Assignment

Filip Alina-Andreea, CR 1.2.A

May 19, 2019

1 Cerinta

O fortareata are multiple tipuri de turnuri: atac si aparare. Intre fiecare pereche de turnuri s-ar putea sa fie sau sa nu fie un tunel. Presupunem ca avem n turnuri si avem o lista de r perechi de turnuri intre care sunt tuneluri. Implementeaza un algoritm care determina daca este posibil sa asignam unele turnuri ca atac si cele ramase ca aparare astfel incat fiecare tunel este intre un turn de aparare si unul de atac. Daca este posibil algoritmul ar trebui sa fie capabil sa il produca.

Din punctul meu de vedere problema ne da n turnuri si r tunele iar algoritmul ar trebui sa produca graful pentru care se pot forma r tunele.

2 Algoritmi

Pentru realizarea acestui program am utilizat mai multi algoritmi impartiti in mai multe coduri sursa.

2.1 Algoritmul de setare a valorii in matrice

set-value(graph,i,j,value)

1. position1 = ri* nodes + j;
2. position2 = j*nodes+i;
3. *(mat+ position1) = value;
4. *(mat+ position2) = value;

Algoritmul trece in matrice valoarea pe care dorim sa o adaugam (1 sau 0) pe pozitia calculata si pe pozitia simetrica celei calculate. Deoarece am ales sa lucrez cu un graf neorientat matricea lui va fi simetrica fata de diagonala principala. Urmeaza urmatorii pasi:

- Calculeaza pozitia 1.
- Calculeaza pozitia 2 (simetrica).
- Ii confera matricii valoarea data la pozitiile calculate.

Acesta este un algoritm implementat dupa modelul celui predat la laborator.

2.2 Algoritmul de initializare al grafului

INIT(G, n, m , r)

1. nodes=n;
2. init = 1;
3. for (i = 0; i ¡ nodes; i++)
4. for (j = i; j ¡ nodes; j++)
5. if(i¡=m-1 and j¿=m and r!=0)
6. aux=1;
7. set-value(g, i, j, aux);
8. r–;
9. else
10. aux=0;
11. set-value(g, i, j, aux);

Algoritmul seteaza valoarea 1 pentru matrice daca exista o legatura intre nodul i si nodul j pana cand se obtine numarul de legaturi cerute si 0 daca nu mai este nevoie de noduri. El functioneaza astfel:

- Forurile iau in considerare dor jumatatea de deasupra diagonalei principale.
- Daca este mai mic sau egal cu numarul de turnuri de atac minus 1 si j este mai mare decat numarul de noduri de atac si inca nu s-au creat toate tunelurile cerute, atunci auxiliarul primeste valoarea 1 si se scade un tunel.
- Daca nu aux primeste valoarea 0 si nu se scade nici un tunel.

Am pus conditia ca i sa mearga pana la m-1 pentru ca porneste de la 0 si daca avem m=2 turnuri de atac atunci doar pe liniile 0 si 1 se vor forma legaturi. J ia valoare mai mare decat m deoarece el reprezinta restul de noduri de aparare ale grafului deci cu ele se vor lega nodurile i. Complexitatea algoritmului este:

$$T(n) = O(n^2)$$

In cel mai bun caz complexitatea poate fi O(1) cand graful nu are nici un nod.

2.3 Algoritmul de printare a legaturilor intre noduri

PRINT(G, atac, aparare)

1. if (init == 1)
2. for(i=0;i¡nodes;i++)
3. for (j = 0;j¡ nodes;j++)
4. aux = get-value(G,i,j)
5. if(aux==1)
6. print j+1;

Algoritmul afiseaza legaturile fiecarui nod cu celelalte noduri. Algoritmul urmeaza urmatorii pasi:

- Daca graful a fost initializat.
- Iteratorul liniilor merge de la 0 la n-1.
- Iteratorul coloanei merge de la 0 la n-1.
- aux primeste valoarea returnata de apelul functiei get-value
- Daca aux este egal cu 1 atunci se scrie indicele coloanei adunata cu 1 si el reprezinta nodul cu care este legat nodul i.

Am modificat algoritmul sa imi arate doar pozitiile pe coloana la care se afla 1 pentru fiecare linie in parte. Initial algoritmul afisa matricea pur si simplu.

Complexitatea algoritmului este :

$$T(n) = O(n^2)$$

In cel mai bun caz, complexitatea algoritmului este O(1) cand graful nu are nici un nod.

