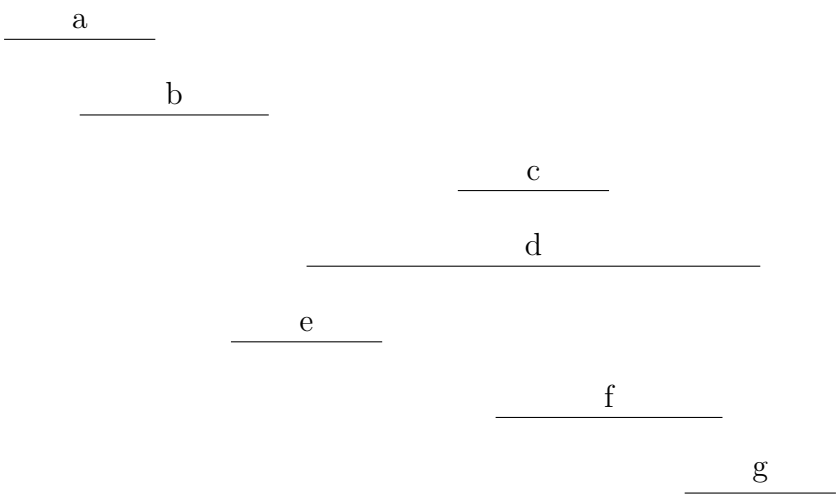
Algorithmes gloutons : ordonnancement de tâches

Florian Bourse

1 Les épreuves dans le gymnase (ordonnancement de tâches non pondérées)

Dans un gymnase doivent se dérouler une série d'épreuves. Les épreuves ne sont pas seulement caractérisées par leurs durées : chaque épreuve est caractérisée par une date de début d_i et une date de fin f_i. On souhaite "caser" le plus possible d'épreuves, deux épreuves ne pouvant avoir lieu en même temps (leurs intervalles de temps doivent être disjoints).

Exemple : Considérons les intervalles de temps suivant :



Le choix optimal contient les intervalles a, e, c, et g.

Ce choix peut être obtenu par les 4 stratégies gloutonnes suivantes :

Stratégie commune : on choisit à chaque étape une épreuve parmi celles qui sont compatibles avec les épreuves déjà choisies, jusqu'à ce qu'il n'en reste plus ;

Stratégie 1 : on choisit l'épreuve la plus courte ;

Stratégie 2 : on choisit l'épreuve commençant le plus tôt ;

Stratégie 3 : on choisit l'épreuve terminant le plus tôt ;

Stratégie 4 : on choisit l'épreuve qui est compatible avec le plus d'autres épreuves.

Question 1. Pour chacune de ces stratégies, montrer qu'elle est optimale ou donner un contre-exemple.

On souhaite implémenter une solution en langage C. Nous utiliserons l'exemple ci-dessus pour tester notre algorithme.

Exemple — C

```c
double debut[7] = {0,1,6,4,3,6.5,9};
double fin[7] = {2,3.5,8,10,5,9.5,11};
```

Question 2. Écrire une fonction `void swap(double tab[], int i, int j)` qui échange les valeurs en positions i et j du tableau passé en entré.

Nous allons trier les intervalles en utilisant un tri par séléction (selection sort), un tri glouton qui à chaque étape cherche le plus petit élément du tableau qui n'est pas encore à sa place, et qui le met dans la bonne position.

Question 3. Écrire une fonction `void sort(double debut[], double fin[], int n)` qui réarrange les tableaux de taille n donnés en argument pour que les épreuves que l'on considère en premier soit au début de la liste.
Quelle est la complexité de cet algorithme dans le pire des cas ?

Question 4. Écrire une fonction `void solution(double debut[], double fin[], int n)` qui affiche la solution du problème.
Quelle est la complexité de cet algorithme dans le pire des cas ?

2 Autre problème d'ordonnancement

Étant donné un ensemble $E = \{1, 2, \ldots, n\}$ de tâches unitaires (toutes ont pour durée une unité de temps) de dates échues $d_1, d_2, \ldots, d_n$ avec $1 \leq d_1 \leq d_2 \leq \ldots \leq d_n \leq n$ et de pénalités $w_1, w_2, \ldots, w_n$, on cherche un ordonnancement des tâches $1, 2, \ldots, n$ qui minimise le cumul des pénalités pour non respect des dates échues.

Un ordonnancement est une permutation des tâches $1, \ldots, n$. Un ensemble de tâches F est indépendant s'il existe un ordonnancement des tâches de F tel qu'aucune ne soit en retard sur sa date échue.

Question 5. Montrez que le calcul d'un ordonnancement optimal équivaut à trouver un ensemble indépendant de tâches de E dont la somme des pénalités soit le plus grand possible.

Question 6. Soit i une tâche de pénalité maximale. Existe-t-il un ensemble indépendant optimal qui contient i ?

Question 7. En déduire une méthode glouton qui fournit une solution optimale.