Algoritmi Fundamentali Tema 1

Ștefan-Octavian Radu

November 19, 2020

Exercițiul 1

Să consideram ce se întâmplă când vrem să adăugăm o muchie nouă (u, v), cu $u, v \in E$, unui graf $G = (V, E)$, astfel încât acesta să rămână valid. Vom analiza două cazuri:

1. u și v aparțin aceleiași componentă conexă
2. u și v aparțin unor componente conexe disjuncte

Cazul 1.

Fie $C \subset V$ o componentă conexă a.î. $u \in C$ și $v \in C$. De aici rezultă că există un drum $(a_1 = u, a_2, \dots, a_k = v)$ (de la u la v), cu $a_i \in C$. Adăugând muchia u, v am obține un ciclu, ceea ce reprezintă o *contradicție*.

Cazul 2.

Fie $C1 \subset V$, cu $u \in C1$ și $C2 \subset V$ cu $v \in C2$. Din ipoteză reiese că $C1 \cap C2 = \emptyset$. De aici rezultă că nu există un drum între u și v, deci adăugând muchia (u, v) nu vom obține un ciclu.

Știm că $\forall x_1 \in C1, \exists(x_1, x_2, \dots, x_k = u)$, cu $x_i \in C1$, un drum de la x_1 la u. Știm și că $\forall y_1 \in C2, \exists(v = y_k, y_{k-1}, \dots, y_1)$, cu $y_i \in C2$, un drum de la v la y_1. Adăugând muchia (u, v) ne rezultă că $\forall x \in C1$ și $\forall y \in C2$ $\exists(x, \dots, u, v, \dots, y)$, un drum între x și y, deci am obținut o nouă componentă conexă formată din nodurile $C1 \cup C2$.

Observația 1: O muchie care ne păstrează proprietățile grafului se poate adăuga doar între noduri din componente conexe diferite. În consecință, pentru adăugarea unei noi muchii, este necesar ca în graf să existe minim două componente conexe.

Observația 2: La adăugarea unei noi muchii, numărul de componente conexe ale grafului scade cu 1.

Concluzie

Dacă considerăm un graf cu n noduri și nicio muchie, ajungem imediat la concluzia problemei, deoarece putem ajunge la numărul minim de o componentă conexă prin adăugare de maxim $n - 1$ muchii noi.

Exercițiul 2

Modelare

Să considerăm cazul particular al cubului Rubik clasic de 3x3. Știind că numărul de piese care își pot schimba pozițiile (colțurile și marginile) este de $6 \cdot 8 = 48$, putem modela o stare a cubului prin intermediul unui $48 - tuplu$, $(v_0, v_1, \dots, v_{47})$, unde:

- $(0, 1, 2, \dots, 47)$ reprezintă starea rezolvată a cubului (Starea 0)
- $v_{i \cdot 8+j}$ reprezintă indicele piesei care se află pe fața i (cu $i \in$ {0 - Alb, 1 - Roșu, …, 5 - Galben}) și pe poziția j (cu $j \in$ {0 - sus stânga, 1 - sus mijloc, …, 7 - mijloc stânga})

Fie $G = (V, E)$ graful cubului Rubik, unde:

- V este definită ca mulțimea tuturor stărilor ($48 - tulurilor$) valide în care se poate afla cubul Rubik
- E este definită ca mulțimea tuturor perechilor (a, b) (cu $a \in V$ și $b \in V$), cu proprietatea că din starea a se poate ajunge în starea b printr-o rotire de 90° a uneia dintre fețele cubului Rubik
- graful este neorientat, întrucât orice rotire a unei fețe este reversibilă

Descrierea soluției

Putem utiliza o căutare în lățime pornind dintr-o starea aleatoare, dar acesta nu ar fii foarte eficient, considerând numărul cardinal imens al mulțimii V

Totuși, inspirându-ne din modul de rezolvare clasic al cubului (cruce albă, colțuri albe, latura din mijloc, etc.), putem introduce niște stări intermediare $s_0, s_1, \dots, s_n, 0$ (unde s_0 este starea aleatoare de la care începem rezolvarea, iar 0 este starea rezolvată) prin care să trecem succesiv pe parcursul rezolvării, astfel încât numărul de mutări necesar pentru a trece din s_i în s_{i+1} să fie considerabil redus. Soluția problmemei inițiale se obține prin concatenarea sulușiilor pentru problemele $s_0 \longrightarrow s_1$, $s_1 \longrightarrow s_2$, …, $s_n \longrightarrow 0$.

