Rapport TP3 graphes et algorithmes

I. Dijkstra dans le métro

Algorithm 1: Dijkstra

Data: E,Γ, $l, n = |E|, x \in E$
Result: Lx
$\overline{S} = \emptyset$;
foreach $y \in E$ **do**
  $Lx[y] = \infty$;
  $\overline{S} = \overline{S} \cup \{y\}$;
end
$Lx[x] = 0; k = 0; \mu = 0;$
while $k < n$ & $\mu \neq \infty$ **do**
  $mini = \infty$; y*=-1 ;
  foreach $y \in E$ **do**
    if $Lx[y] < mini$ & $y \in \overline{S}$ **then**
      y*=y ;
      mini=Lx[y] ;
  end
  $\overline{S} = \overline{S} \setminus \{y^*\}; k++; \mu = Lx[y^*]$;
  foreach $y \in \Gamma(y^*) \cap \overline{S}$ **do**
    Lx[y]=min{Lx[y], Lx[y*] + l(y*, y)} ;
  end
end

II. Chemin le plus court

Pour le chemin le plus court, il faut reprendre l'algorithme de Dijkstra et noter dans un tableau d'entier le chemin emprunté. Puis, dès qu'on rencontre le sommet d'arrivé en y*, on peut remonter ce tableau pour obtenir le chemin.

Algorithm 2: shortest path

```
Data: E,Γ, l, n = |E|, x ∈ E, a ∈ E
Result: Γ_SP
S̄ = Ø ;
foreach y ∈ E do
    Lx[y] = ∞ ;
    S̄ = S̄ ∪ {y} ;
end
Lx[x] = 0; k = 0; μ = 0;
while k < n & μ ≠ ∞ do
    mini = ∞; y*=-1 ;
    foreach y ∈ E do
        if Lx[y] < mini & y ∈ S̄ then
            y*=y ;
            mini=Lx[y] ;
    end
    if y*=a then
        while y* ≠ x do
            ajouter un arc entre predes[y*] et y* dans Γ_SP ;
            y*=predes[y*] ;
        end
        return Γ_SP;
    S̄ = S̄ \{y*}; k++; μ = Lx[y*] ;
    foreach y ∈ Γ(y*)∩S̄ do
        if Lx[y] > Lx[y*] + l(y*, y) then
            Lx[y]=Lx[y*] + l(y*, y) ;
            predes[y] = y* ;
    end
end
```

III. Amélioration de Dijkstra

Algorithm 3: DijkstraBetter

```
Data: E,Γ, l, n = |E|, x ∈ E
Result: Lx
S̄ = Ø; T = Ø ;
foreach y ∈ E do
    Lx[y] = ∞ ;
    S̄ = S̄ ∪ {y} ;
end
Lx[x] = 0; k = 0; μ = 0; T = T ∪ {x};
while k < n & μ ≠ ∞ do
    mini = Lx[T0];y*=T0 ;
    foreach y ∈ T \{T0} do
        if Lx[y] < mini & y ∈ S̄ then
            y*=y ;
            mini=Lx[y] ;
    end
    S̄ = S̄ \{y*};T=T\{y*}; k++; μ = Lx[y*] ;
    foreach y ∈ Γ(y*)∩S̄ do
        Lx[y]=min{Lx[y], Lx[y*] + l(y*, y)} ;
        T = T ∪ {y};
    end
end
```

Temps de traitement pour Dijkstra : 0.000391 secondes

Temps de traitement pour DijkstraBetter : 0.000146 secondes

En passant par une liste T pour les sommets de valeurs finis de S\, on est 2,5 fois plus rapide environ.

IV. Bonus

On peut modifier le réseau et ajouter 20 secondes supplémentaires à chaque temps de trajet dans le fichier metro_complet.graph à l'aide d'un algorithme qui lirait et écrirait dans un autre fichier les lignes du premier jusqu'à ce qu'il arrive à la ligne « arcs values » où alors il doit ajouter 20 à toutes les valeurs jusqu'à arriver à une ligne ou la valeur est 120,0 car ce ne sont pas des trajets en métro.

On peut modifier l'algorithme de Dijkstra pour qu'il ajoute 20 secondes à Lx[y] à chaque fois qu'on rentre dans la dernière boucle de l'algorithme et que l'on doit changer la valeur de Lx[y] sauf si cette valeur vaut 120,0.