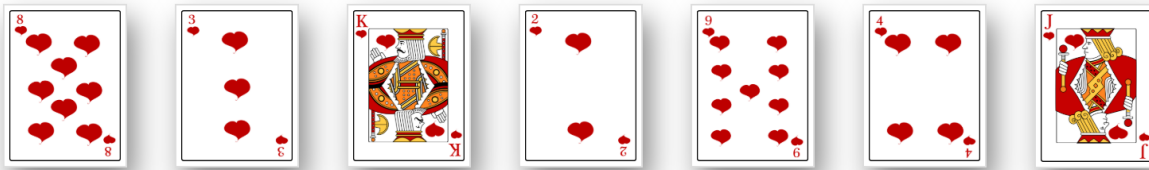


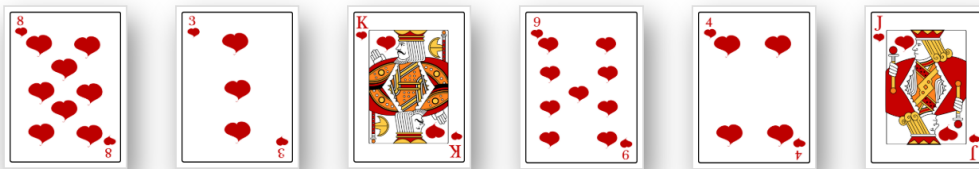
Tri par sélection

C'est le tri qu'utilisent souvent les enfants. Il s'agit de chercher dans les cartes la plus petite, puis la seconde plus petite etc.

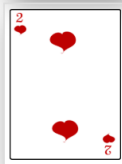


- On cherche la carte ayant la marque la plus petite, on commence un tas :

cartes à
trier

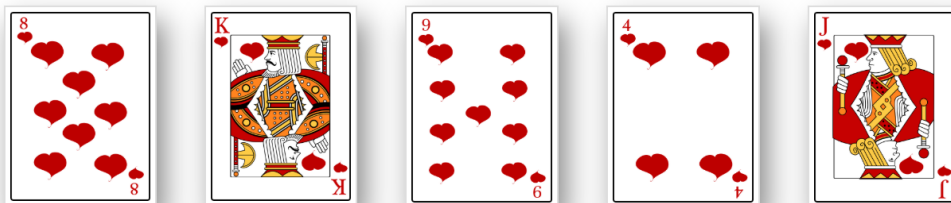


cartes
triées

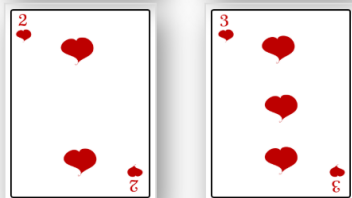


- On cherche la carte ayant la marque la plus petite parmi celles qui restent, on la pose **après** (c'est-à-dire *au-dessus de*) la première dans le tas trié :

cartes à
trier



cartes
triées



- On cherche la carte ayant la marque la plus petite parmi celles qui restent, on la pose **après**

Et ainsi de suite...

Exemple : On veut trier, comme un enfant, le tableau :

main =

4	2	7	1	8	5	3
---	---	---	---	---	---	---

Pour cela, on initialise un tableau vide, nommé tas, et on pose, au fur et à mesure, à la fin du tableau tas, le plus petit élément qu'on a sélectionné dans le tableau main.

- Au départ le tableau tas est vide :

main =

4	2	7	1	8	5	3
---	---	---	---	---	---	---

tas =

- Sur le tas, on place la plus petite des cartes de la main (donc on l'enlève de la main) :

main =

4	2	7	8	5	3
---	---	---	---	---	---

tas =

1

- Puis on sélectionne la plus petite des cartes qui restent dans la main (donc on l'enlève de la main) et on la place sur le tas :

main =

4	7	8	5	3
---	---	---	---	---

tas =

1	2
---	---

- On recommence :

main =

4	7	8	5
---	---	---	---

tas =

1	2	3
---	---	---

- Puis :

main =

7	8	5
---	---	---

tas =

1	2	3	4
---	---	---	---

- Puis :

main =

7	8
---	---

tas =

1	2	3	4	5
---	---	---	---	---

- Puis :

main =

8

tas =

1	2	3	4	5	7
---	---	---	---	---	---

- On s'arrête lorsque la main est vide, le tas est trié :

main =

tas =

1	2	3	4	5	7	8
---	---	---	---	---	---	---

Exercice : Illustrer de même le tri de l'enfant avec le tas de cartes suivant :

tas =

8	1	7	2	4	3	1	6
---	---	---	---	---	---	---	---

(Vous avez les tableaux à compléter sur la page suivante)

main =

8	1	7	2	4	3	1	6
---	---	---	---	---	---	---	---

tas =

• main =

--	--	--	--	--	--	--

tas =

--

• main =

--	--	--	--	--	--

tas =

--	--

• main =

--	--	--	--

tas =

--	--	--

• main =

--	--	--	--

tas =

--	--	--	--

• main =

--	--	--

tas =

--	--	--	--	--

• main =

--	--

tas =

--	--	--	--	--	--

• main =

--

tas =

--	--	--	--	--	--	--

• main =

tas =

--	--	--	--	--	--	--	--

Fastoche...

