

# Algorithmique

Examen de contrôle continu - 21 Mai 2004

Durée : 2h

- Le seul document autorisé est une feuille A4 recto-verso de notes.
- Les calculatrices, téléphones, ordinateurs etc. . . sont interdits.

Document réécrit par Samuel Robyr <[samuel.robyr@epfl.ch](mailto:samuel.robyr@epfl.ch)>.  
Merci de signaler tout erreur ou remarque.

Version du document : 17 avril 2005

La version courante de ce document est disponible à l'adresse <http://epfl.neoch.net>

**Problème 1 [17 points].** Triez la liste suivante avec heap sort :

12, 47, 41, 89, 2, 17, 38, 55, 19, 17

Expliquez clairement chaque étape de l'algorithme.

**Problème 2 [18 points].** Supposons que nous avons le tableau de clés suivant que nous voulons trier par ordre croissant :

66, 28, 1, 19, 0, 17, 7, 3, 49, 11, 14

- a) Réécrivez ce tableau après les 3 premiers échanges de selection sort.
- b) Même question pour bubble sort.
- c) Même question pour quick sort, où on prend comme pivot le dernier élément du tableau.
- d) Si on utilise quick sort avec comme pivot le dernier élément, quelles seront les deux sous-suites sur lesquelles on utilisera quick sort après la première étape ?
- e) Si on utilise quick sort avec la stratégie 3-médianes pour choisir le pivot, quel élément sera le pivot pour la première étape ?
- f) Supposons que nous utilisons Shell sort avec les incréments 5, 3, 1. Réécrivez le tableau après le passage du premier incrément (i.e. après qu'il a été 5-trié).
- g) Supposons que nous avons une permutation aléatoire uniforme de l'ensemble de clés  $0, \dots, N-1$ . Quelle est la probabilité que la première étape de selection sort nécessitera un échange ?
- h) Quel sont les temps de parcours de quick sort sur un tableau de taille  $N$  *dans le cas moyen et dans le pire des cas* (en notation  $\Theta$ ) ? Pas besoin de justifier.
- i) Quel sont les temps de parcours de heap sort sur un tableau de taille  $N$  *dans le cas moyen et dans le pire des cas* (en notation  $\Theta$ ) ? Pas besoin de justifier.

**Problème 3 [16 points].** Dans l'algorithme 1 (donné en pseudocode)  $S$  est un stack initialement vide, et pour tout sommet  $v$  du graphe d'input  $G$ ,  $N[v]$  est initialisé à l'ensemble des voisins de  $v$ .

---

**Algorithme 1** Algo( $G$ )

---

**Input:** Graphe connexe, orienté  $G = (V, E)$

**Output:** ? (voir question b))

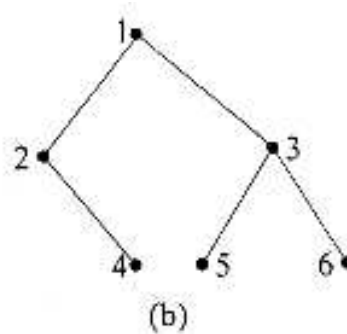
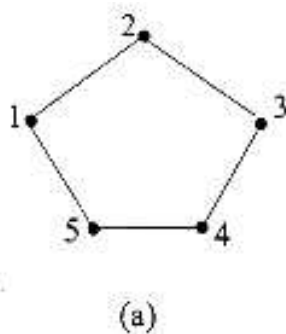
```

1: Choisir  $v \in V$ 
2: Push( $S, v$ )
3: Marquer  $v$ 
4: while Not Stack-Empty( $S$ ) do
5:    $w \rightarrow \text{Top}(S)$ 
6:   Pop( $S$ )
7:   while  $N[w] \neq \emptyset$  do
8:     Choisir  $u \in N[w]$ 
9:      $N[w] \rightarrow N[w] \setminus \{u\}$ 
10:    if  $u$  est marqué then
11:      return TRUE
12:    else
13:      Push( $S, u$ )
14:      Marquer  $u$ 
15:    end if
16:  end while
17: end while
18: return FALSE

```

---

- a) Que se passerait-il si on donnait comme input à cet algorithme les graphes suivants ? Pour chacun des graphes écrire chaque état du stack pendant l'algorithme, et quel serait l'output.

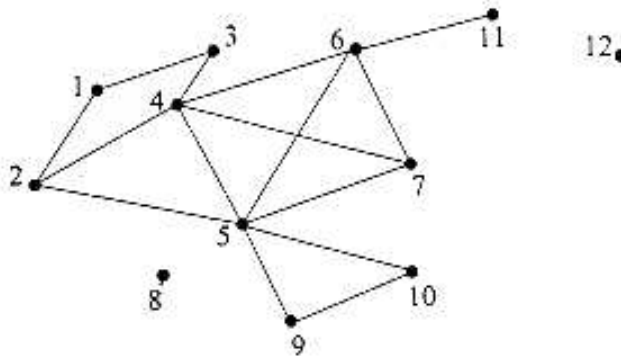


- b) Remplissez le texte manquant pour **Output** dans le pseudocode ci-dessus.
- c) Si on suppose que toutes les opérations individuelles de cet algorithme (push, pop, marquer, choisir, ...) peuvent être réalisées en temps constant, quel est le temps de parcours de cet algorithme dans le pire des cas, en fonction de  $|V|$  ?

