TD: diviser pour régner:

Préambule:

Dans ce TD, les algoritmes utilisant des tableaux de taille n seront indicés de 0 à (n-1).

Exercice 1: palindrome:
Une chaîne qui peut se lire de la même manière dans les deux sens (gauche-droite et droite-gauche) est un palindrome.

1. Expliquer le fonctionnement de l'algorithme 1:

Algorithm 1: Palindrome

```
1 Function palindrome(ch):
2     if longueur(ch) ≤ 1 then
3         return True
4     else
5         return ch[0] = ch[−1] and palindrome(ch[1:-1])
```

2. En quoi cet algorithme utilise la méthode diviser pour régner? Est-il récursif?
3. L'implémenter en python.
4. Quel est l'avantage de l'algorithme 2 sur l'algorithme 1? Quel est son inconvénient?

Algorithm 2: Palindrome

```
1 Function palindrome(ch,g,d):
2     if d-g < 1 then
3         return True
4     else
5         return ch[g] = ch[d] and palindrome(ch,g+1,d-1)
```

5. L'implémenter en python.
6. Proposer une méthode permettant de ne plus avoir les indices g et d dans la fonction palindrome.

Exercice 2: recherche dichotomique:

1. On s'interesse à l'algorithme 3 de recherche dichotomique dans un tableau trié. Expliquer son fonctionnement.

Algorithm 3: Recherche dichotomique

```
1  Function dichotomie(tab,x):
2      g ← 0
3      d ← longueur(tab) − 1
4      while g ≤ d do
5          m ← partie entiere((g+d)/2)
6          if x = tab[m] then
7              return True
8          else
9              if x > tab[m] then
10                 g ← m + 1
11             else
12                 d ← m − 1
13     return False
```

2. En quoi cet algorithme utilise la méthode diviser pour régner? Est-il récursif?

3. L'implémenter en python.

4. Quel est l'avantage de l'algorithme 4 sur l'algorithme 3? Quel est son inconvénient?

Algorithm 4: Recherche dichotomique

```
1  Function dichotomie(tab,x,g,d):
2      if g = d then
3          return tab[g]=x
4      m ← partie entiere((g+d)/2)
5      if x = tab[m] then
6          return True
7      else if x < tab[m] then
8          if g = m then
9              return False
10         else
11             return dichotomie(tab, x, g, m-1)
12     else
13         return dichotomie(tab, x, m+1, d)
```

5. L'implémenter en python.

6. Proposer une méthode permettant de ne plus avoir les indices g et d dans la fonction dichotomie.

7. (bonus) Écrire une fonction de recherche naïve qui testerait toutes les valeurs possibles d'un tableau avant de conclure et qui fonctionnerait de fait pour un tableau non trié.

Exercice 3: tri par insertion:

1. On s'interesse à l'algorithme 5 de tri par insertion. Expliquer son fonctionnement.

Algorithm 5: Tri par insertion

```
1 Function tri(tab):
2     for i ← 1 to longueur(tab) do
3         el ← tab[i]
4         j ← i
5         while j > 0 and tab[j − 1] > el do
6             tab[j] ← tab[j − 1]
7             j ← j − 1
8         tab[j] ← el
9     return tab
```

2. Cet algorithme est-il récursif?

3. L'implémenter en Python.

4. Evaluer le temps d'exécution pire cas du programme Python pour un tableau de 10 000 nombres. On utilisera la commande time.perf_counter() de la bibliothèque time.

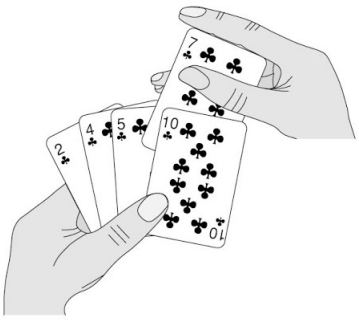

Figure 1: Tri par insertion

Exercice 5: tri fusion:

1. On s'interesse à l'algorithme 7 de tri fusion. Expliquer son fonctionnement.

Algorithm 6: Tri fusion

```
1  Function fusion(tab1,tab2):
2      tab ← [ ]
3      i ← 0
4      j ← 0
5      while i < longueur(tab1) and j < longueur(tab2) do
6          if tab1[i] ≤ tab2[j] then
7              on ajoute tab1[i] à tab
8              i ← i + 1
9          else
10             on ajoute tab2[j] à tab
11             j ← j + 1
12     while i < longueur(tab1) do
13         on ajoute tab1[i] à tab
14         i ← i + 1
15     while j < longueur(tab2) do
16         on ajoute tab2[j] à tab
17         j ← j + 1
18     return (tab)
19
20 Function tri(tab):
21     if longueur(tab) ≤ 1 then
22         return tab
23     else
24         m ← partie entiere(longueur(tab)/2
25         tab1 ← tri(tab[: milieu])
26         tab2 ← tri(tab[milieu :])
27         return fusion(tab1,tab2)
```

2. En quoi cet algorithme utilise la méthode diviser pour régner? Est-il récursif?

3. L'implémenter en python.

4. Evaluer le temps d'exécution pire cas du programme Python pour un tableau de 10 000 nombres. On utilisera la commande time.perf_counter() de la bibliothèque time.

Exercice 4: tri par sélection:

1. On s'interesse à l'algorithme 6 de tri par sélection. Expliquer son fonctionnement.

Algorithm 7: Tri par sélection

```
1  Function tri(tab):
2      i ← 0
3      while i < longueur(tab) − 1 do
4          j ← i + 1
5          min ← i
6          while j ≤ longueur(tab) − 1 do
7              if tab[j] ≤ tab[min] then
8                  min ← j
9              j ← j + 1
10         if min ≠ i then
11             echanger tab[i] et tab[min]
12         i ← i + 1
13     return tab
```

2. Cet algorithme est-il récursif?

3. L'implémenter en Python.

4. Evaluer le temps d'exécution pire cas du programme Python pour un tableau de 10 000 nombres. On utilisera la commande time.perf_counter() de la bibliothèque time.

Exercice 6: bilan sur les tris: On prend pour exemple les 3 tris suivants:

Data	8	7	1	2
Pass 1	7	8	1	2
Pass 2	1	7	8	2
Pass 3	1	2	7	8

Data	8	7	1	2
Pass 1	1	7	8	2
Pass 2	1	2	8	7
Pass 3	1	2	7	8

Data	6	4	1	3	2	5	9	7
Pass 1	4	6	1	3	2	5	7	9
Pass 2	1	3	4	6	2	5	7	9
Pass 3	1	2	3	4	5	6	7	9

1. Nommer chacun des tris.

2. Montrer les étapes des tris par insertion, sélection et fusion pour le tableau suivant: $t = [1, 5, 9, 21, 8, 11, 15]$.

NB: les tris sont visibles à `http://lwh.free.fr/pages/algo/tri/tri.htm`