Nom : Prénom :

Université Clermont Auvergne — ISIMA 1A par apprentissage
Algorithmique — **Devoir Surveillé 1** — 2023-2024

- Durée : **1h**.
- Téléphones portables et objets connectés : interdits, même pour savoir l'heure.
- Veuillez répondre directement sous chaque question dans l'espace prévu à cet effet. Si vous avez besoin de plus de place, demandez une feuille.
- La notation tiendra compte de la rédaction.
- Veuillez écrire votre nom en haut de la première feuille.
- Le barème n'est qu'indicatif et pourra évoluer.

Exercice 1 (Complexité du tri par sélection, 5 points)**.**

1. Expliquer en quelques phrases le principe de l'algorithme du tri par sélection ci-dessous (1pt).
2. Exprimer le plus précisément possible, la complexité en temps $T(n)$ de l'algorithme, où n est la taille du tableau (4pts). Utiliser la formule $\sum_{i=0}^{s} i = \frac{s(s+1)}{2}$.

`triSelection(`T`): tri d'un tableau` T `d'entiers par ordre croissant`

`Entrée : un tableau` $T[1...n]$ `non trié de` n `entiers`

`Sortie : le tableau` T `trié`

- `Pour` i `allant de` 1 `à` $n-1$ `faire :`
  - ⋆ `indiceMin` $=$ i
  - ⋆ `valeurMin` $=$ $T[i]$
  - ⋆ `Pour` j `allant de` $i+1$ `à` n `faire :`
    - `Si` $T[j]$ $<$ `valeurMin alors :`
      `indiceMin` $=$ `j`
      `valeurMin` $=$ $T[j]$
  - ⋆ T[`indiceMin`] $=$ $T[i]$
  - ⋆ $T[i]$ $=$ `valeurMin`
- `Retourner` T

Solution.

1. Dans le tri par sélection, on parcourt le tableau du début à la fin. On maintient une partie triée en début de tableau. A chaque étape, on choisit le plus petit élément dans la partie non triée et on le met en fin de la partie triée. On s'arrête quand la partie triée comprend tout le tableau.

2.

triSelection(TAB): tri d'un tableau TAB d'entiers par ordre croissant

Entrée : un tableau $TAB[1...n]$ non trié de n entiers

Sortie : le tableau TAB trié

- Pour i allant de 1 à $n-1$ faire : //4 opérations par tour (test, $n-1$, incrémentation et réaffectation de i)
  // -1 car pas d'incrémentation le premier tour mais +1 pour le dernier test après le dernier tour
  - ⋆ indiceMin = i //1 opération
  - ⋆ valeurMin = $TAB[i]$ // 2 opérations
  - ⋆ Pour j allant de $i + 1$ à n faire : //4 opérations par tour, comme précédemment
    - Si $TAB[j] <$ valeurMin alors : //3 opérations
      indiceMin = j //2 opérations
      valeurMin = $TAB[j]$ //2 opérations
  - ⋆ TAB[indiceMin] = $TAB[i]$ //3 opérations
  - ⋆ $TAB[i]$ = valeurMin //2 opérations
- Retourner TAB //1 opération

La boucle principale est exécutée $n-1$ fois. La boucle imbriquée est exécutée au pire $n-i$ fois (mais le test de la boucle imbriquée est exécuté une fois supplémentaire sans y rentrer).

$$T(n) \leq \sum_{i=1}^{n-1} \left(4+1+2+3+2+\sum_{j=i+1}^{n} (4+3+2+2) \right) + 1$$

Si on utilise le fait que $n-i \leq n-1$ on obtient :

$$\begin{aligned}
T(n) &\leq \sum_{i=1}^{n-1} \left(12+\sum_{j=i+1}^{n} 11 \right) + 1 \\
&= \sum_{i=1}^{n-1} \left(12+\sum_{j=i+1}^{n} 11 \right) + 1 \\
&= \sum_{i=1}^{n-1} \left(12+\sum_{j=1}^{n-i} 11 \right) + 1 \\
&= (n-1)\left(12+11(n-1)\right)+1 \\
&= 12n-12+11(n-1)^2+1 \\
&= 12n-11+11(n^2-2n+1) \\
&= 11n^2-10n
\end{aligned}$$

Cependant, on peut faire une meilleure estimation ci-dessous. (On utilise le fait que $\sum_{i=1}^{s} i = \frac{s(s+1)}{2}$ dans le passage de la 4e à la 5e ligne.)

$$
\begin{aligned}
T(n) &\leq \sum_{i=1}^{n-1} \left(4 + 1 + 2 + 3 + 2 + \sum_{j=i+1}^{n} 11 \right) + 1 \\
&= \sum_{i=1}^{n-1} \left(12 + \sum_{j=1}^{n-i} 11 \right) + 1 \\
&= 1 + 12(n-1) + 11 \sum_{i=1}^{n-1} (n-i) \\
&= 1 + 12(n-1) + 11 \sum_{i=1}^{n-1} i \\
&= 1 + 12(n-1) + 11 \frac{n(n-1)}{2} \\
&= \frac{11}{2} n^2 + \frac{13}{2} n - 11
\end{aligned}
$$

Exercice 2 (Notations asymptotiques, 8 points)**.**

1. Montrer que $1000n^2 + 1000\log_2(n) - 100 \in O(n^3)$ (1pt)
2. Montrer que $\frac{n}{100} + \sqrt{n} \in \Theta(n)$ (2pts)
3. Montrer que $n^{100} \in O(2^n)$ (2pts)
4. Montrer que $100n \notin \Omega(n^3)$ (3pts)

Solution.

