



Master BioInformatique

ANNÉE : 2011/2012

SESSION DE AVRIL 2012

**PARCOURS** : Master 1

**UE J1BS8203** : Méthodes et outils pour la biologie des systèmes

**Épreuve** : Examen

**Date** : Mardi 10 avril 2012

**Heure** : 10 heures

**Durée** : 2 heures

Documents : autorisés

Épreuve de M. Alain GRIFFAULT

## Code d'anonymat :

### **Avertissement**

- La plupart des questions sont indépendantes.
- L'espace laissé pour les réponses est suffisant (sauf si vous utilisez ces feuilles comme brouillon, ce qui est fortement déconseillé).

| Question                                     | Points | Score |
|--|--------|-------|
| Automates de recherche de motifs             | 4      |       |
| Parcours en profondeur de graphes            | 4      |       |
| Graphes pondérés                             | 4      |       |
| Variantes plus court chemin à origine unique | 8      |       |
| Total:                                       | 20     |       |

### **Exercice 1: Automates de recherche de motifs** **(4 points)**

(a) (2 points) Pour les mots sur l'alphabet  $\Sigma = \{a, b\}$ , dessinez l'automate de recherche du motif *aabab*.

(b) (2 points) On dit d'un motif  $P$  qu'il est *non recouvrable* si  $(P_k \supseteq P_q) \Rightarrow (k = 0 \wedge k = q)$ , c'est à dire que si les  $k$  premières lettres du motif forment un suffixe des  $q$  premières lettres de ce même motif, alors soit  $k = 0$ , soit  $k = q$ . Donnez la particularité de l'automate d'un motif non recouvrable.

**Exercice 2: Parcours en profondeur de graphes****(4 points)**

Donnez un graphe orienté  $G$  tel qu'il existe deux sommets  $u$  et  $v$  vérifiant :

- $u \rightsquigarrow v$
- $u.debut < v.debut$
- $v$  n'est pas un descendant de  $u$  dans la foret en profondeur obtenue lors du parcours en profondeur.

**Exercice 3: Graphes pondérés****(4 points)**

Vous admettrez la propriété suivante :

**Propriété 1** Soit  $G(S, A, w)$  un graphe non orienté pondéré avec  $w : A \rightarrow N$ . Soient  $T \subseteq A$  et  $T' \subseteq A$  deux arbres couvrants de poids minimal. Soient  $L_T = (a_0, \dots, a_n)$  et  $L_{T'} = (a'_0, \dots, a'_n)$  les listes triées par poids croissant des arêtes de  $T$  et  $T'$ . Alors :  $\forall i \in [0..n], w(a_i) = w(a'_i)$ . Informellement, la liste ordonnée des poids des arêtes constituant un arbre couvrant est unique.

Soit  $G(S, A, w)$  un graphe non orienté pondéré. Montrer que pour chaque arbre couvrant de poids minimal  $T$  de  $G$ , il existe un moyen pour que l'algorithme de Kruskal retourne comme résultat  $T$ .

**Exercice 4: Variantes plus court chemin à origine unique****(8 points)**

L'algorithme de Dijkstra

```

Initialisation(G,s){ // G(S,A,w) oriente
    pour u dans S faire {
        u.d <- MAXINT; // infini
        u.pere <- nil;
    }
    s.d <- 0;
}

Relacher(u,v,w){
    si (v.d > u.d + w(u,v))
    alors {
        v.d <- u.d + w(u,v);
        v.pere <- u;
    }
}

Dijkstra(G,s){ // G(S,A,w) oriente
    Initialisation(G,s);
    F <- CreerTas(S); // F est un tas construit avec les valeurs de s.d
    tant que (F != {}) faire {
        u <- ExtraireMinimum(F);
        pour v element de u.adjacents faire {
            Relacher(u,v,w);
        }
    }
}

```

vu en cours calcule :

- Pour chaque sommet  $u$ , la distance  $d(s,u)$  de  $u$  à la racine  $s$ .
- $E$  : Une arborescence des plus courts chemins issus de  $s$ .

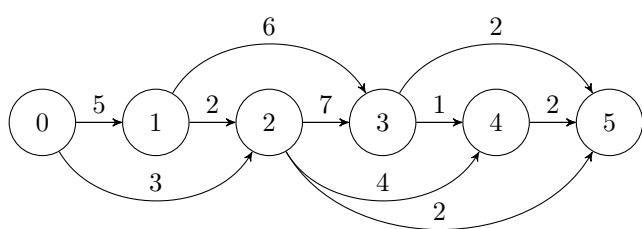
Soit  $G(S, A, w)$  un graphe orienté pondéré **acyclique**. Il est alors possible d'améliorer l'algorithme de Dijkstra pour calculer une arborescence des plus courts chemins.

```

Dijkstra-acyclique(G){ // G(S,A,w) oriente acyclique
    L <- TriTopologique(G); // L contient la liste chainee triee
    Initialisation(G,Tete(L));
    tant que (L != nil) faire {
        u <- Tete(L);
        L <- Queue(L);
        pour v element de u.adjacents faire {
            Relacher(u,v,w);
        }
    }
}

```

- (a) (2 points) Donnez le résultat ( $u.d$  et  $u.pere$  pour chaque sommet) de l'algorithme **Dijkstra-acyclique** sur le graphe suivant.



| sommet | 0   | 1 | 2 | 3 | 4 | 5 |
|--------|-----|---|---|---|---|---|
| u.d    | 0   |   |   |   |   |   |
| u.pere | nil |   |   |   |   |   |

FIGURE 3 – Un graphe orienté pondéré acyclique

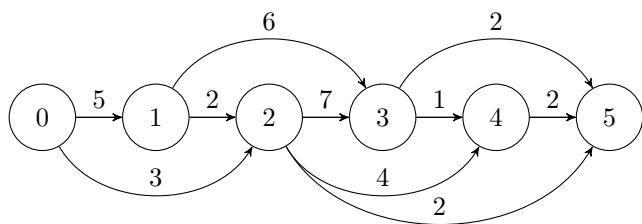
- (b) (2 points) Donnez la complexité de l'algorithme **Dijkstra-acyclique**.

Une variante de cet algorithme est utilisable pour calculer les chemins critiques dans un graphe  $G(S, A, w)$  lorsque :

- Les arcs représentent les tâches à faire.
  - Les pondérations représentent le temps nécessaire pour effectuer la tâche.
  - les arcs  $a_1 = (u, v)$  et  $a_2 = (v, w)$  représentent deux tâches qui doivent être effectuées dans l'ordre  $a_1, a_2$ .
- Un *chemin critique* est un *plus long* chemin dans le graphe, qui correspond au temps maximum requis pour effectuer une séquence ordonnée de tâches. Le poids d'un chemin critique est une borne inférieure du temps total nécessaire à l'exécution de toutes les tâches.

(c) (2 points) Adaptez les algorithmes précédents pour calculer les chemins critiques d'un graphe orienté pondéré acyclique.

(d) (2 points) Donnez le résultat (u.d, u.pere pour chaque sommet et le chemin critique) de votre algorithme sur le graphe suivant.



| sommet | 0 | 1 | 2 | 3 | 4 | 5 |
|--------|---|---|---|---|---|---|
| u.d    |   |   |   |   |   |   |
| u.pere |   |   |   |   |   |   |

FIGURE 4 – Un graphe orienté pondéré acyclique

**Chemin critique :**