Algorithme
2D PACKING
Rakotoarimalala Tsinjo

1 Définition

Le problème de 2D-packing, ou de placement bidimensionnel, est un problème d'optimisation combinatoire qui consiste à agencer des objets bidimensionnels dans un contenant bidimensionnel de manière à minimiser un certain critère, comme l'espace gaspillé, ou à maximiser l'utilisation de l'espace disponible. Ce problème a de nombreuses applications pratiques, notamment dans l'industrie manufacturière, la conception de circuits intégrés, et la logistique.

2 1D Packing

2.1 Définition

Le problème consiste à placer un ensemble d'objets de tailles différentes dans un nombre minimal de bacs (ou conteneurs) de taille fixe. Chaque objet doit être entièrement contenu dans un seul bac.

2.2 Formulation

Le problème de 1D packing peut être formulé de la manière suivante :

Entrée

- Un ensemble d'objets $O = \{o_1, o_2, \ldots, o_n\}$, où chaque objet o_i a une taille s_i telle que $s_i \in \mathbb{R}^+$.
- Une taille de bac fixe $B \in \mathbb{R}^+$.
- Le nombre de bac n'est pas défini en avance, on ajoutera des bacs au fur et à mesure des besoins.

Sortie

Affectation de chacun des objets avec le numéro de bac dans lequel il est placé.

Objectif

Minimiser le nombre de bacs utilisés pour contenir tous les objets.

2.3 Algorithmes de placement

- **First-Fit (FF)** : Insérer chaque objet dans le premier bac disponible où il peut rentrer.
- **Best-Fit (BF)** : Insérer chaque objet dans le bac qui laissera le moins d'espace restant après insertion.
- **Worst-Fit (WF)** : Insérer chaque objet dans le bac qui laissera le plus d'espace restant après insertion.

2.4 Questions

1. Implémenter les trois types d'algorithmes de placement dans la section précédente.
2. Pour chaque algorithme de placement énoncé ci-dessus, trouver un exemple de données d'entrées pour lequel l'algorithme ne donne pas la solution optimale c'est-à-dire il utilise plus de bac que nécessaire.
3. Trouver un algorithme brute force (essayer toutes les possibilités) pour résoudre le problème de **1D Packing**

3 2D Packing avec une seule forme sans rotation

3.1 Définition

Le problème consiste à placer un ensemble de rectangles $\mathcal{R}$ de dimensions fixes dans un grand rectangle R de taille fixe de manière de telle sorte qu'on maximise l'utilisation de l'espace disponible dans R.

3.2 Formulation

Entrée

- Un ensemble de rectangles $\mathcal{R} = \{(w_1, h_1), (w_2, h_2), \ldots, (w_n, h_n)\}$, où chaque rectangle i a une largeur w_i et une hauteur h_i.
- Un rectangle de dimension fixe (W, H).

Objectif

Placer les rectangles dans $\mathcal{R}$ en maximisant l'utilisation de l'espace disponible dans R. Certains rectangles de $\mathcal{R}$ peuvent ne pas être utilisés. Les rotations ainsi que les chevauchements des rectangles sont interdits.

3.3 Algorithmes de placement

- **Next-Fit Decreasing Height (NFDH)**
- **Best-Fit (BF)**
- **First-Fit Decreasing Height (FFDH)**

3.4 Questions

1. Renseigner vous sur les trois types d'algorithmes de placement dans la section précédente.
2. Implémenter ces trois algorithmes et utiliser un interface graphique pour saisir les entrées (ajouter les rectangles dans $\mathcal{R}$ et les données sur R) ainsi que pour afficher les résultats pour chacun des algorithmes
3. Pour chaque algorithme de placement énoncé ci-dessus, trouver un exemple de données d'entrées pour lequel l'algorithme ne donne pas la solution optimale c'est-à-dire il utilise plus de bac que nécessaire.
4. Trouver un algorithme brute force (essayer toutes les possibilités) pour résoudre ce problème
5. Et si on autorise les rotations de $\frac{\pi}{2}$ radians, qu'est-ce qui change? implémenter une solution améliorée de la question précédente en autorisant les rotations de $\frac{\pi}{2}$ radians des rectangles.

4 2D Packing avec trois forme avec rotation de $\frac{\pi}{2}$ ou de π

Le problème est le même que dans la partie 3. Mais cette fois-ci on peut avoir plusieurs objets mais uniquement trois formes:

- cercle défini par son rayon
- rectangle défini par sa hauteur et sa largeur
- un triangle isocèle défini par sa base

On autorise aussi les rotations de $\frac{\pi}{2}$ et de π.

4.1 Questions

1. Implémenter un algorithme de placement heuristique (donnant un résultat approximatif) pour résoudre ce problème et utiliser un interface graphique pour saisir les entrées (ajouter les objets dans les entrées) ainsi que pour afficher le résultat.
2. Trouver un exemple de données d'entrées pour lequel l'algorithme ne donne pas la solution optimale c'est-à-dire il utilise plus de bac que nécessaire.
3. Trouver un algorithme brute force (essayer toutes les possibilités) pour résoudre ce problème (les entrées et les résultats utilisent toujours un interface graphique)

5 Règles

1. Interdiction d'utiliser des librairies existantes pour les algorithmes
2. Projet à faire en groupe de 3 personnes