36
37      pos_ver=[]
38      for i in range(0,m):
39          pos_ver.insert(i,[])
40      seged=0
41      while(seged<m):
42          temp=m+1
43          db=0
44          last_index=0
45          for i in range(1+seged,len(labels),temp):
46              pos_ver[seged].insert(i,labels[i])
47              if db==l_1:
48                  last_index=i
49                  break
```

```
50              else:
51                  db=db+1
52          if l_0!=0:
53              temp=m
54              db=0
55              for i in range(last_index+temp,len(labels),temp):
56                  pos_ver[seged].insert(i,labels[i])
57                  if db==l_0:
58                      break
59                  else:
60                      db=db+1
61              seged=seged+1
62          else:
63              seged=seged+1
64      pos_dict={}
65      tul=[]
66      j=0
67      db=0
68      for i in range(m+1,len(labels),m+1):
69          if db<l_1:
70              tul.insert(j,labels[i])
71              j=j+1
72              db=db+1
73          else:
74              break
75      for i in range(0,l_0+l_1):
76          if len(tul)!=l_0+l_1:
77              tul.insert(len(tul),-5)
78          else:
79              break
80      labels_0=labels.pop(0)
81      temp_0=0
82      r=2
83      j=0
84      while(temp_0<len(pos_ver)):
85          for i in pos_ver[temp_0]:
86              x = float(r*cos(pi/2 + ((2*pi)/len(pos_ver[temp_0]))*j))
87              y = float(r*sin(pi/2 + ((2*pi)/len(pos_ver[temp_0]))*j))
88              j=j+1
89              pos_dict[i] = [x,y]
90          temp_0=temp_0+1
91          r=r+2
92      j=0
93      for i in tul:
94          x = float(r*cos(pi/2 + ((2*pi)/len(tul))*j))
95          y = float(r*sin(pi/2 + ((2*pi)/len(tul))*j))
```

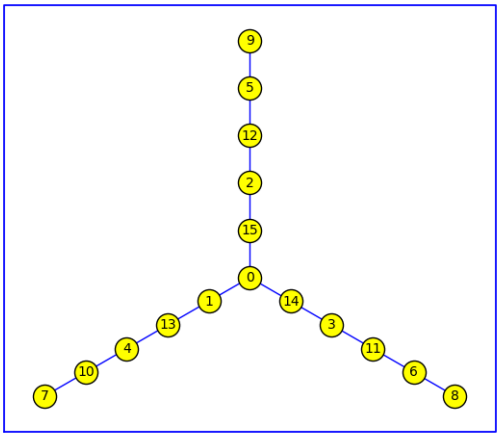
```
96          j=j+1
97          pos_dict[i]=[x,y]
98      pos_dict[labels_0]=[0,0]
99      if tul.count(-5)>0:
100         del pos_dict[-5]
101     for u,v,l in G.edges(): G.set_edge_label(u,v,abs(u-v))
102     G.graphplot(pos=pos_dict,edge_color='blue',
103             vertex_color='yellow',vertex_size=300).show(figsize=[10,5])
```



5.3. ábra. $S_{2,3,4}$ gráf



5.4. ábra. $S_{4,4,4}$ gráf



5.5. ábra. $S_{0,3,4}$ gráf

5.3. Petárda gráfok

5.3.1. Definíció. Legyen $F = (V, E, \varphi)$ véges, egyszerű gráf. Az F gráfot petárda gráfnak nevezzük, ha egy tetszőleges k hosszú útgráf minden egyes csúcsához, egy $m-1$ méretű, de különböző csillag gráfokat illesztünk.

5.3.1. Tétel. *Minden petárda gráfnak van graceful címkézése.*

Bizonyítás. Legyen F egy petárda gráf. Legyen $P(F)_k$ az F gráf részgráfja, amely speciálisan egy k hosszú útgráf. Továbbá, mindegyik csillag gráf legyen $m-1$ csúcsú. Így az F csúcsainak a száma az éppen km lesz.