L'algorithme pourrait s'écrire en langage profane :

Pour i variant de 0 à n-1 :

Rechercher l'indice du plus petit élément de main

Retirer cet élément de main

Placer cet élément à la fin de tas

Écrivons-le de manière plus formelle :

VAR main, tas en liste d'entier

VAR i, k en entier

DEBUT

i=0

Pour i allant de 0 à longueur(main)-1

 k=main[0]

Pour j allant de 0 à longueur(main)-1

Si k >= main[j]

 k=main[j]

 index_kill=j

fin si

fin pour

 tas[i]=k

 delete(main[index_kill]) // on supprime la case de la liste

fin pour

FIN

On parcourt la liste autant de fois qu'il y a de case. A chaque parcours, on recherche la plus petite valeur directement supérieure ou égale à k. dès qu'on l'a trouvée, on affecte cette valeur à k et on supprime la case de main.

Vous pouvez, si vous avez des difficultés à comprendre cet algorithme, le faire tourner « à la main » comme nous avons fait avec l'algorithme de tri par insertion. Desfois que je vous demande ça en interro...

TRI SUR PLACE :

On souhaite maintenant de pas utiliser une liste supplémentaire pour faire le tri, mais trier la liste sur place (c'est-à-dire ne pas prendre les cartes une à une dans la main mais les trier alors qu'on les a toutes en main).

main =

4	2	7	1	8	5	3
---	---	---	---	---	---	---

- L'indice du plus petit élément de la main est 3 :

main =

2	4	7	1	8	5	3
---	---	---	---	---	---	---

- On échange cet élément avec main[0] :

main =

1	4	7	2	8	5	3
---	---	---	---	---	---	---

- L'indice du plus petit élément de main[1..] est à nouveau 3:

main =

1	4	7	2	8	5	3
---	---	---	---	---	---	---

- On échange cet élément avec main[1] :

main =

1	2	7	4	8	5	3
---	---	---	---	---	---	---

- L'indice du plus petit élément de main[2..] est 6 :

main =

1	2	7	4	8	5	3
---	---	---	---	---	---	---

- On échange cet élément avec main[2] :

main =

1	2	3	4	8	5	7
---	---	---	---	---	---	---

- L'indice du plus petit élément de main[3..] est 3 :

main =

1	2	3	4	8	5	7
---	---	---	---	---	---	---

- On échange cet élément avec main[3] (même si, ici, cela ne sert à rien) :

main =

1	2	3	4	8	5	7
---	---	---	---	---	---	---

- L'indice du plus petit élément de main[4..] est 5 :

main =

1	2	3	4	8	5	7
---	---	---	---	---	---	---

- On échange cet élément avec main[4] :

main =

1	2	3	4	5	8	7
---	---	---	---	---	---	---

- L'indice du plus petit élément de main[5..] est 6 :

main =

1	2	3	4	5	8	7
---	---	---	---	---	---	---

- On échange cet élément avec main[4] :

main =

1	2	3	4	5	7	8
---	---	---	---	---	---	---

Exercice :

Illustrer de même le tri avec la main suivante :

main =

8	1	7	2	4	3	1	6
---	---	---	---	---	---	---	---

Étape n°0 : main =

8	1	7	2	4	3	1	6
---	---	---	---	---	---	---	---

 indice du minimum :

Étape n°1 : main =

--	--	--	--	--	--	--	--

 indice du minimum :

Étape n°2 : main =

--	--	--	--	--	--	--	--

 indice du minimum :

Étape n°3 : main =

--	--	--	--	--	--	--	--

 indice du minimum :

Étape n°4 : main =

--	--	--	--	--	--	--	--

 indice du minimum :

Étape n°5 : main =

--	--	--	--	--	--	--	--

 indice du minimum :

Étape n°6 : main =

--	--	--	--	--	--	--	--

 indice du minimum :

Étape n°7 : main =

--	--	--	--	--	--	--	--

L'algorithme pourrait s'écrire en langage profane :

Pour i variant de 0 à n-1 :

Rechercher l'indice du plus petit élément de tab[i..]

Échanger cet élément avec tab[i]

Fin Pour

Ou en magnifique (attention, comme ça se fait parfois, la première case du tableau est d'indice 1 ici. On rappelle qu'il n'y a pas de règles en rapport à ça, et que je ne saurais vous dire si en examen vous aurez le premier indice à 0 ou à 1, donc autant s'habituer à jongler entre les deux):

var t : tableau d'entiers

var i, j, min : nombres entiers

DEBUT

i←1

tant que i<longueur(t): //boucle 1

j←i+1

min←i

tant que j<=longueur(t): //boucle 2

si t[j]<t[min]:

min←j

fin si

j←j+1

fin tant que

si min≠i :

échanger t[i] et t[min]

fin si

i←i+1

fin tant que

FIN

Le déroulé « à la main » :

t = [12, 8, 23, 10, 15]



Complexité de cet algorithme :

Vous vous en souvenez, pour trouver la complexité asymptotique temporelle, on ne s'embête pas avec les coefficients et les constantes, on y va « à la louche »

Ici, on s'aperçoit que plus le nombre de valeurs sera grand, plus le nombre de fois qu'on parcourra la liste sera grand lui aussi.

A la louche, on pourrait dire qu'on parcourt **n fois la liste**. On peut donc dire qu'on fera **n^2 opérations élémentaires**

On dira donc ici que la complexité s'écrit **$O(n^2)$**
 Entre nous, on dira que la complexité est en **n^2**
 On dit aussi qu'elle est **quadratique**

Exercice suivant :

Implanter cet algorithme de tri en Python 😊