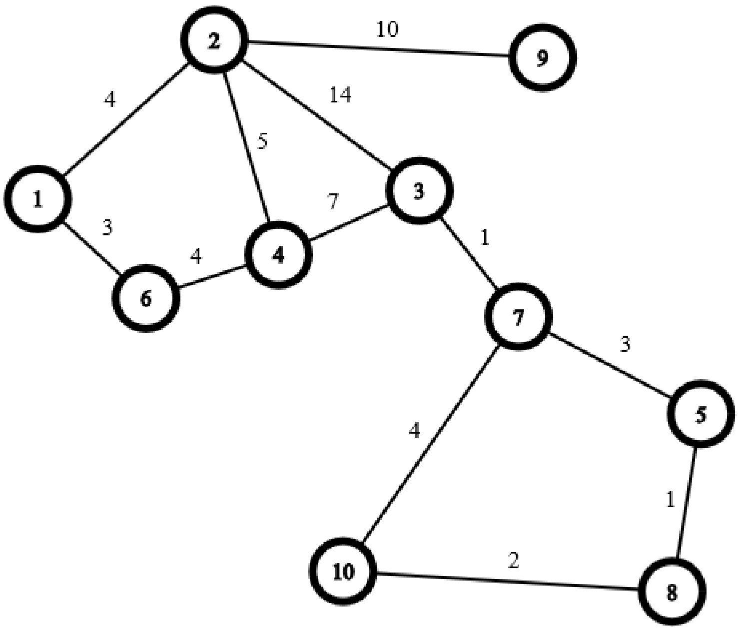
Examen Algoritmi Fundamentali - Varianta 2

Daria Pirvulescu

January 2024



Graful pentru exercitiile 1-6

Problema 1

Eliminati un numar minim de varfuri (posibil 0) astfel incat graful obtinut sa fie bipartit si indicati o bipartitie a acestuia (2-colorare).

Rezolvare:

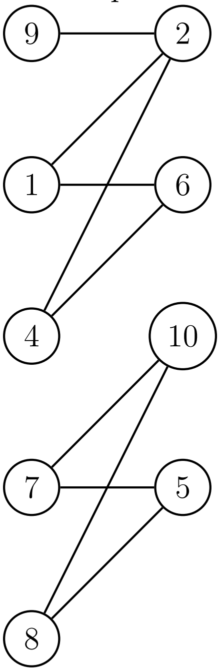
Conform *teoremei lui Konig* un graf este bipartit daca si numai daca toate ciclurile sale elementare au lungime para.

In graful din enunt exista urmatoarele cicluri de lungime impara:

- $3 - 2 - 4 - 3$
- $3 - 2 - 1 - 6 - 4 - 3$

Prin eliminarea nodului 3 ambele ciclui vor avea cu un element mai putin, deci vor avea lungime para (deci nu vom mai avea cicluri de lungmie impara si astfel graful va deveni bipartit).

O bipartitie a acestuia este:



Cele doua multimi de noduri sunt $\{1, 4, 7, 8, 9\}$, $\{2, 6, 10, 5\}$

Problema 2

Exemplificati (cu explicatii) cum functioneaza parcurgerea in latime bf(4), ilustrand si arborele bf asociat.

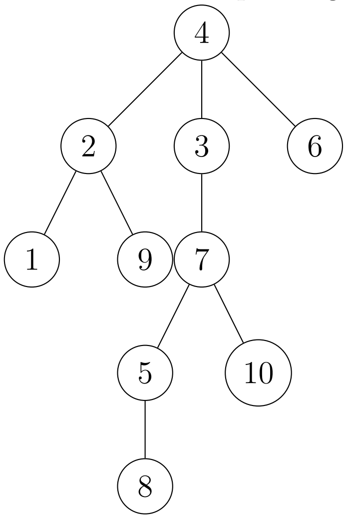
Rezolvare:

Algoritmul **BFS** parcurge mai intai toti vecinii nodului de start, abia apoi vecinii vecinilor s.a.m.d. Algoritmul utilizeaza o coada pentru a pastra ordinea nodurilor.

Mai intai vom adauga in coada nodul 4, si il marcam ca vizitat.

- Scoatem din coada nodul 4 vom parcurge nodurile $2, 3, 6$ si le marcam ca vizitate. Coada devine $2, 3, 6$.
- Scoatem din coada nodul 2 vom adauga nodurile vecine $1, 9$. Coada devine $3, 6, 1, 9$.
- Scoatem din coada nodul 3. Adaugam vecinul 7. Coada devine $6, 1, 9, 7$.
- Scoatem din coada nodul 6. Toti vecinii sai au fost deja vizitati.
- Scoatem din coada nodul 1. Acesta nu are vecini care nu au fost deja vizitati.
- Scoatem din coada nodul 9. Singurul sau vecin este 2, care a fost deja vizitat. In acest moment, in coada mai este doar nodul 7.
- Scoatem din coada nodul 7. Ii punem in coada vecinii. Coada este: $5, 10$.
- Scoatem din coada nodul 5. Ii adaugam in coada singurul vecin nevizitat, adica 8.
- Scoatem din coada nodul 10. Nu are vecini care sa nu fi fost vizitati.
- Scoatem din coada nodul 8, care nu mai are vecini nevizitati. Acesta a fost ultimul, deci ne oprim.

Arborele asociat parcurgerii in latime $bf(4)$ este:



Ordinea parcurgerii $bf(4)$: 4,2,3,6,1,9,7,5,10,8

Problema 3

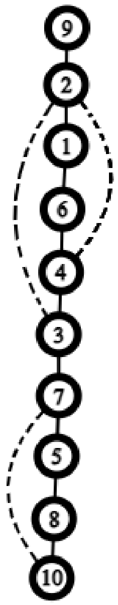
O muchie a unui graf conex se numeste critica daca prin eliminarea ei graful nu mai este conex. Descrieti pe scurt un algoritm eficient de determinare a muchiilor critice din graf si exemplificati (cu explicatii).

Rezolvare:

Pentru determinarea muchiilor critice, vom folosi o varianta modificata a algoritmului de parcurgere in adancime. In plus, vom avea nevoie de doi vectori, nivel si niv_minim. De asemenea, avem nevoie de un vector tata pentru a retine nodul parinte al fiecarui nod.