Az útgráf első csúcsának legyen a címkéje 1, a második csúcsának legyen a címkéje $1+(k-1)m$, a harmadik csúcsának legyen a címkéje $1+m$, a negyedik csúcsának a címkéje legyen $1+(k-2)m$,és folytassuk ezt a címzést addig ameddig csak lehet balról jobbra.
Ezután vegyük az első csillag központi csúcsát és címkézzük el km címkével, a második csillag központi csúcsát m címkéve, a harmadik csillag központi csúcsát $(k-1)m$ címkével, a negyedik csillag központi csúcsát $2m$ címkével,.... Így maradt összesen $k(m-2)$ darab csúcs aminek nincs még címkéje.
Vegyük az i-edik csillag gráfot. Ha i páros, akkor legyen az első csúcs címkéje $2+im$, a második csúcs címkéje $3+im$, ..., az m-edik csúcs címkéje $(m-1)+im$.
Ha i páratlan, akkor legyen az első csúcs címkéje $2+(k-i)m$, a második csúcs címkéje $3+(k-i)m$,..., az m-edik csúcs címkéje $m-1+(k-i)m$.
Az így megkonstruált címkézés minden csúcshoz különböző csúcscímkét rendel, amely egy élcímkézést indukál, amely bijektív, így teljesül a graceful tulajdonság, tehát a gráf graceful gráf. □

Az implementálás során részlistákat fogok képezni, amibe címkék kerülnek. Három részlistát használok, amelyikbe kerülnek az út csúcscímkéi, a központi csillag csúcscímkéi, illetve a csillag csúcscímkéi, leszámítva a központi csúcsot, és az úton fekvő csúcsot.

```
1  import copy
2  def Labels(m,k):
3      L=[]
4      pos=0
5      temp=0
6      temp_0=1
7      for i in range(0,k):
8          if i%2==0:
9              L.insert(pos,1+temp)
10             pos=pos+1
11             temp=temp+m
12         else:
```

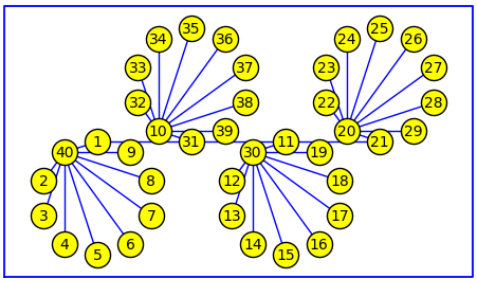
```
            L.insert(pos,1+(k-temp_0)*m)
            temp_0=temp_0+1
            pos=pos+1
    K=[]
    pos=0
    temp_0=0
    temp_1=1
    for i in range(0,k):
        if i%2==0:
            K.insert(pos,(k-temp_0)*m)
            temp_0=temp_0+1
            pos=pos+1
        else:
            K.insert(pos,temp_1*m)
            temp_1=temp_1+1
            pos=pos+1
    T=[]
    pos=0
    length=len(L)
    temp=0
    db=1
    while(length>0):
        for i in range(2+temp,m+temp):
            T.insert(pos,i)
            pos=pos+1
        temp=temp+m
        if(len(T)==(m-2)*k):
            break
        for i in range(2+(k-db)*m,m+(k-db)*m):
            T.insert(pos,i)
            pos=pos+1
        db=db+1
        length=length-1
        if(len(T)==(m-2)*k):
            break
    return L,K,T
```

A pozicionálás során először az út csúcsait helyezem el, majd a pozíciók alapján egy körön ábrázolom egyenként a csillagok csúcsait.

