FIȘĂ DE LUCRU · CLASA A XI-A · FIȘA 11 · ACOPERĂ LECȚIA XI.19

Grafuri cu costuri

Durată: 110 min Nivel: intensiv

CE VEI ȘTI DUPĂ LECȚIE

- Urmărești pas cu pas Dijkstra, Roy-Floyd, Prim și Kruskal și scrii starea vectorilor după fiecare pas.
- Recunoști greșelile clasice: 0 în loc de INF, k în bucla interioară, criteriul lui Dijkstra pus în Prim.
- Alegi algoritmul potrivit și îl adaptezi: drum cu escală, centru, muchii deja construite, taxe pe noduri.

Ce trebuie să știi (o pagină)

Algoritm	Ce face	Rândul-cheie	Complexitate
matricea costurilor	`0` pe diagonală, `INF` = nu există muchie	`c[i][j] = (i == j) ? 0 : INF;`	O(n²) memorie
Dijkstra (costuri ≥ 0)	drum minim de la `s` la toate	fixezi nodul nefixat cu `d` minim; `d[x] + c[x][y] < d[y]`	O(n²)
drumul, nu doar costul	`tata[y] = x` la fiecare îmbunătățire	`void drum(int x) { if (x != s) drum(tata[x]); cout << x; }`	
Roy-Floyd	drum minim între toate perechile	`k` **exterior**: `c[i][k] + c[k][j] < c[i][j]`	O(n³)
APM	conectezi toate nodurile, cost total minim	are exact n − 1 muchii	
Kruskal	muchiile, crescător după cost; iei muchia dacă leagă două componente	`comp[x] != comp[y]`, apoi reetichetezi	O(m log m + n·m)
Prim	arborele crește dintr-un nod	ca Dijkstra, dar compari doar `c[x][y] < d[y]`	O(n²)
eliminări până la arbore	graf conex cu m muchii	m − (n − 1)	

Graful **H** („harta”), folosit în mai multe probleme: n = 6, muchiile (neorientate) 1–2 (4), 1–3 (1), 2–3 (2), 2–4 (5), 3–4 (8), 3–5 (10), 4–5 (2), 4–6 (6), 5–6 (3).

ATENȚIE — GREȘELI FRECVENTE

Cu `INF = 1000000000`, `INF + INF` încape încă în `int`, dar `INF + INF + INF` nu. Testează `c[i][k] < INF && c[k][j] < INF` înainte să aduni, și folosește `long long` pentru totaluri mari.

Nivelul 1 · Prezice

★ citește și prezice

P1. Dijkstra pe H, din nodul 1. Completează tabelul (un rând pentru fiecare nod fixat), apoi vectorul `tata` și drumul minim de la 1 la 6. Ce valoare a lui `d` se îmbunătățește **după** ce părea „bună"?

pas	nod fixat	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
start	1	0	4	1	∞	∞	∞
1							
…							

P2. De câte ori se schimbă? Graf **orientat**, n = 5: 1→2 (2), 1→3 (6), 1→5 (20), 2→3 (3), 2→5 (15), 3→4 (1), 3→5 (9), 4→5 (4). Dijkstra din 1: scrie toate valorile prin care trece `d[5]` și ordinea în care se fixează nodurile.

P3. Roy-Floyd pe etape. Graf orientat, n = 4: 1→2 (3), 2→3 (4), 3→1 (1), 3→4 (2), 4→2 (1), 1→4 (10). Scrie matricea costurilor. Ce celule se schimbă după pasul `k = 1`? Dar după `k = 2`? Scrie matricea finală.

P4. Kruskal pe H. Muchiile sunt sortate crescător (la cost egal, în ordinea din listă). Completează tabelul până la a 5-a muchie luată, cu vectorul `comp` din lecție (`vechi = comp[y]`, `nou = comp[x]`, toți `vechi` devin `nou`). Care este costul APM?

muchie (cost)	ia / respinge	comp[1..6] după
1–3 (1)		
…		

P5. Prim pe H, pornind din nodul 4. În ce ordine intră nodurile în arbore și prin ce muchie? Scrie `d` după fiecare pas. Ce cost total obții? Compară cu P4.

