Algorithmes de tri

Table des matières

Introduction

On désigne par "*tri*" l'opération consistant à ordonner un ensemble d'éléments en fonction de critères sur lesquelles est définie une relation d'ordre.
Les algorithmes de tri ont une grande importance pratique. Ils sont fondamentaux dans l'informatique où l'on tri de manière quasi-systématique des données avant de les utiliser.
L'étude du tri est également intéressante en elle-même car il s'agit sans doute du domaine de l'algorithmique qui a été le plus étudié et qui a conduit à des résultats remarquables sur la construction d'algorithmes et l'étude de leur complexité.
J'espère que ces quelques exemples vous permettrons de vous forger une idée. Il y a bien sûr d'autres algorithmes que ceux présentés ici, mais ce document n'a pas l'ambition de dresser une liste exhaustive de ce qui existe mais d'avoir un aperçu de ce problème qui est plus complexe qui n'y paraît au premier abord. Tout le monde est confronté un jour ou l'autre à trier des affaires et l'on sait que c'est loin d'être facile.

1 Tri par insertion

C'est le tri du joueur de cartes. On fait comme si les éléments à trier étaient donnés un par un, le premier élément constituant, à lui tout seul, une liste triée de longueur 1. On range ensuite le second élément pour constituer une liste triée de longueur 2, puis on range le troisième élément pour avoir une liste triée de longueur 3 et ainsi de suite ....
À chaque itération , on place l'élément à sa place en décalant ceux qui sont plus grands.
Le principe du tri par insertion est donc d'insérer à la $n^{\text{ième}}$ itération le $n^{\text{ième}}$ élément à la bonne place.

1.1 Exemple

Soit à trier la liste suivante : 86 - 8 - 79 - 21 - 58 - 49 - 42 - 22 - 28 - 79

On prend les deux premiers éléments : 86 - 8, comme 8 est plus petit, on décale 86 est l'on met 8 devant. Cela donne : 8 - 86. On prend l'élément suivant 79, il est plus petit que 86 mais plus grand que 8, on décale 86 et l'on insère 79, ce qui donne : 8 - 79 - 86. Voici alors les différentes étapes du processus.

Étape	Ordre des éléments de la liste									
0	86	8	79	21	58	49	42	22	28	79
1	8	86	79	21	58	49	42	22	28	79
2	8	79	86	21	58	49	42	22	28	79
3	8	21	79	86	58	49	42	22	28	79
4	8	21	58	79	86	49	42	22	28	79
5	8	21	49	58	79	86	42	22	28	79
6	8	21	42	49	58	79	86	22	28	79
7	8	21	22	42	49	58	79	86	28	79
8	8	21	22	28	42	49	58	79	86	79
9	8	21	22	28	42	49	58	79	79	86

1.2 Algorithme

Le but de cet algorithme est de générer 20 nombres aléatoires entre 1 et 20 de les trier et de visualiser cette liste tout au long du processus.

```
NORMAL FLOTT AUTO RÉEL RAD MP
PROGRAM:TRICARTE
:EffDess
:EffListe L1
:0→L1(1)
:For(I,2,21)
:nbrAléatEnt(1,20)→L1(I)
:Ligne(I-1,0,I-1,L1(I))
:End
:For(I,3,21)
:L1(I)→A
:I-1→J
:While J>1 et L1(J)>A
:L1(J)→L1(J+1)
:J-1→J
:End
:A→L1(J+1)
:EffDess
:For(K,2,21)
:Ligne(K-1,0,K-1,L1(K))
:End
:End
```

