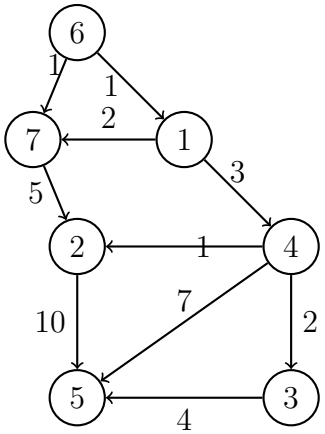
Examen Algoritmi Fundamentali - Varianta 1

Daria Pirvulescu

January 2024



Graful pentru exercitiile 1-6

Problema 1

Definiti notiunea de sortare topologica. Care dintre urmatoarele secvente reprezinta o sortare topologica pentru acest graf? (pot fi mai multe)

a) $1, 4, 7, 2, 3, 5$

b) $6, 1, 4, 7, 2, 3, 5$

c) $6, 7, 1, 2, 4, 3, 5$

d) $6, 1, 7, 4, 3, 2, 5$

Rezolvare:

Fie $G = (V, E)$ un graf orientat. Se numeste sortare topologica o ordonare a varfurilor astfel incat daca exista un arc $u \to v$, u se va afla inainte de v.

Sortarea topologica trebuie sa inceapa de la varfurile cu gradul interior 0. Singurul astfel de varf este 6. Apoi, la fiecare pas, odata adaugat un varf in coada, gradul interior al tuturor varfurilor spre care are arc va scadea cu 1.

Dupa ce l-am pus pe 6 in coada, gradul interior al varfului 1 va deveni 0, deci acesta va fi pus urmatorul.

Dupa ce am adaugat varful 1, gradul interior al varfurilor 4 si 7 a devenit 0. In acest moment, l-am putea adauga pe oricare dintre ele in coada (putem avea $6, 1, 4, 7$ sau $6, 1, 7, 4$). Dupa adaugarea lor, gradul interior al varfurilor 2 si 3 va scadea cu 1, devenind 0. Din nou, le putem adauga in ce ordine dorim in coada, fara a fi vreo diferenta. Ultimul varf ramas este 5.

Asadar, atat sortarea de la punctul b), cat si cea de la punctul d) sunt corecte.

Problema 2

Exemplificati (cu explicatii) cum functioneaza parcurgerea in adancime $df(6)$, ilustrand si arborele df asociat.

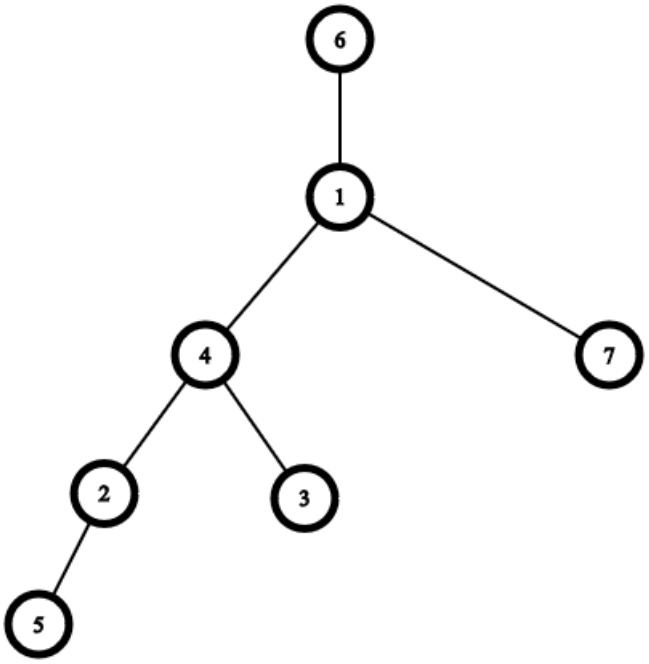
Rezolvare:

Parcurgerea in adancime se face recursiv. Se porneste dintr-un varf, in acest caz 6. Se verifica, vom presupune ca in ordine lexicografica, vecinii varfului curent. In acest caz, se va verifica varful 1. Se repeta procedeul, ceea ce inseamna ca vom avea intai un lant care se va duce in adancime pana cand nu mai exista muchii pe care sa inainteze. Cand se ajunge in acest punct, se intoarce la nodul anterior in recursivitate si se verifica urmatorul vecin.

