Examen Algorithmique 1 2017/2018

Partie 1 : Algorithmes gloutons

On considère le problème d'ordonnancement défini de la manière suivante :

- En entrée, un ensemble d'intervalles $\{Int_i\}_{i \leq n}$ à bornes dans $\mathbb{N}$ avec $Int_i = [s(i), f(i)]$;
- En sortie, un sous-ensemble $J \subseteq \{1, \ldots, n\}$ avec $|J|$ maximal tel que pour tout $i, j \in J$, avec $i \neq j$, $Int_i \cap Int_j = \emptyset$.

Algorithme 1 : Un algorithme glouton générique

```
Glouton(n, s, f)
Input : n un entier ; s, f, deux tableaux entiers indicés par {1, ..., n}
Output : J un sous-ensemble de {1, ..., n}
Data : i, j des indices ; I un ensemble
I ← {1, ..., n} ; J ← ∅
while I ≠ ∅ do
    Sélectionner i ∈ I selon le critère C
    J ← J ∪ {i} ; I ← I \ {i}
    for j ∈ I do
        if s(i) ≤ f(j) and s(j) ≤ f(i) then I ← I \ {j}
    end
end
return J
```

L'algorithme 1 dépend d'un critère de choix $\mathcal{C}$. Cependant les questions 1 et 2 se démomtrent pour tout critère $\mathcal{C}$.

Question 1. Démontrer la terminaison de l'algorithme 1.

Question 2. Etablir un invariant de la boucle **while** de l'algorithme 1 qui permet d'affirmer qu'à la fin de l'algorithme :

- $J \subseteq I$;
- pour tout $i, i' \in J$, avec $i \neq i'$, $Int_i \cap Int_{i'} = \emptyset$;
- pour tout $i \in \{1, \ldots, n\} \setminus J$, il existe $i' \in J$, $Int_i \cap Int_{i'} \neq \emptyset$.

Question 3. On choisit comme critère $\mathcal{C}$, l'un des $i \in I$ tels que $s(i)$ soit minimal. Exhiber un exemple pour lequel le J renvoyé par l'algorithme 1 n'est pas de taille maximale.

Question 4. On choisit comme critère $\mathcal{C}$, l'un des $i \in I$ tels que $f(i) - s(i)$ soit minimal. Exhiber un exemple pour lequel le J renvoyé par l'algorithme 1 n'est pas de taille maximale.

Etant donné $I \subseteq \{1, \ldots, n\}$, on définit pour $i \in I$,
$nc(i, I) = |\{j \in I \setminus \{i\} \mid s(i) \leq f(j) \textbf{ and } s(j) \leq f(i)\}|$.

Question 5. On choisit comme critère $\mathcal{C}$, l'un des $i \in I$ tels que $nc(i, I)$ soit minimal. Exhiber un exemple pour lequel le J renvoyé par l'algorithme 1 n'est pas de taille maximale.

On choisit comme critère $\mathcal{C}$, l'un des $i \in I$ tels que $f(i)$ soit minimal. Soit J^* un sous-ensemble qui répond au problème énoncé plus haut. Notons :

— $J = \{j_1, \ldots, j_k\}$ tel que j_ℓ est inséré à la $\ell^{ième}$ itération du **while** ;

— $J^* = \{j_1^*, \ldots, j_{k'}^*\}$ tel que $\ell \leq \ell'$ implique $f(j_\ell^*) \leq f(j_{\ell'}^*)$.

Par définition de J^*, $k \leq k'$.

Question 6. Montrer par induction que pour tout $\ell \leq k$, $f(j_\ell) \leq f(j_\ell^*)$. En déduire que $k = k'$ et que pour ce critère, l'algorithme 1 renvoie un sous-ensemble maximal.

Partie 2 : Programmation dynamique

On considère le problème d'ordonnancement défini de la manière suivante :

— En entrée, un ensemble d'intervalles $\{Int_i\}_{i \leq n}$ à bornes dans $\mathbb{N}$ avec $Int_i = [s(i), f(i)]$ où chaque intervalle Int_i a une valeur $v(i)$ dans $\mathbb{N}$;

— En sortie, un sous-ensemble $J \subseteq \{1, \ldots, n\}$ avec $\sum_{i \in J} v(j)$ maximal tel que pour tout $i, j \in J$, avec $i \neq j$, $Int_i \cap Int_j = \emptyset$.

On suppose dans la suite que $i \leq j$ implique $f(i) \leq f(j)$.

On définit $p(i) = \max(\{j < i \mid f(j) < s(i)\})$ avec la convention $\max(\emptyset) = 0$.

Question 7. Que représente $p(i)$?

Question 8. Proposer un algorithme de calcul de p en $O(n)$.

Algorithme 2 : Un algorithme récursif

Data : v, p, deux tableaux entiers indicés par $\{1, \ldots, n\}$

`OptRec`(i)
Input : $i \leq n$ un entier
Output : un entier

if $i = 0$ **then return** 0
if $p(i) = i - 1$ **then return** $v(i) +$ `OptRec`$(p(i))$
return $\max(v(i) +$ `OptRec`$(p(i)),$ `OptRec`$(i-1))$

Question 9. Démontrer la terminaison de l'algorithme 2.

Soit J_i^* une solution du problème ci-dessus lorsqu'on se restreint aux i premiers intervalles et $opt_i = \sum_{j \in J_i^*} v(j)$.

Question 10. Démontrer par induction sur i que l'algorithme 2 calcule opt_i.

Question 11. Exhiber une famille d'ensembles indicée par $n \in \mathbb{N}$ avec n, le nombre d'intervalles, telle que l'algorithme 2 s'exécute en temps exponentiel par rapport à n.

Question 12. Employer la technique de mémoïsation pour concevoir un algorithme en temps linéaire.

Question 13. Compléter l'algorithme de la question 12 (en utilisant les valeurs du tableau associé à la mémoïsation) afin de concevoir un algorithme en temps linéaire pour obtenir une solution J_n^*.

Partie 3 : Structure de données

On considère n immeubles le long d'un axe, définis pour tout $i \leq n$ par l'intervalle $[s(i), f(i)[$ des abcisses de l'immeuble i et $h(i)$ la hauteur de l'immeuble i. On définit la fonction de $\mathbb{R}$ dans $\mathbb{N}$, $hm(x) = \max(\{h(i) \mid s(i) \leq x < f(i)\})$ avec la convention $\max(\emptyset) = 0$.

Question 14. Quelle est la forme de $y = hm(x)$? Proposer une représentation finie de cette fonction.

Question 15. Concevoir un algorithme « naïf » de calcul de votre représentation de hm en $O(n^2)$.

On considère l'ensemble des $2n$ triplets $\{(s(i), 1, i), (f(i), 0, i)\}_{i \leq n}$ qu'on trie selon l'ordre lexicographique afin de construire une représentation de hm par la suite des points dont les abcisses sont les éléments de $Ab = \{s(i), f(i) \mid i \leq n\}$. Cette suite sera ordonnée par abcisse croissante. Pour ce faire on maintient l'ensemble *Actif* des immeubles qui couvrent l'abcisse courante de Ab et on détermine ainsi l'immeuble de hauteur maximale pour cette abcisse.

Question 16. Choisir une structure de données pour *Actif* (on justifiera son choix).

Question 17. Concevoir un algorithme de calcul de cette représentation de hm en $O(n \log(n))$ à l'aide de la structure de données pour *Actif*.