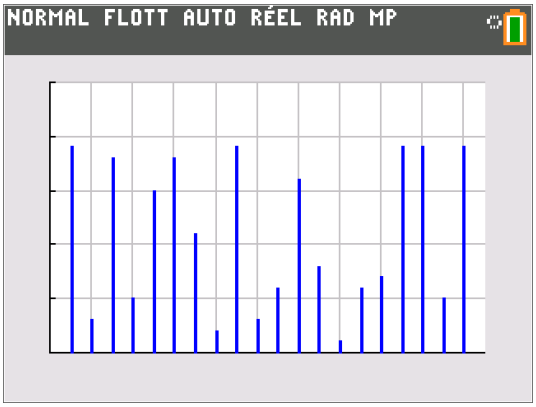
Variables : I, A, J, K : entiers L_1 : liste
Entrées et initialisation
Effacer dessin
Effacer liste L_1
$0 \to L_1(1)$ *
pour I de 2 à 21 **faire**
AléaEnt$(1,20) \to L_1(I)$
Tracer segment$(I-1,\ 0\ , I-1,\ L_1(I))$
fin
Traitement et sorties
pour I de 3 à 21 **faire**
$L_1(I) \to A$ *on prend l'élément suivant*
$I-1 \to J$
tant que $J > 1$ et $L_1(J) > A$ **faire**
$L_1(J) \to L_1(J+1)$
on effectue le décalage pour insérer le nouvel élément
$J-1 \to J$
fin
$A \to L_1(J+1)$ *on insère l'élément*
Effacer dessin
pour K de 2 à 21 **faire**
Tracer
segment$(K-1,\ 0\ , K-1,\ L_1(K))$
fin
fin

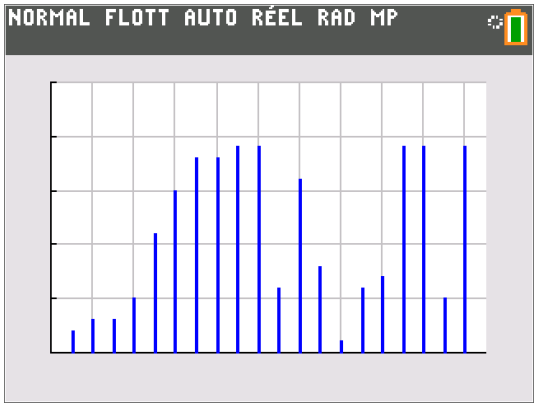
*Pour des raisons de dimension, à cause du test du tant que, on pourrait être amené à calculer $L_1(0)$ si I commençait à 1. On crée alors un élément nul en $L_1(1)$ et la liste réelle va de 2 à 21. On rétablit la liste de 1 à 20 lorsqu'on trace les segments.

On peut prendre pour visualiser le tri, la fenêtre suivante :

$$\begin{array}{ll} X_{min} = 0 & Y_{min} = 0 \\ X_{max} = 21 & Y_{max} = 25 \\ X_{grad} = 2 & Y_{grad} = 5 \end{array}$$

On obtient la première et la dixième visualisations suivantes :





2 Tri par sélection

C'est le tri que l'on fait sur une feuille de papier. Le principe du tri par *sélection* ou par *extraction* est d'aller chercher le plus petit élément d'une liste pour le mettre en premier, puis de cherche le plus petit élément de cette liste en excluant le premier pour le placer en second, puis le troisième en excluant les deux premiers et ainsi de suite.
À chaque itération , on place un élément supplémentaire dans le tri. On échange ainsi un élément avec le plus petit de la liste considérée

2.1 Exemple

Soit à trier la liste suivante : 86 - 8 - 79 - 21 - 58 - 49 - 42 - 22 - 28 - 79

8 est le plus petit élément, on le place en premier. 21 est le plus petit élément en excluant 8, on le place en second, ...

Étape	Ordre des éléments de la liste									
0	86	8	79	21	58	49	42	22	28	79
1	8	86	79	21	58	49	42	22	28	79
2	8	21	79	86	58	49	42	22	28	79
3	8	21	22	86	58	49	42	79	28	79
4	8	21	22	28	58	49	42	79	86	79
5	8	21	22	28	42	49	58	79	86	79
6	8	21	22	28	42	49	58	79	86	79
7	8	21	22	28	42	49	58	79	86	79
8	8	21	22	28	42	49	58	79	86	79
9	8	21	22	28	42	49	58	79	79	86

2.2 Algorithme

Le but de cet algorithme est de générer 20 nombres aléatoires entre 1 et 20 de les trier et de visualiser cette liste tout au long du processus.