Vom parcurge graful in adancime. In graf sunt doua tipuri de muchii: inainte si inapoi. Muchiile inainte sunt muchiile arborelui. La fiecare nod, daca nu a fost deja vizitat, vom aduna 1 la nivelul parintelui sau. In cazul in care avem muchiile intre noduri care au fost deja vizitate, aceste muchii sunt inapoi si ne ajuta sa calculam nivelul minim de care este legat un nod. Muchiile inapoi inchid de fapt cicluri. Astfel, daca exista o muchie inapoi dintr-un nod sau dintr-un descendent al nodului care il uneste de un stramos al lui, atunci inseamna ca muchia dintre nodul curent si parintele sau nu este critica (se afla amandoi intr-un ciclu).

Fiind un algoritm bazat pe recursivitate, dupa ce nu mai putem inainta in graf, ne vom intoarce in nodurile prin care am trecut pe acel drum. La fiecare, vom verifica daca niv_min al copilului din care am venit este mai mic decat nivelul nodului cu curent. Daca da, inseamna ca muchia dintre ele nu este critica. Vom actualiza apoi niv_min al nodului curent pentru a fi minimul dintre niv_min al sau si niv_min al copilului.



- Pasul 1: Incepem din nodul 9, acesta va avea $nivel[9] = niv\_min[9] = 1$. Acum gasim un vecin al lui. Acesta este nodul 2, pentru care punem $nivel[2] = niv\_min[2] = 2$

- Pasul 2: Nodul curent este nodul 2. Ii verificam vecinii, iar primul in ordine lexicografica este nodul 1, care va avea $nivel[1] = niv\_min[1] = 3$
- Pasul 3: Nodul curent este 6 cu $nivel[6] = niv\_min[6] = 4$.
- Pasul 4: Nodul curent este 4 cu $nivel[4] = niv\_min[4] = 5$. Acum ii verificam vecinii. Primul este 2. Muchia $(4, 2)$ este muchie inapoi, deci $niv\_min[4] = 2$. Deoarece 2 era deja vizitat, vom cauta urmatorul vecin.
- Pasul 5: Nodul curent este 3 cu $nivel[3] = niv\_min[3] = 6$. In ordine lexicografica, ii gasim vecinul 2. Deci $niv\_min[3] = 2$. Aceasta muchie inapoi inchide si ea un ciclu.
- Pasul 6: Nodul curent este 7 cu $nivel[7] = niv\_min[7] = 7$.
- Pasul 7: Nodul curent este 5 cu $nivel[5] = niv\_min[5] = 8$.
- Pasul 8: Nodul curent este 8 cu $nivel[8] = niv\_min[8] = 9$.
- Pasul 9: Nodul curent este 10 cu $nivel[10] = niv\_min[10] = 10$. Verificam vecinii nodului 10. Il gasim pe 7, deci $niv\_min[10] = 7$.
- Deoarece nu mai are alti vecini spre care sa inainteze, se intoarce. Comparam si obtinem $nivel[8] > niv\_min[10]$, deci muchia nu este critica, iar $niv\_min[8] = 7$ (minimul dintre ele).
- Ne intoarcem spre 5, facand acelasi lucru, observam ca nici aceasta muchie nu este critica si actualizam $niv\_min[5]$.
- La fel pentru muchia $(5, 7)$.
- Ne intoarcem spre nodul 3. Observam ca $niv\_min[7] > nivel[3]$ $(7 > 2)$, ceea ce inseamna ca muchia $(3, 7)$ este critica.

- Ne intoarcem spre nodul 4. Observam ca muchia nu este critica, iar $niv\_min[4]$ ramane 2.
- Pentru nodurile $6, 1$ si 2 muchiile pe care le-am parcurs cu *df* nu sunt critice, iar pentru fiecare *niv_min* va fi transformat in 2.
- Ne intoarcem de la nodul 2 la nodul 9: $niv\_min[2] > nivel[9]$ $(2 > 1)$, ceea ce inseamna ca muchia este critica.

In concluzie, graful are muchiile critice $(3, 7)$ si $(2, 9)$.

Problema 4

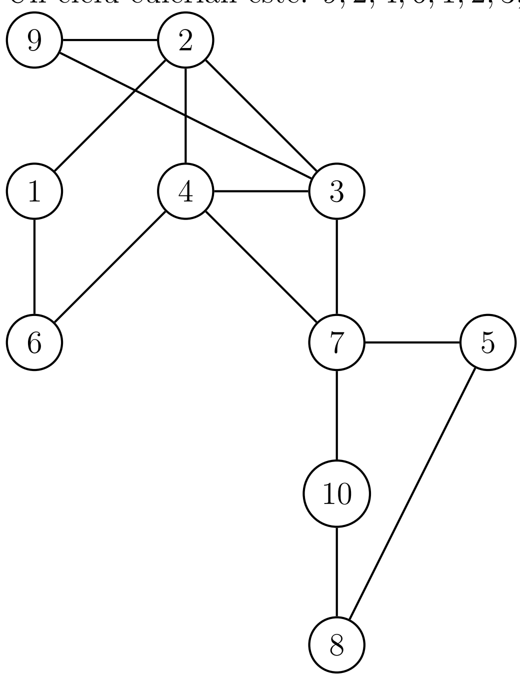
Este graful eulerian? Daca nu adaugati un numar minim de muchii astfel incat graful format sa fie eulerian (fara a avea bucle sau arce multiple), descriind si strategia dupa care ati adaugat arcele. Indicati un circuit (ciclu) eulerian in graful obtinut. Enuntati o conditie necesra si suficienta ca un graf orientat sa fie eulerian.

Rezolvare:

Conditie: Fie un graf neorientat $G = (V, E)$. G este eulerian (are un ciclu eulerian) $\iff$ toate nodurile au grad par.

In acest moment, singurele noduri care au grad impar sunt: $3, 4, 7$ si 9. Asadar, pentru a face graful sa fie eulerian, trebuie sa facem gradele acestor noduri sa fie pare. Din fiecare va trebui sa porneasca o noua muchie. Fiind 4 noduri, putem avea doar doua muchii si gradele vor fi pare. Asadar, vom adauga muchiile $(3, 9)$ si $(4, 7)$. Acum toate nodurile au grad par, ceea ce inseamna ca noul graf este eulerian.