1. On va donner une valeur de c qui marche. Pour $r = 1$ et $c = 2000$, on a bien pour tout $i \geq r$, $1000i^2 + 1000\log_2(i) - 100 \leq 2000 \cdot i^3$ (car $i^2 \leq i^3$ et $\log_2(i) \leq i^3$), donc $1000n^2 + 1000\log_2(n) - 100 \in O(n^3)$.
2. On donne les bonnes valeurs de c pour montrer que $\frac{n}{100} + \sqrt{n} \in O(n)$ et $\frac{n}{100} + \sqrt{n} \in \Omega(n)$.
   Pour $r = 1$ et $c = 2$, on a, pour tout $i \geq r$, $\frac{i}{100} + \sqrt{i} \leq c \cdot i$ (car $\frac{i}{100} \leq i$ et $\sqrt{i} \leq i$), donc $\frac{n}{100} + \sqrt{n} \in O(n)$.
   Pour $r = 1$ et $c = \frac{1}{100}$, on a, pour tout $i \geq r$, $\frac{i}{100} + \sqrt{i} \geq c \cdot i$, donc $\frac{n}{100} + \sqrt{n} \in \Omega(n)$.
   Donc, $\frac{n}{100} + \sqrt{n} \in \Theta(n)$.
3. Ici on doit jouer sur r. A partir de quel r a-t-on $r^{100} \leq 2^r$?

$$\begin{aligned} r^{100} &\leq 2^r \\ \log_2(r^{100}) &\leq r \\ 100\log_2(r) &\leq r \\ 100 &\leq r/\log_2(r) \end{aligned}$$

   La fonction $r/\log_2(r)$ est croisssante, donc si on trouve un r qui marche, les valuers suivantes marcheront aussi. L'équation est vraie par exemple quand $r = 1024$, puisque $100 \leq 1024/10 = 102.4$ (en effet $2^{10} = 1024$, donc $\log_2(1024) = 10$).
   Donc, pour $r = 1024$ et $c = 1$, on a, pour tout $i \geq r$, $i^{100} \leq c \cdot 2^i$, donc $n^{100} \in O(2^n)$.
4. On suppose par l'absurde que $100n \in \Omega(n^3)$. Il existe sonc c et r tel que, pour tout $i \geq r$:

$$\begin{aligned} 100i &\geq c \cdot i^3 \\ 100 &\geq c \cdot i^2 \\ 100/c &\geq i^2 \end{aligned}$$

   Or lorsque i augmente, i^2 va forcément dépasser $100/c$ à un moment (peu importe la valeur de c), mais de par notre hypothèse l'équation devrait être vraie pour tous les i plus grands que r. C'est donc absurde, et $100n \notin \Omega(n^3)$.

Exercice 3 (Complexité asymptotique du maximum itératif, 2 points)**.**

À quelle classe de complexité *asymptotique* appartient l'algorithme ci-dessous ? Justifier.

```
maxiter(T): Recherche du maximum d'un tableau T
Entrée : un tableau T[1...n] non trié de n entiers
Sortie : le plus grand entier de T
  • max = T[1]
  • Pour i allant de 2 à n :
      ⋆ Si T[i] > max alors :
        max = T[i]
  • Retourner max
```

Solution.

Il y a une boucle sur n éléments, à chaque tour de boucle, il ya un nombre constant d'opérations de base, donc la complexité est dans $\Theta(n)$ (complexité linéaire).

Exercice 4 (Algorithme pour la factorielle, 5 points)**.**

1. Exprimer le plus précisément possible, la complexité en temps $T(n)$ de l'algorithme ci-dessous. (2pts)
2. Quelle est sa complexité asymptotique ? (1 pt)
3. Montrer que l'algorithme se termine en décrivant un convergent. (2 pts)

```
fact(n): calcul de n!
Entrée : un entier positif n
Sortie : n!
  • i = 2
  • f = 1
  • Tant que i ≤ n :
      f = f × i
      i = i + 1
  • Retourner f
```

Solution.

1.

```
fact(n): calcul de n!
Entrée : un entier positif n
Sortie : n!
  • i = 2 // 1 opération
  • f = 1 //1 opération
  • Tant que i ≤ n : // 1 opération (plus un test supplémentaire
    après le dernier tour)
      f = f × i //2 opérations
      i = i + 1 // 2 opérations
  • Retourner f //1 opération
```

   On a $n-1$ tours de boucle, donc on a $T(n) = 2 + 5(n-1) + 1 + 1 = 5n - 1$.
2. Complexité linéaire, c'est-à-dire $\Theta(n)$.
3. Le convergent est la suite des valeurs de i, qui commence à 1 et est incrémenté de 1 à chaque tour de boucle, et converge vers n.