Recherche Operationnelle ISIMA - ZZ2F4

TP3 - Depth First Search and Dijkstra

Renaud Chicoisne *2 février 2023*

Étant donné un graphe $G = (V, E)$ l'objectif de ce TP est d'implémenter en C:

1. l'algorithme DFS de recherche en profondeur (c.f. Algorithme 1), ou la routine explore(i) est définie

Algorithm 1: DFS

Data: A graph $G = (V, E)$

Result: An arborescence father$_i$ with dates $[l_i, u_i]$ for each $i \in V$

```
1 t ← 0;
2 for i ∈ V do
3     visited_i ← 0;
4     father_i ← −1;
5 for i ∈ V do
6     if visited_i = 0 then
7         explore(i);
8 return father_i and [l_i, u_i] for each i ∈ V;
```

dans l'Algorithme 2.

Algorithm 2: Explore

Data: A node $i \in V$

Result: Modifies t, father$_i$, visited$_i$ and $[l_i, u_i]$ for each $i \in V$

```
1 visited_i ← 1;
2 l_i ← ++ t;
3 for (i, j) ∈ E do
4     if visited_j = 0 then
5         visited_j ← 1;
6         father_j ← i;
7         explore(j);
8 visited_i ← 2;
9 u_i ← ++ t;
```

2. l'algorithme de Dijkstra de plus courts chemins (c.f. Algorithme 3).

Algorithm 3: Dijkstra

Data: A graph $G = (V, E)$ with edge costs $(c_e)_{e \in E}$ and a source node $s \in V$

Result: An s-shortest paths tree defined by father_i for each $i \in V | \{s\}$

```
for i ∈ V do
    father_i ← −1;
    d_i ← +∞;
father_s ← s;
d_s ← 0;
S ← {s};
while S ≠ ∅ do
    Select i ∈ S such that d_i = min_{k∈S} d_k;
    S ← S|{i};
    for (i, j) ∈ E do
        if j ∉ S then
            S ← S ∪ {j};
        if d_j > d_i + c_ij then
            d_j = d_i + c_ij;
            father_j ← i;
```

Votre code source devra comporter deux fichiers principaux: `TP3.c` et `TP3Functions.c` que vous compilerez avec la commande `gcc TP3.c -o TP3`, ce qui génèrera un fichier éxécutable que vous éxécuterez avec la commande `./TP3`.

Votre code devra lire un fichier d'instance `TP3instance.csv` qui contient toutes les informations du graphe considéré sous le format suivant:

1. La première ligne `n,m` contient le nombre de noeuds $n = |V|$ et d'arètes $m = |E|$
2. Chacune des m lignes suivantes `i,j,c` contient l'information d'une arète $(i, j) \in E$ et son poids $c_{ij} =$`c`

Votre code devra retourner les informations relatives à

1. l'arbre de DFS trouvé dans un fichier `TP3DFS.csv` contenant en premiere ligne le nombre de noeuds n et chacune des n lignes suivantes $i, \text{father}_i, l_i, u_i$.
2. l'arbre de plus courts chemins trouvé dans un fichier `TP3dijkstra.csv` contenant en première ligne le nombre de noeuds n et chacune des n lignes suivantes i, father_i, d_i.

Testez vos implémentations en partant du noeud 0 sur le réseau donné en instance sur l'espace de cours.