Un ciclu eulerian este: $9, 2, 4, 6, 1, 2, 3, 7, 5, 8, 10, 7, 4, 3, 9$.



Problema 5

Exemplificati (cu explicatii) pasii algoritmului lui Dijsktra pentru a calcula distanta de la varful 2 la varful 5.

Rezolvare:

Incepem prin adaugarea nodului 2 in coada. Cu ajutorul algoritmului lui Dijkstra, vom calcula de fapt cea mai mica distanta de la 2 la fiecare nod. Asadar, vom adauga 2 in coada initial. Apoi, la fiecare pas, vom alege nodul cu eticheta d minima (distanta minima de la nodul de start pana la acel nod pana in acel moment). Avand in vedere ca oricare nod pe care l-am putea selecta va avea distanta deja mai mare, inseamna ca nu vom putea obtine o distanta mai mica pana la nod trecand prin altul. Dupa ce am selectat urmatorul nod curent, vom verifica pentru fiecare nod daca suma dintre distanta pana la nodul curent si ponderea de pe muchia dintre nodul curent si vecinul verificat este mai mica decat vechia distanta pana la acel vecin. Daca da, se actualizeaza distanta si nodul curent este pus in vectorul tata la indicele vecinului.

d/tata	1	2	3	4	5	6	7	8	9	10
d/ tata	$\infty/0$	0/0	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$
Selectam 2	4/2	-	14/2	5/2	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	10/2	$\infty/0$
Selectam 1	-	-	14/2	5/2	$\infty/0$	7/1	$\infty/0$	$\infty/0$	10/2	$\infty/0$
Selectam 4	-	-	12/4	-	$\infty/0$	7/1	$\infty/0$	$\infty/0$	10/2	$\infty/0$
Selectam 6	-	-	12/4	-	$\infty/0$	-	$\infty/0$	$\infty/0$	10/2	$\infty/0$
Selectam 9	-	-	12/4	-	$\infty/0$	-	$\infty/0$	$\infty/0$	-	$\infty/0$
Selectam 3	-	-	-	-	$\infty/0$	-	13/3	$\infty/0$	-	$\infty/0$
Selectam 7	-	-	-	-	16/7	-	-	$\infty/0$	-	17/7
Selectam 5	-	-	-	-	-	-	-	17/5	-	17/7
Selectam 8	-	-	-	-	-	-	-	-	-	17/7
Selectam 10	-	-	-	-	-	-	-	-	-	-

Vectorii d/tata final arata asa:
$d/tata = [4/2, 0/0, 12/4, 5/2, 16/7, 7/1, 13/3, 17/5, 10/2, 17/7]$
De fapt, in momentul in care am selectat nodul 5 ne putem opri, deoarece, odata selectat un nod, distanta pana la el nu se va mai modifica. Asadar, distanta intre nodurile 2 si 5 va fi 16.

Problema 6

Exemplificati (cu explicatii) pasii algoritmului lui Prim pornind din varful 2.

Rezolvare:

d/tata	1	2	3	4	5	6	7	8	9	10
d/tata	$\infty/0$	0/0	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$
Selectam 2	4/2	–	14/2	5/2	$\infty/0$	$\infty/0$	$\infty/0$	$\infty/0$	10/2	$\infty/0$
Selectam 1	-	-	14/2	5/2	$\infty/0$	3/1	$\infty/0$	$\infty/0$	10/2	$\infty/0$
Selectam 6	-	-	14/2	4/6	$\infty/0$	-	$\infty/0$	$\infty/0$	10/2	$\infty/0$
Selectam 4	-	-	7/4	-	$\infty/0$	-	$\infty/0$	$\infty/0$	10/2	$\infty/0$
Selectam 3	-	-	-	-	$\infty/0$	-	1/3	$\infty/0$	10/2	$\infty/0$
Selectam 7	-	-	-	-	3/7	-	-	$\infty/0$	10/2	4/7
Selectam 5	-	-	-	-	-	-	-	1/5	10/2	4/7
Selectam 8	-	-	-	-	-	-	-	-	10/2	2/8
Selectam 10	-	-	-	-	-	-	-	-	10/2	-
Selectam 9	-	-	-	-	-	-	-	-	-	-

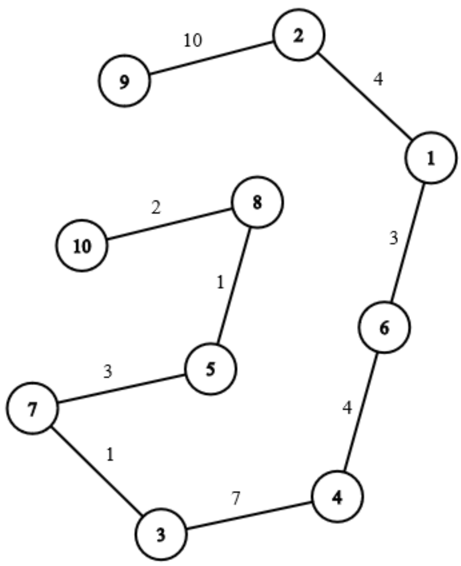
Vectorii d/tata final arata asa:
$d/tata = [4/2, 0/0, 7/4, 4/6, 3/7, 3/1, 1/3, 1/5, 10/2, 2/8]$

Explicatii:

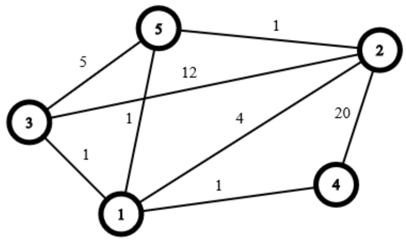
Vectoruii d si tata vor fi initializati cu infinit, respectiv 0 pentru nodul de pornire. In vectorul d retinem: d[nod] = distanta minima de la nod la un nod din APM-ul curent.La fiecare pas se extrage nodul cu eticheta cea mai mica si care nu a mai fost extras pana atunci, incepand cu nodul 2. Apoi, pentru fiecare muchie a nodului, vectorul d si vectorul tata se vor modifica doar daca d[nodul-vecin] este mai mare decat distanta de la nodul extras la nodul-vecin. In vectorul d se va pune noua distanta, iar in vectorul tata nodul parinte a nodului-vecin. O data ce un nod a fost extras, in dreptul lui va aparea simbolul $-$, marcand ca nu se mai poate selecta inca o data, valorile lui din vectori ramanad acelasi pana la finalul algoritmului. Algoritmul se opreste cand s-au vizitat toate nodurile.

