FIȘĂ DE LUCRU · CLASA A XI-A · FIȘA 6 · ACOPERĂ LECȚIILE XI.11–XI.12

Parcurgeri BFS/DFS și grafuri orientate

Durată: 100 min Nivel: mate-info

CE VEI ȘTI DUPĂ LECȚIE

- Scrii de mână ordinea DFS și BFS, vectorul distanțelor `d` și vectorul de tați.
- Citești un graf orientat din matrice: grade, drumuri, circuite, componente tare conexe.
- Folosești BFS/DFS ca unealtă: lanț minim, vârfuri la distanța k, tare conexitate, ordine topologică.

Ce trebuie să știi (o pagină)

Regula	Cod / formulă	Observație
DFS: intri în primul vecin nevizitat, apoi revii	`viz[x] = 1; for y … if (a[x][y] && !viz[y]) dfs(y);`	stiva este chiar recursivitatea
BFS: coadă; marchezi când **pui** în coadă	`if (a[x][y] && d[y] == -1) { d[y] = d[x] + 1; t[y] = x; c[++u] = y; }`	`d[start] = 0`, restul `-1`
BFS dă distanțele minime (în muchii)	lanțul: urci pe `t[]` de la destinație	afișezi recursiv, ca să iasă de la start
ordinea depinde de ordinea vecinilor	matrice: crescător; liste: ordinea citirii	la BAC: „vecinii în ordine crescătoare”
componente conexe	`for i … if (!viz[i]) { nr++; dfs(i); }`	
orientat: o singură atribuire	`a[x][y] = 1;`	m = numărul de 1 din matrice
$d^+(i)$ = suma **liniei** i, $d^-(j)$ = suma **coloanei** j	$\Sigma d^+ = \Sigma d^- = m$	sursă: $d^- = 0$, destinație: $d^+ = 0$
arce: maxim n(n − 1); turneu n(n − 1)/2	grafuri orientate distincte: 2^(n(n−1))	fără circuite: cel mult n(n − 1)/2
tare conex ⇔ din 1 ajungi la toți **și** toți ajung la 1	DFS în `a` și DFS în transpusă	minim n arce (un circuit prin toate)
matricea drumurilor (Roy-Warshall)	`for k … for i … for j … if (d[i][k] && d[k][j]) d[i][j] = 1;`	`k` este bucla **exterioară**
ordine topologică (Kahn)	scoți mereu un vârf cu $d^- = 0$ și scazi d^- vecinilor	dacă rămân vârfuri, există circuit

ATENȚIE — GREȘELI FRECVENTE

Lanț = în graf neorientat; **drum** = în graf orientat, doar în sensul arcelor. Din „se ajunge de la x la y" **nu** rezultă „se ajunge de la y la x".

Nivelul 1 · Prezice

★ citește și prezice

Graful **G** (neorientat), folosit în mai multe probleme: n = 8, muchiile [1,2], [1,5], [2,3], [2,6], [3,4], [3,7], [5,6], [6,7], [7,8], [4,8].

P1. În G (vecinii în ordine crescătoare), scrie ordinea DFS din 1 și ordinea BFS din 1. Completează pentru BFS distanța `d` și tatăl `t` al fiecărui vârf.

vârf	1	2	3	4	5	6	7	8
`d`								
`t`								

P2. BFS din 8 în G. Completează tabelul (câte rânduri sunt necesare) și ordinea finală. Care vârf este cel mai departe de 8?

scot	adaug în coadă	coada după pas
—	8	8
8		

P3. Graful are n = 5 și muchiile citite **în această ordine**: [1,4], [3,1], [1,2], [2,5], [4,5], [3,5], memorate în liste de adiacență cu `L[x].push_back(y); L[y].push_back(x);`. Scrie listele, apoi ordinea DFS din 1 și BFS din 1 folosind **listele**. Ce ordini ai obține cu matricea de adiacență?

P4. Graful are n = 10 și muchiile [1,4], [4,7], [7,10], [9,10], [2,5], [5,8], [8,2], [3,6]. Ce afișează programul? Ce calculează, în cuvinte?

```
int dfs(int x) {
    viz[x] = 1; int s = 1;
    for (int y = 1; y <= n; y++) if (a[x][y] && !viz[y]) s += dfs(y);
    return s;
}
// în main:
for (int i = 1; i <= n; i++) if (!viz[i]) cout << dfs(i) << ' ';
```

Graful orientat **D**, folosit în mai multe probleme, are n = 7 și matricea de adiacență:

	1	2	3	4	5	6	7
1	0	1	0	0	0	1	0
2	0	0	1	0	0	0	0
3	1	0	0	1	0	0	0
4	0	0	0	0	1	0	0

	1	2	3	4	5	6	7
5	0	0	0	0	0	1	0
6	0	0	0	1	0	0	0
7	0	0	1	0	1	0	0

P5. Câte arce are D? Scrie d^+ și d^- pentru fiecare vârf. Care vârfuri sunt surse? Dar destinații? Scrie circuitele elementare.

