Sup Galilée

INFO1
Année 2024-2025

Algorithmique
Devoir

1 Modalités

Le devoir est à déposer *individuellement* et *obligatoirement* sur l'ENT au plus tard le vendredi **20 décembre 2024**. Il sera impossible de rendre ce devoir après cette date, et cela entraînera donc la note de 0 (tout plagiat sera noté de la même manière). N'attendez donc pas la dernière minute pour le déposer !

Dans le cas de devoirs trop semblables, la note obtenue sera divisée par le nombre d'étudiants concernés.

Un seul fichier sera déposé (**.tar.gz** ou **.zip**). Il contiendra :

- un rapport succinct en pdf (réponses aux questions du devoir). Le document peut être une numérisation d'un rapport manuscrit mais attention à la taille du fichier.
- les sources (.c et .h) et makefile éventuel.
- les tests numériques (à inclure dans le rapport ou sous la forme de fichiers texte).
- Un fichier texte contenant des indications sur la façon de compiler/lancer le programme sera le bienvenu (arguments nécessaires, fichiers d'entrée utilisés, etc...).

  Aucun fichier binaire (exécutable) ne devra être fourni.

2 Chasse au trésor

Un chasseur de trésors se trouve dans un bâtiment rectangulaire d'un seul étage constitué d'un ensemble de salles de même taille et formant donc une grille $n \times m$. Chaque salle est repérée par ses coordonnées (i, j) et contient des objets d'une valeur entière `valeur[i,j]`.

Voici ci-dessous une instance possible de bâtiment. Dans chaque salle est inscrite la valeur des objets qui s'y trouvent.

0	3	5	4	2	3	2	4
2	3	6	1	1	1	1	6
4	1	1	2	4	3	3	2
4	2	5	3	3	4	3	5
2	1	4	4	2	3	2	3
2	3	4	3	1	2	1	2
5	4	1	2	3	1	1	1
5	3	2	3	2	1	2	0

Le chasseur est initialement dans la salle $(1, 1)$ (sans pouvoir ressortir du bâtiment par celle-ci) située dans le coin supérieur gauche du bâtiment et doit atteindre l'unique sortie par la salle (n, m) située dans le coin en bas à droite en se déplaçant successivement d'une salle (i, j) vers la salle $(i, j + 1)$ ou vers la salle $(i + 1, j)$ (i.e. vers la droite ou le bas depuis la salle où il se trouve en considérant le bâtiment vu du dessus).

Le chasseur collecte les objets dans les salles qu'il traverse. Son objectif est bien sûr de collecter un ensemble d'objets d'une valeur totale maximale.

2.1 Borne sur le nombre de solutions admissibles

1. Combien y a-t-il de chemins possibles pour le chasseur entre la case (i, i) et la case $(i + 1, i + 1)$?
2. Combien y a-t-il de chemins possibles pour le chasseur entre la case (i, i) et la case $(i + 2, i + 2)$ qui passent par la case $(i + 1, i + 1)$?
3. Combien y a-t-il de chemins possibles pour le chasseur entre la case $(1, 1)$ et la case (n, n) qui passent par toutes les cases de la diagonale (i, i) ?
4. Quel est le nombre de chemins possibles pour le chasseur entre la case $(1, 1)$ et la case (n, m) ?
5. Peut-on envisager une approche par énumération pour résoudre ce problème ?

2.2 Approche récursive

Soit `tresor(i,j)` la valeur maximale qu'il est possible d'accumuler en partant de la salle (i, j) et en allant jusqu'à la sortie. Rappel : on note `valeur[i,j]` la valeur des objets se trouvant dans la salle de coordonnées (i, j).

1. En supposant que le déplacement depuis (i, j) est vers le bas, que vaut `tresor(i,j)` en fonction de `valeur[i,j]` et d'un appel récursif ?
2. En supposant que le déplacement depuis (i, j) est vers la droite, que vaut `tresor(i,j)` en fonction de `valeur[i,j]` et d'un appel récursif ?
3. Que valent `tresor(n,m)`, `tresor(i,m)`, `tresor(n,j)` ?
4. Dans le cas général, que vaut `tresor(i,j)` ?
5. En déduire un algorithme récursif qui calcule `tresor(1,1)`.
6. Montrer que la complexité de l'algorithme récursif est au moins exponentielle. On pourra considérer le cas particulier où $n = m$, et on comptera le nombre de comparaisons (ou de max).

2.3 Programmation dynamique

On utilise à présent un tableau `gain` où `gain[i,j]` contiendra la valeur `tresor(i,j)`.

1. Modifier l'algorithme récursif précédent pour utiliser le tableau `gain` (on supposera que le tableau est initialisé avec une valeur adéquate, par exemple -1).
2. Dans quel ordre peut-on calculer les cases de `gain` pour être sûr de n'avoir besoin à chaque fois que de résultats déjà calculés ?
3. En déduire un algorithme itératif fondé sur la programmation dynamique pour calculer le tableau `gain`. Quelle est sa complexité ?
4. Compléter l'algorithme afin d'obtenir également un chemin optimal pour le chasseur de trésors.

2.4 Section bonus : variante du problème

On considère à présent que les déplacements peuvent aussi s'effectuer en $(i-1, j)$ et $(i, j-1)$ à partir de (i, j) (lorsque les valeurs de i et j le permettent). De plus, on suppose que le chasseur décide de visiter au plus k salles (k entier positif donné) avant de se rendre à la sortie. En effet, il n'est probablement pas seul dans le bâtiment et craint de faire de mauvaises rencontres.

1. A quelle condition sur la valeur de k le chasseur peut-il sortir du bâtiment ?
2. Pour quelles valeurs de k une solution optimale existe et est évidente ?
3. Proposer des adaptations des algorithmes des sections 2.2 et 2.3.

2.5 Tests numériques

1. Implémenter en C l'algorithme récursif de la section 2.2 ainsi que celui issu de la programmation dynamique (version itérative) de la section 2.3.
2. Question bonus : si vous avez traité la section 2.4 alors vous pouvez également implémenter ces versions.
3. Donner et comparer les résultats obtenus vos algorithmes pour l'instance de la section 2 puis les tester sur d'autres instances que vous proposerez.