# Skupina 18: Kemijski grafi

*Avtorja: David Planinšek Šilc, Lenart Žerdin*
*Datum: 20. 12. 2024*

## Opis problema

Najina naloga temelji na raziskovanju kemijskih grafov in njihovem $\sigma_t^{f(n)}(G)$ indeksu. Zanima naju, kako se indeks v odvisnosti od različnih $f(n)$ spreminja in katera stopenjska zaporedja grafa pridelajo njegov maksimum. Omejila sva se na funkcije $f(n) = \frac{1}{n}$ in $f(n) = c$ za $c \in (0, 1)$, kjer sva podrobneje gledala tiste $c$, ki so blizu 0 in 1.

## Definicije

1. Graf je kemijski, če je neusmerjen, povezan in so vsa njegova vozlišča stopnje največ 4. Če ima kemijski graf $a_i$ vozlišč stopnje $i$, $1 \leq i \leq 4$, potem njegovo stopenjsko zaporedje označimo kot $(1^{a_1}, 2^{a_2}, 3^{a_3}, 4^{a_4})$.

2. Definiramo totalni $\sigma$-indeks iregularnosti, v angleščini 'Total $\sigma$-irregularity', $\sigma_t^{f(n)}(G)$ kot:

$$\sigma_t^{f(n)}(G) = \sum_{\{u,v\} \subseteq V(G)} |d_G(u) - d_G(v)|^{f(n)},$$

   kjer je $n = |V(G)|$ in je $f(n)$ funkcija, definirana za $n \geq 4$.

## Izrek

Naj bo $n \geq 7$, $f(n) \leq \log_3\left(\frac{3n^2}{3n^2-8}\right)$, in naj bo $(1^{a_1}, 2^{a_2}, 3^{a_3}, 4^{a_4})$ stopenjsko zaporedje kemijskega grafa $G$ z maksimalno vrednostjo $\sigma_t^{f(n)}(G)$. Potem velja:

1. Če $n = 4k - 1$, potem $a_1 = a_3 = a_4 = k$ in $a_2 = k - 1$.
2. Če $n = 4k$, potem $a_1 = a_2 = a_3 = a_4 = k$.
3. Če $n = 4k + 1$, potem $a_1 = a_2 = a_3 = k$ in $a_4 = k + 1$.
4. Če $n = 4k + 2$, potem velja bodisi $a_1 = a_3 = k$ in $a_2 = a_4 = k + 1$, bodisi $a_1 = a_3 = k + 1$ in $a_2 = a_4 = k$.

Ker je za kemijske grafe razlika med stopnjami vozlišč omejena, domnevamo naslednje:

- **Domneva 1:** Isti grafi, kot v Izreku, imajo maksimalno vrednost za $\sigma_t^{f(n)}$, če je $f(n) = \frac{1}{n}$.
- **Domneva 2:** Isti grafi, kot v Izreku, imajo maksimalno vrednost za $\sigma_t^{f(n)}$, če je $f(n) = c$, kjer je $c$ konstanta v intervalu $(0, 1)$.

## Algoritmi in psevdokode

Za preverjanje domnev sva napisala algoritme, ki generirajo kemijske grafe in izračunajo njihov $\sigma_t^{f(n)}$ indeks. Najprej sva se lotila sistematičnega iskanja za grafe z $n$ vozlišči, kjer je $n \in [7, 14]$, $n \in \mathbb{N}$. Za večje grafe sva zaradi predolgega trajanja iskanja uporabila algoritma Hill climbing in Simulated annealing. Natančneje za grafe velikosti $n \in [15, 20] \cup [50, 53] \cup [100, 103]$.

### Sistematično iskanje

Pri sistematičnem iskanju sva generirala vsa možna stopenjska zaporedja z uporabu knjižnice *nauty_geng*, za različne $n$ in izračunala $\sigma_t^{f(n)}(G)$ za različne $f(n)$. Nato sva izbrala tiste grafe, kjer je bila pri določenih $f(n)$ vrednostjo $\sigma_t^{f(n)}(G)$ največja.