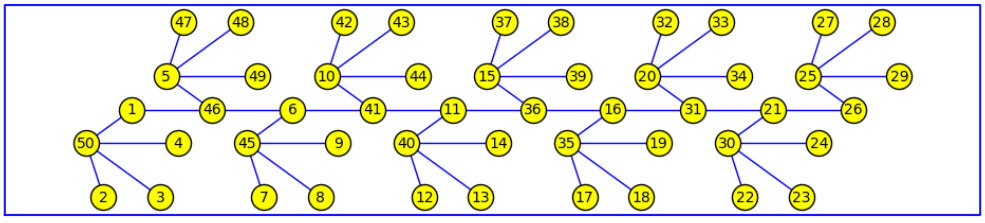
```
def firecracker(n,k):
    if n>=2 and k>=2:
        L,K,T=Labels(n,k)
        labels=[]
        for i in range(0,len(L)):
            labels.insert(i,L[i])
```

```
        for i in range(0,len(K)):
            labels.insert(i+len(L),K[i])
        for i in range(0,len(T)):
            labels.insert(i+len(L)+len(T),T[i])
        G=Graph()
        G.add_vertex(k-1)
        for i in range(0,k-1):
            G.add_edges([(i,i+1)])
        d=0
        for i in range(k,2*k):
            G.add_vertex(i)
            G.add_edges([(d,i)])
            d=d+1
        vmi=k
        db=0
        for i in range(k,2*k):
            hossz=n-2
            while(hossz>0):
                G.add_edges([(i,i+k+db)])
                hossz=hossz-1
                db=db+1
            db=db-1
        G.relabel(labels,inplace=True)
        pos_dict = {}
        S=[]
        Z=[]
        pos=0
        temp=0
        kis=0
        for i in range(0,k):
            S.insert(pos,L[i])
            pos=pos+1
            S.insert(pos,K[i])
            pos=pos+1
            for j in range(0,n-2):
                S.insert(pos,T[j+temp])
                pos=pos+1
            temp=temp+n-2
            W = copy.deepcopy(S)
            Z.insert(kis,W)
            kis=kis+1
            S.clear()
        j=0
        Z_hossz=len(Z)
        R=10
        fix=1
```

```
53          for t in range(0,len(Z)):
54              if t % 2 == 0:
55                  for i in Z[t]:
56                      x = 3*float(2*cos(pi/2 + ((2*pi)/(n))*j))+R*fix
57                      y = 3*float(2*sin(pi/2 + ((2*pi)/(n))*j))
58                      j=j+1
59                      pos_dict[i] = [x,y]
60                  j=0
61                  fix=fix+1
62              else:
63                  for i in Z[t]:
64                      x = 3*float(2*cos(pi/2 + ((2*pi)/(n))*j))+R*fix
65                      y = 3*float(2*sin(pi/2 + ((2*pi)/(n))*j))-12
66                      j=j+1
67                      pos_dict[i] = [x,-y]
68                  j=0
69                  fix=fix+1
70          for u,v,l in G.edges(): G.set_edge_label(u,v,abs(u-v))
71          G.graphplot(pos=pos_dict,edge_color='blue',
72                  vertex_color='yellow',vertex_size=300,
73                          graph_border=True).show(figsize=[5,5])
74      else:
75          print("Az n vagy a k értéke túl kicsi.")
```



5.6. ábra. $F_{10,4}$ gráf



5.7. ábra. $F_{5,10}$ gráf

5.4. Álrács gráfok

5.4.1. Definíció. Legyen $G = (V, E, \varphi)$ véges, egyszerű gráf. Legyenek a csúcsai rendre $v_{1,1}, v_{1,2}, \ldots, v_{1,n-1}, v_{1,n}, \ldots, v_{m,n-1}, v_{m,n}$. Azaz $|V(G)| = nm$. Továbbá, az i-edik csúcs legyen szomszédos az $i + (n+1)$-edik csúccsal, ha $i \equiv 0 \pmod{n}$. Ha $i \equiv n-1 \pmod{n}$, akkor az i-edik csúcs legyen szomszédos az $i+(n-1)$-edik csúccsal. Ha $i \equiv 2 \pmod{n}$, vagy $i \equiv 3 \pmod{n}$,..., vagy $i \equiv n-2 \pmod{n}$ teljesül, akkor az i-edik csúcs legyen szomszédos az $i + (n+1)$-edik, illetve az $i + (n-1)$-edik csúccsal. Az így megkonstruált gráfot álrács gráfnak nevezzük.

