IHDC B331 - Algorithmique II
Projet : Méthodes de Conception d'Algorithmes

Elise Hallaert

March 13, 2023

Contents

1 Introduction

Ce document décrit les spécifications du projet d'Algorithmique II (IHDC B331). L'objectif est d'appliquer les concepts acquis dans le cadre du cours pour résoudre les problèmes proposés.

Les compétences requises sont:

- Maîtrise du langage Java
- Utilisation du langage JML pour spécifier le comportement des classes et des interfaces Java
- Approche de résolution algorithmique diviser pour régner
- Approche gloutonne
- Approche par programmation dynamique

Le projet est à réaliser par groupe de 3 étudiants ou individuellement pour les étudiants le requérant.

2 Dates importantes

Pour le **17 mars**, les groupes formés doivent être renseignés sur Webcampus dans la rubrique appropriée.

Pour le **6 mai à 23h59**, les programmes ainsi qu'un rapport décrivant la méthodologie, une description de vos programmes, leur structure et points-clés, et toute information vous semblant pertinente, sont à rendre. Rigueur, clarté, définitions et justifications sont donc attendus. Relisez-le avant la soumission!

3 Énoncés

3.1 Diviser pour régner: l'invasion de plante carnivore

Afin de déterminer la biodiversité de ses champs en friche, Bob possède une base de données qu'il remplit suivant la constitution de son champ. Pour chaque rangée, il indique, une par une, les plantes qu'il rencontre. Une séquence dans ses notes peut ressembler à ceci:

$$S = coquelicot, rose, chardon, coquelicot, marguerite, coquelicot, pissenlit, coquelicot$$

Une plante est considérée envahissante si elle est présente strictement plus de M/2 fois dans une ligne de M plantes.

Pour aider Bob, l'objectif est de parvenir à trouver **la** plante envahissante, s'il y en a une. L'algorithme prendra en entrée un fichier dont la première ligne contiendra le nombre de rangées n, suivie de 2*n lignes constituées, pour chaque rangée, du nombre de plantes sur une ligne et de la séquence des plantes sur la suivante.

Pour chaque rangée, l'algorithme renvoie le nom de la plante envahissante s'il y en a une, ou *null* sinon.

3.1.1 Exemple

Entrée :

2
9
coquelicot rose chardon coquelicot marguerite coquelicot pissenlit coquelicot coquelicot
5
kudzu violette pissenlit pissenlit ambroisie

Sortie :

coquelicot
null

3.1.2 Analyse, Spécification et Implémentation

Le problème que vous devez analyser, spécifier et implémenter est un problème destiné à l'approche diviser pour régner. Vous suivrez les étapes suivantes:

1. Analysez l'énoncé: détectez les ambiguités, les contradictions, etc...,
2. Spécifiez le problème en utilisant JML (pré et post conditions),

3. Construisez une solution naïve permettant de trouver la plante envahissante,
4. Exprimez l'invariant et variant de boucle. Prouvez la correction de l'invariant et la terminaison de boucle,
5. Donnez la complexité de cet algorithme naïf,
6. Construisez une solution plus efficace,
7. Spécifiez la fonction récursive en utilisant JML (pré et post conditions),
8. Implémentez la solution basée sur le principe "diviser pour régner",
9. Donnez la complexité de votre algorithme basé sur l'approche "diviser pour régner".