2.4 Algoritmul de scoatere a valorii din matrice

get-value(G, i, j)

1. if (init == 1)
2. p = i * nodes + j;
3. return *(mat + p)

Acesta este singuril algoritm pe care nu l-am modificat si l-am luat exact cum ne-a fost dat la laborator. Functioneaza astfel:

Dca graful a fost initializat atunci calculeaza pozitia la care se gaseste valoarea ceruta. In memorie matricea se gaseste ca un sir de pointeri dispusi ca intr-un vector de aceea in functia apelata valoarea este ceruta ca si cand ar face parte dintr-o matrice apoi este nevoie sa i se calculeze pozitia in vector.

2.5 Algoritmul principal

MAIN()

1. max=-5;
2. for(m=n/2;m¿=1;m–)
3. p=n-m
4. nr-max-tunele=m*p;
5. if(nr-max-tunele¿=r)
6. nr-tatac=m;
7. nr-taparare=p;
8. max=nr-tatac*nr-taparare;
9. if(max¿=r)
10. INIT(g,n,nr-tatac,r);
11. print(gr, i,nr-tatac, nr-taparare);
12. return 0;

Am ales sa scriu doar codul care este mai important din tot cel existent in program si codul care are nevoie de explicatii.

Max primeste valoarea -5 pentru ca in cazul in care nu se intra in if-ul din for sa nu se continue cu initializarea si printarea matricei.

For-ul merge de la n/2 la 1 deoarece la n/2 de turnuri de aparare si de atac se obtine numarul maxim de tunele. Eu am dorit ca numarul de tunele de atac sa fie mai mic decat cel de aparare insa pot schimba asta usor. Oricum , for-ul are o logica inversa fata de cum am considerat in calculele facute pe foaie.

Practic cat timp numarul maxim de tunele scade dar este totusi mai mare decat tunelele cerute se scade numarul de tunele de atac si se creste cel de aparare pana la intalnirea unei combinatii care este mai mica decat tunelele cerute.Astfel la iesirea din for avem un numar minim de turnuri de atac, restul turnurilor fiind de aparare.

Am introdus aldoilea daca pentru a preveni executarea urmatoarelor instructiuni daca nu s-a indeplicit conditia sa existe macar o combinatie de turnuri mai mare decat r. In codul sursa se mai regaseste un alt if pentru ce se intampla atunci cand nu se indeplineste conditie pentru a evita orice actiune inutila.

3 Date experimentale

Datele experimentale au fost create cu uneui program ce scrie date random in 10 fisiere sursa ce sunt folosite la inceputul programului ca date de intrare.

3.1 Algoritmul de randomizare a datelor de intrare

Acest algoritm se gaseste in alt program pus in fisierul cu codurile sursa.

1. for(j=1;j¡=10;j++)
2. sprintf(temp-name-in, "teste test-procentd.in", j+1);
3. FILE* test-in = fopen(temp-name-in, "wt");
4. n=rand();
5. r=rand();
6. if(n mai mic = 0 —— n ==1)
7. n=rand();
8. if(r mai mic=0)
9. k=rand()
10. fprintf(test-in, " procentd", n);
11. fprintf(testi-n, "procentd", r);
12. fclose(test-in);

Daca numarul de noduri este egal cu 0 sau 1 atunci nu este posibil sa cream nici o legatura intre noduri deci se alege alta valoare. Daca k este mai mic decat 0 sau egal cu 0 atunci nu se cere defapt sa se creeze nici un tunel. Fisierele se regasesc impreuna cu codul sursa intr-un fisier numit teste. Nr de noduri primeste de 10 ori valori random pana la 100, fiecare valoare fiind scrisa in fisierul sursa. Se creaza 10 fisiere sursa. Teste-in are rol de identificator al fisierului in program. Fopen deschide fisierul iar wt inseamna ca in program se poate doar scrie in fisier iar daca fisierul nu este creat atunci se creaza in momentul deschiderii.

4 Rezultate si concluzii

Nu am reusit sa fac un program cu complexitatea O(n+r) deoarece nu am reusit niciodata sa inteleg grafurile si nu am avut niciodata sansa sa lucrez asa mult cu ele.

M-am gandit ca trebuie sa reprezint graful cu ajutorul listelor de adiacenta dar oriunde am cautat nu am reusit sa ma lamuresc cum se face. Am gasit pe un site ca se poate obtine lista de adiacenta din matricea de adiacenta insa nici asta nu am reusit sa fac si cred ca nu avea nici un rost.

Acum cred ca am inteles gresit si problema caci era vorba de o lista de r perechi de turnuri iar eu am considerat ca ni se dau tunelurile ce trebuie create.

Am lucrat mult la determinarea numarului de turnuri astfel incat sa iese asa cum am vrut.

In concluzie, problema asta m-a invatat foarte multe si m-a facut sa inteleg unele aspecte legate de grafuri.