Informatique PCSI
TP 6 : algorithmes gloutons

1 Énoncé des exercices

1.1 Exercice 1

On doit rendre la monnaie avec les pièces dont on dispose dans la caisse. Quelle affirmation est exacte ?

- Avec uniquement des pièces de $(1, 4, 5, 10)$, un algorithme glouton est optimal.
- Avec uniquement des pièces de $(1, 3, 4, 10)$, un algorithme glouton est optimal.
- Avec uniquement des pièces de $(1, 3, 4, 5)$, un algorithme glouton est optimal.
- Avec uniquement des pièces de $(1, 3, 5, 10)$, un algorithme glouton est optimal.

1.2 Exercice 2

On considère la fonction `monnaie` donnée dans le document préparatoire.

Cette fonction utilise une liste `solution` pour représenter les pièces rendues. Écrire une fonction qui utilise, à la place de cette liste, un dictionnaire dont les clés sont les valeurs des pièces rendues et les valeurs associées aux clés sont le nombre de pièces correspondantes. Tester la fonction pour rendre 48 euros avec le système $(1, 2, 5, 10, 20, 50, 100)$.

1.3 Exercice 3

On considère la fonction `monnaie` donnée dans le document préparatoire.

Écrire une version récursive de cette fonction.

1.4 Exercice 4

Couleur d'un point

On reprend le programme donné dans le document préparatoire avec les principales fonctions concernant la recherche du plus court chemin passant par des points donnés dans le plan.

1. Importer la fonction `randint` et la fonction `sqrt`.

   Créer une variable `nbpoints = 40` et une variable `dim = 20`, puis une liste de couleurs pour le graphique : `couleurs = ["r", "g", "b"]`.
2. Modifier la fonction `points` afin que les points dans la liste renvoyée par la fonction soient des listes de trois valeurs, les deux coordonnées et la couleur choisie au hasard.
3. Créer un nouveau point comme le point de départ du chemin mais en lui ajoutant la couleur noire. Tracer tous les points dans un graphique.
4. Reprendre la fonction `distance` qui renvoie la distance entre deux points et procéder à la modification nécessaire. Reprendre les fonctions `distances` et `indice` telles quelles.
5. Écrire une fonction `proches_voisins` qui prend en paramètres le tableau des distances et un entier `k` et qui renvoie la liste des indices des `k` plus proches voisins. Il s'agit de s'inspirer grandement de la fonction `plus_court`.
6. Terminer le programme afin que le nouveau point prenne la couleur dominante parmi ses quatre voisins les plus proches et afficher les points sur un graphique.