SUPINFO Academic Dept.

SUPINFO International University

INSTITUTE OF INFORMATION TECHNOLOGY

Algorithmique avancée

Mini - Projet

Empilement de boîtes

Version 1.0
Last update: 30/03/2017
Use: Students/Staff
Author: Laurent GODEFROY

SOMMAIRE

0 PREAMBULE 3

1 GENERALITES SUR CE PROJET 4

2 METHODE GLOUTONNE 5

3 RECURRENCE ET RECURSIVITE NAÏVE 5

3.1 UNE FORMULE DE RECURRENCE 5

3.2 UN ALGORITHME RECURSIF NAÏF 6

4 PROGRAMMATION DYNAMIQUE 6

4.1 APPROCHE TOP DOWN 6

4.2 APPROCHE BOTTOM UP 6

5 APPLICATIONS 6

6 BAREME INDICATIF 7

0 PREAMBULE

Cet examen est à réaliser par groupes de deux étudiants. Dans l'unique cas où le nombre d'étudiants de la promotion est impair, un et un seul groupe de trois est autorisé.

Toute forme de plagiat ou utilisation de codes disponibles sur internet ou tout autre support, même de manière partielle, est strictement interdite et se verra sanctionnée d'un 0, d'une mention « cheater », et le cas échéant d'un conseil de discipline.

Ce mini-projet donnera lieu à des soutenances. Vos horaires de passages vous seront communiqués par votre campus.

Les soutenances sont également par groupes de deux. Elles dureront **20 minutes** pendant lesquelles vous montrerez à votre examinateur le bon fonctionnement de votre programme en n'en faisant la démonstration. Si vous n'avez pas implémenté le projet dans sa totalité, vous exposerez les parties fonctionnelles.

Pour appuyer votre présentation, vous devrez préparer un fichier de type Powerpoint, dans lesquels vous expliquerez les points du code que vous jugez les plus importants et significatifs. Il n'est pas nécessaire d'envoyer ce fichier à votre examinateur, ce dernier le découvrira le jour de la soutenance. Une communication précisant tout cela vous sera envoyé au retour des vacances de printemps.

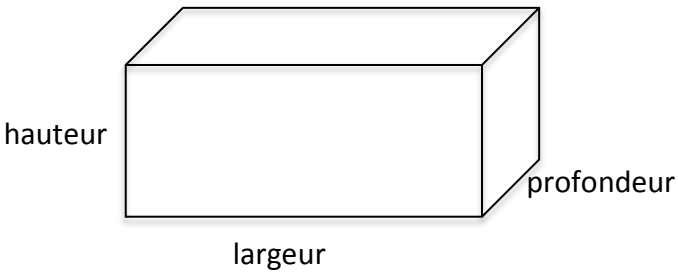
Un barème **indicatif** vous est donné dans la dernière partie de ce sujet.

1 GENERALITES SUR CE PROJET

Remarque importante : aucun code n'est demandé dans cette partie qui n'est qu'explicative.

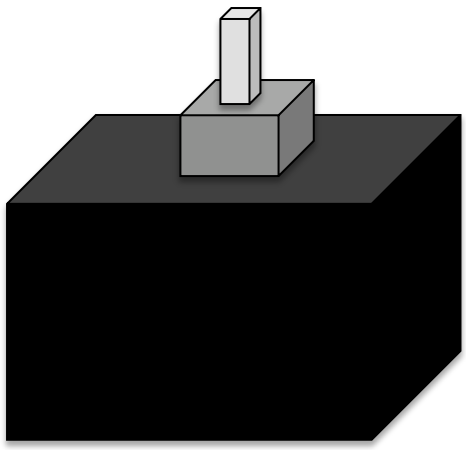
Le but de ce mini-projet est d'écrire en langage Python divers algorithmes permettant de maximiser la hauteur d'un empilement de boîtes.

La donnée du problème est un ensemble de **n** boîtes, plus précisément des pavés droits, chacune étant caractérisée par un triplet (hauteur,largeur,profondeur).



On dispose d'un nombre illimité d'exemplaires de chaque boîte et l'on pourra effectuer toutes les rotations que l'on souhaite, ce qui permettra de permuter les paramètres hauteur, largeur et profondeur.

On ne peut empiler une boîte A au dessus d'une boîte B que si les dimensions de la base de A (largeur **et** profondeur) sont **strictement plus petites** que celles de B.



On cherche à réaliser l'empilement de **hauteur totale maximale**.

Structure de données : notre ensemble de boîtes sera implémenté par une liste de triplets (hauteur,largeur,profondeur).

Indication 1 : on pourra compléter la liste précédente pour tenir compte des rotations et du fait que l'on dispose de plusieurs exemplaires de chaque boîte.

Indication 2 : on considérera les boîtes par ordre décroissant de la valeur de leur surface.

Remarque : dans la suite, lorsque l'on demande une répartition de boîtes il s'agira de retourner une liste de triplets (hauteur,largeur,profondeur).

Exemple : avec les boîtes (2,7,5), (7,6,3), (10,20,5) et (3,4,5) la hauteur maximale d'un empilement est de 39 et correspond à l'empilement (du haut vers le bas) [(7,2,5),(7,3,6),(20,5,10),(5,10,20)].

2 METHODE GLOUTONNE

Proposer et implémenter en Python une méthode gloutonne pour calculer la hauteur maximale d'un empilement (bien penser aux indications 1 et 2) pour un ensemble de boîtes donné. Est-elle optimale ? Si oui justifier pourquoi, sinon proposer un contre-exemple.

Cette fonction retournera la hauteur totale de l'empilement ainsi que la répartition des boîtes correspondante.

3 RECURRENCE ET RECURSIVITE NAÏVE

3.1 UNE FORMULE DE RECURRENCE

Soit **H[i]** la hauteur **maximale** de l'empilement comportant la i-ème boîte à son sommet.

Etablir une formule de récurrence permettant de calculer **H[i]** pour **i** compris entre **1** et **n**, où **n** est

le nombre total de boîtes (bien penser aux indications 1 et 2). Justifier précisément son raisonnement.

3.2 UN ALGORITHME RECURSIF NAÏF

Se servir de la relation de récurrence précédente pour implémenter en Python un algorithme récursif naïf permettant de calculer la hauteur maximale d'un empilement pour un ensemble de boîtes donné.

Estimer la complexité de cet algorithme.

4 PROGRAMMATION DYNAMIQUE

4.1 APPROCHE TOP DOWN

Améliorer l'algorithme de la sous-partie 3.2 en adoptant une approche dynamique Top Down.

Estimer la complexité de cet algorithme.

4.2 APPROCHE BOTTOM UP

Implémenter en Python un algorithme dynamique Bottom Up permettant de calculer la hauteur maximale d'un empilement pour un ensemble de boîtes donné.

Compléter cet algorithme afin de pouvoir obtenir la répartition correspondant à la hauteur maximale.

Estimer la complexité de cet algorithme.

5 APPLICATIONS

Appliquer les algorithmes précédents à l'ensemble de boîtes dont les caractéristiques sont dans le fichier "boxes.txt" joint à ce sujet.

6 BAREME INDICATIF

Ce barème peut-être amené à évoluer, il n'est donc qu'i**ndicatif**.

- Partie 2 : 5 points
- Partie 3: 10 points
- Partie 4 : 20 points
- Partie 5 : 5 points
- Soutenance (à venir) : 20 points

Ce qui fait un total de 60 points. La note totale sera alors ramenée sur 20 points par proportionnalité.