Mini-Projet d'Algorithme
Confiture

Fangzhou YE, Shihao ZHANG, Qijia HUANG

December 3, 2018

Table des Matières

1 Algorithme I: Recherche exhaustive **3**

2 Algorithme II: **5**

3 Algorithme III: **6**

Liste des Algorithmes

1 **Question 4 b)** 6
2 **Question 7** 6

1 Algorithme I: Recherche exhaustive

Question 1 $P(n)$ **RechercheExhaustive** (k, V, n) retourne bien le nombre minimal de bocaux utilisés pour la quantité n et elle se termine.

Base: $P(0)$ **RechercheExhaustive**$(k, V, 0)$=0 donc $P(0)$ vraie. Donc elle est valide et se termine.

Induction: Supposons que $\forall n \in [0, S]$, $P(n)$ vraie. Montrons que $P(n+1)$.

Dans **RechercheExhaustive**$(k, V, n+1)$, NbCount est initialisé par $n+1$, ce qui correspond à l'utilisation de $n+1$ bocaux de 1 dg.

Dans la boucle for: $\forall i \in [1, k]$, x=**RechercheExhaustive**$(k, V, n+1-V[i])$. x est le nombre minimal de bocaux utilisés pour remplir $n+1-V[i]$ dg de confitures. Comme $n+1-V[i] \leq n$. D'après **HR**, pour $\forall i \in [1, k]$, **RechercheExhaustive**$(k, V, n+1-V[i])$ sont valides et se terminent. Concernant la solution pour la quantité $n+1$ dg, on obtient le résultat $x+1$ en rajoutant un bocal de capacité $V[i]$ à la base de la solution pour la quantité $n+1-V[i]$. Puis, on compare $x+1$ avec $NbCount$ (ce qui est la meilleure soltion pour la quantité $n+1$ dg jusqu'à présent) et on garde ce qui plus petit dans la variable $NbCount$. On répète cette opération pour toute taille de bocal disponible. Donc à la fin de boucle.**RechercheExhaustive**$(k, V, n+1)$ correspond au nombre minimal de boucaux pour la quantité n+1. La boucle forá un nombre fini itérations, et dans chaque itération **RechercheExhaustive**$(k, V, n+1-V[i])$ se termine d'après **HR**. La fonction est valide et se termine. Donc $P(n+1)$ est vraie.

Conclusion: La fonction **RechercheExhaustive**(k, V, S) est valide et se termine.

Question 2

a)

$$a(s) = \begin{cases} 0, & \text{si } s = 0 \\ 2, & \text{si } s = 1 \\ a(s-1) + a(s-2) + 2, & \text{si } s \geq 2 \end{cases}$$

b) i. Montrons que $b(s) \leq a(s)$ par récurrence.

$P(n) : a(s) - b(s) \geq 0$.

Base:

- $P(0) : a(0) - b(0) = 0 - 0 = 0$. Elle est vraie.
- $P(1) : a(1) - b(1) = 2 - 2 = 0$. Elle est vraie.

Induction:

Supposons $\forall n \in [0, S]$, $P(n)$ vraie, montrons $P(n+1)$ vraie.

$$\begin{aligned} a(s+1) - b(s+1) &= a(s) + a(s-1) + 2 - (b(s-1) + b(s-1) + 2) \\ &= a(s) - b(s-1) + a(s-1) - b(S-1) \\ &\geq a(s) - b(s) + a(s-1) - b(S-1) \end{aligned}$$

On a $a(s) - b(s) \geq 0$ et $a(s-1) - b(s-1) \geq 0$ d'après **HR**. Donc $a(s+1) - b(s+1) \geq 0$. $P(n+1)$ est vraie.

Conclusion: $a(s) \geq b(s)$ pour tout entier $S \geq 0$.

ii. Montrons que $c(s) \geq a(s)$ par récurrence.

$P(n) : c(s) - a(s) \geq 0$.

Base:

- $P(0) : c(0) - a(0) = 0 - 0 = 0$. Elle est vraie.
- $P(1) : c(1) - a(1) = 2 - 2 = 0$. Elle est vraie.

Induction:
Supposons $\forall n \in [0, S]$, $P(n)$ vraie, montrons $P(n+1)$ vraie.

$$\begin{aligned} c(s+1) - a(s+1) &= c(s) + c(s) + 2 - (a(s) + a(s-1) + 2) \\ &= c(s) - a(s) + c(s) - a(s-1) \\ &\geq c(s) - a(s) + c(s) - a(s) \end{aligned}$$

On a $c(s) - a(s) \geq 0$ d'après **HR**. Donc $c(s+1) - a(s+1) \geq 0$. $P(n+1)$ est vraie.
Conclusion: $b(s) \leq a(s) \leq c(s)$ pour tout entier $s \geq 0$.

c) On suppos $P(s) : c(s) = (2^s - 1) \times 2$

Base:

- $c(0) = (2^0 - 1) \times 2 = 0$. Elle est vraie.
- $c(1) = (2^1 - 1) \times 2 = 2$. Elle est vraie.