3.2 Programmation dynamique: le roi des bleus

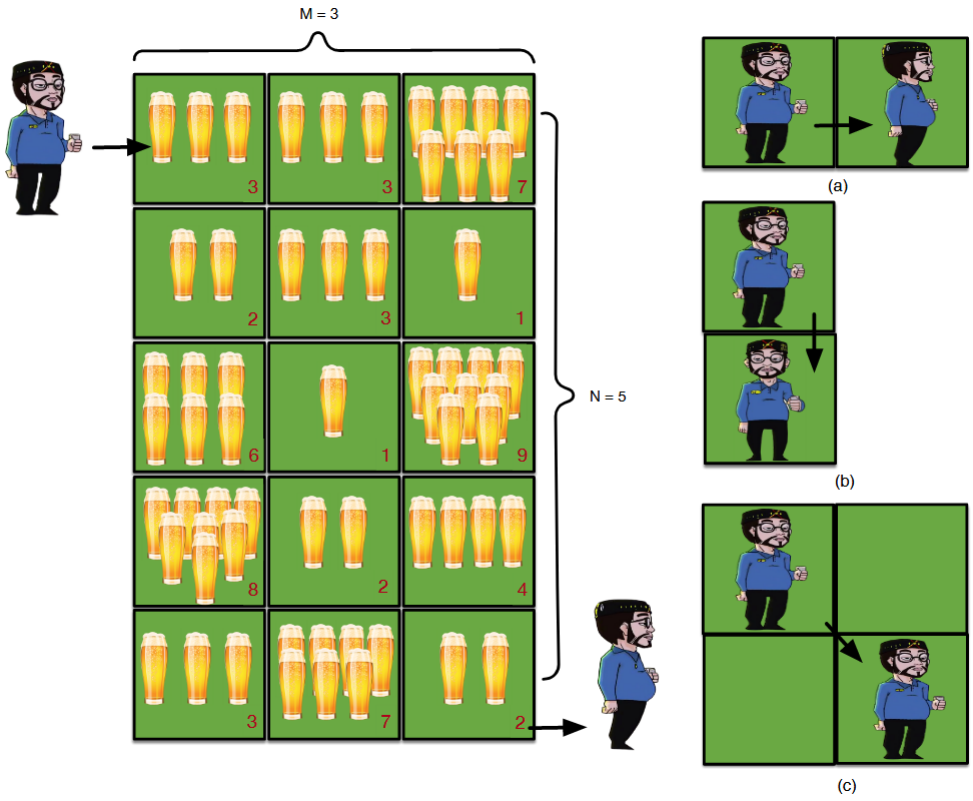
L'objectif du problème est de développer et d'implémenter un algorithme permettant d'aider le futur Roi des Bleus (Frank) à s'entraîner à trouver le nombre maximum (au plus $\leq$ MAX) de bières qu'il peut boire en faisant des pas vers le bas, vers la droite et en diagonale vers le bas à droite sur une grille ($N \times M$) conçue dans le cercle de la faculté d'informatique. Ce problème vous permettra de mettre en œuvre les techniques de résolution de problèmes vues au cours théorique.

La semaine prochaine aura lieu le concours déterminant le prochain roi des bleus et Frank veut participer et tout faire pour s'assurer la première place. L'épreuve aura lieu au cercle où le sol est divisé en grille de $N \times M$ cases, c'est à dire N lignes et M colonnes. Chaque case contient un certain nombre de bières que chaque participant doit boire s'il se déplace sur cette case lors de son épreuve. Le participant commence en (0,0), c'est-à-dire en haut à gauche et doit arriver en $N \times M$ en buvant toutes les bières qu'il rencontre sur son passage. Le participant peut donc se déplacer vers la droite, le bas, ou en diagonale vers le bas à droite mais ne peut jamais sortir du tableau.

Seulement, tous les participants auront une limite K ($\leq$ K) de bières qu'ils ne pourront pas dépasser sous peine d'être disqualifié.

L'objectif, pour aider Frank, est de trouver le nombre maximum de bières qu'il peut boire sans dépasser la quantité maximale de K. Si aucune solution n'est possible, l'algorithme renvoie -1

L'algorithme prendra en entrée un fichier dont la première ligne est un entier n reprenant le nombre de cas à tester. Ensuite, pour chaque cas, la première ligne est constituée de 3 naturels *N, M et K* représentant le nombre de lignes, de colonnes et la limite à ne pas dépasser. Les N lignes suivantes contiennent M naturels représentant le nombre de bières dans chaque case.



3.2.1 Exemple

Entrée :

```
4
5 3 10
3 3 7
2 3 1
6 1 9
8 2 4
3 7 2
1 1 7
8
3 3 7
2 3 1
6 1 9
8 2 3
3 4 9
2 3 4 1
6 5 5 3
5 2 3 4
```

Sortie :

10
-1
6
-1

3.2.2 Analyse, Spécification et Implémentation

Le problème que vous devez analyser, spécifier et implémenter est un problème destiné à l'approche diviser pour régner. Vous suivrez les étapes suivantes:

1. Analysez l'énoncé: détectez les ambiguités, les contradictions, etc...,
2. Spécifiez le problème en utilisant JML (pré et post conditions),
3. Construisez une solution naïve permettant de trouver le nombre maximum de bières que Frank peut boire,
4. Exprimez l'invariant et variant de boucle. Prouvez la correction de l'invariant et la terminaison de boucle,
5. Donnez la complexité de cet algorithme naïf,
6. Déterminez la sous-structure optimale dont on a besoin pour résoudre ce problème,
7. Caractérisez (par une équation récursive) cette sous-structure optimale,
8. Spécifiez ce problème en utilisant JML (pré et post conditions),
9. Ecrivez un algorithme base sur le principe de programmation dynamique pour trouver le nombre maximum de bières que Frank peut boire
10. Donnez la complexité de votre algorithme basé sur l'approche "diviser pour régner".