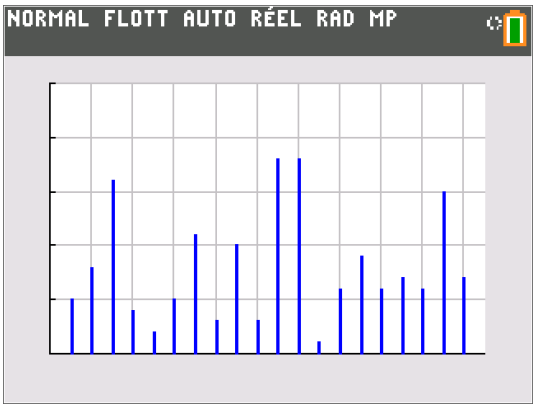
On peut prendre pour visualiser le tri, la fenêtre suivante :

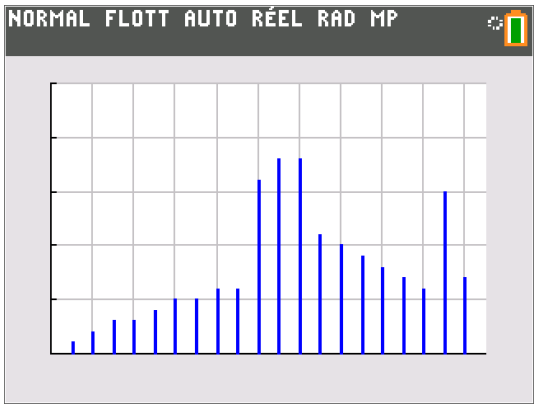
$$\begin{array}{ll} X_{min} = 0 & Y_{min} = 0 \\ X_{max} = 21 & Y_{max} = 25 \\ X_{grad} = 2 & Y_{grad} = 5 \end{array}$$

```
NORMAL FLOTT AUTO RÉEL RAD MP
PROGRAM:TRISELEC
:EffDess
:EffListe L1
:For(I,1,20)
:nbrAléatEnt(1,20)→L1(I)
:Ligne(I,0,I,L1(I))
:End
:For(I,1,19)
:I→K
:For(J,I+1,20)
:If L1(J)<L1(K)
:Then
:J→K
:End
:L1(I)→A
:L1(K)→L1(I)
:A→L1(K)
:I→K
:End
:EffDess
:For(J,1,20)
:Ligne(J,0,J,L1(J))
:End
:End
```

Variables : I, A, J, K : entiers L_1, L_2 : listes
Entrées et initialisation
Effacer dessin
Effacer liste L_1
pour I de 1 à 20 **faire**
AléaEnt$(1,20) \to L_1(I)$
Tracer segment$(I\ ,\ 0\ ,\ I\ ,\ L_1(I))$
fin
Traitement et sorties
pour I de 1 à 19 **faire**
$I \to K$
pour J de $I+1$ à 20 **faire**
si $L_1(J) < L_1(K)$ **alors**
$J \to K$
fin
$L_1(I) \to A$
$L_1(K) \to L_1(I)$ *on échange $L_1(I)$ avec $L_1(K)$*
$A \to L_1(K)$
$I \to K$
fin
Effacer dessin
pour J de 1 à 20 **faire**
Tracer segment$(J\ ,\ 0\ ,\ J\ ,\ L_1(J))$
fin
fin

On obtient la première et la dixième visualisations suivantes :





3 Tri à bulle

Le principe du tri à bulle est de comparer deux éléments consécutifs e_1 et e_2 et de les permuter si $e_1 > e_2$.
On trie jusqu'à ce qu'il n'y ait plus de permutation.

⚠ Pour permuter deux éléments par exemple $L(I)$ et $L(J)$, on doit crée une variable provisoire, par exemple A, pour y stocker un des deux éléments à permuter. On a alors le schéma :

$$\begin{aligned} L(J) &\to A \\ L(I) &\to L(J) \\ A &\to L(I) \end{aligned}$$

3.1 Exemple

Soit à trier la liste suivante : 14 - 21 - 8 - 15 - 35 - 59 - 63 - 9 - 42 - 69

Étape	Ordre des éléments de la liste
0	14 - 21 - 8 - 15 - 35 - 59 - 63 - 9 - 42 - 69
1	14 - 8 - 15 - 21 - 35 - 59 - 9 - 42 - 63 - 69
2	8 - 14 - 15 - 21 - 35 - 9 - 42 - 59 - 63 - 69
3	8 - 14 - 15 - 21 - 9 - 35 - 42 - 59 - 63 - 69
4	8 - 14 - 15 - 9 - 21 - 35 - 42 - 59 - 63 - 69
5	8 - 14 - 9 - 15 - 21 - 35 - 42 - 59 - 63 - 69
6	8 - 9 - 14 - 15 - 21 - 35 - 42 - 59 - 63 - 69

⚠ Au premier passage, on est sûr que le plus grand nombre sera mis à la fin de la série. Au second passage l'avant dernier nombre sera à sa place et ainsi de suite.