Ordinea parcurgerii:

$6, 1, 4, 2, 5, 3, 7$

Arborele $df(6)$:



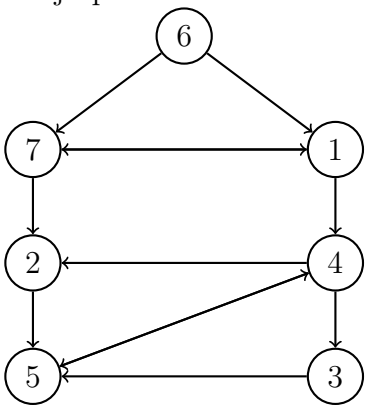
Dupa ce se ajunge la varful 5, nu mai avem muchii pe care sa continuam, asa ca ne intoarcem la varful 2. Nici de aici nu mai avem muchie, deci ne intoarcem la 4. De aici, putem continua spre varful 3. Nu putem continua catre un varf care nu a fost deja vizitat, asa ca ne mai intoarcem un nivel, ajungand inapoi la 4, iar ulterior inapoi la 1. De aici avem muchie spre 7. Ne intoarcem la 1, apoi la 6. Nu mai exista niciun varf nevizitat, deci s-a terminat parcurgerea in adancime.

Problema 3

Adaugati cel mult 3 arce in graf astfel incat graful obtinut sa aiba exact 3 componente tare conexe. Exemplificati (cu explicatii) cum functioneaza algoritmul lui Kosaraju de determinare a componentelor tare conexe pe graful nou obtinut.

Rezolvare:

Vom adauga arcele $(7,1)$, $(5,4)$. Componentele tare conexe vor fi: 6, apoi $1,7$, apoi $2,3,4,5$. In continuare, vom parcurge pasii algoritmului lui Kosaraju pentru determinarea componentelor tare conexe.

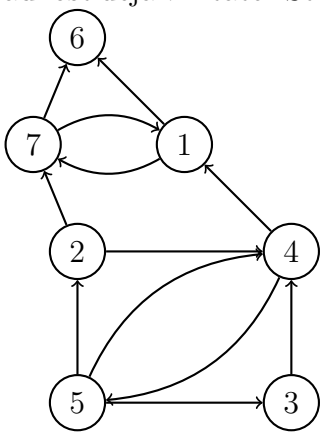


Pasii algoritmului lui Kosaraju:

- Se realizeaza un DFS pe graf si pentru fiecare nod se calculeaza timpul de finalizare al acestuia.
- Se determina graful transpus.
- Se realizeaza un DFS in ordinea inversa a timpilor de finalizare. (vom utiliza o stiva pe care o vom completa cu nodurile in ordinea descrescatoare)

Facand un *DFS* din nodul 1. Parcurgem nodurile $4,2,5,3,7$ Stiva devine: $5,2,3,4,7,1$.

Pornim un DFS din nodul 6 (nu a fost vizitat). Singurul nod din aceasta parcurgere este 6, intrucat $1,7$ au fost deja vizitate. Stiva devine. $5,2,3,4,7,1,6$.



Realizam graful transpus:
Incepem din varful stivei:

- Scoatem nodul 6. Nu putem ajunge nicaieri, deci este singur in CTC.

- Scoatem nodul 1. Putem ajunge doar in 6, care a fost deja verificat. Mai putem ajunge in 7 care nu a mai fost vizitat. Din 7 mai putem ajunge in $6, 1$ care au fost deja vizitate. Deci avem componenta tare conexa $\{1, 7\}$.
- Scoatem nodul 7. A fost deja vizitat, deci nu continuam DFS-ul din el.
- Scoatem nodul 4. Putem ajunge in nodul 5, pe care il vizitam (nu a fost vizitat anterior). Din 5 putem ajunge in $2, 3$ care nu au fost vizitate anterior. Nici din 2, nici din 3 nu putem ajunge in alte noduri nevizitate, deci avem componenta $\{4, 2, 5, 3\}$.
- Scoatem pe rand fiecare nod ramas in stiva si observam ca au fost deja vizitate.

