Algoritmi fundamentali
Tema 1

Laura-Maria Tender

November 2020

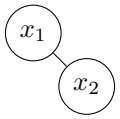
Exercițiul 1

Să se demonstreze că orice graf neorientat cu n noduri fără cicluri are cel mult $n-1$ muchii.

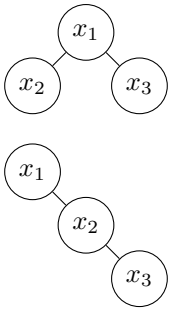
Soluție propusă

Fie $G = (V, E)$ un graf neorientat aciclic, unde $V = \{V_1, V_2, ..., V_n\}$.
Fie P(n): Un graf neorientat cu n noduri fără cicluri are cel mult $n-1$ muchii.

Etapa de verificare
P(2): Un graf neorientat cu 2 noduri fără cicluri are cel mult 1 muchie.

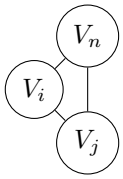


Vom analiza și P(3) pentru a vedea mai bine cum se comportă grafurile.
P(3): Un graf neorientat cu 3 noduri fără cicluri are cel mult 2 muchii.



Putem observa că oricum am încerca să trasăm a treia muchie am obține un ciclu.

Etapa de demonstrare
$P(n-1) \rightarrow P(n)$
Fie $P(n-1)$: Un graf neorientat cu $n-1$ noduri fără cicluri are cel mult $n-2$ muchii.
La acest graf vom mai adauga un nod. Putem trasa o muchie între noul nod și orice alt nod. Trasăm muchia $V_nV_i, i \in \overline{(1, n-1)}$. Dacă graful avea număr maxim de muchii, muchia nu poate fi trasată între nodurile deja existente. Presupunem că putem trasa muchia $V_nV_j, j \in \overline{(1, n-1)}, j \neq i$. Cum graful cu $n-1$ noduri are număr maxim de muchii atunci exista drum de la V_i la V_j. Astfel am obține un ciclu asemănător cu cel de mai jos:



Pentru un graf cu $n-1$ noduri care conține m muchii, m mai mic decât numărul maxim de muchii $n-2$. O muchie poate uni noduri care fac parte din aceeași componentă conexă sau din componente conexe diferite. Dar într-un graf aciclic, o muchie unește noduri din componente conexe diferite. Astfel, într-un graf aciclic, o muchie scăde cu 1 numărul de componente conexe dintr-un graf. Dacă graful cu n muchii conține m muchii atunci acesta conține $n-m$ componente conexe. Graful conține număr maxim de muchii când devine o singură componentă conexă, deci când conține $n-1$ muchii.

Întrucât nu putem adauga a n-a muchie în graful cu n noduri pentru ca acesta să rămână aciclic, pot exista maxim $n-1$ muchii.
În concluzie, conform principiului inducției matematice, $P(n)$ este adevărată.

Exercițiul 2

Fie problema rezolvării cubului Rubik. Modelați-o ca o problemă de grafuri.
1. Care sunt mulțimile de noduri și muchii (V, E)? Este graf sau neorientat?
2. Descrieți un algoritm eficient care găsește o rezolvare a cubului.

Soluție propusă

1. Mulțimea de noduri este mutimea stărilor posibile ale cubului. Muchiile sunt neorientate și vecinătatea a 2 noduri reprezintă că se poate ajunge dintr-o stare în cealaltă printr-o singură mutare.
2. A rezolva cubul Rubik înseamnă a găsi un drum de la starea inițială la starea finală. Cubul are foarte multe stări așa că nu este eficient să le reținem pe toate. Un algoritm care ar rezolva problema ar fi un BFS de la starea inițială la starea

finală. S-a demonstrat că orice configurație a cubului poate fi rezolvată în cel mult 20 de mișcări, echivalent cu 20 de nivele în parcuregerea BFS. Pentru a optimiza memoria putem reține doar stările prin care trecem generându-le și putem porni două parcurgeri BFS, una din nodul inițial, una din cel final și soluția este găsită atunci când găsim un nod care apare în ambele parcurgeri.

Exercițiul 3

Fie $G = (V, E)$ un graf neorientat conex, unde $V = \{1, 2, ..., n\}$. Care este numarul maxim de muchii ce pot fi șterse din G pentru care noul graf $G' = (V, E')$ $(E' \subseteq E)$ să rămână conex și să respecte $d_G(1, i) = d_{G'}(1, i)$ $\forall$ $1 \leq i \leq n$, unde $d_G(1, i)$ este distanța dintre a și b pe graful G.