Un rezumat ar includerii nodurilor in APM ar fi: am inclus pe rand: 2,1,6,4,3,7,5,8,10,9
Costul este: 35

APM-ul va fi:



Problema 7



Graf pentru problema 7:

Fie W matricea costurilor si n numarul de varfuri. Ce valoare va afisa secventa de cod? Justificati.

```
D=W
t=3

pentru k -> 1, t executa
    pentru i -> 1, n executa
        pentru j -> 1, n executa
            daca D[i][j] > D[i][k] + D[k][j] atunci
                D[i][j] = D[i][k] + D[k][j]
scrie D[2][4]
```

Rezolvare:

Codul de mai sus este o varianta a algoritmului Roy-Warshall pentru calcularea distantelor minime intre oricare noduri. Valorile pe care le ia k reprezinta, de fapt, nodurile folosite ca noduri intermediare pe drumurile dintre doua noduri. Secventa de cod va calcula distantele minime dintre oricare doua noduri, pe drumurile care trec doar prin nodurile $1, 2$ si 3, iar la final va afisa aceasta distanta intre 2 si 4.

La inceput, matricea D este initializata cu matricea W a costurilor muchiilor. Initial, nu avem noduri intermediare pe drumuri, deci se poate ajunge doar daca exista muchie intre cele doua noduri, cu costul asociat acesteia. Apoi, la fiecare iteratie de la 1 la 3, se verifica daca distanta de la i la j trecand prin nodul k este mai mica decat distanta calculata anterior. In caz afirmativ, se actualizeaza distanta.

Initializare: D = W =

	1	**2**	**3**	**4**	**5**
1	0	4	1	1	1
2	4	0	12	20	1
3	1	12	0	∞	5
4	1	20	∞	0	∞
5	1	1	5	∞	0

La pasul $k = 1$, adica folosind nodul 1 ca nod intermediar, matricea distantelor se modifica astfel:

W =

	1	**2**	**3**	**4**	**5**
1	0	4	1	1	1
2	4	0	**5**	**5**	1
3	1	**5**	0	**2**	**2**
4	1	**5**	**2**	0	**2**
5	1	1	**2**	**2**	0

La pasul $k = 2$, adica folosind nodurile 1 si 2 ca noduri intermediare, matricea distantelor se modifica astfel:

W =

	1	**2**	**3**	**4**	**5**
1	0	4	1	1	1
2	4	0	5	5	1
3	1	5	0	2	2
4	1	5	2	0	2
5	1	1	2	2	0

La pasul $k = 3$, adica folosind nodurile $1, 2$ si 3 ca noduri intermediare, matricea distantelor se modifica astfel:

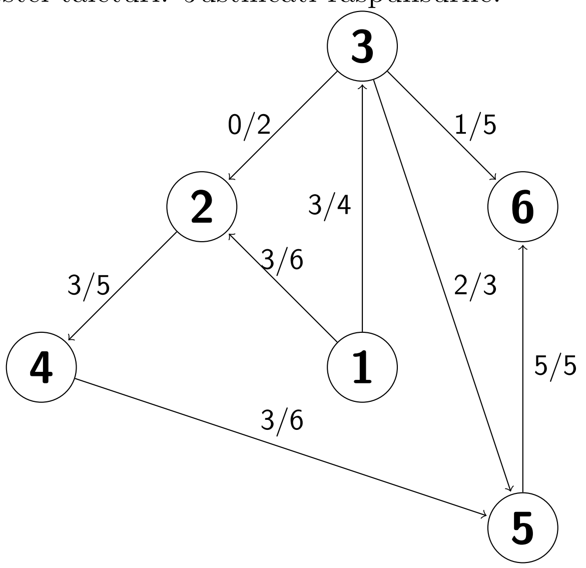
W =

	1	**2**	**3**	**4**	**5**
1	0	4	1	1	1
2	4	0	5	5	1
3	1	5	0	2	2
4	1	5	2	0	2
5	1	1	2	2	0

Asadar, D[2][4] = 5.

Problema 8

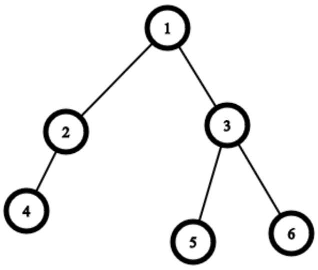
Sursa este varful $s = 1$, iar destinatia $t = 6$. Ilustrati pasii algoritmului Ford-Fulkerson pentru aceasta retea pornind de la fluxul indicat si alegand la fiecare pas un s-t lant f-nesaturat de lungime minima (algoritm Edmonds-Karp). Indicati o taietura (s-t taietura) minima in retea (se vor indica varfurile din bipartitie, arcele directe, arcele inverse) si determinati capacitatea acestei taieturi. Justificati raspunsurile.



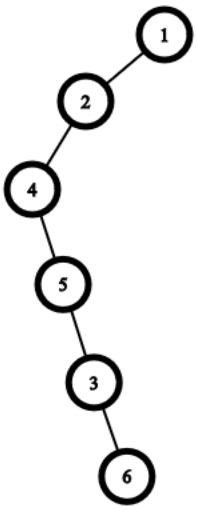
Rezolvare:

Se va face repetat BFS-uri din nodul 1 (sursa) la nodul 6 (destinatie). Daca muchia este saturat nu mai putem merge pe ea (se vor include si muchiile de intoarcere).