In concluzie, componentele tare conexe ale grafului modificat sunt $\{6\}$, $\{1, 7\}$, $\{4, 2, 5, 3\}$

Problema 4

Este graful eulerian? Daca nu adaugati un numar minim de arce 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:

O conditie necesara si suficienta este **teorema lui Euler**: *Un graf este eulerian daca si numai daca toate nodurile sale au gradul interior egal cu cel exterior..*

Determinam gradul nodurilor:

nod	d^+ (nod)	d^- (nod)
1	1	2
2	2	1
3	1	1
4	1	3
5	3	0
6	0	2
7	2	1

Nodul 3 satisface aceasta conditie, asa ca si pe acesta il vom ignora in adaugarea arcelor. Pentru ca nodul 4 sa aiba gradul interior egal cu cel

exterior trebuie sa mai adaugam 2 arce care intra in nodul 4. Nodurile $5, 7$ mai au nevoie de arce care sa iasa din ele. Vom adauga arcele $(7, 4), (5, 4)$.

nod	d^+ (nod)	d^- (nod)
1	1	2
2	2	1
3	1	1
4	3	3
5	3	1
6	0	2
7	2	2

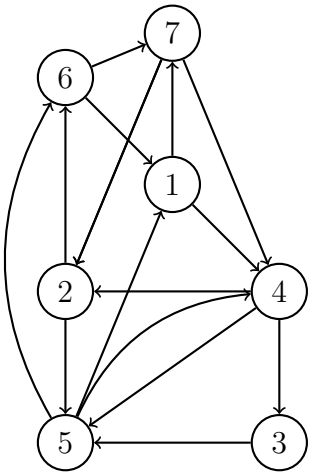
Observam ca arcul $(5, 6)$ ar ajuta ambele noduri si am ajunge sa avem urmatoarele grade:

nod	d^+ (nod)	d^- (nod)
1	1	2
2	2	1
3	1	1
4	3	3
5	3	2
6	1	2
7	2	2

Observam ca varful 1 mai are nevoie de un arc de intrare, iar varful 5 mai are nevoie nevoie de un arc de iesire. In acelasi timp, varful 6 mai are nevoie de un arc de intrare, iar 2 mai are nevoie de un arc de iesire. Astfel construim arcele $(5, 1), (2, 6)$. Astfel, vom avea gradele interioare egale cu cele exterioare:

nod	d^+ (nod)	d^- (nod)
1	2	2
2	2	2
3	1	1
4	3	3
5	3	3
6	2	2
7	2	2

Graful rezultat este:



Ciclul euelerian este: 3,5,4,5,1,7,4,2,5,6,7,2,6,1,4,3

Problema 5

Exemplificati (cu explicatii) pasii algoritmului de determinare a unui drum minim in grafuri aciclice (= graf fara circuite = DAGs) pentru a calcula distanta de la varful 1 la varful 5.

Rezolvare:
Pasii algoritmului sunt:

- Verificam daca graful nu are circuite. In caz contrar algoritmul nu se poate aplica.
- Determinam o sortare topologica a grafului.
- Pastram nodurile din sortare incepand de la nodul *sursa*.
- Initializam cu ∞ vectorul de distante in toate pozitiile, mai putin nodul sursa (unde punem 0).
- Parcurgem varfurile in ordinea data de sortarea topologica.
- Pentru fiecare varf, relaxam arcele spre vecinii sai.

Conform *problemei 1*, o sortare topologica este: $6, 1, 4, 7, 2, 3, 5$
Realizam initializarea.

nod	1	2	3	4	5	6	7
$d[nod]$	0	∞	∞	∞	∞	∞	∞

Parcurgem nodurile in aceasta ordine incepand de la 1 si relaxam arcele spre vecini:

- Nodul 1: vecinii sai sunt $7, 4$ cu costurile $2, 3$. Ambele costuri sunt mai bune decat ∞, deci vom actualiza distanta catre ambele. Vectorul de distante devine:

nod	1	2	3	4	5	6	7
$d[nod]$	0	∞	∞	3	∞	∞	2

- Nodul 4: vecinii sai sunt $2, 3, 5$ cu costurile $1, 2, 7$. Astfel distantele candidat pentru nodurile $2, 3, 5$ sunt $4, 5, 10$. Am obtinut distante mai bune pentru toate. Actualizam vectorul de distante:

nod	1	2	3	4	5	6	7
$d[nod]$	0	4	5	3	10	∞	2

- Nodul 7: singurul sau vecin este 2 cu costul 5. Distanta sa candidat este $2 + 5 = 7$ si este mai mare decat 4 (distanta de pana acum), motiv pentru care nu vom actualiza distantele.