Această abordare poate fi generalizată și utilizată pentru cuburi rubice de orice dimensiune.

Exercițiul 3

Pentru obținerea distanței minime pornind din nodul 1 către toate celelalte noduri ale grafului, putem utiliza BFS în cazul unui graf fără greutăți pe muchii, respectiv algoritmul lui Dijkstra în cazul unui graf cu greutăți pe muchii.

Modelarea grafului

Să considerăm graful $G' = (V, E')$, cu $E' \in E$, obținut în urma rulării unuia dintre algoritmii menționați mai sus, și păstrarea muchiilor de pe drumurile minime.

- Din modul în care este construit graful, știm că $\forall i \in V \setminus \{1\}$ există un drum de la 1 la i, și fiind un graf neorientat, deducem că este și conex.
- Din modul în care este construit graful, știm că $\forall i \in V \setminus \{1\}$ $\exists!$ (j, i) $\in E'$ cu proprietatea că suma dintre distanța minimă până la j și lungimea muchiei (j, i) este egală cu distanța minimă până la i. În consecință, graful are $|V| - 1$ muchii.
- Graful obținut este un arbore, deci este și aciclic, ceea ce ne încadrează în restricțiile exercițiului 1, de unde deducem că $|E'|$ este minim, întrucât eliminarea unei muchii ar rezulta în separarea grafului în două componente conexe.

Descrierea soluției

Graful ales corespunde restricțiilor și are numărul minim de muchii. În consecință, numărul maxim de muchii pe care îl putem elimina din graful inițial este egal cu $|E| - (|V| - 1)$.

Exercițiul 4

Observație: Să spunem că ne dorim ca $ax + by$ să fie divizibil cu d, este echivalent cu a spune că ne dorim ca $ax + by \equiv 0 \pmod{d}$, sau cu $\widehat{ax + by} = 0 \pmod{d}$.

Modelarea grafului

Pornind de la observația precedentă ne vine ideea să modelăm problema sub forma unui graf $G = (V, E)$ după cum urmează:

- $V = \{0,\ 1,\ \ldots,\ d - 1\}$
- $E = \{(u, \widehat{u + a}),\ (u, \widehat{u + b}) \mid u \in V\}, \pmod{d}$

Mulțimea V a nodurilor din graf reprezintă mulțimea claselor de echivalență modulo d, iar mulțimea E a muchiilor din graf reprezintă toate tranzițiile posibile între aceste clase de echivalență utilizând valorile a și b.

Graful este orientat, întrucât nu există certitudinea că operația de adunare modulo d este inversabilă.

Graful obținut modelează corect problema inițială, întrucât existența unei perechi (x, y) pentru care $\widehat{ax + by} = 0 \pmod{d}$, este echivalentă cu existența în graf a unui drum de la $\widehat{a + b}$ la 0.

Descrierea soluției

Rezolvarea problemei poate utiliza o parcurgere în lățime pentru determinarea distanței minime de la nodul $\widehat{a + b}$ la nodul 0, care corespunde sumei căutate $x + y$.

Exercițiul 5

Caz general (Bonus)

Fie $G' = (V, E'),\ cu\ E \subseteq E'$, un graf cu arborele *DFS* identic cu arborele inițial G.

Fie $d_G : V \longrightarrow \mathbb{N},\ d_G(u)$ = distanța din rădăcină (nodul 1) până în nodul u în arborele *DFS* al grafului G'.

Putem analiza schimbările rezultate ca urmare a introducerii unei noi muchii (a, b) (unde $d_G(1, a) \geq d_G(1, b)$) în G' considerând următoarele două cazuri:

1. b se află pe drumul de la rădăcină la a. În acest caz, muchia se poate adăuga în lista de adiacență astfel încât în parcurgerea *DFS* să apară ca muchie de întoarecere și să fie ignorată.
2. b **NU** se află pe drumul de la rădăcină la a. În acest caz, nodurile unite fac parte din subarbori diferiți ai arborelui *DFS* inițial. O data cu introducrea muchiei (a, b), cele două noduri ar apărea in același subarbore în parcurgerea *DFS*, care va fi diferită de arborele inițial G și, în consecință, nu convine.