Megjegyzés. Ezeket a gráfokat $PG_{m,n}$-nel jelöljük. A $PG_{m,n}$ gráfok ábrázolhatóak, úgy mint a rácsgráfok, és így úgy néz ki a gráf, mintha összefüggő lenne, de nem az, két komponensből állnak. A tétel, illetve a bizonyítása Meghpara Meera cikkében [17] olvasható.

5.4.1. Tétel. *Az $PG_{m,n}$ gráfok graceful gráfok, pontosan akkor, ha $m, n \geq 3$,és m, illetve n páratlan számok.*

Bizonyítás. Tegyük fel, hogy az $SG_{m,n}$ gráf, egy rácson van ábrázolva. Legyen $v_{i,j}$ a gráf i-edik sorának, és j-edik oszlopának a csúcsa, ahol $1 \leq i \leq m$, illetve $1 \leq j \leq n$. Tegyük fel, hogy m, és n is páratlan számok, illetve $m, n \geq 3$. Definiáljuk az $f : V(SG_{m,n}) \to \{0, 1, \ldots, mn-1\}$ csúcscímkézést a következőképpen:
Ha $1 \leq i \leq m$, és i páratlan, akkor

$$f(v_{i,1}) = \frac{i-1}{2},$$

illetve

$$f(v_{i,2}) = f(v_{m,n}) + \frac{nm-n}{2} - \frac{i-1}{2}.$$

Ha $1 \leq i \leq m$, i páratlan, $3 \leq j \leq n$, és j páratlan, akkor

$$f(v_{i,j}) = f(v_{i,j-2}) + (m-1).$$

Ha $1 \leq i \leq m$, i páratlan, $4 \leq j \leq n-1$, és j páros, akkor

$$f(v_{i,j}) = f(v_{i,j-2}) - (m-1).$$

Ha $2 \leq i \leq m-1$, és i páros, akkor

$$f(v_{i,2}) = 2(m-1)(n-1) - \frac{i-2}{2},$$

illetve

$$f(v_{i,n}) = f(v_{2,n-1}) + \frac{i}{2}.$$

Ha $2 \leq i \leq m-1$, és i páros, $4 \leq j \leq n-1$, és j páros, akkor

$$f(v_{i,j}) = f(v_{i,j-2}) - (m-1).$$

Ha $2 \leq i \leq m-1$, és i páros, $1 \leq j \leq n-2$, és j páratlan, akkor

$$f(v_{i,j}) = f(v_{i,j+2}) - (m-1).$$

Az így definiált f csúcscímkézés egy f^* élcímkézést indukál, amely bijektív, így teljesül a graceful tulajdonság, tehát a gráf graceful gráf. □

```
def temp_f(m,n):
    L_paratlan=[]
    L=[]
    temp=0
    szamlalo=0
    szamlalo_2=0
    mid=[]
    q=2*(m-1)*(n-1)
    for i in range(1,m+1):
        if i%2!=0:
            for j in range(1,n+1):
                if j%2!=0:
                    if j==1:
                        L_paratlan.insert(szamlalo,(i-1)/2)
                        szamlalo=szamlalo+1
                    else:
                        L_paratlan.insert(szamlalo,L_paratlan[szamlalo-1]+
                                +(m-1))
                        szamlalo=szamlalo+1
        else:
            for j in range(2,n+1):
                if j==2:
                    mid.insert(szamlalo_2,q-(i-2)/2)
                    szamlalo_2=szamlalo_2+1
    L_2=[]
    for i in range(1,m+1):
        if i%2!=0:
            for j in range(1,n+1):
                if j==2:
                    L_2.insert(szamlalo,L_paratlan[len(L_paratlan)-1]
                            +n*(m-1)/2-(i-1)/2)
                    szamlalo=szamlalo+1
    szamlalo=0
    kor=0
    for i in range(1,m+1):
```

```
        if i%2!=0:
            for j in range(4,n+1):
                if j%2==0:
                    if j==4:
                        L.insert(szamlalo,L_2[kor]-m+1)
                        szamlalo=szamlalo+1
                        kor=kor+1
                    else:
                        L.insert(szamlalo,L[szamlalo-1]-m+1)
                        szamlalo=szamlalo+1
    szamlalo=0
    tomb=[]
    temp_0=0
    for i in range(2,m,2):
        for j in range(4,n,2):
            if j==4:
                tomb.insert(szamlalo,mid[temp_0]-(m-1))
                szamlalo=szamlalo+1
                temp_0=temp_0+1
            else:
                tomb.insert(szamlalo,tomb[len(tomb)-1]-(m-1))
                szamlalo=szamlalo+1
    utolso_oszlop=[]
    szamlalo=0
    if n==5:
        for i in range(2,m,2):
            utolso_oszlop.insert(szamlalo,tomb[0]+i/2)
            szamlalo=szamlalo+1
    elif n==3:
        for i in range(2,m,2):
            if szamlalo==0:
                utolso_oszlop.insert(szamlalo,mid[0]+i/2)
                szamlalo=szamlalo+1
            else:
                utolso_oszlop.insert(szamlalo,mid[0]+i/2)
                szamlalo=szamlalo+1
    else:
        for i in range(2,m,2):
            utolso_oszlop.insert(szamlalo,tomb[1]+i/2)
            szamlalo=szamlalo+1
    tomb_2=[]
    szamlalo=0
    temp_0=0
    if n==3:
        for i in range(2,m,2):
            for j in range(n-2,0,-2):
```

```
                tomb_2.insert(szamlalo,utolso_oszlop[temp_0]-(m-1))
                szamlalo=szamlalo+1
                temp_0=temp_0+1
    else:
        for i in range(2,m,2):
            for j in range(n-2,0,-2):
                if j==n-2 :
                    tomb_2.insert(szamlalo,utolso_oszlop[temp_0]-(m-1))
                    szamlalo=szamlalo+1
                    temp_0=temp_0+1
                else:
                    tomb_2.insert(szamlalo,tomb_2[len(tomb_2)-1]-(m-1))
                    szamlalo=szamlalo+1
    return L_paratlan, L_2,L,mid,tomb,utolso_oszlop,tomb_2
```

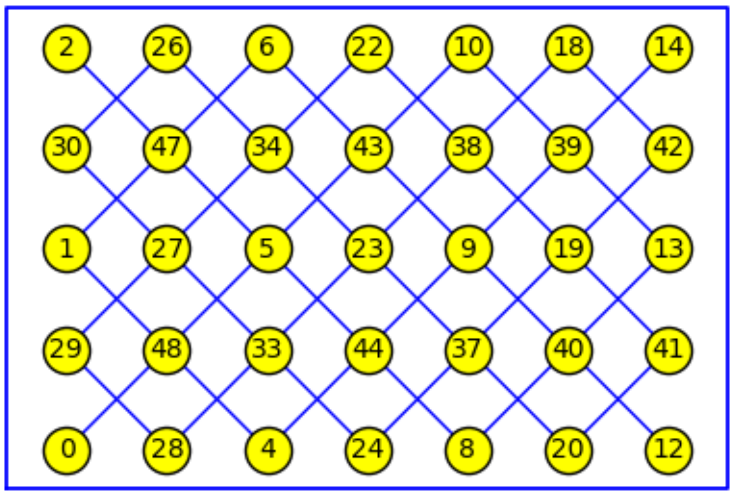
A könnyebb ábrázolás miatt a részlistákat rendezni kell, és a listákba most már soronként kerülnek majd az értékek. Ezen élekkel az ábrázolás már sokkal könnyebben történik, hiszen a sorokat már egyszerűen csak soronként kell feltölteni.