3.3 Le marchand

Afin de vendre ses marchandises, le marchand de la tribu de la musaraigne grise doit marcher deux jours dans la pampa. Étant donné que le trajet lui semble très long, il espère gagner un maximum d'argent en remplissant son sac à dos d'objets de grande valeur. Malheureusement, certains objets valant cher sont très lourds et le marchand fera le trajet à pied. L'objectif est de trouver et implémenter un algorithme permettant d'aider le marchand à gagner un maximum d'argent en trouvant une solution optimale. L'algorithme devra trouver une valeur optimale qui peut être contenue dans le sac à dos sans dépasser un poids K ($\leq$ K).

L'algorithme prendra en entrée un fichier dont la première ligne contiendra un naturel représentant le nombre de cas test. Les lignes suivantes seront, pour chaque cas test, une ligne avec deux naturels N et M représentant, respectivement, le nombre de produits possibles à vendre et le poids maximal à ne pas dépasser, suivie par N lignes constituées de deux naturels I J correspondant, quant à eux, à la valeur et au poids de chaque objet. La sortie sera la valeur optimale maximale que le marchand peut emporter.

3.3.1 Exemple

Entrée :
2
5 17
3 8
5 1
12 13
15 1
8 3
2 3
1 1
5 2

Sortie :

10
-1
6
-1

3.3.2 Analyse, Spécification et Implémentation

Le problème que vous devez analyser, spécifier et implémenter est un problème destiné à l'approche gloutonne. Vous suivrez les étapes suivantes:

1. Analysez l'énoncé: détectez les ambiguités, les contradictions, etc...,
2. Spécifiez le problème en utilisant JML (pré et post conditions),
3. Construisez une solution naïve permettant de trouver le montant maximal des objets que le marchand peut porter,
4. Exprimez l'invariant et variant de boucle. Prouvez la correction de l'invariant et la terminaison de boucle,
5. Donnez la complexité de cet algorithme naïf,
6. Déterminez la propriété gloutonne dont vous avez besoin pour résoudre le problème,
7. Montrez que dans chaque étape de la récursivité, le choix glouton est correct (mène vers une solution optimale) et démontrez la récursivité,
8. Spécifiez ce problème en utilisant JML (pré et post conditions),
9. Ecrivez un algorithme basé sur le principe glouton pour trouver le montant maximal que le marchand peut porter,
10. Donnez la complexité de votre algorithme basé sur l'approche gloutonne.

4 Contraintes techniques, organisationnelles et évaluation

Il vous est demandé de coder en Java (version 8) et d'utiliser JML pour les 3 problèmes décrits ci-dessus. Le projet est à réaliser par groupe de **3 personnes** ou individuellement si cela est requis. La participation de tous les membres à l'élaboration des algorithmes est requise.

5 Rapport écrit

Le rapport devra être soumis en version **électronique** sur WebCampus avec le code et sera rédigé dans un français clair, correct et précis (Relisez avant de soumettre!). Un soin particulier sera apport à ce que le rapport soit pratique et agréable à lire. Le rapport sera écrit en Latex et contiendra les éléments suivants:

1. Titre, nom et prénom des étudiants
2. Introduction
3. un calendrier du projet, qui peut être extrait à partir de votre repository git à condition qu'il soit remis en page pour être lu aisément,
4. Une description de votre démarche générale et votre méthodologie pour analyser, spécifier et implémenter vos algorithmes ainsi que vos choix d'implémentation. Jetez également un regard critique sur le déroulement de votre projet,
5. Une description de vos structures de données, fonctions et classes pour chaque algorithmes et la justification de vos choix,
6. Une conclusion reprenant ce que vous avez appris,
7. Toute information qui vous semble pertinente.

Attention: l'utilisation d'un outil de génération de code implique que vous n'êtes **pas** l'auteur de votre code. L'entièreté de vos délivrables doivent être le fruit de votre propre travail.

6 Outils à utiliser

Git pour faciliter vos développements avec un repository **privé**. En cas de fuite ou de plagiat, tous les étudiants impliqués seront sanctionnés.

Le langage de programmation Java.

Le langage OpenJML pour spécifier le comportement des classes et des interfaces Java (Specification avec JML).

Le solveur Z3 pour vérifier la spécification écrite en JML et l'implémentation.

Latex pour la rédaction du rapport