Tema 1 Seminar

Dumitrescu Delia Ioana

November 2020

Exercitiul 1

Fie G = (V, E) un graf aciclic conex. Presupunem ca putem construi un arbore cu n noduri si mai mult de (n-1) muchii. La adaugarea muchiei (a, b), la un oarecare moment, se pot intampla urmatoarele:

- Daca nodurile a si b fac parte din aceeasi componenta conexa, adaugarea muchiei (a, b) va forma un ciclu si deci nu vom mai avea un graf aciclic. Astfel, vom trata doar urmatorul caz, anume:
- Nodurile a si b fac parte din componente conexe diferite. Adaugarea muchiei (a, b) va uni cele doua componente conexe. Aplicam acelasi rationament la fiecare pas:
  - Pasul 0: n componente conexe, 0 muchii
  - Pasul 1: n - 1 componente conexe, 1 muchie
  - Pasul 2: n - 2 componente conexe, 2 muchii
    .
    .
    .
  - Pasul n - 1: 1 componenta conexa, (n-1) muchii
  - Pasul n: 1 componenta conexa, n muchii. Muchia cu numarul n in ordinea adaugarii va forma un ciclu. Astfel, presupunerea facuta este falsa, deci orice graf aciclic conex are cel mult n - 1 muchii.

Exercitiul 2

Pentru rezolvarea problemei, vom descrie graful G = (V, E) astfel:

- Nodurile reprezinta toate starile posibile ale cubului
- Muchiile se adauga intre 2 noduri daca se poate ajunge dintr-o stare in alta printr-o singura miscare

Graful este unul orientat, deoarece miscarile sunt reversibile: putem realiza o miscare pentru a ajunge din starea s1 in starea s2, dar putem de asemenea sa ajungem din starea s2 inapoi in starea s1 cu o singura miscare.

Rezolvarea problemei se rezuma acum la gasirea unui drum intre nodul ce reprezinta starea initiala, respectiv nodul ce reprezinta starea finala(cubuln Rubik rezolvat). In functie de scopul nostru, putem aborda diferite metode:

- Putem parcurge graful printr-un BFS care porneste din starea initiala a grafului si se opreste cand ajunge in starea finala, astfel obtinand cel mai scurt drum(adica numarul minim de miscari necesare pt rezolvarea cubului)

  Observatie: Stim ca numarul de stari este unul foarte mare. O alta abordare ar fi sa pornim din starea finala si sa generam, pe parcurs, urmatorul set de stari pana dam de starea initiala, ignorand miscarile care ne duc inapoi. Mai exact, la primul pas generam starile cubului care necesita 1 miscare pentru a ajunge in starea finala, la al doilea pas starile care necesita 2 miscari si tot asa.

- In cazul in care consideram timpul necesar realizarii unei miscari, putem construi un graf cu costuri si calcula drumul de cost minim intre starea initiala si cea finala prin diferiti algoritmi precum Dijsktra sau Roy-Floyd. Vom obtine astfel timpul minim necesar pentru a rezolva cubul, insa, in mod evident, aceasta metoda nu ne asigura un numar minim de pasi.

Exercitiul 3

Consideram G' arborele obtinut prin rularea unui BFS pe graful G. Arborele G' va pastra distanta dintre nodul 1 si orice nod i din multimea nodurilor(din modul in care functioneaza efectiv algoritmul). Numarul de muchii al arborelui G' este $|E'| = |V| - 1$, fiind numarul minim de muchii din G necesare pentru a pastra distanta. (Daca scoatem fix 1 muchie, vom obtine 2 componente conexe, pierzand distantele.)

Astfel, putem scoate maxim $|E| - |E'| = |E| - |V| + 1$ muchii.

Exercitiul 4

Pentru rezolvarea problemei, pentru a si b, vom considera resturile impartirii unui numar la d: 0, 1, .. d-1. (Spre exemplu, pentru d = 3, cazul a = 3 si b = 4 este echivalent cu a = 0 si b = 1, nefiind necesar sa fie modelat in graf deoarece suma a+b va fi mereu mai mica in cazul a = 0 si b = 1). Pentru a gasi in cel mai scurt timp x si y care satisfac cerinta, vom consider pasul minim (adaugam sau scadem 1 pentru a ajunge dintr-un nod in altul).

Fie G = (V, E) definit astfel:

- V = { (a, b) | a, b ϵ { 0, 1, .. , d - 1} }
- E = { ((a1, b1), (a2, b2)) | (a1, b1), (a2, b2) in V, | a1 - a2 | + + | b1 - b2 | = 1 }
- Graful este unul neorientat, deoarece putem ajunge si din nodul v1 in v2, si din v2 in v1 fara a incalca definitia grafului,

Rezolvarea problemei se rezuma acum la pornirea unui BFS din (0, 0) si oprirea la prima pereche de x si y pentru care ax + by e divizibil cu d. Deoarece ne miscam cu minimul posibil la fiecare pas (1), solutia generata de BFS va fi cea mai mica pereche (x, y) pentru care ax + by e divizibil cu d, deci x + y va fi minim.

Exercitiul 5

Este necesar ca graful initial sa contina, pe langa muchiile alese de DFS, doar muchii de intoarcere, deci nu muchii care leaga subarbori diferiti(altfel DFS-ul nu ar mai fi acelasi). Astfel, vom alege toate muchiile de intoarcere si vom avea grija la ordinea nodurilor pentru a ne asigura ca pastram DFS-ul in forma actuala.