```
def Labels(m,n):
    csucsok=[[]]
    S,MID,L,mid,tomb,utolso_oszlop,tomb_2=temp_f(m,n)
    for i in range(0,(m+1)/2-1):
        csucsok.insert(i,[])
    szamlalo=0
    kor=0
    temp=0
    while(kor!=len(csucsok)):
        for i in range(0+temp,(n+1)/2+temp):
            csucsok[kor].insert(i,S[i])
            szamlalo=szamlalo+1
        kor=kor+1
        temp=temp+(n+1)/2
    for i in range(0,len(csucsok)):
        csucsok[i].insert(1,MID[i])
    k_t=(n-3)/2
    l_t=(m+1)/2
    l=l_t
    szamlalo=0
    temp_1=0
    while(l>0):
        temp_0=3
        k=k_t
        while(k>0):
```

```
            csucsok[szamlalo].insert(temp_0,L[temp_1])
            temp_0=temp_0+2
            temp_1=temp_1+1
            k=k-1
        szamlalo=szamlalo+1
        l=l-1
    ismetlodes_hossza=(n-3)/2+1
    ismetlodes_szama=(m+1)/2-1
    temp=[]
    szamlalo=0
    kor=0
    temp_1=0
    while(kor!=ismetlodes_szama):
        szamlalo=0
        for i in range(0+temp_1,len(tomb_2)):
            if szamlalo!=ismetlodes_hossza:
                temp.insert(0+temp_1,tomb_2[i])
                szamlalo=szamlalo+1
        kor=kor+1
        temp_1=temp_1+ismetlodes_hossza
    csucsok_2=[]
    for i in range(0,len(mid)):
        csucsok_2.insert(i,[])
    szamlalo=0
    lep=0
    szamlalo=1
    szamlalo_2=0
    for j in range(0,len(mid)):
        if j==0:
            csucsok_2[j].insert(0, temp[0])
        else:
            csucsok_2[j].insert(0,temp[szamlalo])
            szamlalo=szamlalo+1
        csucsok_2[j].insert(1, mid[lep])
        for i in range(2,n-1):
            if i%2==0:
                csucsok_2[j].insert(i,temp[szamlalo])
                szamlalo=szamlalo+1
            else:
                csucsok_2[j].insert(i,tomb[szamlalo_2])
                szamlalo_2=szamlalo_2+1
        csucsok_2[j].insert(6,utolso_oszlop[lep])
        lep=lep+1
    osszes_csucs=[]
    szamlalo=0
    szamlalo_1=0
```

```
72      for i in range(0,len(csucsok)+len(csucsok_2)):
73          if i%2==0:
74              osszes_csucs.insert(i,csucsok[szamlalo])
75              szamlalo=szamlalo+1
76          else:
77              osszes_csucs.insert(i,csucsok_2[szamlalo_1])
78              szamlalo_1=szamlalo_1+1
79      return osszes_csucs
```

```
1   def pseudogrid(m,n):
2       if (m>2 and n>2) and (m%2!=0 and n%2!=0):
3           csucsok=Labels(m,n)
4           labels=[]
5           szamlalo=0
6           for i in range(0,len(csucsok)):
7               for j in range(0,len(csucsok[i])):
8                   labels.insert(szamlalo,csucsok[i][j])
9                   szamlalo=szamlalo+1
10          G=Graph()
11          for i in range(0,n*m):
12              G.add_vertex(i)
13          for i in range(0,(m-1)*n):
14              if i%n==0:
15                  G.add_edges([(i,i+(n+1))])
16              elif i%n==n-1:
17                  G.add_edges([(i,i+(n-1))])
18              else:
19                  G.add_edges([(i,i+(n-1))])
20                  G.add_edges([(i,i+(n+1))])
21          G.relabel(labels,inplace=True)
22          pos_dict={}
23          x=0;y=0
24          for i in csucsok:
25              x=0
26              y=y+100
27              for j in i:
28                  x=x+100
29                  y
30                  pos_dict[j]=[x,y]
31          G.graphplot(pos=pos_dict,edge_color='blue',
32                          vertex_color='yellow').show(figsize=[5,5])
33      else:
34          print("Az n,m>2 vagy m,n páratlan paritása nem teljesül.")