P6. În D: ce vârfuri se pot atinge din 1? Din ce vârfuri se poate ajunge în 1? Care sunt componentele tare conexe? Care este numărul minim de arce de adăugat ca D să devină tare conex? Dă un exemplu.

P7. Completează, fără desen.

Întrebare	Răspuns	Întrebare	Răspuns
arce maxim, n = 6		turneu cu 5 vârfuri: arce	
grafuri orientate distincte cu 3 vârfuri		fără circuite, n = 7: arce maxim	
… dintre ele cu exact 2 arce		tare conex, n = 9: arce minim	
n = 8, fiecare vârf are $d^+ = 3$: arce		turnee cu 3 vârfuri / câte au circuit	

P8. Sarcinile pentru proiect sunt vârfurile 1..6, iar arcul (x, y) înseamnă „x trebuie făcută înaintea lui y”: (1,3), (2,3), (3,4), (2,5), (5,4), (4,6), (5,6). Ce afișează algoritmul lui Kahn din lecție (coada pornește cu vârfurile de grad interior 0, în ordine crescătoare; vecinii se parcurg crescător)? Este ordinea topologică unică? Câte sunt?

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Graful are muchiile [1,2], [1,3], [2,4], [3,4]. BFS-ul de mai jos afișează `1 2 3 4 4`. De ce apare 4 de două ori? Repară mutând o singură instrucțiune.

```
c[u++] = start;
while (p < u) {
    int x = c[p++];
    viz[x] = 1; cout << x << ' ';
    for (int y = 1; y <= n; y++) if (a[x][y] && !viz[y]) c[u++] = y;
}
```

P10. Un elev verifică tare conexitatea doar cu `dfs(1)` în graful dat și afișează `DA` dacă toate vârfurile au fost vizitate. Dă un graf orientat cu 3 vârfuri pentru care răspunde greșit. Ce trebuie adăugat?

P11. *Puzzle cu rânduri amestecate.* Pune în ordine BFS-ul cu distanțe și tați, pornit din `s`. Un rând este în plus. Apoi: ce afișează `drum(8)` pentru graful G și `s = 1`, dacă `void drum(int x) { if (x == 0) return; drum(t[x]); cout << x << ' '; }`?

```
A)  while (p <= u) {
B)      int x = c[p++];
C)      for (int y = 1; y <= n; y++)
D)          if (a[x][y] && d[y] == -1) {
E)              d[y] = d[x] + 1; t[y] = x; c[++u] = y;
F)  } }
G)          if (a[x][y] && d[y] != -1) {
H)  d[s] = 0; c[1] = s; p = u = 1;
```

P12. Completează golurile algoritmului Roy-Warshall (`d` pornește egală cu matricea de adiacență). Câte valori 1 are matricea drumurilor pentru D? Câte sunt pe diagonală și ce înseamnă `d[i][i] = 1`?

```
for (int k = 1; k <= n; k++)
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            if (d[i][j] == 0 && __________ && __________) d[i][j] = 1;
```

P13. *Ce greșeală a făcut elevul?* Dă câte un contraexemplu.

a. „Pe un arbore, BFS și DFS din rădăcină dau mereu aceeași ordine.”

b. „Un graf orientat cu n vârfuri și n arce este tare conex.”

c. „Graful orientat are matricea cu 10 valori de 1, deci are 10 / 2 = 5 arce.”

P14. Elevul a uitat `d[start] = 0` (toate valorile pornesc de la −1). Pe graful de la P9, din 1, ce ordine afișează și ce vector `d` obține? Ce vârf „se descoperă” de două ori?

```
for (int i = 1; i <= n; i++) d[i] = -1;
c[u++] = start;
while (p < u) {
    int x = c[p++]; cout << x << ' ';
    for (int y = 1; y <= n; y++) if (a[x][y] && d[y] == -1) { d[y] = d[x] + 1; c[u++] = y;
}
}
```

Nivelul 3 · Scrie cod

★★ scrie cod

Programele citesc `n m`, apoi `m` muchii (sau arce), apoi datele cerute. Vecinii se parcurg crescător. Verifică **toate** testele.

P15. Gradele de separare. Într-o rețea de prieteni (neorientat), afișează distanța de la `x` la `y` și, pe rândul următor, un lanț minim (cel găsit de BFS din `x`), sau `-1`. Teste pe G: `1 8` → `4` / `1 2 3 4 8`; `8 5` → `3` / `8 7 6 5`; `5 5` → `0` / `5`. Test separat: `4 1` · `1 2` · `1 4` → `-1`.

