IPS 30 octobre 2020
Examen intermédiaire Durée: 3h

Algorithme et complexité
Examen 3 novembre 2020

Les documents sont autorisés. La note prendra en compte la clarté des explications. Il est conseillé d'effectuer les exercices dans l'ordre.

1 Partie d'A. Chateau

Exercice 1 Récurrence.

Démontrer par récurrence que pour tout entier $n \geq 1$, on a :

$$\sum_{k=1}^{n} 3^k k^2 = \frac{3}{2}(3^n(n^2 - n + 1) - 1)$$

□

Exercice 2 Ensembles.

— Dessinez un ensemble $E = \{a, b, c, d, e\}$ et deux ensembles A et B, tels que $A \subseteq E$, $B \subseteq E$, $|A \cap B| = 2$ et $|A| < |B|$.
— Écrivez en extension $\mathcal{P}(A)$.

□

Exercice 3 Applications. Soit $f : \mathbb{N} \to \mathbb{N}^2$ définie pour $n \in \mathbb{N}$ par $f(n) = (n, (n+1)^2)$ et $g : \mathbb{N}^2 \to \mathbb{N}$ définie pour $(n, m) \in \mathbb{N}^2$ par $g(n, m) = nm$.

— calculez $f(n)$ pour $n \leq 4$.
— calculez $g(0, 0)$, $g(0, 1)$, $g(1, 0)$ et $g(1, 1)$.
— f est-elle injective ?
— f est-elle surjective ?
— g est-elle injective ?
— g est-elle surjective ?

□

2 Partie de R. Giroudeau

Exercice 4 Parcours en largeur et en profondeur

1. Rappeler les structures de données utilisées pour les parcours en profondeur et en largeur.
2. Pour le graphe donné par la figure 1 à 14 sommets, les voisins de chaque sommet sont supposés écrits dans l'ordre croissant de leurs numéros. Ainsi nous obtenons :

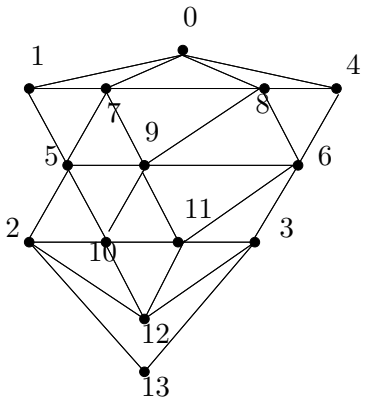

FIGURE 1 – *Graphe à explorer.*

— 0 admet pour voisins $1, 4, 7$ et 8,
— 1 a pour voisin $0, 5, 7$,
— 2 admet pour voisins $5, 10, 12, 13$,
— et ainsi de suite.

En partant du sommet 0, procéder à une exploration en profondeur de ce graphe, en utilisant l'ordre des voisins tel qu'il a été proposé ci-avant. Donner l'arbre obtenu.

3. Procéder à une exploration en largeur toujours en partant du sommet 0.

□

Exercice 5 Complexité algorithmique

1. Déterminer la complexité des algorithmes 1, 2, 3, et 4 suivants (par rapport au nombre d'itérations effectuées), où m et n sont deux entiers positifs.

Algorithm 1 Algorithme 1

$i := 1; j := 1;$
while $i \leq m$ et $j \leq n$ **do**
  $i := i + 1$
  $j := j + 1$
end while

Algorithm 2 Algorithme 2

$i := 1; j := 1;$
while $i \leq m$ ou $j \leq n$ **do**
  $i := i + 1$
  $j := j + 1$
end while

□

Exercice 6

Algorithm 3 Algorithme 3

```
i := 1; j := 1;
while j ≤ n do
  if i ≤ m then
    i := i + 1
  else
    j := j + 1
  end if
end while
```

Algorithm 4 Algorithme 4

```
i := 1; j := 1;
while  j ≤ n do
  if i ≤ m then
    i := i + 1 ;
  else
    j := j + 1 ;
    i := 1 ;
  end if
end while
```

1. Donner la matrice d'adjacence et la matrice d'incidence.
2. Donner la fermeture transitive du graphe donné par la figure 2.
3. Donner les composantes fortement connexes pour le même graphe.
4. Existe-t'il existe un cycle hamiltonien dans ce graphe ?

□

Exercice 7

Soit T un trié où un élément peut-être en plusieurs occurrence par exemple

$$T = [2, 5, 7, 7, 10, 13, 13, 13, 13, 17]$$

.

1. Donner un algorithme $occurrence(T, x)$ qui retourne l'indice de la première occurrence de x dans le tableau T (0 si $x \notin T$). Sur l'exemple $occurrence(T, 7)$ donne 3, $occurrence(T, 5)$ donne 2, et $occurrence(T, 14)$ donne 0. Donner la complexité de votre algorithme en fonction n où n est le nombre d'éléments du tableau.
2. En déduire un algorithme $nbreocc(T, x)$ qui retourne le nombre d'occurrences de x dans T. Sur l'exemple $nbreocc(T, 7)$ donne 2, $nbreocc(T, 5)$ donne 1 et $nbreocc(T, 14)$ donne 0. L'algorithme utilisera l'idée suivante : on cherche la première occurrence et on compte le nombre x. Donner la complexité de votre algorithme en fonction de n et du nombre d'occurrences m de x. Quelle est l'ordre de grandeur de la complexité dans le pire des cas en fonction de n.
3. Comment peut-on améliorer l'algorithme précédent pour obtenir un algorithme dont la complexité dans le pire des cas est en $\theta(\log n)$.

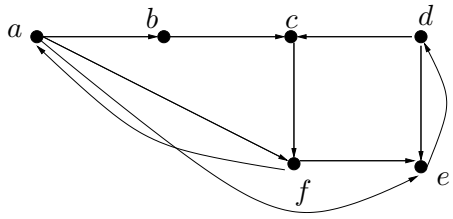

FIGURE 2 – *Graphe*

□

Exercice 8

Dans la suite, nous supposerons que le graphe est représenté par soit une matrice d'adjacence soit par une liste d'adjacente.

1. Donner les complexités pour les deux structures de données précédentes pour les fonctions suivantes :
   — Complexité en espace, accéder à un sommet, parcourir tous les sommets, parcourir toutes les arêtes et vérifier l'existence d'une arête (u, v).
   — Expliquer les avantages et les inconvénients des deux structures de données.

□