Algorithmes 2bis

I- Recherche par dichotomie

Une **liste** de nombres est générée par la commande

```
liste=[randint(0,10)]
for i in range(1,100000) :
  liste.append(liste[i-1]+randint(1,10))
```

On souhaite rechercher si un nombre est présent dans cette liste le plus rapidement possible.
1) D'après ce code, la liste est-elle déjà ordonnée ? Si oui, en ordre croissant ou décroissant ?
2) Est-il possible que la liste contienne deux fois le même nombre ?
3) Écrire une fonction **recherche_dicho(nombre,liste)** qui recherche par dichotomie le **nombre** dans **liste** et retourne si elle le trouve son indice et si elle ne le trouve pas les 2 indices encadrant la position qu'il aurait dû avoir
4) Utiliser le benchmark de la séance *algorithmes01* pour vérifier si cette méthode est réellement plus rapide que l'algorithme **indice(liste,valeur)** écrit précédemment.
Attention : votre fonction **bench** génère par défaut une liste aléatoire, ce qui n'est pas le cas de la liste ci-dessus, modifiez la fonction en conséquence !

II- Algorithmes gloutons

1) Les fractions égyptiennes

Si les égyptiens ne connaissaient pas la notation décimale, ils connaissaient les fractions, sous une forme particulière (voir Wikipédia) : leurs fractions étaient de la forme $\frac{1}{p}$ où $p \in \mathbb{N}^*$. On parle de fraction unitaire.

Ils pouvaient donc écrire n'importe quel nombre rationnel (un nombre qui peut s'écrire comme le quotient de 2 nombres entiers) en le décomposant d'une infinité de façons à l'aide de fractions unitaires.

Par exemple $\frac{2}{5}=\frac{1}{5}+\frac{1}{6}+\frac{1}{30}=\frac{1}{5}+\frac{1}{8}+\frac{1}{20}+\frac{1}{40}$.

Cette solution les satisfaisait probablement d'autant plus qu'il n'est pas sûr qu'ils savaient que certains nombres ne pouvaient être rationnels, $\sqrt{2}$ ou π !

Il vous faudrait écrire un algorithme glouton **egypte(nombre)** qui décompose n'importe quel nombre 0<x<1 en somme de fractions unitaires…

Remarque : cette façon de décomposer un nombre décimal devrait vous rappeler quelque chose, puisque c'est de cette façon que l'on détermine l'écriture binaire d'un décimal, en utilisant des fractions unitaires de la forme $\frac{1}{2^n}$

2) Le voleur

Marcel Lupin est un voleur prévoyant. Avant de perpétrer ses larcins, il prend soin de lister les objets de valeur présents chez ses futures malheureuses victimes. Il n'emporte sur les lieu qu'un seul sac de 50kg, il doit donc être sélectif !
Il établit donc une liste (enfin, nous dirions plutôt *dictionnaire* mais Marcel n'est pas informaticien) des objets présents, en indiquant tout d'abord leur masse en kg puis leur valeur en Brouzoufs (c'est la monnaie d'Andorrsbourg, riche petite principauté voisine de l'Absurdistan dans laquelle se déroule notre histoire).
Cette liste est disponible ici : http://raspberry.bernon.fr/bernon/voleur.txt)

Proposez une fonction **a_voler(dictionnaire)** qui pour un **dictionnaire** donné retourne la liste des items à voler en priorité afin d'avoir la plus grande valeur dans le sac.
Cette fonction calcule également la valeur du contenu du sac.