```



5.8. ábra. $PG_{5,7}$ gráf

Irodalomjegyzék

[1] A. Rosa, „On certain valuations of the vertices of a graph," *Journal of Graph Theory - JGT*, 01 1967.

[2] S. W. Golomb, „How to number a graph," in *Graph Theory and Computing*, pp. 23 – 37, Academic Press, 1972.

[3] E. Gnang, „On graceful and harmonious labelings of trees." 11 2018.

[4] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 9.0)*, 2020. https://www.sagemath.org.

[5] Wikipedia contributors, „Hadamard matrix — Wikipedia, the free encyclopedia," 2020.

[6] S. Vaidya and N. Shah, „Graceful and odd graceful labeling of some graphs," *International Journal of Mathematics and Soft Computing*, vol. 3, p. 61, 01 2013.

[7] S. Vaidya and L. Bijukumar, „Some new graceful graphs," *Advances and Applications in Discrete Mathematics*, vol. 6, 01 2010.

[8] R. L. Watson, „A survey on the graceful labeling of graphs," Master's thesis, University of Colorado at Denver, 2000.

[9] S. Guruswamy and S. Kuppusamy, „Subdivision of wheel is graceful and cordial," *International Journal of Mathematical Analysis*, vol. 9, pp. 817–822, 01 2015.

[10] V. Srinivasan and N.Amirthavahini, „New results on some vertex labeling of graphs," *International Journal of Pure and Applied Mathematics*, vol. 118, pp. 891–898, 02 2018.

[11] A. Elumalai and S. Guruswamy, „Gracefulness of a cycle with parallel chords and parallel p k -chords of different lengths," *Ars Combinatoria*, vol. 104, 04 2012.

[12] A. Solairaju, S. Subbulakshmi, and R. Kokila, „Graceful labelings for sun graph, sun extension graph, double fan, and leg graph with star," vol. 13, pp. 1357–1364, 2017.

[13] A. Elumalai and A. Ephremnath, „Gracefulness of a cycle with zigzag chords," *International Journal of Pure and Applied Mathematics*, vol. 101, pp. 629–635, 01 2015.

[14] K. M. Koh and K. Y. Yap, „Graceful numberings of cycles with a p-chord," *Bull. Inst.Math. Acad. Sinica*, vol. 12, pp. 41–48, 1985.

[15] P. Bahls, S. Lake, and A. Wertheim, „Gracefulness of families of spiders," *Involve*, vol. 3, 10 2010.

[16] E. Robeva, „An extensive survey of graceful trees," 07 2011. Undergraduate Honours thesis.

[17] M. Meghpara, „Graceful labeling for grid related graphs," *International Journal of Mathematics and Soft Computing*, vol. 5, pp. 111–117, 04 2015.

[18] J. Gallian, „A dynamic survey of graph labeling," *Electron J Combin DS6*, vol. 19, 11 2000.

[19] C. M. Cavalier, „Graceful labelings," Master's thesis, University of South Carolina, 2009.

[20] J. Maowa, „Study on graceful labeling of trees," Master's thesis, Bangladesh University of Engineering and Technology (BUET), 07 2016.

[21] R. M. Zhou, „Graceful labeling of graphs," Master's thesis, UFRJ/COPPE, Rio de Janeiro, 12 2016.

[22] A. Graf, „Graceful labelings of pendant graphs," *Rose-Hulman Undergraduate Mathematics Journal*, vol. Vol. 15, 2014.

[23] V. Kaneria and H. Makadia, „Graceful labeling for cycle of graphs," vol. 6, p. 173–178, 2014.