Soluție generală

Soluția problemei se obține adăugând la numărul de muchii din arborele *DFS* numărul ne muchii de întoarcere ce pot fi adăugate și este egală cu:

$$|V| - 1 + \sum_{u \in V \setminus \{1\}} (d_G(u) - 1) = \sum_{u \in V \setminus \{1\}} d_G(u)$$

Soluție particulară

Utilizând formula de mai sus, numărul maxim de muchii pe care il poate avea un graf, cu arborele *DFS* identic cu cel din figură, este 24, și corespunde grafului cu listele de adiacență:

- 1: 3, 6, 4, 8, 5, 2, 11, 7, 10, 9
- 2: 5, 4, 3, 1
- 3: 4, 11, 1, 8, 5, 2
- 4: 8, 5, 3, 1, 2
- 5: 2, 4, 3, 1
- 6: 7, 1, 10, 9
- 7: 10, 9, 6, 1
- 8: 4, 3, 1
- 9: 7, 6, 1
- 10: 7, 6, 1
- 11: 3, 1

Ordinea în care sunt procesați vecinii unui vârf corespunde cu ordinea din lista de adiacență corespunzătoare.

Exercițiul 7

Modelare

Fie graful $G = (V, E)$ modelat după cum urmează:

- $V = \{a_i \mid 1 \leq i \leq n\} \cup \{D\}$
- $E' = \{(a_i,\ a_j,\ b_i - a_i) \mid b_i \leq a_j \text{ și } \nexists\ k \text{ a.î } b_i \leq a_k < a_j,\ \forall\ 1 \leq i, j, k \leq n\}$
- $E = E' \cup \{(a_i,\ D,\ b_i - a_i) \mid \forall\ 1 \leq i \leq n\}$

V reprezintă mulțimea tuturor capetelor stânga din mulțimea segmentelor $[a_i, b_i]$. E reprezintă toate perechile *ordonate* de segmente $(([a_i, b_i], [a_j, b_j])$ cu proprietatea că $[a_j, b_j]$ e cel mai apropiat segment la dreapta de $[a_i, b_i]$) care respectă restricțiile de nesuprapunere.

Nodul D, precum și muchiile suplimentare care ajung în acesta au rolul de a suplimenta ultimul segment din cadrul unui drum.

Graful este orientat și aciclic întrucât parcurge segmentele decât într-o direcție.

Graful modelează corect problema deoarece:

- suma costurilor de pe muchiile unui drum care se termină în nodul D corespunde suma lunigimilor unor segmente care se suprapun în maxim un punct
- mulțimea drumurilor care se termină în nodul D din graful astfel descris este echivalentă cu mulțimea submulțimilor S care respectă restricțiile

Soluție

Soluția se poate obține parcurgând sortarea topoligică a grafului transpus (întrucât și acesta este un *DAG*), pornind din nodul D și reținând drumul de lungime maximă.

Exercițiul 9

Modelare

Fie graful $G = (V, E)$ modelat după cum urmează:

- $V = \{1, \dots, n\}$
- $E = \{(p_i, p_j, \sqrt[2]{(x_i - x_j)^2 + (y_i - y_j)^2}) \mid i \neq j, i, j \in \{1, \dots, n\}\}$

V este mulțimea puntelor de interes, iar E este mulțimea tripletelor formate din puncte distincle de interes și distranța euclidiană dintre acestea în plan.

Graful astfel construit modelează corect problema întrucât un drum, precum și lungimea sa din enunț este echivalent cu același drum parcurs pe graful descris anterior.

Soluție

Drumul minim care parcurge toate nodurile și se întoarce în punctul de plecare este reprezentat de un ciclu Hamiltonian de cost minim

Putem demonstra că un arbore parțial de cost minim $G' = (V, E' \subset E)$ al grafului complet G, are costul mai mic sau egal cu cel mai scurt ciclu Hamiltonian, deoarece prin eliminarea unei muchii din acesta am obține un arbore parțial cu cost mai mic decât cel inițial, ceea ce este imposibil.

Putem așadar obține soluția problemei printr-o parcurgere DFS dus-întors a arborelui parțial de cost minim al grafului inițial. Acest APM se poate obține cu ajutorul algoritmului lui Prim, care este mai eficient decât cel al lui Kruskal, deoarece lucrăm cu un graf complet.