Institut Galilée
Sup Galilée

INFO1
Année 2022-2023

Algorithmique
Devoir

1 Modalités

Le devoir est à déposer *individuellement* et *obligatoirement* sur l'ENT avant le vendredi **10 Février 2023 à 19h**. Il sera impossible de rendre ce devoir après cette date et heure, et cela entraînera donc la note de 0 (tout plagiat sera noté de la même manière). Dans le cas de devoirs trop semblables, la note obtenue sera divisée par le nombre d'étudiants concernés.

On déposera **un seul fichier archive (.tgz, .tar.gz, .zip, gzip)** contenant :

- un **rapport en pdf** contenant uniquement les réponses aux questions du devoir (produit par un logiciel de traitement de texte, depuis LaTeX, ou simplement un document manuscrit numérisé). Il n'est pas utile de décrire/commenter le code dans ce rapport.
- les **sources (.c et .h)** et makefile éventuel
- les **tests numériques** (à inclure dans le rapport ou sous la forme de **fichiers texte**)
- Eventuellement un fichier texte contenant des indications sur la façon de compiler/lancer le programme (arguments nécessaires, fichiers d'entrée utilisés, etc...).

Ne pas déposer de fichier binaire (exécutable), ni de doc, odt, docx, TeX...

2 Description du problème à résoudre

Planification de calculs sur deux machines

On veut exécuter un ensemble de calculs numériques pour lesquels on a besoin de machines spécifiques. Par chance, on dispose d'un accès à deux super calculateurs, notés A et B. Cependant, notre travail n'est pas prioritaire. Ainsi, une partie du temps de calcul est déjà réservée pour d'autres applications pour chaque minute sur chacun des calculateurs. On ne dispose donc que de ce qui reste.

Nos calculs ne sont pas parallélisables donc on ne peut pas utiliser simultanément les deux calculateurs.

On peut choisir de faire "migrer" à chaque minute le calcul en cours d'une machine à l'autre. Cependant, le temps de déplacer les données et le contexte (sauvegarde et restitution), il faut une minute complète pour faire la migration.

Enfin, la charge (i.e. le temps de calcul utilisé par les autres travaux qui sont prioritaires) des deux calculateurs est connue à l'avance pour chaque minute sur une période donnée de n minutes.

L'objectif est d'optimiser les changements de calculateurs (les migrations de l'un à l'autre) afin d'avoir le plus de temps de calcul disponible sur une période donnée de n minutes.

3 Etude préliminaire et approche heuristique

3.1 Chacun des tableaux suivants correspond à une instance du problème (trois avec $n = 5$ et une avec $n = 10$). Pour chaque minute (valeur de t), on indique le nombre de secondes qui sont disponibles pour notre calcul sur les calculateurs A (valeurs de $A[1 \dots n]$) et B (valeurs de $B[1 \dots n]$). Déterminer dans chaque cas une solution optimale (on ne demande pas de prouver l'optimalité mais quelques commentaires/justifications sont les bienvenus).

t	1	2	3	4	5
A	40	40	10	30	30
B	20	20	60	20	20

t	1	2	3	4	5
A	50	10	10	30	30
B	10	20	50	20	40

t	1	2	3	4	5
A	40	40	10	10	60
B	20	30	60	60	10

t	1	2	3	4	5	6	7	8	9	10
A	10	10	10	60	10	10	10	50	40	10
B	20	40	20	10	10	60	20	10	10	60

3.2 Afin de tenter de résoudre le problème pour une valeur de n donnée, on décide de faire appel deux fois à l'algorithme ci-dessous : on lance $Planif(n, A, B, 1)$ puis $Planif(n, A, B, 0)$ et on conserve la meilleure des deux valeurs renvoyées.

Algorithme 1 Planif(n : entier, $A[1..n]$, $B[1..n]$: tableaux d'entiers, calculateur : booléen) : entier

```
Temps ← 0 ; t ← 1 ;
tantque t ≤ n faire
  si calculateur == 1 alors
    Afficher("A la minute",t,"calcul sur A") ;
    Temps ← Temps + A[t] ;
    si t ≥ n − 1 ou A[t + 1] + A[t + 2] > B[t + 2] alors
      t ← t + 1 ;
    sinon
      t ← t + 2 ; calculateur ← 0 ;
    finsi
  sinon
    Afficher("A la minute",t,"calcul sur B") ;
    Temps ← Temps + B[t] ;
    si t ≥ n − 1 ou B[t + 1] + B[t + 2] > A[t + 2] alors
      t ← t + 1 ;
    sinon
      t ← t + 2 ; calculateur ← 1 ;
    finsi
  finsi
fin tantque
Retourner(Temps) ;
```

Donner la complexité de cette approche puis établir qu'elle ne donne pas toujours une solution optimale en exhibant une instance bien choisie.

4 Algorithmes de résolution exacte

4.1 Combien y a-t-il de choix d'exécutions possibles pour n minutes ?

4.2 On note $f(t, x)$ le temps maximum disponible entre la minute 1 et la minute t (incluse) en supposant qu'à la minute t le calcul est fait sur la machine x ($x = A$ ou $x = B$). Que vaut $f(1, A)$? $f(1, B)$? $f(2, A)$? $f(2, B)$?

4.3 On suppose qu'à la minute $t - 1$ on a effectué le calcul sur la machine A. Que vaut alors $f(t, A)$?

4.4 On suppose qu'à la minute $t - 2$ on a effectué le calcul sur la machine B puis qu'à la minute $t - 1$ on a migré sur A. Que vaut alors $f(t, A)$?

4.5 Que vaut $f(t, A)$ dans le cas général ? $f(t, B)$?

4.6 Par commodité, on note $\overline{A} = B$ et $\overline{B} = A$ (on considère A et B comme des valeurs booléennes *vrai*/*faux*). Donner alors l'expression de $f(n, x)$ dans le cas général.

4.7 Écrire un algorithme récursif qui calcule $f(n, x)$. Montrer que sa complexité est exponentielle.

4.8 Le temps de calcul que l'on cherche à calculer est $\max(f(n, A), f(n, B))$. Écrire un algorithme de programmation dynamique fondé sur le calcul de $f(t, A)$ et $f(t, B)$ pour t dans $\{1, \ldots, n\}$. On utilisera un tableau à deux dimensions pour calculer ces valeurs. Donner la complexité de l'algorithme.

4.9 Implémenter votre algorithme en C et le tester sur les instances de l'énoncé (et sur d'autres si vous le souhaitez !). On affichera la valeur optimale ainsi que la solution correspondante, i.e. le placement du calcul sur A ou B en fonction de $t \in \{1, \ldots, n\}$.

5 Pour les courageux

On généralise le problème à m super calculateurs $A_1, \ldots, A_m$. On suppose qu'ils sont "en ligne" et que le temps de migration de A_i à A_j est $|i - j|$ minutes.

5.1 Donner l'expression récursive du temps maximal disponible.

5.2 Donner un algorithme de programmation dynamique qui calcule cette valeur et renvoie la solution correspondante. Implémenter et tester votre algorithme.