Primul BFS:
Q: 1,2,3,4,5,6 → STOP am ajuns la destinatie
lant: 1→3→6
minimul care mai poate fi bagat pe muchii va fi 1 (ne uitam la muchiile (1,3) si (3,6))



Al doilea BFS:
Q: 1,2,4,5,3,6 → STOP am ajuns la destinatie (cu 5-3 muchie de intoarcere)
lant: 1→2→4→5→3→6
minimul care mai poate fi bagat pe muchii va fi 2 (ne uitam la muchiile (1,2),(2,4),(4,5) la muchia de intoarcere 5-3 si vedem ca maximul care mai poate fi bagat este 2)

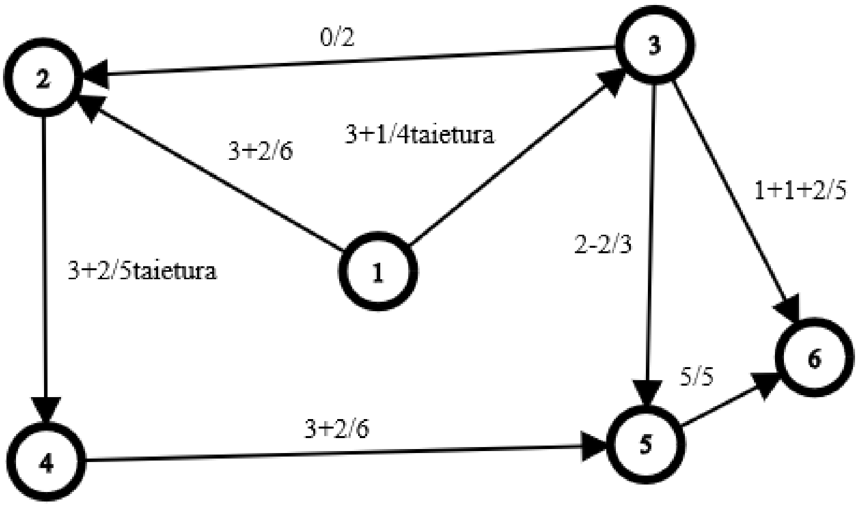


Al treilea BFS:
Q: 1,2 → nu putem ajunge la destinatie, deci algoritmul se opreste, fluxul este maxim
Fluxul maxim este: 4+5=9 (arcele care intra in 6).
Taietura minima se afla prin bipartitionarea grafului: din 1 se poate ajunge la 2, deci A :

$\{1,2\}$ si $B = \{3,4,5,6\}$

Taietura minima e (1,3) si (2,4) = 4+5 =9, deci fluxul este maxim.

Arce inverse: (3,2).

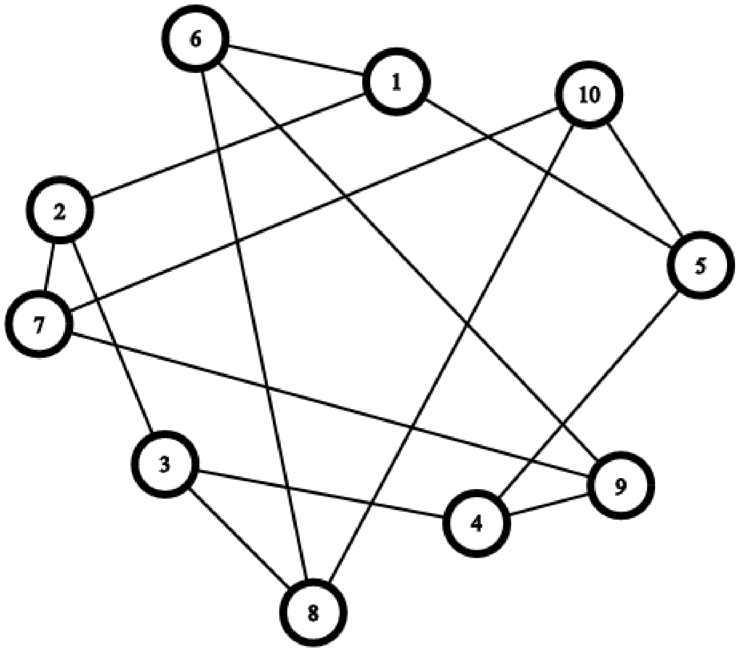


Problema 9

a Fie G un graf planar conex cu $n > 3$ noduri si m muchii cu propietatea ca lungimea minima a unui ciclu din G este 5. Aratati ca $3 \cdot m + 10 \leq 5 \cdot n$.

b Aratati ca graful din figura nu este planar.

c Construiti un graf cu minim 6 varfuri care verifica proprietatile de la a) pentru care are loc chiar inegalitatea $3 \cdot m + 10 = 5 \cdot n$.



Graful este:

Rezolvare:

a) $3m + 10 \leq 5n \iff 3m + 10 - 5n \leq 0 \iff 5(m - n + 2) \leq 2m \iff 5|F| \leq 2m$

Fiecare muchie este numarata de doua ori atunci cand facem suma gradelor fetelor dintr-o harta a unui graf planar, deci:

$2m = \sum d_m(f)$, iar $d_m(F) \geq 5$ (adica fiecare ciclu are lungimea minim 5)

$\implies 5|F| \leq 2m$

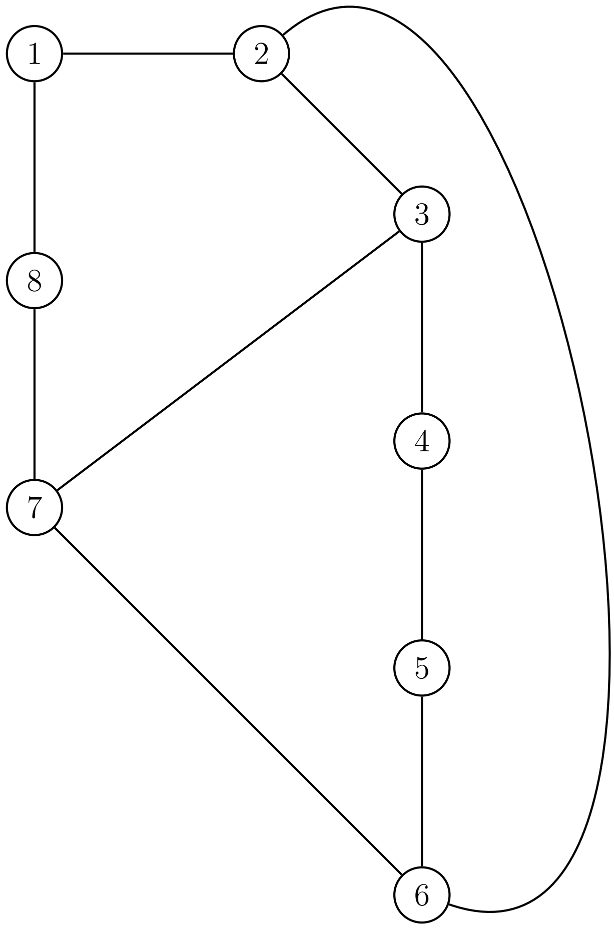
b)