P16. Prietenii de gradul k. Afișează câte vârfuri sunt la distanța **exact** `k` de `x`, apoi vârfurile (crescător). Teste pe G: `1 2` → `2` / `3 6`; `8 3` → `2` / `2 5`; `6 1` → `3` / `2 5 7`; `1 5` → `0`.

P17. Urmăritorii. Arcul (x, y): x îl urmărește pe y. Afișează vârfurile cu cei mai mulți urmăritori (crescător), apoi câte perechi se urmăresc **reciproc**. Teste: `4 5` · `1 2` `2 1` `3 2` `4 2` `2 3` → `2` / `2`; `3 0` → `1 2 3` / `0`; graful D → `3 4 5 6` / `0`.

P18. Ordinea sarcinilor. Afișează o ordine topologică în care, la fiecare pas, alegi **cel mai mic** vârf disponibil (cu toți predecesorii deja aleși), sau `imposibil`. Teste: sarcinile de la P8 → `1 2 3 5 4 6`; `4 2` · `3 1` `2 4` → `2 3 1 4` (Kahn cu coadă ar da `2 3 4 1`!); `3 3` · `1 2` `2 3` `3 1` → `imposibil`.

P19. Orașe cu sens unic. Afișează `DA` dacă graful orientat este tare conex, altfel `NU` și cel mai mic vârf care nu e în aceeași componentă tare conexă cu 1. Teste: D → `NU 4`; `3 3` · `1 2` `2 3` `3 1` → `DA`; `3 2` · `1 2` `2 1` → `NU 3`; `1 0` → `DA`.

P20. Vestea. O veste pornește din `x` la minutul 0 și trece pe fiecare arc într-un minut. Afișează după câte minute o află ultimul care o poate afla și câți nu o află niciodată. Teste pe D: `x = 1` → `3 1`; `x = 7` → `3 0`; `x = 4` → `2 4`.

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Problemă inversă. Construiește un graf neorientat cu 5 vârfuri pentru care BFS din 1 dă `1 2 3 4 5`, iar DFS din 1 dă `1 2 5 3 4` (vecinii crescător). Câte muchii îți trebuie, minimum? Poate DFS-ul să înceapă cu `1 3` dacă BFS-ul începe cu `1 2`? De ce?

P22. De ce coadă? Un elev calculează distanțele cu DFS: `d[y] = d[x] + 1` când coboară din x în y. Dă cel mai mic graf pe care obține o distanță greșită și spune ce valoare greșită apare.

P23. Turneul fără câștigător. Cinci jucători joacă fiecare cu fiecare; arcul (x, y) înseamnă „x l-a învins pe y": (1,2), (1,4), (2,3), (2,5), (3,1), (3,4), (4,2), (4,5), (5,1), (5,3). Câte victorii are fiecare? Găsește un „clasament" `a b c d e` în care fiecare l-a învins pe următorul. De ce există **mereu** un asemenea clasament într-un turneu? (Indiciu: inserezi jucătorii unul câte unul.)

P24. Ce se întâmplă dacă... în Roy-Warshall scrii `k` ca buclă **interioară** (`for i, for j, for k`)? Verifică pe graful cu n = 4 și arcele (2,3), (3,4), (4,1): ce drum nu este descoperit? De ce?

P25. Câte arce lipsesc? Graful orientat cu n = 5 are arcele (1,3), (2,3), (3,4), (3,5). Care este numărul minim de arce de adăugat ca să devină tare conex? Arată că mai puține nu ajung (gândește-te la surse și destinații) și dă o soluție.

Probleme de lucru în clasă (pbinfo)

Le rezolvăm împreună în ședință, în ordinea de mai jos.

- [ ] `#19` **bfs** — ușoară
  https://www.pbinfo.ro/probleme/19/bfs
- [ ] `#539` **dfs** — ușoară
  https://www.pbinfo.ro/probleme/539/dfs
- [ ] `#573` **grade2** — ușoară
  https://www.pbinfo.ro/probleme/573/grade2
- [ ] `#580` **roy-warshall** — ușoară
  https://www.pbinfo.ro/probleme/580/roy-warshall

Temă pentru acasă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] #484 **lantminim** — medie
  https://www.pbinfo.ro/probleme/484/lantminim
- [ ] #4276 **nrcompconexe** — ușoară
  https://www.pbinfo.ro/probleme/4276/nrcompconexe
- [ ] #586 **gradintnul** — ușoară
  https://www.pbinfo.ro/probleme/586/gradintnul
- [ ] #583 **tare-conexitate** — medie
  https://www.pbinfo.ro/probleme/583/tare-conexitate
- [ ] #549 **epidemie** — medie
  https://www.pbinfo.ro/probleme/549/epidemie
