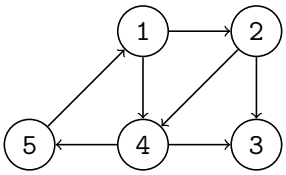
TD 5-6 : Graphes

18 octobre 2018

1 Algorithme de Kosaraju

Question 1 – Appliquer l'algorithme de Kosaraju au graphe ci-dessous.



2 Algorithme de Dijkstra

Question 2 – Exécuter l'algorithme de Dijkstra sur le graphe $\mathbb{G}_1$ à partir du sommet C. On pourra représenter les différentes étapes de l'algorithme avec un tableau montrant l'évolution de la distance estimée pour chaque nœud.

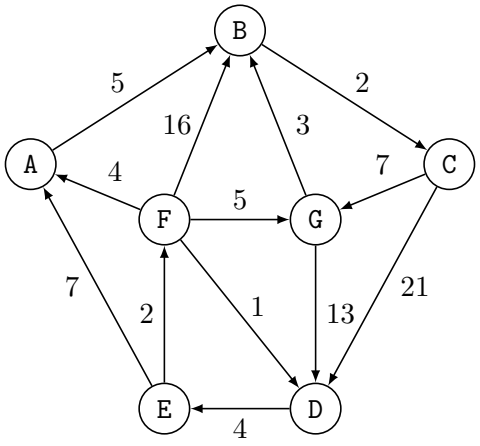


Figure 1 – Graphe exemple $\mathbb{G}_1$

QUESTION 3 – On modifie $\mathbb{G}_1$: l'arc (E,A) est maintenant valué -7 au lieu de 7. Exécuter de nouveau l'algorithme de Dijkstra à partir du sommet F. La correction de l'algorithme est-elle conservée ?

Soit $G = (S, A)$ un graphe et s un sommet de G. Nous considérons $pred$ le tableau des prédécesseurs calculé durant l'éxécution de l'algorithme de Dijkstra. Soit $G_{pred} = (S_{pred}, A_{pred})$ le *sous-graphe des prédécesseurs* de G tel que :

$$S_{pred} = \{v \in S : pred(v) \neq []\} \cup \{s\}$$

et

$$A_{pred} = \{(pred(v), v) \in A : v \in S_{pred} \setminus \{s\}\}$$

QUESTION 4 – Montrer que G_{pred} est un arbre.

3 Algorithme A*

L'algorithme A* est une modification de l'algorithme de Dijkstra utile quand on cherche un plus court chemin entre deux sommets donnés. Cet algorithme suppose qu'il existe une heuristique, h, telle que pour tout couple de sommets i, j, $h(i, j)$ renvoie une estimation de la plus courte distance entre i et j. Cette estimation peut être plus ou moins précise, mais en aucun cas elle ne doit surestimer la distance : $h(i, j) \leq \delta(i, j)$.

Parmi les différences avec l'algorithme de Dijkstra, on peut noter :

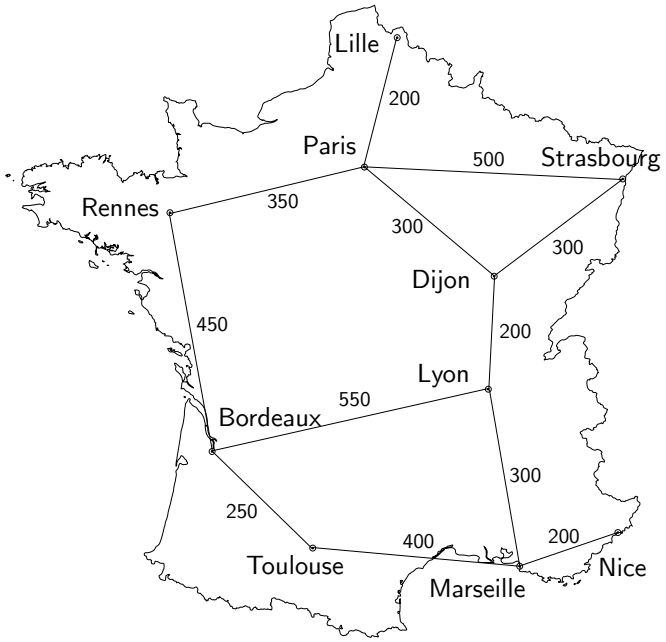
— la file de priorité F est indexée par $D[\cdot] + h(\cdot, t)$,

— METTRE-À-JOUR(F, v) replace v dans la file F.

Fonction A*(s, t)
  pour tout $u \in S$ **faire**
    $D[u] \leftarrow \infty$
    $P[u] \leftarrow nil$
  $D[s] \leftarrow 0$
  $F \leftarrow$ CRÉER-FILE-PRIORITÉ(S)
  u $\leftarrow$ DÉFILER-MIN(F)
  tant que non EST-VIDE(F) **et** $u \neq t$ **faire**
    pour tout $v \in Adj[u]$ **faire**
      si $D[v] > D[u] + w(u, v)$ **alors**
        $D[v] \leftarrow D[u] + w(u, v)$
        $P[v] \leftarrow u$
        METTRE-À-JOUR(F, v)
    u $\leftarrow$ DÉFILER-MIN(F)

On propose d'appliquer $A*$ pour calculer un plus court chemin de Rennes à Nice : le graphe ici est pondéré par les distances routières entre les villes reliées par une autoroute, on propose d'utiliser comme heuristique h la distance en ligne droite.

QUESTION 5 – Exécuter l'algorithme A*(Rennes, Nice) sur le graphe suivant, en utilisant l'heuristique proposée.



h(., Nice)	
Rennes	850
Lille	830
Strasbourg	550
Paris	700
Dijon	450
Bordeaux	700
Toulouse	500
Marseille	150
Lyon	300

4 Problème 2-SAT

Le problème 3-SAT est NP-complet. Dans cet exercice, nous allons montrer que si on se restreint aux formules qui sont des *formes normales conjonctives* d'ordre 2, comme par exemple $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_4) \wedge \dots$ alors le problème de satisfiabilité est dans P.

On définit le problème 2-SAT de la manière suivante :

— entrée : une formule $\phi(x_1, \dots, x_n)$ sous forme normale conjonctive d'ordre 2 ;

— sortie : oui ssi il existe une valuation pour $x_1 \dots x_n$ qui rende ϕ vraie.

QUESTION 6 – En observant que la formule $x \vee y$ est équivalente à $\neg x \implies y$, donner un problème équivalent sur les graphes orientés résoluble en temps polynomial.

QUESTION 7 – Conclure.

5 2-colorabilité

Une *k-coloration* d'un graphe non-orienté $G = (S, A)$ est une fonction $c : S \to [\![0, k-1]\!]$ telle que $c(u) \neq c(v)$ pour toute arête $(u, v) \in A$.

QUESTION 8 – Montrer que tout arbre est *2-coloriable*.

QUESTION 9 – Proposer un algorithme permettant de décider si un graphe est *2-coloriable*.