**Psevdokoda za sistematično iskanje**

```
function GenerateUniqueChemicalGraphsConfigs(n)
    unique_configs ← []
    for all g v graphs.nauty_geng(f"n − c − D4") do
        config ← DegreeConfiguration(g)
        if config ∉ unique_configs then
            Dodaj config v unique_configs
        end if
    end for
    return unique_configs
end function
function SigmaTotalIrregularityFromConfig(config, f_n)
    for all config v degree_list do
        compute σ_t^{f(n)}(G)
    end for
    return (config| max(σ_t^{f(n)}(G)))
end function
```

**Hill climbing algoritem**

Hill climbing algoritem je optimizacijski algoritem, ki iterativno izboljšuje trenutno rešitev tako, da na vsakem koraku poišče sosednjo rešitev z boljšo, v najinem primeru večjo, $\sigma_t^{f(n)}(G)$ vrednostjo. Algoritem sva ustavila po 100000 iteracijah.

**Psevdokoda za Hill climbing algoritem**

```
function GenerateInitialGraph(n)
    Ustvarimo drevo na n vozliščih
end function
function MutateGraph(g, u, v)
    if g vsebuje povezavo (u, v) then
        Odstrani povezavo (u, v) iz g
        if g je povezan then
            return g
        else
            Dodaj nazaj (u, v)
            return g
        end if
    else
        Dodaj povezavo (u, v) v g
        if max(degree(g)) ≤ 4 then
            return g
        else
            Odstrani (u, v)
            return g
        end if
    end if
end function
```

```
function HillClimbing(n, f, iterations)
    current_graph ← GenerateInitialGraph(n)
    degree_counts ← seznam stopenj v current_graph
    for i in range(iterations) do
        vertices ← seznam vozlišč v current_graph
```

```
        (u, v) ← naključno izbran par vozlišč iz vertices
        original_contribution ← 0
        for all w v vertices do
            if w ≠ u then
                org_contribution+ = |degree_counts[u] − degree_counts[w]|^{f_n}
            end if
            if w ≠ v in w ≠ u then
                org_contribution+ = |degree_counts[v] − degree_counts[w]|^{f_n}
            end if
        end for
        new_graph ← MutateGraph(current_graph, u, v)
        new_degree_counts ← seznam stopenj v new_graph
        new_contribution ← 0
        for all w v vertices do
            if w ≠ u then
                new_contribution+ = |degre_counts[u] − degree_counts[w]|^{f_n}
            end if
            if w ≠ v in w ≠ u then
                new_contribution+ = |degree_counts[v] − degree_counts[w]|^{f_n}
            end if
        end for
        if new_contribution > original_contribution then
            current_graph ← new_graph
            degree_counts ← new_degree_counts
        end if
    end for
    σ_t^{f(n)}(G) ← 0
    for all (x, y) ∈ pari vozlišč v current_graph do
        σ_t^{f(n)}(G)+ = |degree(x) − degree(y)|^{f_n}
    end for
    return (current_graph, σ_t^{f(n)}(G))
end function
```

### Simulated annealing algoritem

Problem pri Hill climbing algoritmu je, da se lahko zatakne v lokalnem maksimumu. Simulated annealing algoritem je pristop, ki se temu izogne, tako da včasih sprejema tudi slabše rešitve. Algoritem sva ustavila po 100000 iteracijah.

### Psevdokoda za Simulated annealing algoritem

```
function SIMULATED ANNEALING(n, f_n, iterations, T, α)
    T = 1
    α = 0.99
    Δ ← new_contribution − original_contribution
    if Δ > 0 or random() < exp(Δ/T) then
        current_graph ← new_graph
        degree_counts ← new_degree_counts
    end if
    T ← T · α
end function
```