Listele de adiacenta vor fi:

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

Astfel, numarul maxim de muchii va fi 24.

Bonus: Pentru arbori arbitrari, aplicam acelasi procedeu, adica adaugam toate muchiile de intoarcere posibile(de la fiecare nod la fiecare stramos al sau).

Totusi, trebuie sa avem grija la ordinea procesarii muchiilor: primul nod procesat de un parinte este fiul sau din arborele DFS, iar apoi descendentii aflati mai jos in arbore (daca parintele proceseaza un descendent aflat mai jos care a fost procesat deja de parintele lui, DFS-ul nu se schimba).

Exercitiul 6

Arborele generat de BFS este identic cu cel generat DFS pentru un graf G daca si numai daca G este arbore. Din cerinta, G este neorientat si conex.

- Presupunem ca G poate fi un graf neorientat si conex dar care nu este arbore. Prin urmare, G trebuie sa contina cel putin 1 ciclu. Fie acest ciclu (v1, v2 .., vi, v1). Sa analizam cum se comporta acest ciclu in cele 2 parcurgeri:
  - BFS: In BFS, acest ciclu va aparea pe doua ramuri. Acest lucru se intampla deoarece BFS-ul se uita la toti vecinii. Atunci cand procesam primul nod din ciclu, BFS-ul va pune pe prima ramura primul vecin din ciclu, iar pe a doua ramura al doilea vecin din ciclu.
  - DFS: In DFS, acest ciclu va aparea pe o singura ramura. Acest lucru se intampla deoarece DFS-ul actioneaza in adancime, adica atunci cand procesam primul nod din ciclu, vom procesa apoi nod cu nod tot restul ciclului, fara a ne intoarce.

  Prin urmare, daca G nu este arbore, arborii determinati de parcurgeri vor fi diferiti.
- Presupunem ca graful initial este un arbore. Atat parcurgerea BFS, cat si cea DFS vor contine aceleasi noduri si aceleasi muchii, fiind, in realitate, acelasi arbore.

Exercitiul 7

Pentru rezolvarea problemei, vom descrie graful G = (V, E) astfel:

- Nodurile reprezinta toate intervalele date
- Muchiile se adauga intre nodul a si nodul b daca sunt valabile ambele conditii:
  - Intervalele pe care le reprezinta a si b nu se intersecteaza
  - Extremitatea din dreapta a intervalului reprezentat de b este mai mare decat extremitatea din dreapta a intervalului reprezentat de a (vom avea un fel de "ordine" intre intervale). De aici rezulta ca graful este orientat.
- Costul muchiei de la a la b este lungimea intervalului a.

Pentru a rezolva problema, vom face o sortare topologica, iar apoi vom folosi programarea dinamica pentru a tine minte drumul cu cel mai mare cost care se termina in fiecare nod.

Exercitiul 8

Presupunem ca graful G = (V, E) cu V = { 1, 2, .., n } este format doar din noduri izolate. Astfel, contine n componente conexe, fiecare avand cate 1 singur nod. Vor fi necesare urmatoarele interogari:

- Pentru nodul 1: Q(1, 2), Q(1, 3) ... Q(1, n) - n interogari
- Pentru nodul 2: Q(2, 3), Q(2, 4) .. . Q(2, n) - (n-1) interogari
  .
  .
  .
- Pentru nodul (n-1) : Q(n-1, n) - 1 interogare
- Pentru nodul n : nicio interogare

Astfel, pentru un graf cu n componente conexe si pentru orice algoritm, sunt necesare n(n-1)/2 interogari.

Exercitiul 9

Cel mai bun itinerariu se poate obtine printr-un ciclu hamiltonian de cost minim al grafului. Pentru rezolvarea problemei, vom descrie graful G = (V, E) astfel:

- V = { 1, 2, .. n} (cele n puncte de interes)
- E = { (i, j, d_{ij}) $\Big|$ $i, j \epsilon V$, $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ } (distanta euclideana dintre doua puncte de interes)

Fie A = (V, E') un APM al grafului G. Obtinerea APM-ului se poate face prin Algoritmul lui Prim in $O(n^2)$. Parcurgerea arborelui A incepand din nodul 1, si intorcandu-ne in tot in el are lungimea 2 * lungimea lui A, fie ca folosim BFS sau DFS (fiind arbore, am demonstrat deja ca arborele generat de parcurgerea BFS este acelasi cu cel generat de parcurgerea DFS).

Ne vom folosi de faptul ca intr-un graf cu ponderi pozitive, suma ponderilor unui ciclu hamiltonian al grafului este mai mare sau egala cu suma ponderilor unui APM al grafului.
Demonstratie: Presupunem ca exista un ciclu hamiltonian in graf astfel incat suma ponderilor este mai mica decat suma ponderilor oricarui APM al grafului. Orice muchie am elimina din ciclu, vom obtine un arbore partial cu suma

ponderilor mai mica decat cea a unui APM. *Fals.*

Astfel, suma ponderilor din parcurgerea APM-ului A va fi mai mica sau egala cu suma ponderilor dintr-un ciclu hamiltonian. In concluzie, parcurgerea APM-ului prin BFS sau DFS va genera un itinerariu cel mult de 2 ori mai mare ca distanta minima.