3.2 Algorithme

Le but de cet algorithme est de générer 20 nombres aléatoires entre 1 et 20 de les trier et de visualiser cette liste tout au long du processus.

$P = 1$ lorsqu'il y a permutation et $P = 0$ dans le cas contraire.

On peut prendre pour visualiser le tri, la fenêtre suivante :

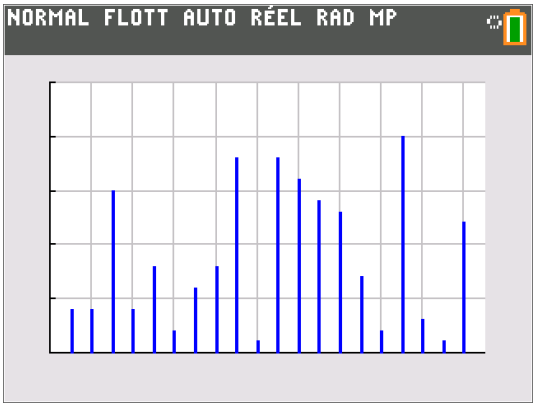
$$\begin{aligned} X_{min} &= 0 & Y_{min} &= 0 \\ X_{max} &= 21 & Y_{max} &= 25 \\ X_{grad} &= 2 & Y_{grad} &= 5 \end{aligned}$$

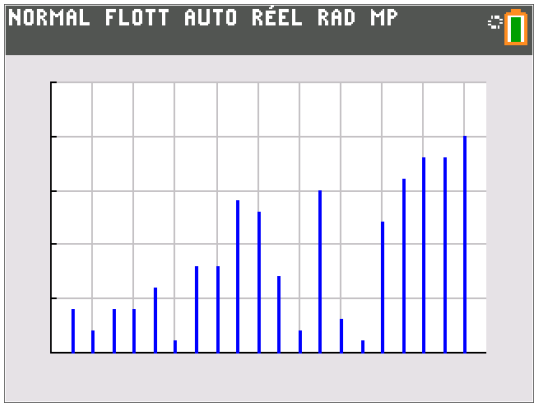
```
NORMAL FLOTT AUTO RÉEL RAD MP
PROGRAM:TRIBULLE
:EffDess
:EffListe L1
:For(I,1,20)
:nbrAléatEnt(1,20)→L1(I)
:Ligne(I,0,I,L1(I))
:End
:1→N
:1→P
:While P=1
:0→P
:For(I,1,20-N)
:If L1(I)>L1(I+1)
:Then
:L1(I+1)→A
:L1(I)→L1(I+1)
:A→L1(I)
:1→P
:End
:End
:EffDess
:For(J,1,20)
:Ligne(J,0,J,L1(J))
:End
:End
```

Variables : I, N, P, A, J : entiers L_1 : liste
Entrées et initialisation
- Effacer dessin
- Effacer liste L_1
- **pour** I de 1 à 20 **faire**
  - AléaEnt$(1,20) \to L_1(I)$
  - Tracer segment $(I\ ,\ 0\ ,\ I\ ,\ L_1(I))$
  - *visualisation en segments de la liste*
- **fin**
- $1 \to N$
- $1 \to P$ *permutation = vrai*

Traitement et sorties
- **tant que** $P = 1$ **faire**
  - $0 \to P$ *permutation = faux*
  - **pour** I de 1 à $20 - N$ **faire**
    - **si** $L_1(I) > L_1(I+1)$ **alors**
      - $L_1(I+1) \to A$
      - $L_1(I) \to L_1(I+1)$
      - $A \to L_1(I)$
      - $1 \to P$ *permutation = vrai*
    - **fin**
  - **fin**
  - $N+1 \to N$ *Les N derniers sont placés*
  - **pour** J de 1 à 20 **faire**
    - Tracer segment $(J\ ,\ 0\ ,\ J\ ,\ L_1(J))$
  - **fin**
- **fin**

On obtient la première et la quatrième visualisation suivantes :





4 Tri à peigne

Le principe du tri à peigne est le même que le tri à bulle mais cette fois-ci on permute des éléments plus lointains puis on raccourcit progressivement par un facteur de réduction, l'intervalle de comparaison jusqu'à 1. Il améliore de façon notable les performance du tri à bulle. Cet algorithme fut conçu en 1980 par W ?odzimierz Dobosiewicz.