Soluție propusă

Fie P(n): Un graf conex cu n noduri conține minim $n - 1$ muchii.

Etapa de verificare
P(2): Un graf conex cu 2 noduri conține cel puțin 1 muchie.
P(3): Un graf conex cu 3 noduri conține cel puțin 2 muchii.
Pentru a vizualiza aceste propoziții putem observa figurile de la exercițiul 1.

Etapa de demonstrare
P($n - 1$) $\rightarrow$ P(n)
Fie P($n - 1$): Un graf conex cu $n - 1$ noduri are cel puțin $n - 2$ muchii.
La acest graf vom mai adauga un nod. Pentru a forma o componentă conexă este suficient să mai trasăm o singură muchie între noul nod și oricare alt nod, întrucât graful anterior este conex.
În concluzie, conform principiului inducției matematice, P(n) este adevărată.
Pornind un BFS din nodul 1 aflăm distanța de la acest nod la toate nodurile. Dacă în G' păstrăm toate muchiile din parcurgerea BFS atunci $d_G(1, i) = d_{G'}(1, i)$ $\forall$ $1 \leq i \leq n$. Muchiile pe care nu le parcurgem în BFS pot fi eliminate și rămânem cu $n - 1$ muchii. În concluzie, putem elimina $|E|$ - $(n - 1) = |E|$ - $n + 1$ muchii.

Exercițiul 4

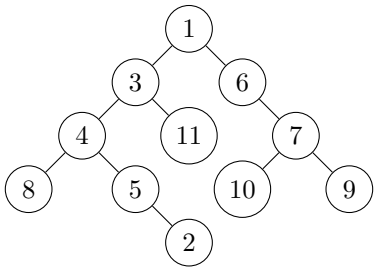
Fie a, b numere naturale pozitive și $1 \leq d \leq 10^6$. Găsiți două numere naturale pozitive x și y a.î. $ax + by$ divizibil cu d și $x + y$ este minim posibil (dacă există). Modelați-o ca o problemă de grafuri.
1. Care sunt mulțimile de noduri și de muchii (V, E)? Este graf orientat sau neorientat?
2. Descrieți un algoritm eficient care găsește o rezolvare a problemei.

Soluție propusă

1. Mulțimea de noduri este reprezentată de mulțimea resturilor la împărțirea cu d. Muchiile sunt orientate. Fie nodul r, $r \in \overline{(0, d-1)}$, graful conține muchie orientată de la r către $(r+a)\%d$ și către $(r+b)\%d$. Muchiile au costul 1.
2. Soluția problemei este drumul de cost minim de la nodul 0 la nodul 0. $x + y =$ numarul de muchii parcurse. Un algoritm eficient care găsește soluția problemei este o parcurgere de tip BFS care pornește din nodul 0 și ajunge din nou în nodul 0. Graful conține d noduri și $2d$ muchii, deci algoritmul obține o complexitate $O(d)$. Însă x și y sunt pozitive, deci nenule, astfel trebuie să verificăm că drumul conține ambele tipuri de muchii), acceptăm soluția dacă conține cel puțin o muchie de la r către $(r+a)\%d$ și cel puțin o muchie de la r' către $(r'+b)\%d$, unde r și r' sunt noduri, $r, r' \in \overline{(0, d-1)}$.

Exercițiul 5

În figura următoare este ilustrat arborele DFS al unui graf $G = (V, E)$, unde $V = \{1, 2, ..., 11\}$. Care este numărul maxim de muchii pe care îl poate avea graful inițial pentru a se obține acest arbore? Trebuie precizată și ordinea în care au fost procesați vecinii unui vârf.
Bonus: Generalizați problema pentru arbori arbitrari.