P6. Vectorul de tați obținut de un Dijkstra din nodul 2 este `tata = (4, 0, 2, 2, 3, 5, 6)`. Ce afișează `drum1(7)`, `drum2(7)` și `drum1(1)`?

```
void drum1(int x) { if (tata[x]) drum1(tata[x]); cout << x << ' '; }
void drum2(int x) { cout << x << ' '; if (tata[x]) drum2(tata[x]); }
```

P7. BAC, doar numere. a) Un graf neorientat conex are 12 noduri și 20 de muchii. Câte muchii are un APM? Câte se elimină? b) Dacă toate muchiile au costul 3, cât costă APM-ul? c) Un graf cu 7 noduri are 3 componente conexe: câte muchii are o pădure parțială de cost minim (câte un APM pe fiecare componentă)?

P8. Alt Floyd. Muchiile sunt poduri, iar numărul este **lățimea** podului: 1–2 (5), 2–3 (3), 1–3 (2), 3–4 (7), 2–4 (1). Inițial `w[i][j]` = lățimea, sau 0 dacă nu e pod. Ce valori au `w[1][4]` și `w[1][3]` la final? Ce calculează programul, în cuvinte?

```
for (int k = 1; k <= n; k++)
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            w[i][j] = max(w[i][j], min(w[i][k], w[k][j]));
```

Nivelul 2 · Găsește și repară

★★ găsește și repară

P9. Un elev a scris Roy-Floyd cu buclele în ordinea `i`, `j`, `k` (restul identic). Pe graful orientat 1→4 (1), 4→3 (1), 3→2 (1), ce valoare are `c[1][2]` la final? Ce ar trebui să fie? De ce se pierde drumul?

P10. Un elev a inițializat matricea cu `0` pentru „fără muchie” și a scos testul `c[x][y] < INF` din Dijkstra. Ce afișează pe H, din nodul 1? Explică primul pas greșit.

P11. Rânduri amestecate. Pune în ordine bucla principală Kruskal (după sortare și `comp[i] = i`). Un rând este în plus: e o greșeală clasică. Ce strică el? Dă un exemplu pe H.

```
A)  if (comp[e[k].x] != comp[e[k].y]) {
B)  for (int k = 0; k < m && luate < n - 1; k++)
C)  total += e[k].cost; luate++;
D)  comp[e[k].y] = comp[e[k].x];
E)  int vechi = comp[e[k].y], nou = comp[e[k].x];
F)  for (int i = 1; i <= n; i++) if (comp[i] == vechi) comp[i] = nou;
G)  }
```

P12. Prim cu criteriul greșit. În codul lui Prim din lecție, un elev a scris `if (!inArbore[y] && d[x] + c[x][y] < d[y]) d[y] = d[x] + c[x][y];`. Ce afișează pe H (pornind din 1)? Ce reprezintă, de fapt, numărul obținut?

P13. Ce greșeală a făcut elevul? Spune pentru fiecare afirmație „adevărat” sau „fals”; pentru „fals”, dă un contraexemplu mic.

a. „APM-ul conține întotdeauna o muchie de cost minim din graf.”

b. „În APM, drumul dintre oricare două noduri este și drumul minim din graf.”

c. „Dacă adun 10 la costul fiecărei muchii, drumurile minime rămân aceleași.”

d. „Dijkstra merge pe un graf neorientat cu o singură muchie de cost negativ, pentru că nu există cicluri negative.”

P14. Completează golurile. Floyd care ține minte drumul: `p[i][j]` = un nod intermediar al drumului minim (0 dacă drumul e chiar muchia). Completează, apoi scrie ce afișează `cout << 3 << ' '; drum(3, 2);` pentru graful din P3.

```
if (c[i][k] < INF && c[k][j] < INF && c[i][k] + c[k][j] < c[i][j])
{ c[i][j] = c[i][k] + c[k][j]; p[i][j] = ____; }

void drum(int i, int j)            // afișează nodurile de după i, până la j
{
    if (p[i][j] == 0) cout << ____ << ' ';
    else { drum(i, ____); drum(____, j); }
}
```

Nivelul 3 · Scrie cod

★★ scrie cod

Graful H se dă la intrare așa: `6 9`, apoi muchiile `1 2 4`, `1 3 1`, `2 3 2`, `2 4 5`, `3 4 8`, `3 5 10`, `4 5 2`, `4 6 6`, `5 6 3`. Toate grafurile sunt neorientate.

P15. Curierul. După graf se citește depozitul `s`. Afișează distanța minimă de la `s` la fiecare nod, sau `-1` pentru nodurile unde nu se poate ajunge. Teste: H, `1` → `0 3 1 8 10 13`; `5 3` / `1 2 5 2 3 1 4 5 2`, `2` → `5 0 1 -1 -1`; `1 0`, `1` → `0`.

