Algorithmes d'ordonnancement

Épreuve pratique d'algorithmique et de programmation

Juillet 2002

On veut exécuter un ensemble $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ de $n = 200$ tâches. Les durées de ces tâches sont stockées dans un tableau W de taille n. Le tableau est initialisé comme suit. On définit la suite récurrente d'entiers $(x_i)_{0 \leq i \leq n}$ par :

– x_0 est égal au numéro inscrit sur votre table d'examen (on pourra prendre toute valeur entre 23 et 30, par exemple),

– $x_i = (x_{i-1} \times x_{i-1})$ modulo 3953 pour i compris entre 1 et n.

Puis, pour $1 \leq i \leq n$, on pose $W[i] = (x_i \text{ modulo } 10) + 1$.

Partie I. Préliminaires

Question I.1. Combien d'éléments du tableau W sont égaux à 5 ? S'il en existe, quel est l'indice de l'avant-dernier d'entre eux ?

Question I.2. Quelles sont la ou les valeurs qui apparaissent le plus souvent dans le tableau W ?

On dispose d'un ensemble $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ de m machines identiques, capables d'exécuter n'importe quelle tâche. Une machine ne peut exécuter qu'une tâche à la fois. L'exécution débute au top $t = 0$. Si une machine commence l'exécution de la tâche T_i au top t, elle l'exécute sans interruption dans l'intervalle $[t, t + W[i][$; au top $t + W[i]$ l'exécution est terminée, et la machine devient disponible pour une autre tâche.

On veut construire un ordonnancement, c'est-à-dire associer à chaque tâche T_i un top de début d'exécution $\sigma(i)$ et un numéro de machine d'affectation $\mu(i)$, compris entre 1 et m, qui détermine la machine sur laquelle T_i est exécutée. On note D la durée totale de l'exécution : $D = \max_{1 \leq i \leq n} \sigma(i) + W[i]$.

Partie II. Tâches indépendantes

On suppose dans cette partie que chaque tâche peut être exécutée à n'importe quel top. Le principe général des algorithmes étudiés est de parcourir la liste des tâches selon un certain ordre ; la tâche courante sera exécutée sur une machine donnée en fonction des décisions déjà prises par l'algorithme pour les tâches précédentes.

On commence par l'algorithme *glouton* suivant. On prend les tâches dans l'ordre initial, pour i variant de 1 à n. On affecte T_i à la machine qui permet de commencer son exécution au plus tôt, compte tenu des décisions déjà prises. En cas d'égalité, on affecte T_i à la machine d'indice le plus petit.

Par exemple, suivons le début de l'algorithme avec $m = 3$ machines. On aura toujours $\sigma(1) = \sigma(2) = \sigma(3) = 0$, $\mu(1) = 1$, $\mu(2) = 2$, et $\mu(3) = 3$. Si $W[1] = 3$ et $W[2] = W[3] = 1$, on aura $\sigma(4) = 1$ et $\mu(4) = 2$ (cas d'égalité au top 1). Ainsi de suite ...

Question II.1. On suppose $m = 2$ dans cette question. On cherche à calculer D.

1. Expliquer brièvement l'algorithme que vous mettez en oeuvre.
2. Définir parmi les opérations élémentaires effectuées par votre programme celles qui sont les plus significatives (justifier votre choix) ; indiquer (en ordre de grandeur) leur nombre en fonction de n.
3. Que valent D, $\sigma(45)$ et $\mu(127)$?

Question II.2. On suppose $m = 5$ dans cette question. Que valent D, $\sigma(45)$ et $\mu(127)$?

Question II.3. On décide de trier les tâches par durée décroissante : on définit une permutation p de $[1..n]$ telle que

$$p[i] < p[j] \Leftrightarrow (W[i] < W[j]) \text{ ou } (W[i] = W[j] \text{ et } i < j).$$

On exécute l'algorithme précédent en considérant les tâches dans l'ordre donné par p. Reprendre les deux questions précédentes, en expliquant brièvement l'algorithme que vous mettez en oeuvre.

Partie III. Tâches indépendantes, avec dates de terminaison imposées

Les tâches ont maintenant des dates de terminaison imposées : pour tout i, on cherche à terminer la tâche T_i au plus tard à la date $Z[i]$, c'est-à-dire obtenir l'inégalité $\sigma(i) + W[i] \leq Z[i]$. Le tableau Z est initialisé comme suit : $Z[i] = i$ si i est pair et $Z[i] = 2i$ si i est impair, où $1 \leq i \leq n$.

Au cours de l'exécution de l'algorithme d'ordonnancement, il se peut qu'une tâche ne puisse être terminée avant sa date de terminaison imposée. Dans ce cas, on supprime la tâche en question (on ne l'exécute pas), on paie une pénalité, et on continue avec la tâche suivante.