Soluție propusă

Putem împărți muchiile posibile în următoarele categorii: muchiile arborelui, muchiile "înapoi" și muchiile "laterale". Vom analiza care dintre ultimele 2 tipuri sunt posibile printr-o parcurgere DFS.
Muchiile "înapoi": poate exista muchia 4 - 1? Dacă 4 se află dupa 3 în lista de adiacența a lui 1 muchia poate exista. Deci, parcurgerea DFS permite astfel de muchii.
Muchiile "laterale": Dacă ar exista o muchie laterală, atunci ar exista o relație de vecinătate între noduri din subarbori diferiți. Atunci când DFS revine la un nod, toți vecinii nodurilor din subarborele său au fost procesați, deci acest nod ar trebui să se afle în același subarbore. Astfel DFS nu permite muchiile "laterale".
Vom adăuga cât mai multe muchii de "întoarcere" în graf. Va exista muchie

între orice rădăcină și nod din subarborele său. Nodurile vor fi procesate în ordinea în care se află în listele de adiacență descrise mai jos:
1: 3 6 11 4 8 2 7 5 9 10
2: 5 4 3 1
3: 1 4 11 8 5 2
4: 3 8 5 2 1
5: 4 2 3 1
6: 1 7 10 9
7: 6 10 9 1
8: 4 3 1
9: 7 6 1
10: 7 6 1
11: 3 1
La cele 10 muchii deja existente le-am adăugat pe următoarele: (1, 4), (1, 11), (1, 8), (1, 5), (1,2), (1, 7), (1, 10), (1, 9) (3, 8), (3, 5), (3, 2), (4, 2), (6, 10), (6, 9). Obținem un număr maxim de 10 + 14 = 24 de muchii.
Bonus: Analog pentru orice graf vom adăuga cât mai multe muchii de "întoarcere" în graf. Va exista muchie între orice rădăcină și nod din subarborele său.

Exercițiul 6

Fie $G = (V, E)$ un graf neorientat conex unde $V = \{1, 2, ..., n\}$ și o relație de ordine $<$ pe V. Se consideră parcurgerile DFS și BFS care pornesc din nodul 1 și viziteaza vecinii unui nod în ordinea dată de $<$. Ce proprietăți trebuie să respecte G pentru ca cele două parcurgeri să obțină același arbore parțial $T_{DFS} = T_{BFS}$.

Soluție propusă

În exercițiul anterior am analizat tipurile de muchii dintr-un arbore DFS. În paralel vom analiza și tipurile de muchii din arborele BFS. O muchie de întoarcere înseamnă o relație de vecinătate între un nod și un strămoș al său. Dar într-un BFS toți vecinii nevizitați ai unui nod sunt procesați imediat devenind copiii lui. Deci BFS nu permite muchii de întoarcere. O muchie laterală reprezintă o relație de vecinătate între noduri din subarbori diferiți. Fie graful $G' = (V', E')$ un graf neorientat conex unde $V' = \{1, 2, ..., 5\}$ și $E' = \{(1, 2), (1, 3), (1, 4), (2, 5), (4, 5)\}$. În acest exemplu, muchia (4, 5) este o muchie laterala. Deci, BFS permite muchiile laterale.

Întrucât $T_{DFS} = T_{BFS}$ graful poate avea doar muchii de arbore. Deci, relația este adevărată pentru arbori.

Exercițiul 7

Se dau n intervale închise $[a_i, b_i], 1 \leq i \leq n$. Trebuie să alegem o submulțime S de intervale astfel încât oricare două să se intersecteze în maximul un punct, iar suma lungimilor intervalelor alese să fie maximă. Modelați-o ca o problemă de grafuri.
1. Care sunt mulțimile de noduri și de muchii (V, E)? Este graf orientat sau neorientat?
2. Descrieți un algoritm eficient care găsește o rezolvare a problemei.
Bonus: Considerăm varianta 2D a problemei (intervalele devin regiuni dreptunghiulare, iar problema cere să maximizăm suma ariilor). Ce problemă de grafuri modeleaza noua problemă?

Soluție propusă

1. Mulțimea nodurilor va fi mulțimea numerelor naturale de la 0 la $2n - 1$. Aceste numere naturale reprezintă o redenumire a nodurilor reale care sunt reprezentate de capetele intervalelor noastre. Aceasta normalizare va fi efectuată prin sortarea mulțimii ce conține capetele intervalelor $\{a_1, b_1, a_2, b_2, ..., a_n, b_n\}$. Astfel cel mai mic număr va fi redenumit 0, al doilea număr 1 și tot așa. Muchiile vor fi orientate și vor fi între normalizarea lui a_i și normalizarea lui b_i cu costul $b_i - a_i$. De asemea vor exista muchii de cost 0 de la nodul j la $j + 1$, unde j este număr natural de la 0 la $2n - 2$.
2. Soluția a problemei este drumul de cost maxim de la 0 la $2n - 1$. Întrucât muchiile sunt orientate "spre dreapta" graful este aciclic. Astfel pentru a afla drumul maxim în grafuri orientate aciclice putem folosi algoritmul ce implică sortare topologică și programare dinamică având o complexitate $O(V + E)$. Dar parcurcând graful de la stânga spre dreapta obținem o sortare topologică ceea ce ușurează implementarea.