Presupunem prin reducere la absurd ca graful de mai sus este planar. De asemenea, se observa ca lungimea oricarui ciclu al acestui graf este minim 5. Deci, conform acestor lucruri, ar trebui sa indeplineasca proprietatea de la subpunctul anterior, adica $3m + 10 \leq 5n$.

Calculand, obtinem ca $n = 10$ si $m = 15$. Ar trebui ca $55 = 3m + 10 \leq 50$, ceea ce nu este adevarat $\implies$ presupunerea facuta este falsa, deci graful nu este planar.

c) Graful trebuie sa aiba $n \geq 6$, sa fie planar conex, sa aiba lungimea minima a unui ciclu egala cu 5 si $3m + 10 = 5n$.

Vom alege $n = 8 \geq 6 \implies m = (40 - 10) : 3 = 10$



Problema 10

Descrieti pe scurt algoritmul de determinare a distantei de editare intre doua cuvinte si explicati relatiile de recurenta pentru calculul acestei distante. Ilustrati algoritmul pentru cuvintele "antrenat" si "entorsa", scriind matricea cu valorile subproblemelor si explicand cum au fost acestea calculate.

Rezolvare:

Pentru a determina distanta de editare intre doua cuvinte (distanta Levenshtein) se iau in calcul urmatoarele situatii pentru fiecare combinatie de litere din cele doua cuvinte:

- Putem scoate o litera (distanta creste cu 1)
- Putem adauga o litera (distanta creste cu 1)
- Putem schimba o litera (distanta creste cu 1)
- Daca cele doua litere sunt egale putem sa nu schimbam nimic (distanta nu creste)

Fie $dist[i][j]$ distanta de editare intre prefixul de lungime i al primului cuvant si cel de lungime j din al doilea. Evident, $dist[i][0] = dist[0][i] = i$, deoarece se vor face i adaugari in cuvantul nenul. Putem realiza urmatoarea recurenta pentru programare dinamica, unde $i, j \neq 0$:

$$dist[i][j] = min \begin{cases} dist[i-1][j] + 1 \\ dist[i][j-1] + 1 \\ dist[i-1][j-1] + (a[i]! = b[j]) \end{cases}$$

Calculam distanta Levenshtein intre *antrenat* si *entorsa*:

		a	n	t	r	e	n	a	t
	0	1	2	3	4	5	6	7	8
e	1	1	2	3	4	4	5	6	7
n	2	2	1	2	3	4	4	5	6
t	3	3	2	1	2	3	4	5	5
o	4	4	3	2	2	3	4	5	6
r	5	5	4	3	2	3	4	5	6
s	6	6	5	4	3	3	4	5	6
a	7	6	6	5	4	4	4	4	5

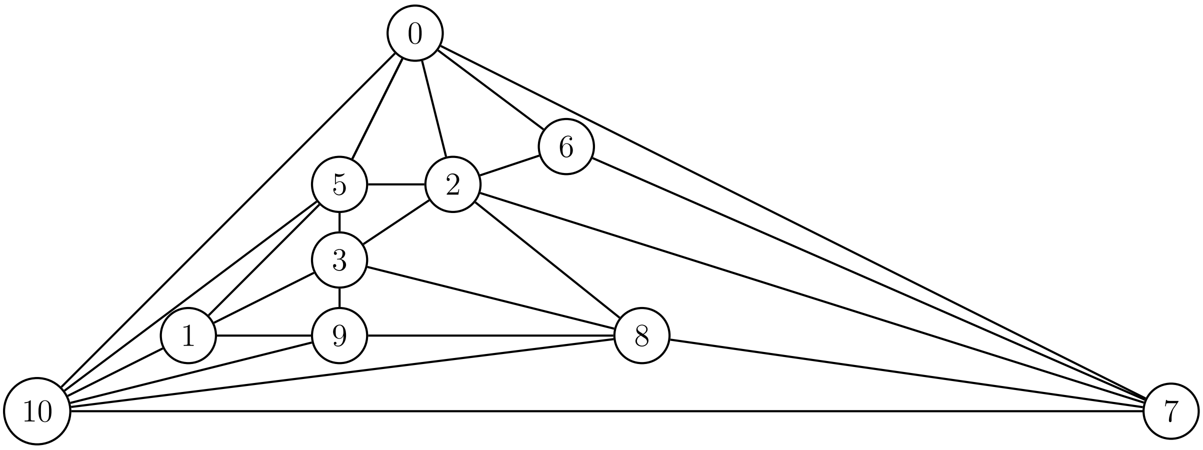
Astfel, distanta de editare dintre *antrenat* si *entorsa* este 5.

Problema 11

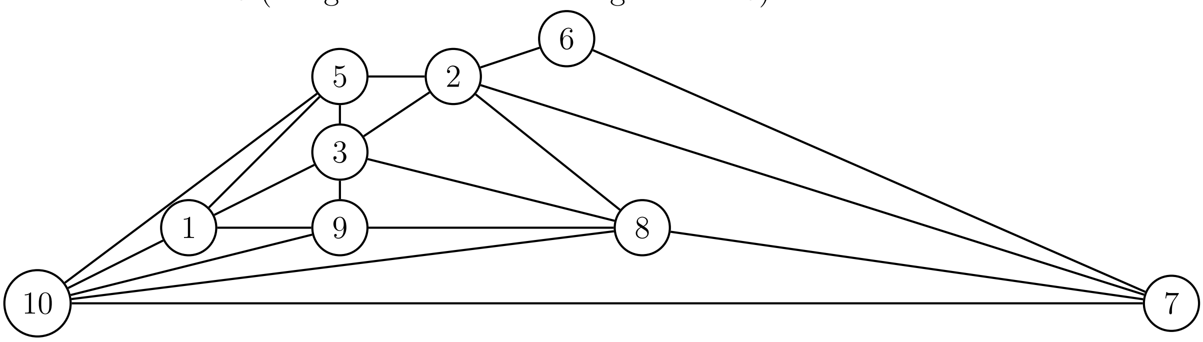
Descrieti algoritmul de 6-colorare a varfurilor unui graf neorientat conex planar si exemplificati acest algoritm.

