Informatique PCSI
Corrigé TP 6 suite

1 Énoncé des exercices

1.1 Exercice 1

Problème du sac à dos

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

```
def glouton(liste, poids_max, reponse, valeur, poids, i):
    if i < len(liste) and poids <= poids_max:
        nom, val, pds = liste[i]
        if poids + pds <= poids_max:
            reponse.append(nom)
            poids = poids + pds
            valeur = valeur + val
        return glouton(liste, poids_max, reponse, valeur, poids, i+1)
    else:
        return reponse, valeur
```

Par exemple, avec `solution(objets, 15, poids)`, on obtient `(['objet 4', 'objet 2', 'objet 3', 'objet 5'], 75)`.

1.2 Exercice 2

Stations d'essence

```
def choix(distances, dmax, i, stations, d):
    n = len(distances)
    if i != n:
        if i < n and distances[i] <= d:
            choix(distances, dmax, i+1, stations, d-distances[i])
        else:
            stations.append(i-1)
            choix(distances, dmax, i, stations, dmax)

def stats(distances, dmax):
    stations = []
    choix(distances, dmax, 0, stations, dmax)
    return stations
```

Pour tester le programme, on reprend l'exemple donné dans le document préparatoire et on écrit `s = stats(tab, res)`.

1.3 Exercice 3

- L'algorithme 1 n'est pas optimal : on choisit le programme le plus court, par exemple de 8h30 à 9h30, alors que 2 autres seraient possibles de 7h à 9h et de 9h15 à 11h.
- L'algorithme 2 n'est pas optimal : on choisit le programme qui commence le plus tôt, par exemple de 6h à 8h30 alors que plusieurs seraient possibles : de 6h30 à 7h30, de 7h45 à 8h15, etc.
- L'algorithme 3 est optimal : on peut le démontrer comme pour le problème des stations d'essence à la fin du cours. Soit `f` la date de fin la plus petite et une solution optimale `[f1,f2,...,fk]`, avec $\texttt{f1} < \texttt{f2} < \ldots < \texttt{fk}$. On remplace `f1` par `f` ($\texttt{f} < \texttt{f1}$) et alors `[f,f2,...,fk]` est aussi optimale. On continue avec la date de fin la plus petite parmi les dates compatibles avec `f` qui remplace `f2`, et ainsi de suite.

Remarques

- Un algorithme glouton est optimal si l'on souhaite maximiser le nombre de programmes visionnés. On peut vérifier que ce n'est pas le cas si l'on souhaite maximiser la durée totale de visionnage.
- On peut écrire un programme semblable à celui qui est écrit pour le problème du sac à dos, chaque programme étant accompagné de son heure de début, son heure de fin et sa durée.