Exercițiul 8

Fie un graf neorientat $G = (V, E)$, unde V = $\{1, 2, ..., n\}$ și E nu este inițial cunoscut. Graful poate fi descoperit prin interogări de tipul $Q(i, j)$ către un oracol, care va răspunde cu 1 dacă muchia neorientată (i, j) există în E și cu 0 altfel. Demonstrați că, pentru orice algoritm A de determinare a componentelor conexe, există un graf G_A pentru care A trebuie să facă cel puțin $\frac{n(n-1)}{2}$ interogări.

Soluție propusă

Presupunem că există un algoritm A care poate să determine toate componentele conexe în mod unic prin cel mult $\frac{n(n-1)}{2}$ - 1 interogări. Algoritmul a făcut $q < \frac{n(n-1)}{2}$ interogări. Pentru algoritmul A există un graf la care oracolul să răspundă 0 la toate interogările. Există $\frac{n(n-1)}{2} - q$ muchii a căror stare nu

este cunoscută. Din acele muchii poate fi aleasă una (u, v) $\forall q$. Nodurile u și v pot face parte din aceeași componentă conexă sau pot fi în componente conexe diferite. Algoritmul nu poate ști răspunsul corect și indiferent de răspunsul său, pentru una dintre cele două variante răspunsul ar fi greșit.
În concluzie, presupunearea făcută este falsă, deci pentru orice algoritm A de determinare a componentelor conexe, există un graf G_A pentru care A trebuie să facă cel puțin $\frac{n(n-1)}{2}$ interogari.

Exercițiul 9

Fie n un punct de interes, reprezentate prin puncte în plan (x_i, y_i) $(1 \leq i \leq n)$. Un turist împătimit pornește din punctul (x_1, y_1) și dorește să viziteze fiecare alt punct de interes, ca mai apoi să se întoarcă în locul de unde a plecat, parcurgând o distanță cât mai mică. După multe încercări, nu a reușit să găsească cel mai bun itinerariu, așa că se multumește ca distanța parcursă să fie de cel mult două ori mai mare decât distanța minimă. Descrieți un algoritm eficient care găsește un itinerariu corespunzător.
Notă: Distanța dintre două puncte de interes (x_i, y_i) și (x_j, y_j) se va considera cea euclideană, egală cu $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Soluție propusă

Fie graful neorientat complet ale cărui noduri sunt punctele de interes, iar costul muchiilor este distanța euclidiană dintre capetele muchiei. Există soluția optimă, un drum de cost minim care este un ciclu. Presupunem că în soluția optimă ar exista repetiții. Fie y un nod care se repetă și apare în secvența x y z . Întrucât costul este minim $cost((x, y)) + cost((y, z)) \leq cost((x, z))$. Conform regulii triunghiului relația este adevărată doar dacă y aparține dreptei xy, caz în care obținem egalitate. Deci putem considera că ciclul optim nu are repetiții. Astfel dacă din ciclul optim am elimina o muchie am obține un lanț.

Conform definiției arborelui parțial de cost minim, distanța lanțului obținut mai devreme este mai mare decât suma muchiilor APM-ului. Prin Parcurgerea Euler a APM-ului obținem un drum care pleacă din (x_1, y_1) se întoarce în (x_1, y_1) și trece de 2 ori prin toate muchiile. Cum costul drumului optim este mai mare decât costului APMului, costul acestei parcurgeri (Euler) este de mai mic decât 2*distanța drumului minim. Obs: putem optimiza aceasta soluție prin eliminarea repetițiilor din parcurgerea Euler conform demonstrației anterioare.

În concluzie, problema se reduce la un algoritm de determinare a unui APM și la parcurgerea Euler a acestuia. Determinând APM-ul prin algoritmul lui Prim implementat cu ajutorul heapurilor Fibonacci obținem o complexitate $O(n^2)$ întrucât graful este complet. Parcurgerea Euler are complexitate $O(n)$

întrucât APMul are $O(n)$ muchii. Astfel soluția oferă răspunsul într-o complexitate $O(n^2)$.