Rezolvare: Algoritmul de 6-colorare este utilizat pentru a colora nodurile unui graf cu 6 culori astfel incat oricare doua noduri vecine vor avea culori diferite. Daca graful are cel mult 6 noduri, atunci asignam fiecarui nod cate o culoare. Altfel, eliminam, pe rand, cate un nod care are gradul mai mic sau egal decat 5, colorand graful ramas (din care am eliminat acest nod) si atribuindu-i nodului o culoare diferita de cea a vecinilor sai.

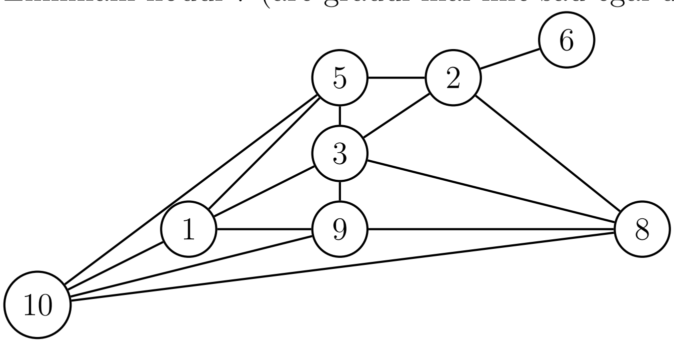
```
colorare(G)
    daca |V(G)| <= 6 atunci
        coloreaza varfurile cu culori distincte din {1,...,6}
    altfel
        alege x cu d(x) <= 5
        colorare(G-x)
        colorează x cu o culoare din {1,...,6}
                diferită de culorile vecinilor
```



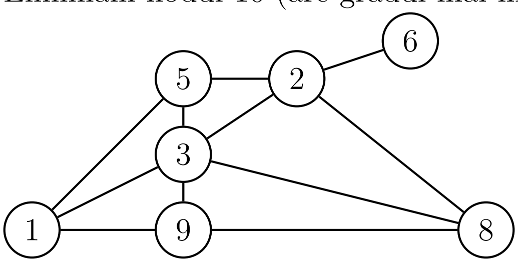
Eliminam nodul 0 (are gradul mai mic sau egal decat 5).



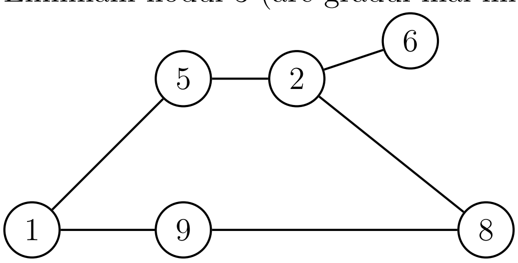
Eliminam nodul 7 (are gradul mai mic sau egal decat 5).



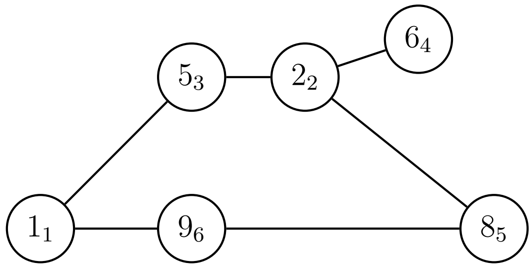
Eliminam nodul 10 (are gradul mai mic sau egal decat 5).



Eliminam nodul 3 (are gradul mai mic sau egal decat 5).

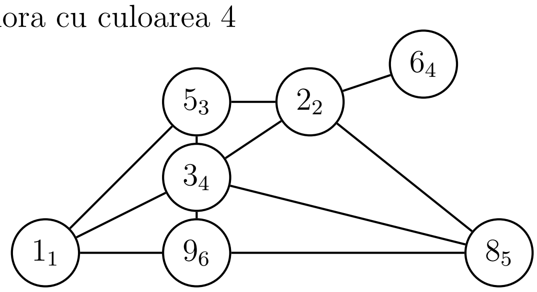


Am ramas cu 6 noduri, motiv pentru care vom aloca fiecarui nod o culoare din cele 6.

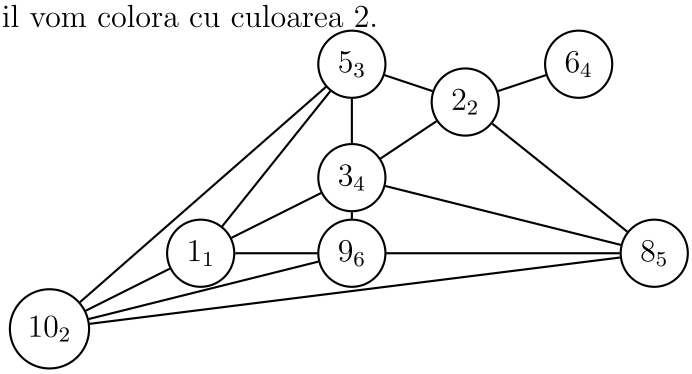


Ne intoarcem in recurenta si adaugam culori nodurilor eliminate la fiecare pas astfel incat sa aiba culori diferite de vecinii lor.

