Course assignment

Student : Wojcicki Andrei Cristian

June 2, 2019

Grupa: 1.3B
Specializare: CR

Enunt (Problema):

Sa se implementeze doi algoritmi difieriti care sa determine componentelele conexe dintr-un graf neorientat.

Descrierea programului:

Cele doua implementari diferite sunt BFS si DFS. Algoritmul genereaza automat un numar random de noduri si de muchii. Principiul de functionare este urmatorul: se parcurge graful (in prima varianta in latime, iar in a 2-a varianta in adancime) si se ia un vector separat. Cand un nod este vizitat, in vector, pe pozitia corespunzatoare nodului se va pune cifra 1. (De exemplu daca sunt vizitate nodurile 1 2 si 4 atunci vectorul va fi 1 1 0 1). Apoi se testeaza daca apare cifra 0 in acel vector, si se reia algoritmul plecand de la nodul corespunzator pozitiei in vector unde a aparut. Acest lucru semnaleaza gasirea unei componente conexe. Se afiseaza nodurile pe masura ce sunt vizitate.

Algoritmul genereaza automat un numar random de noduri (intre 1 si 200000) si un numar random de muchii(intre 1 si noduri*(noduri-1)/2). (noduri*(noduri-1)/2 este numarul maxim de muchii, in cazul in care graful este complet).

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).

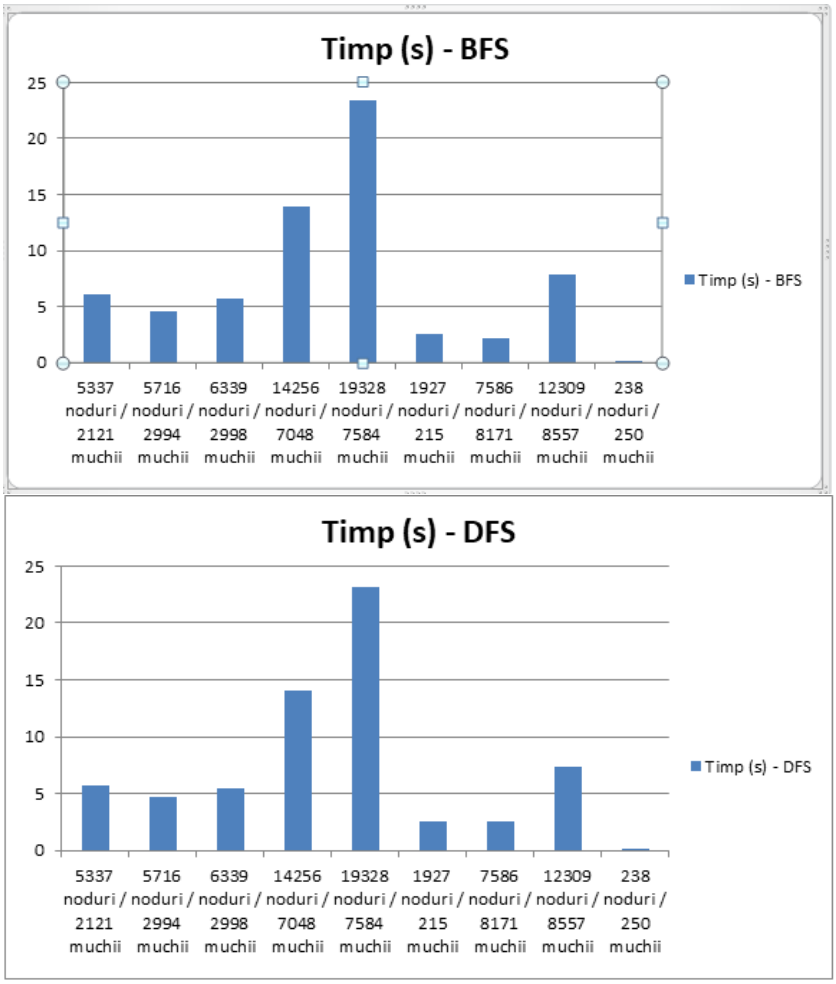
Pseudocod pentru functia GENERATE

```
noduri = rand ()
muchii = rand ()
scrie noduri, muchii
for iterator = 1, muchii, do
    x = rand ()
    y = rand ()
    if x != y
        scrie x, y
    else
        iterator –
```

Pseudocod (pentru prima varianta - BFS) :
SET ADJ MATRIX VALUE
position = row.index * no.nodes + column.index
*(adj.matrix + position) = element.value
GET ADJ MATRIX VALUE
if (graph.init == 1)
    position = row.index * no.nodes + column.index return *(adj.matrix+position)
else
    return -1
INIT GRAPH FILE if (file.in == NULL)
    scrie "nu exista fisierul"
    iesi
citeste in fisier nr de noduri si nr de muchii
graph.init = 1
graph.adj.matrix = calloc (graph.no.nodes * graph.no.nodes, sizeof (int))
for iterator.row = 0, muchii do
    citeste aux si temp in fisier
    SET ADJ MATRIX (graph, aux, temp, 1)
    SET ADJ MATRIX (graph, temp, aux, 1)
POP END LIST
if head.next != NULL
for iterator = head, iterator.next.next != NULL, do
    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
returneaza 0
PUSH BEGINING LIST
new.element.info = new.element.value
new.element.next = head.next
head.next = new.element

```
GRAPH BFS
head.queue.next = NULL
PUSH BEGINING LIST (head.queue, start.node)
visited[start.node] = 1
vector [start.node] = 1
while head.queue.next !=NULL
    current.node = POP END LIST (head.queue)
    scrie current.node
    for iterator = 0, graph.no.nodes
        GET ADJ MATRIX (graph, current.node, column.index)
        if aux != 0 si visited[column.index] == 0
            PUSH BEGINING LIST (head.queue, column.index)
            visited [column.index] = 1
            vector [column.index] = 1
MAIN
GENERATE ()
INIT GRAPH FILE (graph)
GRAPH BFS (graph, 0)
While POZZERO (graph.no.nodes-1) != 0
    GRAPH BFS (graph,POZZERO(graph.no.nodes))
Pseudocod (pentru a 2-a varianta - DFS)
MAIN
GENERATE ()
INIT GRAPH FILE (graph)
GRAPH DFS (graph, 0)
While POZZERO (graph.no.nodes-1) != 0
    GRAPH DFS (graph,POZZERO(graph.no.nodes))
SET ADJ MATRIX (identic cu cel de mai sus)
GET ADJ MATRIX (identic cu cel de mai sus)
INIT GRAPH FILE (identic cu cel de mai sus)
PUSH BEGINING LIST (identic cu cel de mai sus)
POP BEGINING LIST
if head.next != NULL
    deleted.element = head.next
    aux = deleted.element.in
    head.next = deleted.element.next
    returneaza aux
else
    afiseaza "lista este goala"
    returneaza -1
POZZERO (identic cu cel de mai sus)
```

```
GRAPH DFS
head.stack.next = NULL
PUSH BEGINING LIST(head stack, start node)
visited[start.node] = 1
vector[start.node] = 1
for column.index = 0, graph.no.nodes, do
    aux = GET ADJ MATRIX VALUE (graph, current.node, column.index)
    if aux !=0 si visited[column.index] == 0
        PUSH BEGINING LIST (head.stack, column.index)
        visited[column.index] = 1
        vector[column.index] = 1
```



Observatii: Ambele implementari au aproximativ acelasi timp de executie.