Correction CC HAI503I 2023-2024

Sujet A

Exercice 1 :

Algo 1 : $\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{j} 1 + \sum_{i=0}^{n-1} 1 \leq \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1} 1 + \sum_{i=0}^{n-1} 1 = n^3 + n \in O(n^3)$

Algo 2 : le nombre d'itérations est le plus petit entier k tel que $\frac{n}{3^k} \leq 1 \Leftrightarrow k \geq \log_3(n)$ c'est à dire $k = \lceil \log_3(n) \rceil$. La complexité est donc $2 + \lceil \log_3(n) \rceil \in O(\log(n))$.

Algo 3 : Le nombre d'opérations élémentaires T_n vérifie $T_n = 2T_{\lceil n/3 \rceil} + n + 2 = 2T_{\lceil n/3 \rceil} + O(n^1)$. Le master theorem appliqué avec $a = 2$, $b = 3$ et $d = 1$ donne $T_n \in O(n)$.

Exercice 2 :

1.

```
Procédure : videPile(E P : pile)
    Tant que estVide(P) faire
        Depiler(P)
    FinTantQue
```

2. Il y a k éléments qui coûtent c chacun, d'où un coût de $k \times c$ (aussi vrai pour une pile vide).

3.a. On décompose n_E en $E + D + V$ avec :
- E le nombre d'éléments empilés jamais dépilés. Coût : 1 op/élem donc $c \cdot E$
- D le nombre d'éléments empilés puis dépilés par Dépiler. Coût : 2 op/élem donc $2 \cdot c \cdot D$
- V le nombre d'éléments empilés puis dépilés par VidePile. Coût : 2 op/élem donc $2 \cdot c \cdot V$

Le coût total C est le nombre de manipulations d'éléments (les appels ne manipulant pas d'éléments ont un coût nul). Autrement dit,

$C = c \cdot E + 2 \cdot c \cdot D + 2 \cdot c \cdot V \leq 2 \cdot c \cdot (E + D + V) = 2 \cdot c \cdot n_E$.

Le coût amorti par opération est $\frac{C}{n}$. Comme $n_E \leq n$, on obtient $\frac{C}{n} \leq 2c$.

3.b. On note M le solde du compte, k le nombre d'éléments de la pile et var_i la valeur de var à la fin de la i^{eme} opération.

La propriété que l'on veut démontrer par récurrence est P_n : "$M_n \geq 2 \cdot c \cdot k_n$".
- Initialisation : $M_0 = 0$ et $k_0 = 0$ donc P_0 est vraie.
- Hérédité : supposons P_n vraie : $M_n \geq 2 \cdot c \cdot k_n$. On effectue une $(n+1)^{eme}$ opération.
  - si c'est une opération de coût 0 : $M_{n+1} = M_n$ et $k_{n+1} = k_n$ donc $M_{n+1} \geq 2ck_{n+1}$.
  - si c'est Empiler : $k_{n+1} = k_n + 1$ et $M_{n+1} = M_n + c \geq 2 \cdot c \cdot k_n + c = c(2 \cdot k_{n+1} - 1) \geq 2 \cdot c \cdot k_{n+1}$
  - si c'est Dépiler : $k_{n+1} = k_n - 1$ et $M_{n+1} = M_n - c \geq 2 \cdot c \cdot k_n - c = c(2 \cdot k_{n+1} + 1) \geq 2 \cdot c \cdot k_{n+1}$.
  - si c'est VidePile : $k_{n+1} = 0$ et $M_{n+1} = M_n - c \cdot k_n \geq 2 \cdot c \cdot k_n - c \cdot k_n = c \cdot k_n \geq 0 = c \cdot k_{n+1}$.

  Dans tous les cas, P_{n+1} est vraie.
- Conclusion : à tout moment, $M \geq 2 \times k$.
- Deuxième conclusion : en particulier, M est positif, c'est à dire que le coût C des opérations est majoré par ce qu'on crédite sur le compte : $C \leq 2 \cdot c \cdot n_E$. On en déduit une majoration du coût amorti de chaque opération : $\frac{C}{n} \leq \frac{2 \cdot c \cdot n_E}{n} \leq 2c$.

Exercice 3 :

1.a. Le poids total des objets dépasse le poids de charge des camions.

1.b. $s \leq 100 \times m$ avec s : somme des poids des objets.

1.c. Si on choisit 3 objets de poids 60, on a $s = 180$ et $100 \times m = 200$ donc la condition du b) est vérifiée. Cependant il est impossible de rentrer plus d'un objet par camion donc de faire rentrer les 3 objets dans 2 camions.

2.a.

```
    Variable : somme : nombre ; U : tableau de m nombres
    // si on n'a pas m, on recherche le max de S avec un coût theta(n)
    Initialiser U avec des 0
    Pour i de 0 à n-1 faire
       U[S[i]] += T[i]
       si U[S[i]]>100 alors renvoyer False
    finPour
    Renvoyer True
```

F

2.b. Initialisation de U : $\theta(m)$

Boucle pour : $\theta(n)$.

On peut sans perte de généralité supposer $m \leq n$ donc la complexité est $\theta(n)$.

3.a. Chaque élément du tableau a m choix possibles donc il y a m^n n-uplets à parcourir, de $(0, 0, ..., 0)$ à $(m-1, m-1, ..., m-1)$.

3.b.

```
Fonction Suivant (S,m) : n-uplet ou texte
    Variables : T : n-uplet , i : entier
    T <- S
    i <- n-1
    Tant que i>=0 et T[i]=m-1 faire
       T[i]<-0
       i<-i-1
    FinTantQue
    Si i=-1 alors renvoyer "Fini"
    Sinon T[i]<-T[i]+1 ; renvoyer T
    FinSi
```

3.c. Dans le pire cas, on a n itérations dont l'algo est en O(n).

4.a.

```
Fonction Déménagement (T,m) : booléen
    Variables : S : n-uplet
    S<-[0,...,0]
    Tant que S!="Fini" et non EstValide(T,S) faire
       S<-suivant(S)
    FinTantQue
    Renvoyer S!="Fini"
```

4.b. Le nombre d'itérations est le nombre de n-uplets parcourus, qui est majoré par m^n. Comme EstValide et Suivant sont en $O(n)$, l'algorithme est en $O(n \times m^n)$.

4.c.

```
Fonction miniM (T) : entier
    Variables : m : entier
    m<-0
    tant que non Déménagement(T,m)
       m<-m+1
    FinTantQue
    Renvoyer m
```

5. On essaye de mettre le premier objet dans le premier camion ; si c'est ok, on recommence avec les autres objets (appel récursif) ; en cas d'échec, on le met dans le deuxième camion et ainsi de suite.

```
Fonction DémBTAux (T,S,i,m) : booléen
```

```
    Variable i : entier
    si i=n alors renvoyer True
    S[i] <- 0
    Tant que S[i]<n et (non estValide(T,S) ou non DémBTAux(T,S,i+1,m)) faire
        S[i]<-S[i]+1
    finTantQue
    Renvoyer S[i]<n

Fonction DémBT (T,m) : booléen
    Variable S : n-uplet
    S<-[0,...,0]
    renvoyer DémBTAux (T,S,0,m)
```