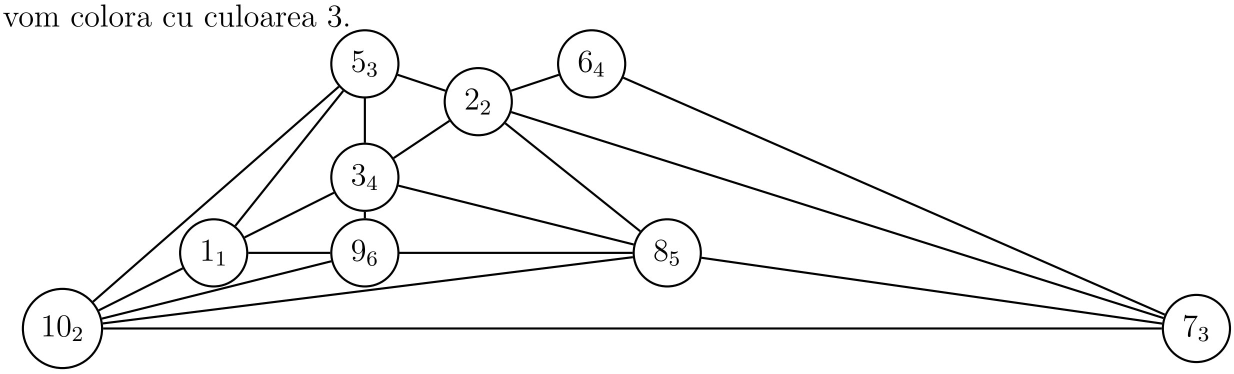
Trebuie sa coloram nodul 3 cu o culoare diferita de cea a nodurilor $1, 5, 2, 8$, asa ca il vom colora cu culoarea 4



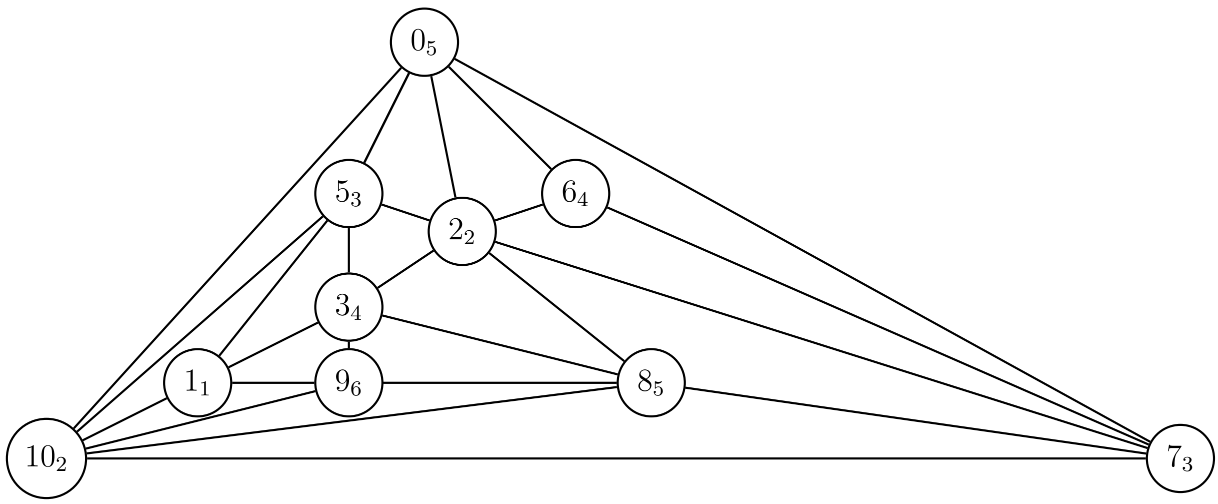
Trebuie sa coloram nodul 10 cu o culoare diferita de cele pe care le au nodurile $1, 5, 9, 8$, asa ca il vom colora cu culoarea 2.



Trebuie sa coloram nodul 7 cu o culoare diferita de a nodurilor $2, 6, 8, 10$, motiv pentru care il vom colora cu culoarea 3.



Trebuie sa coloram nodul 0 cu o culoare diferita de cea a nodurilor $2, 5, 6, 7, 10$. Asa ca il vom colora cu culoarea 5 si vom obtine graful dupa algoritmul de 6-colorare.



Problema 12

In 2024, Eliza doreste sa fie un cetatean exemplar si se hotaraste sa devina observator independent mobil pentru alegeri. Ea obtine o harta a drumurilor bidirectionale dintre sectiile de votare, impreuna cu costurilor asociate acestora. Pentru a verifica ca alegerile se desfasoara corect, se decide sa viziteze exact K statii (oricare), fiecare exact o data! Ajutati-o sa gaseasca cel mai scurt drum care porneste din sectia 1 si viziteaza exact K sectii. (Exista drum doar intre anumite statii, iar Eliza nu poate trece pe langa o sectie fara sa o viziteze).

Rezolvare:
Vom utiliza un algoritm asemanator celui pentru determinarea ciclului Hamiltonian de cost minim. Vom avea o matrice de costuri avand urmatoarea semnificatie: $cost[nod][mask]$ = costul de a ajunge din nodul 1 la nodul nod trecand exact o data prin bitii marcati cu 1 in masca de biti $mask$.

Vom realiza o parcurgere similara algoritmului lui dijkstra:

- Pentru fiecare nod parcurs vom verifica daca venind dintr-un vecin cu o anumita masca de biti, am reusit sa il parcurgem anterior cu acea masca de biti dar cu un cost mai mic. In cazul in care am gasit un cost mai bun pentru combinatia *nod*, *masca*, vom actualiza costul si vom introduce in coada de prioritati tuplul $cost, nod, masca$.

Vom optimiza evitand sa depasim K noduri vizitate in masca. Vom verifica mereu daca bit-ul aferent nodului pe care ne dorim sa il parcurgem a fost deja setat in masca sau nu, pentru a nu trece de mai multe ori prin acelasi nod. Vom actualiza din mers soulutia cea mai buna, de fiecare data cand avem o massca cu exact K noduri;

Complexitate: $O\left(N^2 \cdot S \cdot \log\left(M\right)\right)$

Unde S este numarul submultimilor cu maxim K elemente ale multimii nodurilor.

$$S = C_N^1 + C_N^1 + \cdots + C_N^K$$

S provine din totalitatea mastilor de biti care vor fi parcurse, generandu-se astfel toate submultimile de maxim K noduri. $N^2\log(M)$ este complexitatea clasica a algoritmului de drum Hamiltonian (logaritmul provine de la min-heap), iar ce doi N de la parcurgerea nodurilor.