## Tabele in grafi

### Tabela za sistematično iskanje

| $n$ | $\frac{1}{n}$ | 0.0001 | 0.1 | 0.2 | 0.45 | 0.55 | 0.8 | 0.9 | 0.9995 |
|---|---|---|---|---|---|---|---|---|---|
| 7 | (2, 1, 2, 2) | (2, 2, 2, 1) | (2, 1, 2, 2) | (2, 1, 2, 2) | (2, 1, 2, 2) | (3, 1, 1, 2) | (3, 0, 1, 3) | (3, 0, 1, 3) | (3, 0, 1, 3) |
| 8 | (2, 2, 2, 2) | (2, 2, 2, 2) | (2, 2, 2, 2) | (2, 2, 2, 2) | (3, 1, 1, 3) | (3, 1, 1, 3) | (3, 1, 1, 3) | (4, 0, 0, 4) | (4, 0, 0, 4) |
| 9 | (2, 2, 2, 3) | (2, 3, 2, 2) | (2, 2, 2, 3) | (2, 2, 2, 3) | (3, 2, 1, 3) | (3, 2, 1, 3) | (4, 1, 0, 4) | (4, 1, 0, 4) | (4, 1, 0, 4) |
| 10 | (3, 2, 3, 2) | (3, 2, 3, 2) | (3, 2, 3, 2) | (3, 2, 3, 2) | (4, 1, 2, 3) | (4, 1, 2, 3) | (5, 0, 1, 4) | (5, 0, 1, 4) | (5, 0, 1, 4) |
| 11 | (3, 2, 3, 3) | (3, 3, 3, 2) | (3, 2, 3, 3) | (3, 2, 3, 3) | (4, 2, 2, 3) | (4, 1, 2, 4) | (5, 0, 1, 5) | (5, 0, 1, 5) | (5, 0, 1, 5) |
| 12 | (3, 3, 3, 3) | (3, 3, 3, 3) | (3, 3, 3, 3) | (3, 3, 3, 3) | (4, 2, 2, 4) | (4, 2, 2, 4) | (5, 1, 1, 5) | (5, 1, 1, 5) | (6, 0, 0, 6) |
| 13 | (3, 3, 3, 4) | (3, 3, 3, 4) | (3, 3, 3, 4) | (3, 3, 3, 4) | (4, 3, 2, 4) | (4, 2, 2, 5) | (5, 1, 1, 6) | (6, 1, 0, 6) | (6, 1, 0, 6) |
| 14 | (4, 3, 4, 3) | (4, 3, 4, 3) | (4, 3, 4, 3) | (4, 3, 4, 3) | (5, 2, 3, 4) | (5, 2, 3, 4) | (6, 1, 2, 5) | (7, 0, 1, 6) | (7, 0, 1, 6) |