- Nodul 2: singurul sau vecin este 5 cu costul 10 deci distanta candidat $4 + 10 = 14$. Aceasta este mai mare decat 10 (distanta curenta pentru nodul 5), motiv pentru care nu vom actualiza.

- Nodul 3: singurul sau vecin este 5 cu costul 4, deci distanta candidat $5 + 4 = 9$. Actualizam distanta nodului 5:

nod	1	2	3	4	5	6	7
$d[nod]$	0	4	5	3	9	∞	2

- Nodul 5: nu avem arce care sa iasa din nodul 5. Ne putem opri, deoarece trebuia sa aflam distanta pana la nodul 5.

Concluzie: Distanta de la nodul 1 pana la nodul 5 este: 9.

Problema 6

Exemplificati (cu explicatii) pasii algoritmului lui Kruskal.

Rezolvare:

Conform *Algoritmului lui Kruskal* pentru a obtine informatii despre conexitate, se utilizeaza o structura de tip DSU, si se incepe de la graful fara

muchii. Se parcurg muchiile in ordinea crescatoare a costurilor, iar la fiecare pas se va lua muchia pentru APM daca uneste doua componente conexe (se face UNION).

Complexitate: $O(m \log n)$.

Muchiile in ordine descrescatoare dupa costuri sunt:

- (1,6)- cost 1
- (6,7)- cost 1
- (2,4)- cost 1
- (1,7)- cost 2
- (3,4)- cost 2
- (1,4)- cost 3
- (3,5)- cost 4
- (2,7)- cost 5
- (4,5)- cost 7
- (2,5)- cost 10

Se va exemplifica algoritmul cu ajutorul vectorului r. Atunci cand vectorul r va ajunge sa aiba aceleasi numere peste tot, inseamna ca toate nodurile apartin unei comaponente conexe. Prima data vectorul r este initializat cu nodurile grafului.

$r = [1, 2, 3, 4, 5, 6, 7]$

Pentru muchia (1,6) $r(1) \neq r(6)$, deci vectorul devine:

$r = [1, 2, 3, 4, 5, 1, 7]$

Pentru muchia (6,7) $r(6) \neq r(7)$, deci vectorul devine:

$r = [1, 2, 3, 4, 5, 1, 1]$

Pentru muchia (2,4) $r(2) \neq r(4)$, deci vectorul devine:

$r = [1, 2, 3, 2, 5, 1, 1]$

Pentru muchia (1,7) $r(1) = r(7)$, deci vectorul ramane la fel.

Pentru muchia (3,4) $r(3) \neq r(4)$, deci vectorul devine:

$r = [1, 3, 3, 3, 5, 1, 1]$

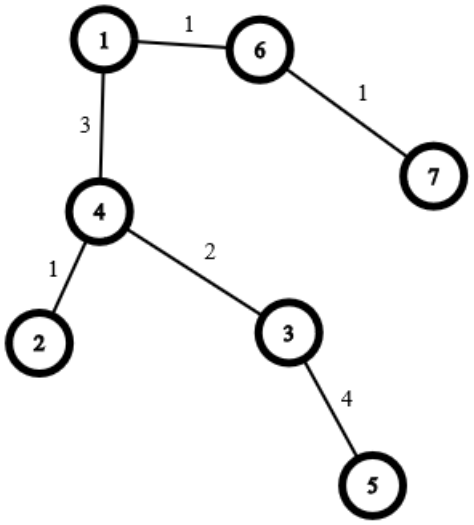
Pentru muchia (1,4) $r(1) \neq r(4)$, deci vectorul devine:

$r = [1, 1, 1, 1, 5, 1, 1]$

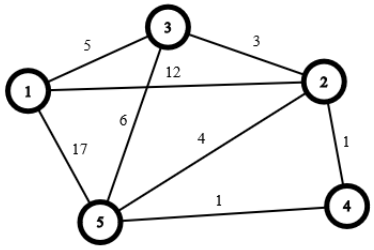
Pentru muchia (3,5) $r(3) \neq r(5)$, deci vectorul devine:

$r = [1, 1, 1, 1, 1, 1, 1]$

STOP → s-a obtinut o componenta conexa, algoritmul se opreste.
Costul este: 1+1+1+2+3+4 = 12



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[1][5]
```

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 1 si 5.

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	12	5	∞	17
2	12	0	3	1	4
3	5	3	0	∞	6
4	∞	1	∞	0	1
5	17	4	6	1	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	12	5	∞	17
2	12	0	3	1	4
3	5	3	0	∞	6
4	∞	1	∞	0	1
5	17	4	6	1	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	12	5	**13**	**16**
2	12	0	3	1	4
3	5	3	0	**4**	6
4	**13**	1	**4**	0	1
5	**16**	4	6	1	0

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

W =

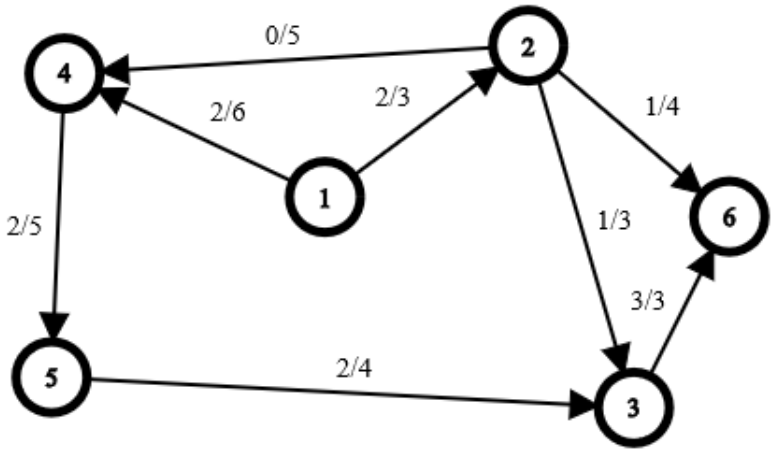
	1	**2**	**3**	**4**	**5**
1	0	**8**	5	**9**	**11**
2	**8**	0	3	1	4
3	5	3	0	4	6
4	**9**	1	4	0	1
5	**11**	4	6	1	0

Asadar, D[1][5] = 11.

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:

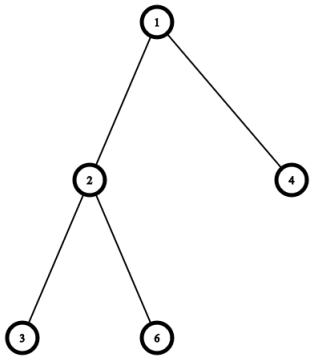
Se va face repetat BFS-uri din nodul 1 (sursa) la nodul 6 (destinatie). Daca muchia este saturat nu mai putem merge pe ea (se vor include si muchiile de intoarcere).

Primul BFS:

Q: 1,2,4,3,6 → STOP am ajuns la destinatie

lant: 1→2→6

minimul care mai poate fi bagat pe muchii va fi 1 (ne uitam la muchiile (1,2) si (2,6) pcel mai mult pe ele se poate baga 1)

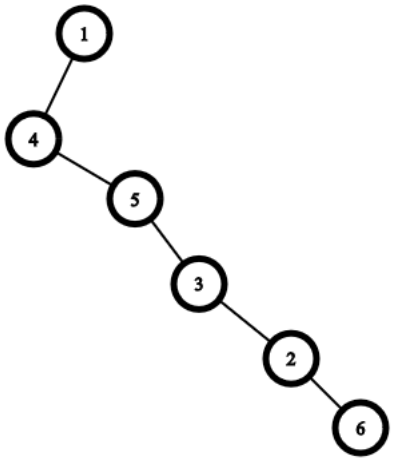


Al doilea BFS:

Q: 1,4,5,3,2,6 → STOP am ajuns la destinatie (cu 3-2 muchie de intoarcere)

lant: 1→4→5→3→2→6

minimul care mai poate fi bagat pe muchii va fi 1 (ne uitam pe muchiile (1,4), (4,5), (5,3) si muchia de intoarcere (3,2))



Al treilea BFS:

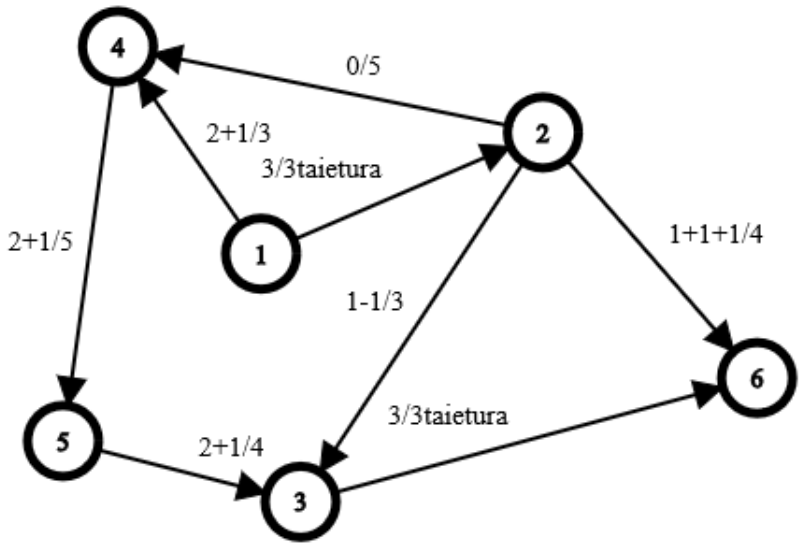
Q: 1,4,3,5 $\rightarrow$ nu putem ajunge la destinatie, deci algoritmul se opreste, fluxul este maxim

Fluxul maxim este: 3+3=6 (arcele care intra in 6).

Taietura minima se afla prin bipartitionarea grafului: din 1 se poate ajunge la 4, 5, 3, deci $A : \{1, 4, 5, 3\}$ si $B = \{2, 6\}$