P16. Taxiul. După graf se citesc `s` și `t`. Afișează costul minim, iar pe rândul următor drumul, de la `s` la `t`; sau `imposibil`.

intrare	ieșire
H, `1 6`	`13` / `1 3 2 4 5 6`
H, `6 1`	`13` / `6 5 4 2 3 1`
`4 2` / `1 2 3` `3 4 1`, `1 4`	`imposibil`

P17. Fibra optică. Afișează costul APM, apoi muchiile alese, în ordinea în care le ia Kruskal (la cost egal, în ordinea din intrare); `imposibil` dacă graful nu e conex. Teste: H → `13`, apoi `1 3`, `2 3`, `4 5`, `5 6`, `2 4`; `4 2` / `1 2 3` `3 4 1` → `imposibil`; triunghi cu toate costurile 5 → `10`, `1 2`, `2 3`.

P18. Unde construim spitalul? Alege nodul pentru care **cea mai mare** distanță până la celelalte noduri e cât mai mică. Afișează nodul (cel mai mic, la egalitate) și acea distanță. Teste: H → `4 8`; `3 2` / `1 2 1` `2 3 1` → `2 1`; `4 4` / `1 2 10` `2 3 1` `3 4 1` `1 4 10` → `2 10`.

P19. Drum cu escală. Se citesc `s`, `k`, `t`: cel mai ieftin drum de la `s` la `t` care trece prin `k` (nodurile se pot repeta). `-1` dacă nu există. Teste: H, `1 6 2` → `23`; H, `1 4 6` → `13`; `4 2` / `1 2 3` `3 4 1`, `1 3 4` → `-1`.

P20. Cabluri deja trase. După graf se citește `p` și `p` perechi de noduri între care cablul există deja (gratuit). Cât mai costă, minim, ca rețeaua să fie conexă? Teste: H, `2` / `3 5` / `1 2` → `6`; H, `0` → `13`; H, `1` / `4 6` → `10`.

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P21. Problema inversă. În H, costul muchiei 3–5 devine `x` (în loc de 10). a) Pentru ce valori ale lui `x` drumul minim de la 1 la 5 se scurtează? b) Pentru ce valori **trebuie** muchia 3–5 să facă parte din orice APM? De ce pragurile sunt diferite?

P22. De ce funcționează? Roy-Floyd scrie rezultatele **în aceeași matrice** pe care o citește. La pasul `k`, se poate ca `c[i][k]` sau `c[k][j]` să fie deja modificate în același pas, și să strice calculul? Argumentează (indiciu: cât este `c[k][k]`?).

P23. Contraexemplu. „Ca să ajungi de la `s` la `t`, mergi mereu pe cea mai ieftină muchie spre un nod nevizitat.” Pe H, de la 1 la 6, regula chiar dă drumul minim. Construiește un graf cu 3 noduri pe care regula greșește.

P24. Ce se întâmplă dacă... Kruskal sortează muchiile **descrescător**? Ce obții pe H (costul și muchiile)? La ce ar putea folosi un astfel de arbore?

P25. Taxe de oraș. În H, la intrarea în orașele 2, 4, 5 plătești o taxă: 5, 1, respectiv 4 (în celelalte, 0; în orașul de start nu plătești). Cum transformi graful ca să poți folosi Dijkstra sau Floyd neschimbat? (Graful rezultat mai este neorientat?) Care este costul minim de la 1 la 6 și pe ce drum?

Probleme de lucru în clasă (pbinfo)

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

- [ ] #589 **roy-floyd** — ușoară
  https://www.pbinfo.ro/probleme/589/roy-floyd
- [ ] #588 **dijkstra** — ușoară
  https://www.pbinfo.ro/probleme/588/dijkstra
- [ ] #590 **prim** — ușoară
  https://www.pbinfo.ro/probleme/590/prim

Temă pentru acasă (pbinfo)

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

- [ ] #592 **kruskal** — ușoară
  https://www.pbinfo.ro/probleme/592/kruskal
- [ ] #1651 **graf** — ușoară
  https://www.pbinfo.ro/probleme/1651/graf
- [ ] #1652 **rf** — ușoară
  https://www.pbinfo.ro/probleme/1652/rf
- [ ] #591 **firma** — medie
  https://www.pbinfo.ro/probleme/591/firma