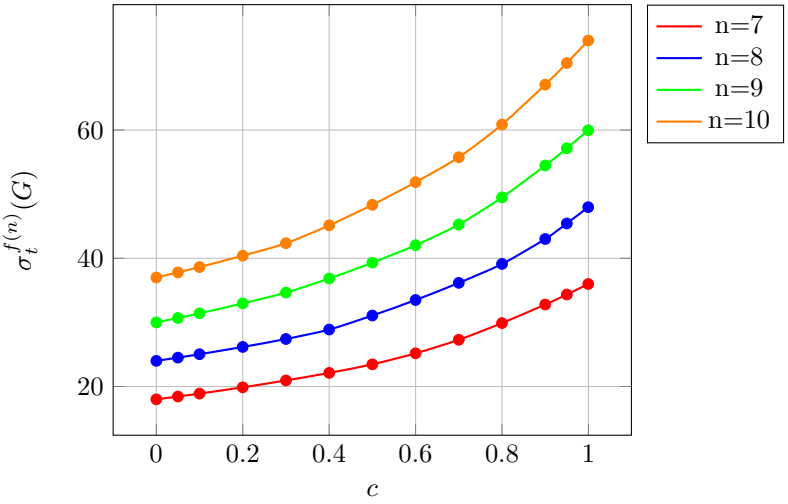
### Tabela za Hill climbing algoritem

| $n$ | $\frac{1}{n}$ | 0.001 | 0.1 | 0.5 | 0.9 | 0.995 |
|---|---|---|---|---|---|---|
| 15 | (4, 3, 4, 4) | (4, 3, 4, 4) | (5, 2, 3, 5) | (5, 2, 3, 5) | (6, 1, 2, 6) | (7, 0, 1, 7) |
| 16 | (4, 4, 4, 4) | (4, 4, 4, 4) | (5, 3, 3, 5) | (5, 3, 3, 5) | (6, 1, 2, 7) | (6, 1, 2, 7) |
| 17 | (4, 4, 4, 5) | (4, 4, 4, 5) | (5, 3, 3, 6) | (6, 3, 2, 6) | (6, 2, 2, 7) | (7, 1, 1, 8) |
| 18 | (4, 5, 4, 5) | (5, 4, 5, 4) | (5, 4, 3, 6) | (7, 2, 3, 6) | (7, 2, 1, 8) | (8, 0, 2, 8) |
| 19 | (5, 4, 5, 5) | (5, 4, 5, 5) | (6, 3, 4, 6) | (7, 3, 3, 6) | (7, 1, 3, 8) | (9, 0, 1, 9) |
| 20 | (5, 5, 5, 5) | (5, 5, 5, 5) | (6, 4, 4, 6) | (7, 3, 3, 7) | (7, 3, 1, 9) | (8, 1, 2, 9) |
| 50 | (12, 13, 12, 13) | (12, 13, 12, 13) | (15, 10, 9, 16) | (17, 8, 9, 16) | (21, 4, 3, 22) | (21, 4, 1, 24) |
| 51 | (13, 12, 13, 13) | (13, 12, 13, 13) | (16, 9, 10, 16) | (17, 8, 9, 17) | (21, 4, 3, 23) | (24, 1, 0, 26) |
| 52 | (13, 13, 13, 13) | (13, 13, 13, 13) | (16, 10, 10, 16) | (17, 9, 9, 17) | (22, 3, 4, 23) | (22, 3, 2, 25) |
| 53 | (13, 13, 13, 14) | (13, 13, 13, 14) | (17, 10, 9, 17) | (18, 9, 8, 18) | (23, 3, 3, 24) | (23, 3, 1, 26) |
| 100 | (25, 25, 25, 25) | (25, 25, 25, 25) | (31, 19, 19, 31) | (34, 16, 16, 34) | (44, 5, 6, 45) | (47, 3, 1, 49) |
| 101 | (25, 25, 25, 26) | (25, 25, 25, 26) | (32, 19, 18, 32) | (34, 17, 16, 34) | (45, 5, 5, 46) | (45, 5, 1, 50) |
| 102 | (25, 26, 25, 26) | (25, 26, 25, 26) | (32, 19, 18, 33) | (34, 17, 16, 35) | (46, 4, 6, 46) | (44, 6, 2, 50) |
| 103 | (26, 25, 26, 26) | (26, 25, 26, 26) | (27, 24, 25, 27) | (35, 16, 17, 35) | (47, 4, 5, 47) | (49, 2, 1, 51) |
| 1000 | (250, 250, 250, 250) | (250, 250, 250, 250) | (264, 236, 236, 264) | (337, 163, 163, 337) | (354, 128, 58, 460) | (353, 112, 37, 498) |
| 1001 | (250, 250, 250, 251) | (250, 250, 250, 251) | (264, 236, 236, 265) | (337, 163, 163, 338) | (361, 129, 51, 460) | (361, 112, 31, 497) |

### Tabela za Simulated annealing algoritem

| $n$ | $\frac{1}{n}$ | 0.001 | 0.1 | 0.5 | 0.9 | 0.995 |
|---|---|---|---|---|---|---|
| 15 | (4, 3, 4, 4) | (4, 3, 4, 4) | (4, 3, 4, 4) | (5, 2, 3, 5) | (6, 1, 2, 6) | (7, 0, 1, 7) |
| 16 | (4, 4, 4, 4) | (4, 4, 4, 4) | (4, 4, 4, 4) | (5, 3, 3, 5) | (7, 1, 1, 7) | (6, 1, 2, 7) |
| 17 | (4, 4, 4, 5) | (4, 4, 4, 5) | (4, 4, 4, 5) | (6, 3, 2, 6) | (6, 2, 2, 7) | (7, 1, 1, 8) |
| 18 | (4, 5, 4, 5) | (5, 4, 5, 4) | (5, 4, 5, 4) | (7, 2, 3, 6) | (6, 2, 2, 8) | (8, 1, 0, 9) |
| 19 | (5, 4, 5, 5) | (5, 4, 5, 5) | (5, 4, 5, 5) | (6, 3, 4, 6) | (8, 1, 2, 8) | (8, 1, 0, 10) |
| 20 | (5, 5, 5, 5) | (5, 5, 5, 5) | (5, 5, 5, 5) | (7, 3, 3, 7) | (9, 1, 1, 9) | (8, 1, 2, 9) |
| 50 | (12, 13, 12, 13) | (12, 13, 12, 13) | (13, 12, 11, 14) | (17, 8, 9, 16) | (20, 4, 4, 22) | (24, 0, 2, 24) |
| 51 | (13, 12, 13, 13) | (13, 12, 13, 13) | (14, 12, 12, 13) | (17, 8, 9, 17) | (21, 4, 3, 23) | (23, 1, 1, 26) |
| 52 | (13, 13, 13, 13) | (13, 13, 13, 13) | (14, 12, 12, 14) | (17, 9, 9, 17) | (22, 3, 4, 23) | (24, 1, 2, 25) |
| 53 | (13, 13, 13, 14) | (13, 13, 13, 14) | (14, 13, 12, 14) | (18, 9, 8, 18) | (23, 3, 3, 24) | (25, 1, 1, 26) |
| 100 | (25, 25, 25, 25) | (25, 25, 25, 25) | (26, 24, 24, 26) | (34, 16, 16, 34) | (42, 7, 6, 45) | (46, 3, 2, 49) |
| 101 | (25, 25, 25, 26) | (25, 25, 25, 26) | (26, 24, 24, 27) | (34, 17, 16, 34) | (46, 5, 4, 46) | (44, 6, 0, 51) |
| 102 | (25, 26, 25, 26) | (25, 26, 25, 26) | (26, 25, 24, 27) | (34, 17, 16, 35) | (45, 6, 5, 46) | (46, 4, 2, 50) |
| 103 | (26, 25, 26, 26) | (26, 25, 26, 26) | (27, 24, 25, 27) | (35, 16, 17, 35) | (47, 4, 5, 47) | (42, 9, 0, 52) |
| 1000 | (250, 250, 250, 250) | (250, 250, 250, 250) | (264, 236, 236, 264) | (337, 163, 163, 337) | (360, 129, 52, 459) | (351, 111, 41, 497) |
| 1001 | (250, 250, 250, 251) | (250, 250, 250, 251) | (264, 236, 236, 265) | (337, 163, 163, 338) | (358, 124, 58, 461) | (356, 112, 34, 499) |