Taietura minima e (1,2) si (3,6) = 3+3 =6, deci fluxul este maxim.

Arce inverse: (2,4).

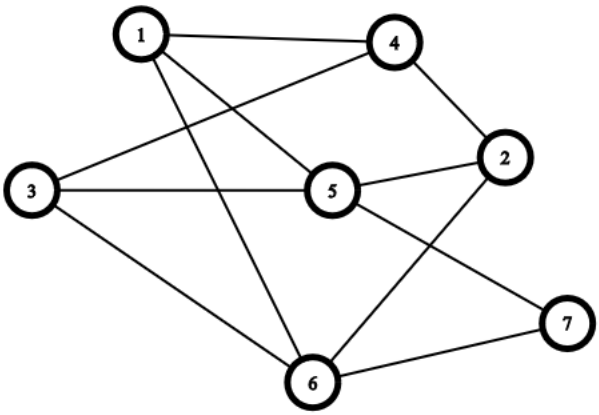


Problema 9

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 4. Aratati ca $2 \cdot n - m \geq 4$.

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 egalitatea $2 \cdot n - m = 4$.



Rezolvare:

a) Suma gradelor este 2m.

$2n - m \geq 4 \iff 2(n-m) + m \geq 4 \iff 2(2 - |F|) + m \geq 4 \iff 4 - 2|F| + m \geq 4 \iff 2|F| \leq m \iff 4|F| \leq 2m$

Lungimea minima a unui ciclu este $4 \implies 4|F| \leq \sum_{ffata} d_M(f)$

$d_M(f) \geq 4 \implies \sum d_M(f) \geq 4|F|$

b) Vom presupune prin reducere la absurd ca graful alaturat este planar. Se poate observa ca nu exista niciun ciclu cu lungimea mai mica de 4, dar exista mai multe cicluri cu lungimea 4 $(1, 4, 2, 5, 4,\ 3, 6, 2, 4, 3$ etc).

Conform presupunerii facute, graful indeplineste cerintele de la punctul a), deci ar trebui ca inegalitatea sa fie adevarata.

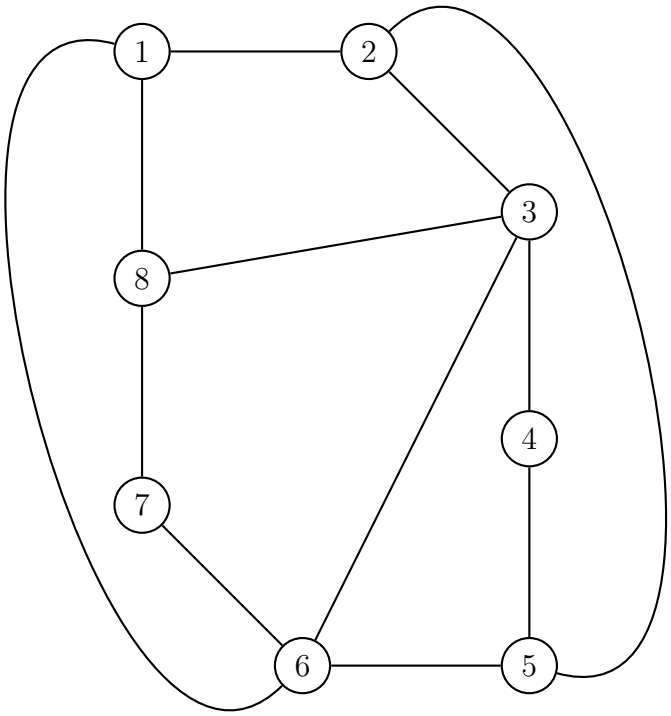
Avem:

- $n = 7$
- $m = 11$

Deci: $3 = 2n - m \leq 4$. Nu este adevarat, deci 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 4 si $2n - m = 4 \iff 2n = m + 4$.

Vom alege $n = 8 \geq 6 \implies m = 16 - 4 = 12$



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 "albastru" si "aleator", scriind matricea cu valorile subproblemelor si explicand cum au fost acestea calculate.

Rezolvare:

Pentru a calcula distanta de editare dintre cuvintele "albastru" si "aleator", vom calcula distanta Levenshtein dintre ele in functie de urmatoarele operatii de cost 1:

- stergerea unei litere
- adaugarea unei litere
- schimbarea unei litere

Calcularea ei se rezolva dinamic, cu ajutorul unei matrice. Initial, vom avea:

- $dp[0][i] = i$
- $dp[i][0] = i$

Formula de recurenta pentru calcularea fiecarei casute este:

```
dp[i][j] = min(dp[i-1][j] + 1,
               dp[i][j - 1] + 1,
               dp[i-1][j-1] + (a[i] != b[j])
               )
```

Practic, atunci cand ne deplasam pe diagonala, inseamna ca modificam o litera (aceasta poate ramane si identica, iar in acest caz costul va fi 0). Daca ne deplasam in jos, inseamna ca am sters o litera, iar daca ne deplasam spre dreapta am inserat-o.

Calculul incepe de pe prima linie, prima coloana si se finalizeaza in dreapta jos.

Matricea pentru calcularea distantei Levenshtein

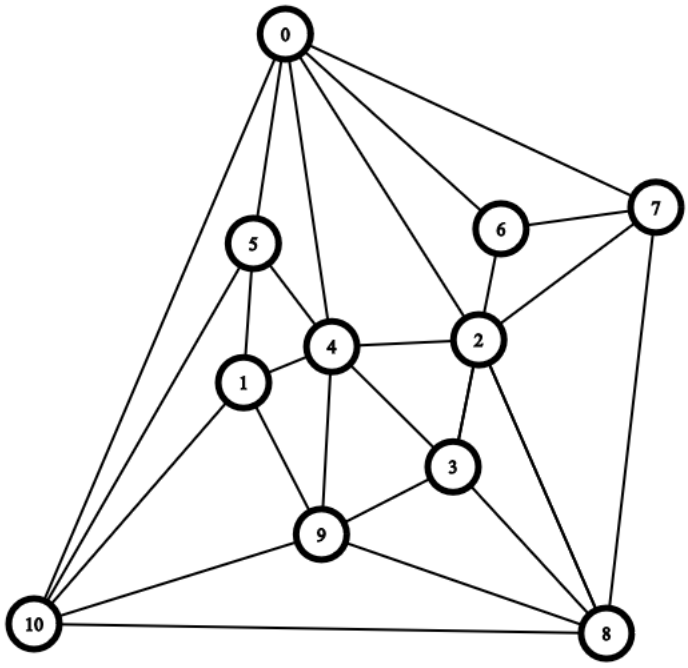
	-	a	l	b	a	s	t	r	u
-	**0**	1	2	3	4	5	6	7	8
a	1	**0**	1	2	3	4	5	6	7
l	2	1	**0**	1	2	3	4	5	6
e	3	2	1	**1**	2	3	4	5	6
a	4	3	2	2	**1**	2	3	4	5
t	5	4	3	3	2	**2**	2	3	4
o	6	5	4	4	3	3	**3**	3	4
r	7	6	5	5	4	4	4	**3**	**4**

Distanta este 4.

Problema 11

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

Graful este:



Rezolvare:

Algoritmul de 6-colorare incepe prin eliminarea nodurilor care au $grad \leq 5$, pana cand ramanem cu mai putin de 6 noduri.Daca graful are cel mult 6 noduri, atunci asignam fiecarui nod cate o culoare diferita. La fiecare pas, odata cu eliminarea nodului, vom elimina si toate muchiile incidente in el. Practic, incepem colorarea cu nodurile cu gradul cel mai mare (si mai mare decat 6), deoarece, nodurile cu $grad \leq 5$, chiar daca au toti vecinii de culori diferite, tot vor putea fi colorate.

nod	0	1	2	3	4	5	6	7	8	9	10
grad	6	4	6	4	6	4	3	4	5	5	5

Vom incepe sa adaugam pe stiva, in ordine, nodurile, de la cel cu gradul cel mai mic. Apoi, vom vedea cel mai mic grad ramas dupa eliminarea nodului. In cazul in care doua noduri ajung cu acelasi grad, vom incerca sa il alegem pe cel care initial avea gradul mai mic, dar nu este important.

$S : 6, 7, 2, 3, 4, 0, 8, 9, 1, 5, 10$

Acum incepem sa le scoatem din stiva si sa le coloram.
$10 \to 1$
$5 \to 2$
$1 \to 3$
$9 \to 2$
$8 \to 3$
$0 \to 3$
$4 \to 1$
$3 \to 4$
$2 \to 2$
$7 \to 1$
$6 \to 4$
Asadar, am folosit doar 4 culori.

