BFS

Ruxandra-Laura Nanu

January 2022

1 Introducere

De multe ori, cand lucram cu grafuri, avem nevoie de modalitati de a le parcurge. O modalitate de a parcurge un graf presupune trecerea prin fiecare nod exact o data intr-o ordine dorita. Un algoritm cu care putem parcurge un graf este BFS (Breadth First Search).

BFS-ul porneste dintr-un nod selectat (radacina sau sursa) si traverseaza graful "pe nivele", luand pe rand vecinii fiecarui nod de pe nivelul curent si apoi avansand la nivelul urmator.

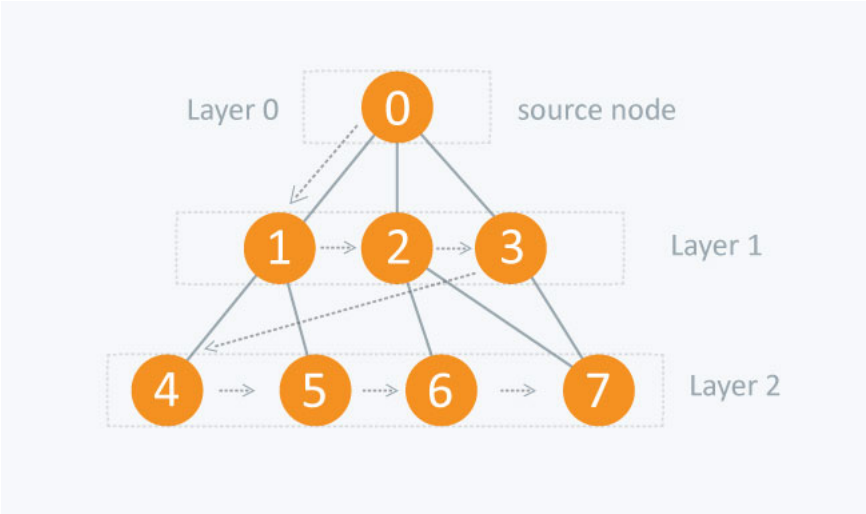
In alte cuvinte, nivelul x reprezinta multimea nodurilor la distanta $x - 1$ fata de nodul sursa. Nivelul 1 este multimea alcatuita din nodul sursa, nivelul 2 este cea alcatuita din vecinii nodului sursa etc.

2 O prima aplicatie

O sa rezolvam problema BFS - Parcurgere in latime de pe infoarena, arhiva educationala.
Problema este cel mai clasic exemplu de aplicatie al acestui algoritm.

In acest caz, nodul sursa se da in input. Cum aflam distanta intre nodul X si nodul S, unde S este nodul sursa? Am stabilit inainte ca nodul X se va afla pe nivelul y daca putem trece prin $y - 1$ muchii ca sa ajungem din X in S. In acelasi timp, stim ca nu vom trece printr-un nod de mai multe ori si ca vom parcurge nivelurile in ordine crescatoare.

Aceste trei observatii sunt suficiente pentru a afirma ca daca nodul X se afla pe nivelul y, atunci distanta minima intre X si S este $y - 1$. Pentru a face asta mai clar ne putem referi la urmatoarea diagrama:



Aici nodul 0 este nodul sursa. Nodul 3 va aparea mai intai pe nivelul 1, fiindca cel mai scurt drum intre 0 si 3 este de lungime 1. Desigur, putem ajunge in nodul 3 si pe drumul 0-2-3 care ar fi de lungime 2, dar fiindca deja am trecut prin nodul 3 atunci cand am evaluat nivelul 2 (traseul cu lungime 1), nu il vom mai reconsidera atunci cand gasim un potential drum mai lung.

Pentru a parcurge nivelurile in ordinea descrisa, o sa folosim un deque si vom adauga mereu la finalul sau nodurile vecine ale nodului de la inceput (desigur, pe care nu le-am mai vizitat, facem asta cu un vector de frecventa) si pentru fiecare nod vom tine si "nivelul" pe care l-am pozitionat. Un exemplu de implementare se afla **aici**.

3 Probleme usoare

3.1 Muzeu

Problema este aproape identica cu cea de pe arhiva educationala, diferenta consta doar in modul in care interpretam enuntul. Astfel, matricea din input poate fi vazuta ca un graf: fiecare celula reprezinta un nod, si o muchie exista intre doua noduri daca putem sa ne deplasam din camera reprezentata de primul in cea reprezentata de al doilea. De exemplu, exista o muchie intre nodul ce reprezinta celula (i,j) si cel ce reprezinta (i, j+1) daca ambele celule sunt in matrice si daca niciuna nu este blocata.

Odata ce am construit graful, observam ca nu are sens sa tratam fiecare paznic separat, ci putem sa consideram ca avem mai multe noduri sursa, fiecare reprezentat de un paznic. Astfel, daca o camera este prima oara pusa pe nivelul x, inseamna ca distanta minima dintre ea si un paznic oarecare este $x - 1$, deci nu ne va mai interesa distanta pana la niciun alt paznic.

Implementarea mea este **aici**.