Question III.1. On suppose $m = 3$ dans cette question. Combien de pénalités paie l'algorithme glouton précédent qui examine les tâches dans l'ordre initial ? Que vaut alors D ? Expliquer brièvement l'algorithme que vous mettez en oeuvre.

Question III.2. On suppose toujours $m = 3$ dans cette question. Combien de pénalités paie l'algorithme qui examine les tâches "les plus urgentes" d'abord, c'est-à-dire dans l'ordre donné par la permutation q de $[1..n]$ telle que

$$q[i] < q[j] \Leftrightarrow (Z[i] < Z[j]) \text{ ou } (Z[i] = Z[j] \text{ et } i < j) \text{ ?}$$

Que vaut alors D ? Expliquer brièvement l'algorithme que vous mettez en oeuvre.

Partie IV. Tâches avec dépendances

On définit une matrice E de taille $n \times n$ dont les éléments valent 0 ou 1 comme suit :

$$\text{Pour } 1 \leq i, j, \leq n, \quad E[i,j] = 1 \Leftrightarrow i < j \text{ et } (x_{(i+j)\div 2} \text{ modulo } 13) \in \{1, 8\}$$

(où la suite (x_i) est celle définie au début de l'énoncé). Si $E[i,j] = 1$, on dit qu'il y a une relation de dépendance de T_i vers T_j : l'exécution de T_i devra être terminée avant que celle de T_j ne puisse commencer. L'ordonnancement doit respecter toutes ces contraintes :

$$E[i,j] = 1 \Rightarrow \sigma(i) + W[i] \leq \sigma(j).$$

Par contre les tâches peuvent toujours être exécutées sur n'importe quelle machine. Lorsque $E[i,j] = 1$ on dit que T_i est un prédécesseur de T_j ; de plus, lorsqu'il n'existe pas d'indice k tel que $E[i,k] = 1$ et $E[k,j] = 1$, on dit que T_i est un prédécesseur direct de T_j.

Question IV.1. Combien y-a-t-il de tâches sans aucun prédécesseur ?

Les tâches ont une priorité donnée par la valeur du tableau P défini plus bas ; T_i est d'autant plus prioritaire que $P[i]$ est grand.

L'algorithme affecte les tâches aux machines en gérant un tableau d'occupation des machines. On dit qu'une tâche (pas encore affectée à une machine) est prête si tous ses prédécesseurs directs ont été exécutés. Au début, seules sont prêtes les tâches sans aucun prédécesseur. On procède comme suit :

- au début toutes les machines sont libres ;
- on considère le premier instant t où une ou plusieurs machines se libèrent (information à extraire du tableau d'occupation des machines) ;
- on met à jour l'ensemble des tâches prêtes ;
- à cet instant t, supposons que q machines se libèrent, d'indices $p_1 \leq p_2 \leq \ldots \leq p_q$. Soit r le nombre de tâches prêtes à l'instant t, et $s = \min(q, r)$;
- on débute au temps t l'exécution des s tâches prêtes de de plus grande priorité : celle de priorité maximale est affectée à la machine d'indice p_1, la seconde à la machine p_2, etc.
- on met à jour le tableau d'occupation des machines, et on recommence tant qu'il reste des tâches non exécutées.

Question IV.2. Pour cette question seulement, on considère le petit exemple illustratif de la Figure 1, où les valeurs de W, E, P n'ont rien à voir avec le reste de l'énoncé. On suppose $m = 3$. Donner les 6 valeurs manquantes de σ et de μ (vous devez trouver que $D = 4$).

Matrice des dépendances : tous les éléments de E sont égaux à 0, sauf les neuf éléments suivants :

$$E[1,2] = E[1,5] = E[3,9] = E[4,5] = E[5,7] = E[6,7] = E[6,8] = E[6,9] = E[7,8] = 1$$

Tâches	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Durée W	1	1	1	1	1	1	1	1	1
Priorité P	5	3	2	6	7	9	8	4	1
Ordonnancement σ	0	1	1	0	1	0			
Machine μ	3	2	3	2	1	1			

FIG. 1 – Exemple pour les tâches avec dépendances

Question IV.3. On suppose $m = 3$ et que $P[i] = i$ pour $1 \leq i \leq n$. Que valent D, $\sigma(55)$ et $\mu(27)$? Expliquer brièvement l'algorithme que vous mettez en oeuvre.

Question IV.4. On suppose $m = 5$ et que $P[i] = n + 1 - i$. Que valent D, $\sigma(55)$ et $\mu(27)$?

Question IV.5. On suppose que $m = 4$ et que $P[i] = 2i + 1$ si i est pair, $P[i] = 2(n + 1 - i)$ si i est impair, $1 \leq i \leq n$. Que valent D, $\sigma(55)$ et $\mu(27)$?