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 toate sectiile nepericuloase, lasand sectiile periculoase sa fie vizitate de alti observatori. Ajutati-o sa gaseasca cel mai scurt drum care porneste din sectia 2 si viziteaza toate sectiile nepericuloase exact o data, evitand complet sectiile periculoase. (Exista drum doar intre anumite statii, iar Eliza nu poate trece pe langa o sectie fara sa viziteze). Cele K sectii periculoase sunt specificate, iar sectia 2 nu este periculoasa.

Rezolvare: Pentru a determina drumul de cost minim vom aplica un algoritm similar celui pentru determinarea drumului hamiltonian de cost minim.

Astfel, vom calcula o matrice de costuri $cost[nod][masca]$, unde nod este nodul pentru care se determina costul, iar masca este masca este masca de biti reprezentand nodurile care au fost deja vizitate.

Pentru a nu parcurge cele K noduri periculoase, le vom elimina din graf, ramanand astfel cu $N - K$ noduri.

Vom determina o masca de biti $mascaPericuloase$ in care vom seta bitii aferenti sectiilor periculoase. Pentru a determina masca aferenta parcurgerii tuturor sectiilor nepericuloase vom face operatia *not* asupra acelei masti (pentru a stii cand am realizat o parcurgere completa).

Vom realiza o parcurgere similara algoritmului lui Dijkstra, pentru care in coada de prioritati vom tine minte si masca de biti aferenta nodului (pe langa distanta). La fiecare pas vom verifica daca avem un rezultat mai bun pentru acel nod cu acea masca de biti.

Vom verifica si daca sectia pe care o parcurgem la pasul curent a fost deja vizitata, verificand daca bit-ul sau era deja setat sau nu.

Vom calcula din mers si rezultatul final care este minimul din matricea de costuri pentru masca opusa mastii celor periculoase.

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

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

$$S = C_N^1 + C_N^1 + \cdots + C_N^{N-K}$$

Prin utilizarea mastilor de biti vom genera toate submultimile pentru cele $N - K$ noduri nepericuloase (de aici provine S). $N^2 \log(M)$ este complexitatea clasica a algoritmului de drum Hamiltonian (logaritmul provine de la min-heap), iar ce doi N de la parcurgerea nodurilor.