**Graf $\sigma_t^{f(n)}(G)$ indeksa v odvisnosti od števila vozlišč $n$**




## Rezultati in ugotovitve

Ugotovila sva, da so največje vrednosti $\sigma_t^{f(n)}(G)$ za $f(n) = \frac{1}{n}$ dosežene pri grafih, ki imajo stopenjsko zaporedje enako kot v izreku. Enako velja za $f(n) = c$, kjer so vrednosti za $c$ blizu 0. Nato se na neki točki začneta notranja člena stopenjskega zaporedja zmanjševati, zunanja pa povečevati. Za $c$ blizu 1 pa so vrednosti $\sigma_t^{f(n)}(G)$ maksimizirane takrat, ko sta zunanja člena stopenjskega zaporedja $(1^{a_1}, 2^{a_2}, 3^{a_3}, 4^{a_4})$ čim večja, notranja pa čim manjša. Najini algoritmi teh vrednosti niso dosegli, saj sva uporabila omejeno število iteracij. Misliva, da za grafe drži naslednja trditev.
Domneva 1 torej drži, domneva 2 pa ne drži, za c blizu 1.

### Trditev

Naj bo $n \geq 7$, $f(n) = c$, za $c$ 'zelo blizu' 1 in naj bo $(1^{a_1}, 2^{a_2}, 3^{a_3}, 4^{a_4})$ stopenjsko zaporedje kemijskega grafa $G$ z maksimalno vrednostjo $\sigma_t^{f(n)}(G)$. Potem velja:

1. Če $n = 4k$, potem $a_1 = a_4 = 2k$ in $a_2 = a_3 = 0$.
2. Če $n = 4k + 1$, potem $a_1 = a_4 = 2k$ in $a_2 = 1$ $(a_3 = 0)$ ali $a_3 = 1$ $(a_2 = 0)$.
3. Če $n = 4k + 2$, potem $a_1 = 2k$, $a_2 = 1$, $a_3 = 0$ in $a_4 = 2k + 1$.
4. Če $n = 4k + 3$, potem $a_1 = 2k - 1$, $a_2 = 0$, $a_3 = 1$ in $a_4 = 2k - 1$.

### Konstruiranje takih grafov

Na naslednji strani, sva narisala grafe, ki imajo maksimalno vrednost $\sigma_t^{f(n)}(G)$ za $f(n) = c$, kjer je c zelo blizu 1. Grafi so narisani za $n \in [16, 19]$ in so le ena od možnosti, zraven pa sva dodala še njihovo konfiguracijo. Z rdečo so označena vozlišča stopnje 1, z roza vozlišča stopnje 2, z zeleno vozlišča stopnje 3 in z modro vozlišča stopnje 4.

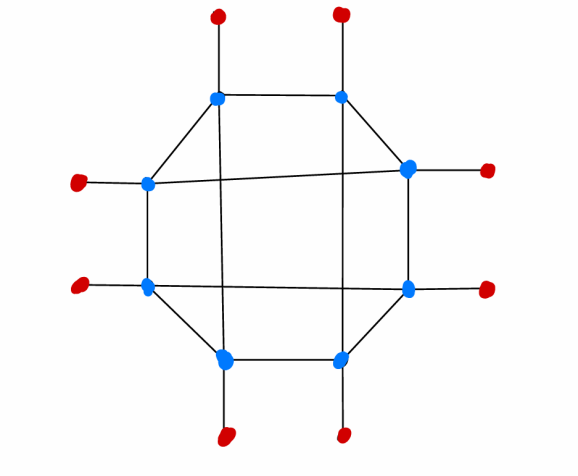
V splošnem lahko take grafe konstruiramo za poljubno velik $n$. Če je $n = 4k$, potem graf konstruiramo tako, da konstruiramo cikel dolžine $2k$, kjer gre ven iz vsakega vozlišča še ena povezava oziroma list. Na koncu vsako vozlišče, ki je del cikla povežemo z nasprotnje ležečim vozliščem.

