Assignment2

-problem2-

Student: Wojcicki Andrei-Cristian

Grupa: 1.3B

Specializare: CR

Enuntul problemei:

O fortareata are multiple tipuri de turnuri: atac si aparare. Intre orice pereche de turnuri poate sa fie sau poate nu un tunel. Presupunem ca avem n turnuri si r perechi de turnuri intre care exista tunele. Implementati un algoritm O(n+r) care sa determine daca este posibil sa se atribuie unor turnuri ca fiind de atac si celor ramase ca fiind de aparare astfel incat fiecare tunel sa fie intre un turn de atac si unul de aparare. Daca este posibil, algoritmul ar trebui sa o faca.

Pseudocodul:

SET\_ADJ\_MATRIX\_VALUE

Position = row\_index \* graph->no\_nodes + column\_index

Element\_value = graph->adj\_matrix + position

GET\_ADJ\_MATRIX\_VALUE

If graph->init == 1

Position = row\_index \* graph->no\_nodes + column\_index

Returneaza graph->adj\_matrix + position

Else

Returenaza -1;

INIT\_GRAPH\_FILE

If file == NIL

Inchide fisierul

Graph->init =1

For i=0, no\_tunnles do

Citeste aux1, aux2

Set\_adj\_matrix\_value (graph, aux1, aux2, 1)

Set\_adj\_matrix\_value (graph, aux2, aux1, 1)

PUSH\_BEGIN\_LIST

New\_element->info = new\_element\_value

New\_element->next = head->next

Head->next = new\_element

POP\_END\_LIST

If head->next != NIL

For iterator = head, iterator->next->next != NIL, iterator = iterator->next

Deleted\_element = iterator->next

Aux = deleted\_element->info

Iterator->next = deleted\_element->next

Returneaza aux

Else

Returneaza -1

POZZERO

For iterator=1, number do

If vector[iterator] == 0

Returneaza iterator

GRAPH\_BFS

Count = 0

K= 2

Check = 0

Head\_queue->next = NIL

Push\_begin\_list (head\_queue, start\_node)

Visited[start\_node] = 1

Vector[start\_node] = 1

While head\_queue->next != NIL

Current\_node = pop\_end\_list (head\_queue)

Count= count +1

For column\_index =0, graph->no\_nodes do

Aux = get\_adj\_matrix\_value (graph, current\_node, column\_index)

If aux !=0 && visited[column\_index] == 0

Push\_begin\_list (head\_queue, column\_index)

Visited[column\_index] = 1

Vector[ column\_index] = k

Check =1

If check == 1

K=k+1

Check = 0

Ok = 1

If graph->init == 1

If count == 1

Ok = 1

Else

For iterator1=1, graph->no\_nodes do

For iterator2=1, graph->no\_nodes, do

Aux = get\_adj\_matrix\_value (graph, iterator1, iterator2)

If (aux == 1 && vector[iterator1] == vector [iterator2] && vector[iterator1] != 0 && vector[iterator2] !=0

Ok=0

If ok != 1

Done = done + 1

MAIN

Citeste start\_node

Genereaza

Graph\_bfs(grap, start\_node)

While PozZero(graph->no\_nodes-1) != 0

Graph\_bfs (graph, PozZero (graph->no\_nodes))

If done !=0

Afiseaza in fisier ’NU’

Altfel

Afiseaza in fisier ’DA’

Descrierea programului :

Programul genereaza automat un numar random de turnuri si tunele. Se creeaza un graf care apoi se parcurge in latime. Principiul de functionare al algoritumului este urmatorul: daca un nod este un turn de atac, automat vecinii lui trebuie sa fie de aparare sau invers. Acest lucru este testat cu ajutorul unui vector. Mai exact, cand primul nod este pus in coada, automat in vector pe pozitia corespunzatoare primului este pusa valoare 1. Se sterge din coada si apoi sunt pusi si vecinii lui, iar in vector, se pune valoarea 2 pe pozitiile corespunzatoare nodurilor. (Daca nodul de inceput este 3 si are 2 vecini, nodul 4 si nodul 1 vectorul va fi 2 0 1 2). Apoi se sterge primul vecin sunt pusi vecinii acestuia. Din nou, in vector se va pune valoare 3 pe pozitiile corespunzatoare vecinilor (Daca nodul 4 are un singur vecin si anume nodul 5, vectorul va fi 2 0 1 2 3). Algoritmul se repeta pana cand fiecare nod va fi vizitat. Apoi se parcurge matricea de adiacenta si se testeaza daca apare valoarea 1 ( asta inseamnand ca este o muchie) si apoi daca in vector valoarea de pe pozitia indexului liniei este egala cu valoarea de pe pozitia indexului coloanei si daca aceste valori sunt diferite de 0 ( adica fac parte din aceeasi componenta conexa).

Daca este adevarat inseamna ca exista un tunel intre 2 turnuri de acelasi fel si variabila ok va deveni 0 ( semn ca nu se respecta conditita).

Vectorul fiind declarat global, este initializat automat cu 0. Daca apare valoarea 0 la un moment dat inseamna ca exista si alte componente conexe, iar atunci graful se va parcurge de la nodul corespunzator pozitiei 0 in vector. (De exemplu daca in vector apare pe pozitia 14 numarul 0, atunci graful se va parcurge de la nodul 14 si se va executa tot algoritmul de mai sus pentru toate nodurile din componenta conexa din care face si acesta parte).

Algoritmul genereaza automat un numar random de turnuri (intre 1 si 200000) si un numar random de tunele (intre 1 si 2\*turnuri). (2\*n este numarul maxim de muchii, in cazul in care graful este complet).

Pseudocod pentru generare:

Towers = random () % 200000

Tunnles = randmom () % (2\*towers)

Afiseaza in fisier towers si tunnles

For iterator = 1, tunnels do

X = random () % towers

Y = random () % towers

If x!=y

Afiseara in fisier x, y

Else

Iterator = iterator – 1

Unde x si y reprezinta turnurile intre care exita un tunel .

Observatii:

* Timpul de executie se modifica in functie de numarul de componente conexe.