Correction des exercices du TD7

29 mars 2020

Exercice 2

```
def triRapide(T) :
    if len(T) <= 1 : return T
    pivot = T[0]
    gauche = [elt for elt in T[1:] if elt <= pivot]
    droite = [elt for elt in T[1:] if elt > pivot]
    return triRapide(gauche) + [pivot] + triRapide(droite)
```

Question 1

— Le pire cas arrive lorsque le pivot sépare le tableau de taille n en une partie vide et en une partie de taille $n-1$. La taille de la pile est alors en $O(n)$.

— Le meilleur cas arrive lorsque le pivot choisi est la médiane. Le tableau est donc séparer en deux parties de même taille. La taille de la pile est alors en $O(\log(n))$.

Question 2

— Sur l'entrée $[1, \ldots, n]$, la pile atteint une hauteur de 1. En effet, la récursion est appelée sur la partie gauche vide.

— Sur l'entrée $[n, \ldots, 1]$, la pile atteint une hauteur en $O(n)$. La récursion est appelée sur la partie gauche de taille $n-1$.

Question 3

Il faut effectuer la récursion sur la partie la plus petite. Le pire cas (taille de la pile) se produit lorsque les deux parties sont de même taille, la pile atteint alors une hauteur en $O(\log(n))$.

Question 4

Le meilleur cas pour la hauteur de la pile se produit sur des entrées comme $[1, \ldots, n]$ ou $[n, \ldots, 1]$, la pile a une hauteur en $O(1)$. Cela correspond à des appels récursifs sur des tableaux de taille inférieur à 1.

Exercice 3

On doit associer chaque écrou à sa vis. Pour cela, on va trier simultanément les vis et les écrous en s'inspirant du *tri rapide*.

1. S'il y a une vis et un écrou alors c'est terminé.
2. On partitionne les vis à l'aide d'un pivot e_i choisi dans les écrous. On mémorise durant cette étape l'unique vis v_i associée à e_i.
3. On partitionne les écrous à l'aide du pivot v_i.
4. À la fin des étapes 1 et 2, les deux partitions coïncident. De plus, on a associé e_i à v_i. On reprend alors à l'étape 1 pour les vis inférieurs à e_i et les écrous inférieurs à v_i, de même pour les parties supérieures.

```
def associe(V, E):
    if len(V) <= 1 and len(E) <= 1:
        return V, E

    pivotVis = E[0] # pivotVis ∈ E
    gaucheVis, droiteVis = [], []
    # Partitionnement de V selon pivotVis
    for elt in V:
        if elt < pivotVis:
            gaucheVis += [elt]
        elif elt > pivotVis:
            droiteVis += [elt]
        else:
            pivotEcrou = elt

    # Partitionnement de E selon pivotEcrou ∈ V
    gaucheEcrou = [elt for elt in E if elt < pivotEcrou]
    droiteEcrou = [elt for elt in E if elt > pivotEcrou]

    a, b = associe(gaucheVis, gaucheEcrou)
    c, d = associe(droiteVis, droiteEcrou)
    return a+[pivotEcrou]+c, b+[pivotVis]+d
```

Exercice 4

Question 1

```
# T un tableau de paire (x, color) et color ∈ {BLEU, BLANC, ROUGE}
def question1(T):
    bleus, blancs, rouges = [], [], []
    for elt in T:
        if   j[1] == Color.BLEU:  bleus  += [j]
        elif j[1] == Color.ROUGE: rouges += [j]
        else:                     blancs += [j]

    return bleus + blancs + rouges
```

Question 2

```
# T un tableau de paire (x, color) et color ∈ {BLEU, ROUGE}
def question2(T):
    bleu, rouge = 0, len(T)-1
    while bleu <= rouge:
        if T[bleu][1]  == Color.BLEU: bleu += 1
        elif T[rouge][1] == Color.ROUGE: rouge -= 1
        else: T[rouge], T[bleu] = T[bleu], T[rouge]
```

Question 3

```
# T un tableau de paire (x, color) et color ∈ {BLEU, BLANC, ROUGE}
def question3(T):
    bleu, blanc, rouge = 0, 0, len(T)-1
    while blanc <= rouge:
        if T[blanc][1] == Color.BLEU:
            T[blanc], T[bleu] = T[bleu], T[blanc]
            blanc += 1
            bleu += 1
        elif T[blanc][1] == Color.ROUGE:
            T[blanc], T[rouge] = T[rouge], T[blanc]
            rouge -= 1
        else:
            blanc += 1
```

Les variables bleu, rouge et blanc agissent comme des frontières :

— bleu est le premier indice après la zone bleue
— blanc est le premier indice après la zone blanche
— rouge est le premier indice avant la zone rouge

Question 4

On peut considérer :
— la zone bleue comme les éléments inférieurs au pivot.
— la zone blanche comme les éléments égaux au pivot.
— la zone rouge comme les éléments supérieurs au pivot.

On effectue alors un appel récursif sur la zone bleue et sur la zone rouge.

```
def partition(T, deb, fin):
    if fin - deb == 2 and T[deb] < T[deb+1]:
        return deb+1, deb+1

    pivot = T[deb]
    gauche, milieu, droite = deb, deb+1, fin-1
    while milieu <= droite:
        if T[milieu] < pivot:
            T[milieu], T[gauche] = T[gauche], T[milieu]
            gauche += 1
            milieu += 1
        elif T[milieu] > pivot:
            T[milieu], T[droite] = T[droite], T[milieu]
            droite -= 1
        else:
            milieu += 1

    return gauche, milieu

def triRapide(T, deb=0, fin=None) :
    if fin is None:
        fin = len(T)
    if deb < fin:
        sepInf, sepSup = partition(T, deb, fin)
        triRapide(T, deb, sepInf)
        triRapide(T, sepSup, fin)
```