**Problème 4 [16 points].** Une *clique* dans un graphe non orienté est un sous-graphe dans lequel chaque sommet est connecté à tous les autres sommets. Formellement, si  $G = (V, E)$  est un graphe non-orienté, alors une clique est un sous-ensemble  $W \subseteq V$  avec

$$\forall w_1, w_2 \in W \quad \text{avec } w_1 \neq w_2 : (w_1, w_2) \in E$$

a) Trouvez 3 cliques de taille  $> 2$  dans le graphe suivant :



Considérer l'algorithme glouton suivant pour trouver une clique dans un graphe donné :

---

**Algorithme 2** FindClique( $G$ )

---

**Input:** Graphe non orienté  $G = (V, E)$

**Output:** Sous-ensemble  $U \subseteq V$  tel que  $\forall u_1, u_2 \in U$  avec  $u_1 \neq u_2 : (u_1, u_2) \in E$

1:  $U \leftarrow V$

2: **while**  $U$  n'est pas une clique **do**

3:   Choisir  $u \in U$  pour lequel il existe un  $u' \in U$  qui n'est pas connecté à  $u$

4:    $U \leftarrow U \setminus \{u\}$

5: **end while**

6: **return**  $U$

---

b) Le problème de trouver *la clique la plus grande* d'un graphe est en général très difficile. Cet algorithme ne le résout pas.

Donnez un exemple d'un graphe, et d'un parcours de cet algorithme sur ce graphe qui ne retourne pas la clique la plus grande de ce graphe.

c) Trouvez une construction qui marche pour tout  $s$  d'un graphe ont la clique la plus grande est de taille  $s$ , mais pour lequel il y a un parcours de cet algorithme qui retourne une clique  $\leq 2$ .

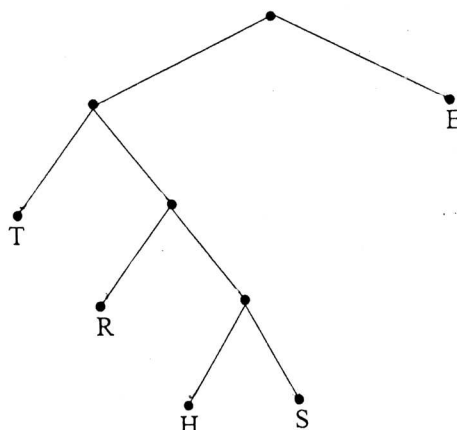
**Problème 5 [16 points].**

- a) Supposons que nous voulons envoyer un message composé à partir de lettres dans l'ensemble suivant :  $C = \{A, B, C, D, E, F\}$ . Supposons aussi que les fréquences moyennes avec lesquelles sont utilisés ces lettres sont (dans le même ordre que ci-dessus) :

$$\frac{3}{35}, \frac{4}{35}, \frac{3}{35}, \frac{7}{35}, \frac{14}{35}, \frac{4}{35}$$

Supposons que les seuls symboles que nous pouvons envoyer sont de 0's et des 1's. Utilisez l'algorithme de Huffman pour construire un encodage sans préfixe de  $C$  dans  $\{0, 1\}^+$  avec la longueur moyenne la plus petite possible.

- b) Supposons que nous envoyons un message de longueur 3500 (i.e., nous envoyons 3500 symboles de  $C$  avec les fréquences données). Quelle est l'espérance du nombre de bits que nous devons envoyer une fois que le message a été encodé avec votre code de la question précédente.
- c) Supposons que nous avons un code binaire pour l'alphabet  $\{E, S, T, H, R\}$  donné par l'arbre suivant :



(Nous supposons que les 0's sont à gauche et les 1's à droite).

- Decodez le message suivant :

0001100101101110001011000111

- Encodez le message suivant :

SETTHETREE

- d) Supposons que nous avons un message dans lequel chacun des symboles dans l'alphabet  $\{E, S, T, H, R\}$  apparaît avec une fréquence  $\frac{1}{5}$ . Quelle serait la longueur moyenne du code dans la question c) ? Serait-il optimal (explique pourquoi, pas de points pour oui/non) ?

**Problème 6 [17 points].**

- a) Donnez un algorithme qui calcule le in-degree de tous les sommets dans un graphe orienté acyclique.
- b) Prouvez qu'un graphe orienté acyclique fini contient toujours un sommet avec in-degree 0.
- c) Un graphe orienté acyclique fini contient-il toujours un sommet avec out-degree 0 ? Justifiez (pas de points pour oui/non).