Če je $n = 4k + 1$, konstruiramo cikel dolžine $2k$, kjer gre ven iz vsakega vozlišča še ena povezava. Izberemo poljubnen list, ki mu dodamo še eno povezavo, da dobimo graf, ki maksimizira $\sigma_t^{f(n)}(G)$.

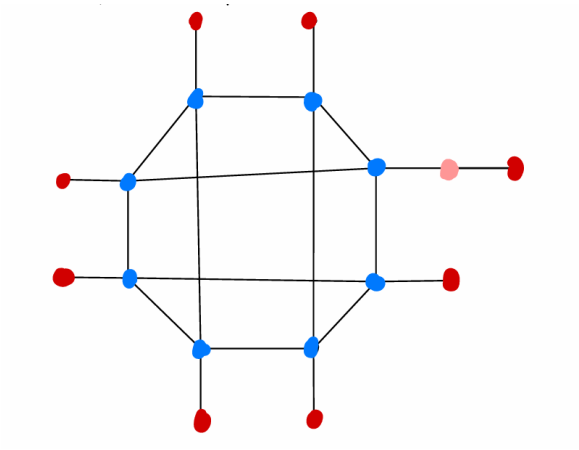
Če je $n = 4k+2$, potem konstruiramo cikel dolžine $2k$. Nasprotno ležeča vozlišča povežemo ter izberemo dve sosednji vozlišči, katerima dodamo novega skupnega soseda. Ostalim tako kot prej dodamo lsite. Iz soseda prej izbranih dveh vozlišč dodamo še dva lista. Iz enega od novo dodanih listov dodamo še eno

povezavo.

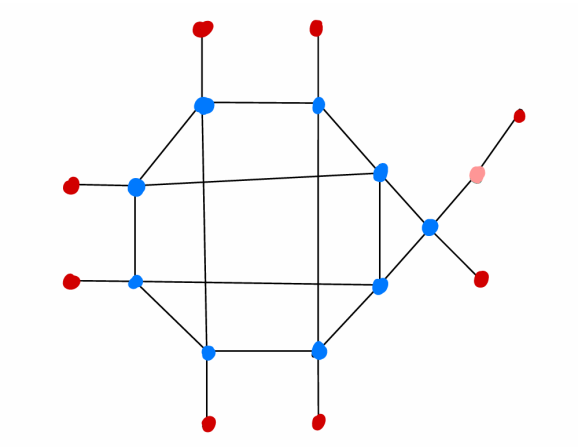
Če je $n = 4k + 3$, konstruiramo cikel dolžine $2k$. Izberemo sosednji vozlišči, ki vsaki dodamo svoj list ter ta lista povežemo med seboj. Nato enemu od listov dodamo še eno povezavo, drugemu pa dve. Na koncu še vsakemu od preostalih vozlišč iz cikla dodamo list.



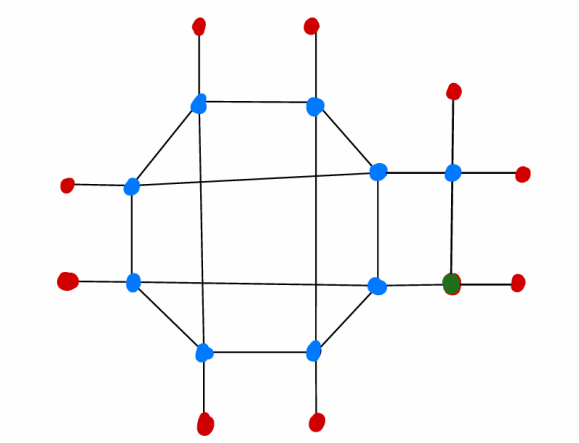
Graf za n = 16; (8, 0, 0, 8)



Graf Za n = 17; (8, 1, 0, 8)



Graf za n = 18; (8, 1, 0, 9)



Graf za n = 19; (9, 0, 1, 9)