Informatique PCSI
TP 6 suite : algorithmes gloutons

1 Énoncé des exercices

1.1 Exercice 1

Problème du sac à dos

Reprendre le code complet donné dans le document préparatoire pour résoudre le problème du sac à dos. L'objectif est de transformer le programme itératif en un programme récursif. Pour cela on découpe la fonction `glouton` en deux parties. Le code de la première partie, qui permet de trier la liste des données suivant le critère choisi, est le suivant :

```
def solution(liste, poids_max, choix):
    copie = sorted(liste, key=choix, reverse=True)
    return glouton(copie, poids_max, [], 0, 0, 0)
```

Écrire la fonction `glouton` dont les paramètres gardent les mêmes noms que dans le document préparatoire et sont dans l'ordre `liste`, `poids_max`, `reponse`, `valeur`, `poids`, `i`.

Attention, contrairement au problème du rendu de monnaie pour lequel on peut prendre plusieurs fois la même pièce, on ne peut prendre un objet qu'une seule fois. On crée une nouvelle liste pour chaque choix possible qui permet de mettre à jour la liste précédente.

1.2 Exercice 2

Stations d'essence

Le programme donné dans le document préparatoire est itératif. Écrire une version récursive.

1.3 Exercice 3

Programme télé

Une personne doit faire une critique des programmes diffusés à la télévision. Elle souhaite regarder le plus grand nombre de programmes possible dans une journée quel que soit le contenu. Chaque programme `pi`, (émission, film, série), est caractérisé par un intervalle `[di, fi]` où `di` est l'heure de début et `fi` l'heure de fin. La différence `fi-di` représente donc la durée du programme `pi`. Les intervalles de temps des différents programmes regardés doivent être disjoints.

Pour organiser le visionnage des programmes, elle utilise un algorithme glouton. Elle trie les programmes suivant un critère (durée, ou début, ou fin). Ensuite elle choisit le premier et supprime parmi les autres ceux qui ne sont pas compatibles avec ce choix. Elle recommence cette étape tant que la liste des programmes n'est pas vide.

Montrer que les algorithmes 1 et 2 qui suivent ne sont pas optimaux en donnant un contre-exemple. Prouver que l'algorithme 3 est optimal.

- Algorithme 1 : elle trie les programmes par durées croissantes, choisit le plus court, puis le plus court parmi ceux qui restent et qui sont compatibles, et ainsi de suite.
- Algorithme 2 : elle trie les programmes par heures de début croissantes, choisit le programme commençant le plus tôt, puis le plus tôt parmi ceux qui restent et qui sont compatibles, et ainsi de suite.
- Algorithme 3 : elle trie les programmes par heures de fin croissantes, choisit le programme qui se termine le plus tôt, puis le programme qui se termine le plus tôt parmi ceux qui restent et qui sont compatibles, et ainsi de suite.