Induction:
Supposons $P(s)$ vraie, montrons $P(s+1)$ vraie.

$$\begin{aligned} c(s+1) &= 2 \times c(s) + 2 \\ &= 2 \times (2^s - 1) \times 2 + 2 \\ &= (2^(s+1) - 2) \times 2 + 2 \\ &= (2^(s+1) - 1) \times 2 \end{aligned}$$

Conclusion: $c(s) = (2^s - 1) \times 2$

d) $P(s) : b(s) = c(\frac{s}{2})$.

Base:

- $P(0) : b(0) = c(0) = 0$. Elle est vraie.
- $P(1) : b(1) = c(\frac{1}{2}) = 0$. Elle est vraie.

Induction:
Suppons $\forall s \geq 0, P(s)$ vraie, montrons $P(s+1)$ vraie.

$$\begin{aligned} b(s+1) &= 2 \times b(s-1) + 2 \\ c(\frac{s+1}{2}) &= 2 \times c(\frac{s+1}{2} - 1) + 2 \\ &= 2 \times c(\frac{s+1}{2}) + 2 \\ \text{D'après } \mathbf{HR}, b(s-1) &= c(\frac{s-1}{2}) \\ \text{Donc } b(s+1) &= c(\frac{s+1}{2}) \end{aligned}$$

Conclusion: $P(s+1)$ vraie. Donc $b(s+1) = c(\frac{s+1}{2})$

e) $b(s) = (2^{\frac{s}{2}} - 1) \times 2$.

D'après les questions précédentes, on a $a(s)$ correspond au nombre d'après récursif réalisé par **RechercheExhaustive**$(2, \frac{1}{2}, s)$, et on a $b(s) \leq a(s) \leq c(s)$. Donc $(2^{\frac{s}{2}} - 1) \times 2 \leq a(s) \leq (2^s - 1) \times 2$. Donc on généralise cette conclusion. Donc **RechercheExhaustive**(k, V, s) où k est le nombre de bocas dispositifs. La complexité est en $O(k^5)$

2 Algorithme II:

Question 3

a) $m(s) = m(s, k)$

b) $P(s)\langle \forall i \in \{1, \ldots, k\}, m(s,i) = \min\{(s, i-1), m(s - v[i], i) + 1\}\rangle$

Base:

$$\begin{aligned} P(0) : m(0,i) &= \min(m(0, i-1), m(0 - v[i], i) + 1) \\ &= \min(0, +\infty) \\ &= 0 \end{aligned}$$

Induction: Suppons $\forall n \in \{0, \ldots, s\}, P(n)$vraie, montrons $P(n+1)$ vraie.

- Cas 1: $0 < s+1 < v[i]$: Le problème revient à déterminer une solution pour remplir s en utilisant que des bocal de $v[1] \ldots v[i-1]$. Donc, $m(s+1, i) = m(s+1, i-1)$.
  Soit $l = \max(h \in \{1, \ldots, i-1\}, \text{tel que } s + 1 - v(j) > 0)$,
  Alors $m(s+1, i-1) = m(s+1-v[i], i) + 1$. Comme $s + 1 - v[j] \leq s$. D'après **HR**, elle est vraie.
  Donc $m(s+1, i) = m(s+1, i-1)$ est vraie pour $s + 1 < v[i]$.
- Cas 2: $s + 1 \geq v[i], m(s+1, i) = m(s+1-v[i], i) + 1$, comme $s + 1 - v[j] \leq s$. D'après **HR**, elle est vraie.
  Donc $m(s+1, i) = m(s+1-v[i], i) + 1$ pour $s + 1 \geq v[i]$.

Conclusion: En combinant cas 1 et cas 2, on a

$$m(s,i) = \begin{cases} 0, & \text{si } s = 0 \\ \min(m(s, i-1), m(s - v[i], i) + 1) & \text{sinon.} \end{cases}$$

Question 4

a) En ordre de postfixe d'arbre récursif.

```
Algorithme : AlgoProgDynIter
Données : k:entier, V:tableau de k entiers, s:entier
Résultat : entier
opt : tableau de s + 1 entiers;
a, j, min, left, right : entier;
Opt[0] = 0;
pour a = 1 → s faire
    J = k;
    tant que V[j] > a faire
        J = j - 1;
    fin
    Left = 1;
    Right = a - 1;
    si a%V[j] == 0 alors
        Min = a/V[j]
    sinon
        Min = a
    fin
    tant que left ≤ right faire
        si min > opt[right] + opt[left] alors
            Min = opt[right] + opt[left];
        fin
        Left = left + 1;
        Right = right + 1;
    fin
    Opt[a] = min;
fin
Retourner opt[s+1];
```

