Tri par sélection

1 Algorithme

Le tri par sélection recherche le plus petit ou le plus grand élément non trié de la liste pour le mettre à son emplacement définitif, qui peut varier si l'on souhaite trier en ordre croissant ou décroissant.

On répète cette recherche autant de fois qu'il y a d'éléments dans la liste. Chaque nouvelle itération s'effectue donc sur une liste plus petite qu'à l'itération précédente, car on ne considère plus les éléments déjà triés.

C'est un tri de complexité $O(n^2)$.

2 Exemple

Voici les étapes du tri par sélection sur la liste [6, 5, 3, 1, 8, 7, 2, 4].
On souhaite trier la liste en ordre croissant et on recherche les minimums pour les placer au début.
Les cases grisées sont les indices déjà triés.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Valeurs	6	5	3	1	8	7	2	4

Indices parcourus : de 0 à 7
Indice du minimum : 3
Indices des valeurs échangées : 0 et 3.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Iter 0	1	5	3	6	8	7	2	4

Indices parcourus : de 1 à 7
Indice du minimum : 6
Indices des valeurs échangées : 1 et 6.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Iter 1	1	2	3	6	8	7	5	4

Indices parcourus : de 2 à 7
Indice du minimum : 2
Indices des valeurs échangées : 2 et 2.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Iter 2	1	2	3	6	8	7	5	4

Indices parcourus : de 3 à 7
Indice du minimum : 7
Indices des valeurs échangées : 3 et 7.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Iter 3	1	2	3	4	8	7	5	6

Indices parcourus : de 4 à 7
Indice du minimum : 6
Indices des valeurs échangées : 4 et 6.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Iter 4	1	2	3	4	5	7	8	6

Indices parcourus : de 5 à 7
Indice du minimum : 7
Indices des valeurs échangées : 5 et 7.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Iter 5	1	2	3	4	5	6	8	7

Indices parcourus : de 6 à 7
Indice du minimum : 7
Indices des valeurs échangées : 6 et 7.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Iter 6	1	2	3	4	5	6	7	8

Il ne reste plus qu'une valeur à trier.
Elle est forcement à sa place donc la liste est triée.

3 Pseudo-code

```
procédure tri_selection(liste l)
    n ← longueur(l)
    pour i de 0 à n - 2
        min ← i
        pour j de i + 1 à n - 1
            si l[j] < l[min], alors min ← j
        fin pour
        si min ≠ i, alors échanger l[i] et l[min]
    fin pour
fin procédure
```

Exercice 1 : Trier dans l'ordre croissant la liste [1, 5, 3, 6, 8, 7, 2, 4] en utilisant l'algorithme de tri par sélection qui recherche les minimums.

indices	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Valeurs	1	5	3	6	8	7	2	4
Iter 0								
Iter 1								
Iter 2								
Iter 3								
Iter 4								
Iter 5								
Iter 6								

Exercice 2 : Coder l'algorithme du tri par sélection en ordre croissant recherchant les minimums en Python. Aider vous du pseudo-code si besoin. Coder le test pour la liste de l'exercice 1.

```
1   def echange(l : list, i : int, j : int):
2       """ Échange les éléments d'indice i et j dans la liste l. """
3       temp = ...
4       ...  = ...
5       ...  = ...
6
7   def tri_par_selection(l : list):
8       """ Tri la liste l avec un tri par sélection
9       >>> ...
10      ...
11      """
12
13      for i in range(..............):
14          indice_min = ...
15          for j in range(.................):
16              if .............. > ..............:
17                  ......... = ...
18          ..............................
19      return .........
```

Bonus : Coder l'algorithme du tri par sélection en ordre croissant recherchant les maximums en Python.

N'hésitez pas à aller sur https://pythontutor.com/ pour tester et visualiser votre code.

Tri a bulles

1 Définition

L'algorithme du tri a bulles parcourt la liste et compare à chaque itération deux éléments consécutifs. C'est un tri de complexité $O(n^2)$. Nous illustrerons ici le tri en ordre croissant.

Il compare d'abord le couple d'indices 0 et 1, puis les couples d'indices 1 et 2, puis 2 et 3, ... jusqu'à atteindre la fin de la liste. Lorsque deux éléments consécutifs ne sont pas dans l'ordre croissant, ils sont permutés.

Après un premier parcours complet de la liste, le maximum est forcement placé en fin de liste. Le prochain parcours s'effectuera toujours en partant du départ mais en s'arrêtant à l'avant dernier élément.

Chaque parcours trie un élément et le parcours suivant s'effectue sur une liste plus petite d'un élément.

Le dernier parcours trie les deux plus petits éléments de la liste.

2 Pseudo_code

```
1  tri_à_bulles(Tableau T)
2     pour i allant de (taille de T)-1 à 1
3         pour j allant de 0 à i-1
4             si T[j+1] < T[j]
5                 (T[j+1], T[j]) ← (T[j], T[j+1])
```

3 Exemple de tri

On trie la liste `[3, 6, 5, 1, 2, 4]` avec l'algorithme du tri a bulle.

i = 5	j = 0	3	6	5	1	2	4	Ordre croissant, on ne fait rien.
	j = 1	3	6	5	1	2	4	6 > 5 donc on les permute.
	j = 2	3	5	6	1	2	4	6 > 1 donc on les permute.
	j = 3	3	5	1	6	2	4	6 > 2 donc on les permute.
	j = 4	3	5	1	2	6	4	6 > 4 donc on les permute.
i = 4	j = 0	3	5	1	2	4	6	Ordre croissant, on ne fait rien.
	j = 1	3	5	1	2	4	6	5 > 1 donc on les permute.
	j = 2	3	1	5	2	4	6	5 > 2 donc on les permute.
	j = 3	3	1	2	5	4	6	5 > 4 donc on les permute.
i = 3	j = 0	3	1	2	4	5	6	3 > 1 donc on les permute.
	j = 1	1	3	2	4	5	6	3 > 2 donc on les permute.
	j = 2	1	2	3	4	5	6	Ordre croissant, on ne fait rien.
i = 2	j = 0	1	2	3	4	5	6	Ordre croissant, on ne fait rien.
	j = 1	1	2	3	4	5	6	Ordre croissant, on ne fait rien.
i = 1	j = 0	1	2	3	4	5	6	Ordre croissant, on ne fait rien.
Fin		1	2	3	4	5	6	La liste est triée.

