

Algorithmique

Correction Partiel n° 1 (P1)

INFO-SUP S1# – EPITA

19 juin 2018 - 9 : 00

Solution 1 (Algorithmes de recherche – 3 points)

1. Recherche séquentielle sans tenir compte de l'ordre : 13
 2. Recherche séquentielle en tenant compte de l'ordre : 9
 3. Recherche dichotomique : $8 = 2 \times 4$
-

Solution 2 (Séquences et ABR – 3 points)

1. Séquences valides :

50, 70, 2048, 75, 1500, 1024 **oui**

50, 75, 2048, 70, 1500, 1024

2048, 50, 70, 75, 1500, 1024 **oui**

50, 75, 70, 2048, 1500, 1024

2. *Principe* :

Au fur et à mesure que l'on avance dans la liste on vérifie que les éléments sont dans l'intervalle courant $[inf, sup]$ (au départ $[-\infty, +\infty]$), si ce n'est pas le cas, la séquence n'est pas valide.

Si la valeur actuelle x est suivie d'une valeur supérieure, alors $inf \leftarrow x$ (on "part" à droite), sinon $sup \leftarrow x$ (on "part" à gauche).

Solution 3 (Types abstraits - 3 points)

1. *Quel est le nom de l'opération mystère ?* **inverse**

2. **Spécifications** :

La fonction `mystery(L)` inverse les éléments de la liste L .

```
1 def mystery(L):
2     n = len(L)
3     for k in range(n // 2):
4         (L[k], L[n-k-1]) = (L[n-k-1], L[k])
```

Solution 4 (What is it? – 3 points)

- `what([(0,0), (10,10), (20,20), (30,30)], 15)` 15.0
 - `what([(0,0), (10,20), (20,40), (30,60)], 24)` 12.0
 - `what([(0,0), (1, 10), (2,100), (3, 1000)], 2.5)` 0.25
 - `what([(0,3), (1,6), (2,9), (3,10), (4,15)], 20)` 5.0
 - Si on considère les couples comme des coordonnées triées par ordre croissant, `what(L, Y)` calcule l'abscisse X correspondant à l'ordonnée Y calculée par interpolation linéaire.
-

Solution 5 (Tri par sélection (Select Sort) – 8 points)

- La fonction `minimum(L, d, f)` détermine la position de la valeur minimum dans la liste L entre les positions d et f comprises (avec $0 \leq d < f \leq \text{len}(L)$).

```
1 def minimum(L, d, f):
2     pos = d
3     for i in range(d + 1, f + 1):
4         if L[i] < L[pos]:
5             pos = i
6     return pos
7
8
9 # a nice version
10 def minimum2(L, d, f):
11     while (d < f):
12         if L[d] < L[f]:
13             f = f - 1
14         else:
15             d = d + 1
16     return d
```

- La fonction `selectsort(L)` trie la liste L en ordre croissant **en place**

```
1 def selectSort(L):
2     n = len(L)
3     for i in range(n - 1):
4         pos = minimum(L, i, n - 1)
5         (L[i], L[pos]) = (L[pos], L[i]) # swap
```

- Soit L une liste de longueur n , le tri par sélection de L effectue :

- $\frac{n(n-1)}{2}$ comparaisons;
- $2(n-1)$ copies d'éléments.