Algorithme 1 : Question 4 b)

3 Algorithme III:

```
Algorithme : AlgoGlouton
Données : k: entier, V: tab de k entiers, S: entier, res:entiers
nb:=entiers;
si k==1 alors
    Retourner S;
sinon
    nb=s/V[k];
    res=s%v[k];
    Retourner AlgoGlouton(k-1, tab, res) + nb
fin
```

Algorithme 2 : Question 7

Question 8 On a exemple que pour $k = 5, tab = [1, 10, 20, 50, 70]$. Pour remplacer $s = 100$. D'après **AlgoGlouton**, on a résultat de 3 correspondant le bocal de [10, 20, 70]. Cependant, la solution de [50, 50] est plus optimal au niveau de nombre de bocaux.

Question 9

a) Montrer par l'absurde. Soit il n'existe pas un plus grand indice j tel que $o_j < k_j$, c'est-à-dire, pour tout $j \in \{1, \ldots, k\}, o_j > k_j$. Donc $\sum_{i=1}^{k} o_i > \sum_{i=1}^{k} g_i$ qui contredit que o est une solution optimale.

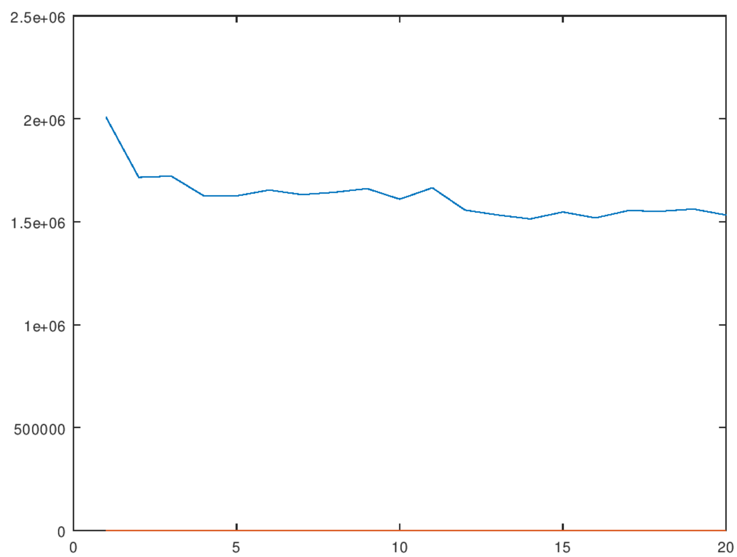
Question 10 $k = 2$ donc $v[1] = 1 = d^0, v = [2] = d = d^1$.
C'est bien un système Expo qui one bien glouton-compatible d'après la Question 9.

Question 11 Pour la première boucle

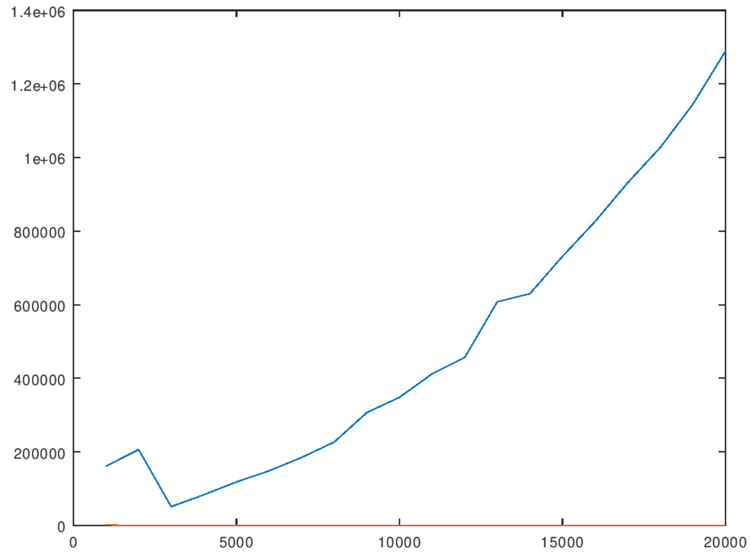
$$v[3] > 3$$
$$v[k-1] < s - 1$$
$$v[k] < s$$

Donc première boucle exécute au plus $2s \times 7$ fois.
Dans la deuxième boucle, la boucle exécute k fois.
AlgoGloutun (s) est en $O(s)$.
Donc $(2s - 7) \times k \times 2O(s)$ dont $k << s$.
Donc $T(n) = O(n^2)$.

Question 12



yefangzhou chajin



yefangzhou chajin