Université Paris 7
Maîtrise d'Informatique – Algorithmique
27 novembre 2002
Durée : 2 heures. Documents autorisés : notes de cours/TD. Le barème est indicatif.

Question 1 (4 points) :

Soit $G = (V, E)$ un graphe orienté. Pour deux sommets $i, j \in V$ on note $d_{\max}(i, j)$ la longueur maximale d'un chemin de i à j ($d_{\max}(i, j) = \infty$, s'il existe un chemin non-simple de i à j et $d_{\max}(i, j) = -\infty$, s'il existe aucun chemin de i à j).

Proposez un algorithme de temps au plus cubique pour calculer $d_{\max}(i, j)$ pour toute paire de sommets $i, j \in V$. Justifier la validité de votre algorithme.

Question 2 (8 points) :
Soit $G = (V, E, p)$ un graphe orienté pondéré par la fonction $p : E \to \{1, \ldots, 1000\}$. On fixe un sommet source $s \in V$ et on veut calculer pour tout sommet $v \in V$ le poids minimal d'un chemin de s à v.

Le but de l'exercice est de proposer une implémentation de l'algorithme de Dijkstra avec un temps d'exécution linéaire (dans $O(|V| + |E|)$). L'algorithme de Dijkstra est rappelé à la fin de l'exercice.

1. Vous devez gérer les sommets v à traîter dans un tableau T. L'entrée $T[k]$ du tableau contient les sommets de distance actuelle $D(v) = k$.
   Précisez quels sont le type et la taille du tableau.
2. Décrivez comment effectuer Relax(u, v, p) sur la structure de données proposée.
3. Pour choisir un sommet v tel que $D(v)$ est minimal on garde comme information supplémentaire le k minimal tel que $\{v \mid D(v) = k\}$ est non-vide.
   Décrivez comment effectuer Delete-Min(T) sur la structure de données proposée.
4. Justifier que Relax(u, v, p) ne change pas la valeur k de la question précédente.
5. Justifier le temps d'exécution linéaire.

```
Dijkstra(G = (V, E), s ∈ V, p : E → ℕ)
debut
    S ⟵ ∅
    T ⟵ V // a detailler
    Tant que (T ≠ ∅) faire
        u ⟵ Delete-Min(T)
        S ⟵ S ∪ {u}
        Pour tout (u, v) ∈ E faire
            // Relax(u, v, p) :
            D(v) ⟵ min(D(v), D(u) + p(u, v))
        FinPour
    FinTantque
fin
```

Question 3 (8 points) :
Soit $G = (V, E, p)$ un graphe non-orienté connexe pondéré, où $p : E \to \mathbb{Z}$ est la fonction associant à chaque arête $e \in E$ un poids entier $p(e) \in \mathbb{Z}$. *On suppose ici que les poids sont tous différents :* $p(e) \neq p(e')$, pour tous $e \neq e'$.

Le but de cet exercice est de montrer que l'algorithme ci-dessous calcule un arbre recouvrant minimal (ARM) de G. L'algorithme opère sur une liste $\mathcal{F}$ d'arbres $T_1, \dots, T_k$, qui sont deux à deux disjoints et qui recouvrent G. Formellement, on a $T_i = (V_i, E_i)$, où V est l'union disjointe des V_i et chaque E_i est un sous-ensemble de E.

La liste $\mathcal{F}$ est initialisée avec les arbres-singletons $(\{v\}, \emptyset)$, $v \in V$. L'algorithme retourne un ensemble d'arêtes $F \subseteq E$.

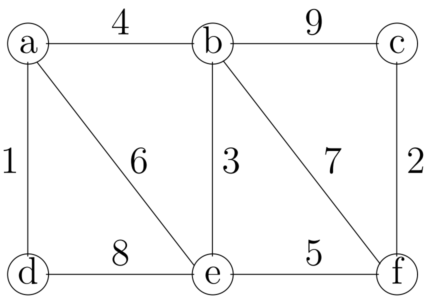
```
fonction ARM (G = (V, E, p)) : retourne un ensemble d'arêtes ;
debut
    Trier les aretes selon le poids (ordre croissant) ;
    𝓕 = ∅ ;
    F = ∅ ;
    Pour tout v ∈ V faire
        rajouter ({v}, ∅) a 𝓕 ;
    Tant que |𝓕| > 1 faire
        X ⟵ ∅ ;
        Pour tout T ∈ 𝓕 faire
            uv ⟵ arête de poids minimal telle que u ∈ T et v ∉ T ; (*)
            X ⟵ X ∪ {uv} ;
        FinPour
        Reorganiser (𝓕, F, X) ;
        F ⟵ F ∪ X ;
```

FinTantque
retourner(F)
fin

On suppose que le graphe G est donné par des listes d'adjacence.

Après l'appel de la procédure Reorganiser ($\mathcal{F}$, F, X), la liste $\mathcal{F}$ contient les composantes connexes de $(V, F \cup X)$ (avant l'appel, $\mathcal{F}$ contient les composantes connexes de (V, F)). On montrera que ces composantes connexes sont toujours des arbres.

1. Montrer l'exécution de cet algorithme sur l'exemple suivant :



2. Montrer qu'après chaque itération de la boucle Pour-tout, le graphe (V, F) est acyclique. (Rappel : les poids des arêtes sont tous différents.)
3. Montrer l'invariant suivant : à chaque instant, l'ensemble des arêtes F peut être étendu par un ARM de G : il existe un ARM (V, F') tel que $F \subseteq F'$.
4. Montrer qu'après k itérations de la boucle Tantque, la liste $\mathcal{F}$ contient au plus $\frac{|V|}{2^k}$ arbres.
5. Décrire une structure de données de type union-find pour gérer la liste $\mathcal{F}$ et les arbres T_i. Cette structure de données sera basée sur l'implémentation de Union-Find par des listes chaînées.
6. Décrire la ligne (*) de l'algorithme ainsi que la procédure Reorganiser ($\mathcal{F}$, F, X) avec des opérations union-find.
7. Montrer qu'en utilisant l'union par longueur on obtient un temps d'exécution de $O((|V| + |E|) \log(|V|))$. Justifier votre réponse, en indiquant :
   - le nombre maximal de fois que chaque arête est considérée dans l'instruction (*),
   - le temps total pour exécuter les instructions (*),
   - le temps total pour exécuter les appels de Reorganiser ($\mathcal{F}$, F, X).