Algorithmes 2

I- Recherche par dichotomie

Vous jouez à essayer de deviner le nombre auquel pense votre adversaire : en début de partie, celui-ci pense à un nombre entre 0 et 100, le note sur une feuille qu'il retourne (pour éviter la triche). Vous devez trouver ce nombre en un minimum de propositions. L'adversaire ne peut dire que "c'est plus", "c'est moins" et "bravo".

1) Essayez de décrire votre stratégie.

2) En justifiant votre réponse, indiquez quel serait le nombre minimum de questions à poser afin de trouver le résultat de façon sûre et certaine.

3) Écrire une fonction akinator(n) qui va chercher à deviner le nombre x auquel vous pensez ($0 \leq x \leq n$) en utilisant la même stratégie que vous.

4) Application : La fonction $\mathbf{f(x)=x^3-1{,}45x^2-11{,}55x+5{,}39}$ admet une solution **s** telle que **f(s)=0** avec **$0 \leq s \leq 1$**
Donner une valeur arrondie à 10^{-3} de **s** à l'aide d'une recherche par dichotomie.
On donne `def f(x):return x**3-1.45*x**2-11.55*x+5.39`

II- Algorithmes gloutons

Vous êtes employé par l'entreprise CashExpress, spécialisée dans la vente de caisses enregistreuses automatiques : pour éviter les contacts entre les employés et la clientèle, le client dépose directement l'argent dans un automate qui se charge de lui rendre la monnaie

On suppose que la monnaie peut-être rendue avec toutes les pièces et billets existant en euros :
[200, 100, 50, 20, 10, 5, 2, 1, 0.5, 0.2, 0.1, 0.05, 0.01]
(En effet, les billets de 500€ ne sont plus produits depuis 2018)

1) Un client vous présente un billet de 50€ pour payer sa chocolatine a 0,85€, indiquez le détail du rendu de monnaie que vous allez faire.

Un algorithme évident afin de rendre systématiquement correctement la monnaie, serait de rendre uniquement des pièces de 1 cent.
Cela présente 2 inconvénients majeurs :

- il est nécessaire de prévoir un très gros volume de pièce de 1 cent afin de pouvoir continuer à rendre la monnaie tout au long de la journée
- Vous risquez d'irriter vos clients

2) Proposez une stratégie vous permettant systématiquement de rendre la monnaie en utilisant un nombre minimal de pièces/billets.

3) Écrire la fonction **rendu_monnaie(montant)** qui retourne un *dictionnaire* contenant le détail du rendu de monnaie pour un **montant** donné.
Exemple : Le client a payé sa baguette à 1,05€ avec un billet de 5€ → l'automate doit lui rendre 3,95€
rendu_monnaie(3,95) →
{'200' :0,'100' :0,'50' :0,'20' :0,'10' :0,'5' :0,'2' :1,'1' :1,'0.5' :1,'0.2' :2,'0.1' :0,'0.05' :1,'0.01' :0}
En effet, le rendu optimal est 2€ + 1€ + 50ct + 2x20ct + 5ct

Une boulangerie basée en Absurdistan souhaite se procurer une de vos machine. Leur monnaie est le Plouk, dont le taux de change est proche de l'Euro : 1 Pk = 1 €.
Leur monnaie est différemment échelonnée : [200, 100, 30, 20, 5, 2, 1, 0.3, 0.1, 0.05, 0.01]

4) En adaptant la fonction **rendu_monnaie(montant)** aux différentes valeurs disponibles dans ce pays, quel retour propose-t-elle pour 45 Pk ?
Ce rendu est-il optimal ?