Université de Lille

FST- Département Informatique

ACT

2018–2019
Master d'Informatique-S1

Fiche TD: Algorithmes gloutons

Exercice 1 : Ordonner des tâches

Les utilisateurs d'une machine soumettent des *tâches* qui doivent s'exécuter sur celle-ci. Une instance du problème est:

- n le nombre de tâches
- $a_0, a_1, \ldots, a_{n-1}$: pour chaque tâche, sa date d'arrivée (en secondes)
- $t_0, t_1, \ldots, t_{n-1}$: pour chaque tâche, sa durée (en secondes)

On cherche un ordonnancement qui minimise le temps de complétion des tâches, c'est à dire la date à laquelle toutes les tâches sont terminées. Une tâche ne peut être interrompue si elle est en cours d'exécution. Une solution peut être vue comme une permutation des n tâches indiquant l'ordre dans lequel elles s'exécutent.

Par exemple, soit $n = 5$, $a_0 = 10, t_0 = 3$, $a_1 = 0, t_1 = 2$, $a_2 = 4, t_2 = 1$, $a_3 = 9, t_3 = 2$, $a_4 = 2, t_4 = 4$. Une permutation possible est $1, 4, 2, 0, 3$. La tâche 1 serait lancée au temps 0, la tâche 4 au temps 2, la tâche 2 au temps 6, la tâche 0 au temps 10, la tâche 3 au temps 13, avec un temps de complétion égal à 15.

Q 1. Dans l'exemple précédent, l'ordre d'exécution proposé est-il optimal? Justifier.

Q 2. Proposez un algorithme en $\Theta(n)$ qui étant donné l'instance du problème et un ordonnancement - une permutation des n tâches-, calcule la date à laquelle toutes les tâches seront terminées. Par exemple, pour l'instance ci-dessus et l'ordonnancement $1, 4, 2, 0, 3$, la sortie attendue est 15. Vous pouvez supposer que l'instance et l'ordonnancement sont représentés par la structure de données de votre choix.

Q 3. Le professeur Ordonnix propose d'utiliser un critère glouton pour trier les tâches afin de trouver une permutation optimale, c'est à dire qui minimise le temps de complétion. Il propose pour cela trois critères de tri:

- par t_i croissant, i.e. par durée croissante
- par a_i croissant, i.e. par date d'arrivée croissante
- par $a_i + t_i$ croissant.

Lequel de ces trois critères donne toujours la solution optimale? Justifiez votre réponse, par une preuve pour le critère qui garantit l'optimalité, par un contre-exemple pour chacun des deux autres critères.

Exercice 2 : Les antennes-relais, version simplifiée

Considérez une route rectiligne avec des habitations dispersées, placées aux positions $\{x_1, \ldots, x_n\}$ (les x_i sont réels et représentent la distance en kilomètres à partir du kilomètre 0 de la route). Une antenne-relais couvre les habitations à une distance au plus égale à k kilomètres d'elle.

Concevez un algorithme qui calcule le nombre minimal d'antennes-relais à placer pour couvrir toutes les habitations et les place. Analysez la complexité et justifiez l'optimalité de la solution.

Exercice 3 : Gardons la tête froide

Un laboratoire doit stocker des produits au frais. Il dispose de réfrigérateurs mais veut minimiser le nombre de ceux qu'il met en marche. La température d'un réfrigérateur peut être réglée à une température entre MIN et MAX (MIN et MAX sont indépendants du réfrigérateur). Chaque produit est muni d'une plage de température de stockage admissible, sous la forme d'un intervalle d'entiers fermé $[t_{min}, t_{max}]$ - avec $MIN \leq t_{min} \leq t_{max} \leq MAX$. On suppose que le volume d'un seul réfrigérateur est suffisant pour contenir tous les produits.

Donnez un algorithme polynomial (glouton) qui détermine le nombre minimal de frigos nécessaires -et leur réglage, pour stocker tous les produits à la bonne température. Justifier.

Exercice 4 : Stockage

Soit n enregistrements de longueur variable stockés dans un fichier à accès séquentiel. Chaque enregistrement e_i est de longueur s_i. Après chaque accès, on suppose que la fenêtre de lecture est repositionnée au début du fichier. On suppose que le temps d'accès au $i$ème enregistrement est proportionnel à la somme des longueurs des i premiers.

Par exemple pour deux enregistrements e_1, e_2, si ils sont stockés dans l'ordre (e_2, e_1), le temps d'accès à l'enregistrement e_2 est $c * s_2$, pour e_1, c'est $c * (s_2 + s_1)$, c étant une constante.

Dans un premier temps, on suppose que les demandes d'accès à chaque enregistrement sont équiprobables. Donc, si les enregistrements sont stockés dans l'ordre $e_{\sigma(1)}, \ldots, e_{\sigma(n)}$ le temps moyen d'accès est

$$M = c * 1/n * \sum_{j=1}^{n} (\sum_{k=1}^{j} s_{\sigma(k)})$$

Par exemple pour deux enregistrements, si les enregistrements sont stockés dans l'ordre (e_2, e_1), le temps moyen est donc $1/2 * c * (s_2 + (s_2 + s_1))$, soit $c * (s_2 + s_1/2)$.

Le problème est donc défini par:

Entrée:

n –le nombre d'enregistrements.

$s_1, s_2, .., s_n$ les longueurs des enregistrements.

Sortie: L'ordre de stockage optimal, i.e. une permutation $\sigma(1), .., \sigma(n)$ telle que $M = c*1/n*\sum_{j=1}^{n}(\sum_{k=1}^{j} s_{\sigma(k)})$ soit minimal.

Q 1. Soit la donnée suivante : $n = 3, s_1 = 12; s_2 = 26; s_3 = 19$.

Donner la meilleure façon de stocker les enregistrements.

Q 2. Proposer une méthode gloutonne qui donnera toujours une solution optimale. Justifier!

On suppose maintenant que pour chaque enregistrement i, on connait la fréquence d'accès, f_i (donc $\sum_i^n f_i = 1$). Le temps moyen d'accès sera donc :

$$M = c * \sum_{j=1}^{n} (f_{\sigma(j)} * \sum_{k=1}^{j} s_{\sigma(k)})$$

Par exemple pour deux enregistrements, si les enregistrements sont stockés dans l'ordre (e_2, e_1), le temps moyen est donc $c * (f_2 * s_2 + f_1 * (s_2 + s_1))$, soit $c(s_2 + f_1 * s_1)$.

Q 3. Soit l'heuristique consistant à ordonner les enregistrements selon les s_i croissants. Donner un exemple qui montre qu'elle ne donne pas toujours la solution optimale.

Q 4. Soit l'heuristique consistant à ordonner les enregistrements selon les f_i décroissants. Donner un exemple qui montre qu'elle ne donne pas toujours la solution optimale.

Q 5. Proposer un algorithme glouton qui donne toujours la solution optimale. Justifier.