3.2 Graf

Pentru inceput, o sa definim ce inseamna ca un nod Z sa se afle pe un drum de lungime minima intre nodurile X si Y. Notam dist(a , b) = distanta minima ca numar de muchii intre nodurile a si b. Astfel, Z se afla pe un drum de lungime minima intre X si Y daca si numai daca $dist(X, Y) = dist(X, Z) + dist(Y, Z)$

Astfel, vom incepe prin a face doua BFS-uri, unul calculand distantele $dist(X, Z)$ si unul calculand $dist(Y, Z)$, pentru toate nodurile Z.

Mai ramane un caz de tratat: daca exista doua noduri distincte Z1 si Z2, astfel incat $dist(X, Y) = dist(X, Z1) + dist(Y, Z1)$ si
$dist(X, Y) = dist(X, Z2) + dist(Y, Z2)$, deci ambele sunt pe drumuri minime, dar $dist(X, Z1) = dist(X, Z2)$, atunci inseamna ca drumurile dintre X si Y ce trec prin Z1, respectiv Z2 sunt distincte, deci nici Z1 si nici Z2 nu respecta cerinta si nu trebuie afisate.

Exemplu de implementare aici

4 Probleme medii

4.1 Sate

Vom incepe prin a interpreta relatiile intre noduri ca pe un graf. Vom trage o muchie intre nodurile A si B cu "costul" C, daca pentru a ajunge din A in B trebuie sa mergem C kilometri la dreapta (C este negativ daca B se afla la stanga lui A).

Odata ce avem acest graf, putem face un BFS din nodul X dat, incercand sa ajungem in Y folosind muchiile date. De data asta, in loc sa ne intereseze "nivelul" pe care se afla un nod, ne intereseaza suma costurilor muchiilor pe care le-am parcurs pana la el.

Atentie: doar pentru ca in problema asta folosim un graf cu "costuri" pe muchii, nu inseamna ca putem folosi BFS-ul pentru a afla distanta dintre doua noduri intr-un graf cu costuri pe muchii in mod normal. Problema este un caz special pentru ca distanta dintre doua sate este definita clar dinainte, nu exista doua drumuri de lungimi diferite intre doua sate.

Exemplu de implementare aici

4.2 Reinvent

Problema seamana cu problema Muzeu, insa de data asta, desi este clar ca avem mai multe noduri sursa, destinatiile in care vrem sa ajungem apartin tot multimii de noduri sursa.

O sa folosim o strategie usor similara cu cea de la problema Muzeu. O sa facem o parcurgere BFS in care vom incepe cu mai multe noduri sursa (nodurile din setul de X date). Totusi, de data asta nu este suficient ca in deque sa tinem doar nodurile, ci trebuie sa tinem minte si "radacina" fiecarui nod.

Definim "radacina" unui nod = nodul sursa din care porneste drumul (de lungime minima) pe care am ajuns la nodul curent = cel mai apropiat nod de nodul curent din cele X date

Cum ne ajuta informatiile astea? Ne dam seama ca in momentul in care un nod, fie el numit A cu radacina R1, ar ajunge "candidat" sa fie bagat in deque, dar el a aparut anterior in deque cu radacina R2, nodurile R1 si R2 sunt cele mai apropiate din setul de X.

Acest lucru este adevarat din cauza modului "pe nivele" in care facem parcurgerea. Daca alte doua radacini R3 si R4 ar fi fost mai apropiate, nodul AUX pentru care $dist(R3,\text{AUX}) + dist(R4,\text{AUX}) <= dist(R1, A) + dist(R2, A)$ ar fi fost gasit mai devreme.

Exemplu de implementare aici

5 Problema mai dificila

5.1 Amici2

Incepem prin a crea un graf pe baza retelei definite in enunt. La fiecare pas, deci, apar muchii intre nodurile aflate la distanta 2 si se cere numarul de zile pana cand graful devine complet.

Dupa ziua 1, nodurile aflate la distanta 2 in graful initial devin conectate. Dupa ziua 2, nodurile aflate la distanta 2 in graful nou devin conectate. Deci, nodurile care devin conectate in ziua 2 sunt cele care in graful initial erau la o distanta <= 4. Urmarind pattern-ul, vedem ca dupa ziua k, exista muchie intre oricare doua noduri care, in graful initial, erau la o distanta $<= 2^k$

Intuitiv ne dam seama ca nodurile care vor fi "legate" ultimele sunt cele doua noduri A si B care se aflau la cea mai mare distanta unul de celalat in graful initial. $dist(A, B)$ se numeste si **diametrul grafului**, sa notam aceasta valoare cu D.

Solutia noastra este cel mai mic k astfel incat $D <= 2^k$

Totusi, ca sa calculam diametrul unui graf, ar trebui sa facem n BFS-uri (n = numarul de noduri din graf) ceea ce ar fi neoptim. Totusi, remarcam ca nu trebuie sa afisam rezultatul exact, si ca raspunsul nostru poate fi cu 1 mai mic sau mai mare decat cel corect, deci putem incerca alta abordare pentru a aproxima D

Fie X prima zi in care un nod A are muchie directa cu toate celelalte noduri. Este evident ca dupa acest punct, procesul mai poate dura o singura zi in plus, deoarece orice pereche de noduri neadiacente va folosi nodul A pentru a adauga muchiile necesare. Raspunsul real al problemei este deci fie X, fie X + 1.

De aceea, ii vom da lui A o valoare aleatoare, vom gasi cel mai departat nod de A, fie el A' si vom aproxima $D = dist(A, A') * 2$ si vom incerca sa gasim valoarea k dorita apoi.

Exemplu de implementare aici