COMPTE RENDU BE3

Problème de la monnaie rendue

I – Préambule

On définit les constantes suivantes :

- ⇨ monnaie = [200, 100, 50, 20, 10, 5, 2, 1]
- ⇨ dispo = [10, 10, 10, 10, 10, 10, 10, 10]

et on importe la librairie numpy (as **np**) pour utiliser np.inf lors de la phase PrD

II – Solution gloutonne

1) Sans prise en compte de la disponibilité

On commence par implémenter la fonction **gloutonne**(S, M) :

- Retourne un tuple T, sum(T), représentant le vecteur solution et le nombre de pièces qu'il comporte.
- **S** est la liste des monnaies disponibles
- **M** est la valeur du retour de monnaie à effectuer, **exprimé en centimes**
- On utilise **assert(**M**==int(**M**))** pour s'assurer que M est entier.

Il s'agit d'un algorithme glouton récursif, on essaie à chaque étape de rentre un maximum de monnaie sous la forme de « Grosses pièces », on calcule le reste à rendre, puis on recommence.

2) Avec prise en compte de la disponibilité

On définit la fonction **gloutonne_dispo**(S,D,M) :

- **D** est la liste des disponibilités des différentes pièces.
  ⇨ D[i] est la disponibilité de S[i]
- On modifie le test pour déterminer la plus petite valeur disponible : On s'assure que la pièce dont l'indice a été choisi est bien disponible
- Lors de l'exécution, on tronque le nombre de pièces utilisé avec la valeur maximale de pièces disponibles.
  ⇨ Ainsi, on a bien ∀i T[i]<=D[i]
- On retire ensuite les pièces utilisées des pièces disponibles (peu d'intérêt ici)
- On gère le cas où il est impossible de rendre la monnaie avec une exception

Le reste du déroulement de la fonction est identique à **gloutonne**(S, M)

II – En programmation dynamique (PrD)

1) Sans connaitre les pièces utilisées

On définit la fonction **PrD**(S,M) :

- On commence par construire la matrice des résultats dans une taille adaptée
- On peuple les valeurs simples (autour des conditions aux limites) avec soit l'infini, soit « 0 »
- On réalise l'opération de calcul : si l'on peut faire en ajoutant une pièce, ou si on se réfère au cas précédent
- On stocke l'opération de calcul à la place qui convient dans la matrice
- On lit la valeur finale de l'algorithme (en bas à droite de la matrice)

III – Glouton optimal

On implémente le principe de l'algorithme glouton en utilisant la méthode proposée :

- Pour chaque pièce, on effectue le calcul en interdisant d'utiliser ladite pièce. On retient le vecteur solution qui contient le moins de pièces.
- Ainsi, on s'assure que de choisir une pièce « n'occultera » pas un meilleur chemin