Exercice 1 : Appliquer à la main le tri à bulle sur [8, 7, 4, 3, 6, 5, 2, 1] en suivant le modèle de l'exemple.

		8	7	4	3	6	5	2	1
Fin									

Exercice 2 : Coder l'algorithme du tri à bulle en Python. Aider vous du pseudo-code si besoin. Coder le test pour la liste de l'exercice 1.

```
1   def tri_à_bulles(l) :
2       """
3       Effectue un tri à bulles en ordre croissant sur la liste l
4       >>> ........................................
5       ........................................
6       """
7
8       for ....................................
9           for ................................
10              if .............................
11                  .............................
12                  .............................
13                  .............................
14      return l
```

Bonus : Quelle modification doit-on effectuer pour que la liste soit triée en ordre décroissant ?

Tri par insertion

1 Description

Le tri par insertion considère un par un chaque élément de la liste et l'insère à la bonne place parmi les éléments déjà triés. Ainsi, au moment où on considère un élément, les éléments qui le précèdent sont déjà triés, tandis que les éléments qui le suivent ne sont pas encore triés. C'est un tri de complexité $O(n^2)$.

Pour trouver la place où insérer un élément parmi les précédents, il faut le comparer à ces derniers, et les décaler afin de libérer une place où effectuer l'insertion. Le décalage occupe la place laissée libre par l'élément considéré. En pratique, ces deux actions s'effectuent en une passe, qui consiste à faire « remonter » l'élément au fur et à mesure jusqu'à rencontrer un élément plus petit.

2 Exemple

Voici les étapes de l'exécution du tri par insertion sur la liste `[6, 5, 3, 1, 8, 7, 2, 4]`.
La liste est représentée au début et à la fin de chaque itération.

i = 1 :	6	5	3	1	8	7	2	4	→	5	6	3	1	8	7	2	4
i = 2 :	5	6	3	1	8	7	2	4	→	3	5	6	1	8	7	2	4
i = 3 :	3	5	6	1	8	7	2	4	→	1	3	5	6	8	7	2	4
i = 4 :	1	3	5	6	8	7	2	4	→	1	3	5	6	8	7	2	4
i = 5 :	1	3	5	6	8	7	2	4	→	1	3	5	6	7	8	2	4
i = 6 :	1	3	5	6	7	8	2	4	→	1	2	3	5	6	7	8	4
i = 7 :	1	2	3	5	6	7	8	4	→	1	2	3	4	5	6	7	8

Élément déjà trié Élément à trier

Détaillons l'itération 6 de l'exemple. La valeur à trier est le 2 et on compare les valeurs précédemment triées de la liste (en gris).

Tant que valeur comparée est plus grande que 2, on la décale (copie) à l'indice +1.
Si valeur est plus petite que 2 ou que l'on est arrivé au début de la liste, le 2 est à sa place.

j = 6	1	3	5	6	7	8		4
j = 5	1	3	5	6	7		8	4
j = 4	1	3	5	6		7	8	4
j = 3	1	3	5		6	7	8	4
j = 2	1	3		5	6	7	8	4
j = 1	1	3	3	5	6	7	8	4
	1	2	3	5	6	7	8	4

8 est plus grand que 2, on le décale.
7 est plus grand que 2, on le décale.
6 est plus grand que 2, on le décale.
5 est plus grand que 2, on le décale.
3 est plus grand que 2, on le décale.
1 n'est pas plus grand que 2, on insère 2.

Élément déjà trié Élément à trier Cellule vide

3 Pseudo-code

```
procédure tri_insertion(liste l)

    pour i de 1 à taille(l) - 1

        # mémoriser l[i] dans x
        x ← l[i]

        # décaler les éléments l[0]..l[i-1] qui sont plus grands que x,
        # en partant de l[i-1]
        j ← i
        tant que j > 0 et l[j-1] > x
            l[j] ← l[j-1]
            j ← j - 1

        # placer x dans le "trou" laissé par le décalage
        l[j] ← x
```

Exercice 1 : Trier la liste suivante en utilisant le tri par insertion.

i = 1 :	8	4	7	5	2	3	1	6	→									
i = 2 :									→									
i = 3 :									→									
i = 4 :									→									
i = 5 :									→									
i = 6 :									→									
i = 7 :									→									

Exercice 2 : Coder l'algorithme du tri par insertion en Python. Aider vous du pseudo-code si besoin. Coder le test pour la liste de l'exercice 1.

```
1   def tri_insertion(l : list):
2       """
3       Effectue un tri par insertion sur la liste l
4       >>> ........................................
5       ........................................
6       """
7
8       for i in ..............:
9           # mémoriser l[i] dans x
10          x = .........
11          # décaler les éléments l[0]..l[i-1] qui sont plus grands que
12          # x, en partant de l[i-1]
13          j = ...
14          while ........................:
15              ...... = ......
16              ...... = ......
17          # placer x dans le "trou" laissé par le décalage
18          ...... = ......
19      return l
```