Le choix du facteur de réduction est primordial pour l'efficacité de cet algorithme. En général, on prend un facteur de réduction compris entre 1.25 et 1.33. On prendra ici 1,3 comme facteur de réduction.

4.1 Exemple

Reprenons la liste suivante : 14 - 21 - 8 - 15 - 35 - 59 - 63 - 9 - 42 - 69

Ici $N = 10$, on prend la partie entière de $\frac{10}{1,3}$ soit 7

$$14 - 21 - 8 - 15 - 35 - 59 - 63 - 9 - 42 - 69$$

Seuls 14 et 9 sont à permuter, on calcule la partie entière de $\frac{7}{1,3}$ soit 5, on a alors :

$$9 - 21 - 8 - 15 - 35 - 59 - 63 - 14 - 42 - 69$$

Aucun élément n'est à permuter, on calcule la partie entière de $\frac{5}{1,3}$ soit 3, on a alors en ne visualisant que les paires à permuter :

$$9 - 21 - 8 - 15 - 35 - 59 - 63 - 14 - 42 - 69$$

Deux paires sont à permuter, on calcule la partie entière de $\frac{3}{1,3}$ soit 2, on a alors en ne visualisant que les paires à permuter :

$$9 - 21 - 8 - 15 - 14 - 42 - 63 - 35 - 59 - 69$$

Quatre paires sont à permuter. on calcule $\frac{2}{1,3} < 1$, on prend 1, on a alors en ne visualisant que les paires à permuter :

$$8 - 15 - 9 - 21 - 14 - 35 - 59 - 42 - 63 - 69$$

Trois paires sont à permuter. On observe que l'on peut encore permuter deux nombres consécutifs :

$$8 - 9 - 15 - 14 - 21 - 35 - 42 - 59 - 63 - 69$$

On obtient alors la liste triée :

$$8 - 9 - 14 - 15 - 21 - 35 - 42 - 59 - 63 - 69$$

4.2 Algorithme

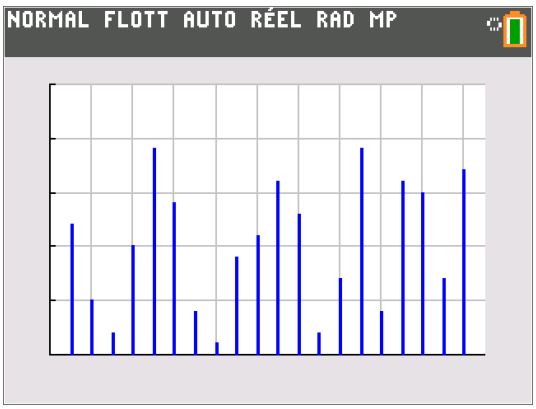
Le but de cet algorithme est de générer 20 nombres aléatoires entre 1 et 20 de les trier et de visualiser cette liste tout au long du processus.

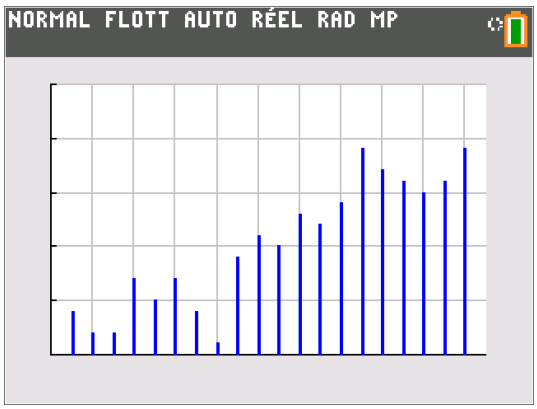
$P = 1$ lorsqu'il y a permutation et $P = 0$ dans le cas contraire.

```
NORMAL FLOTT AUTO RÉEL RAD MP
PROGRAM:TRIPEIGN
:EffDess
:EffListe L1
:For(I,1,20)
:nbrAléatEnt(1,20)→L1(I)
:Ligne(I,0,I,L1(I))
:End
:20→N
:While P=1 ou N>1
:0→P
:ent(N/1.3)→N
:If N<1
:1→N
:For(I,1,20-N)
:If L1(I)>L1(I+N)
:Then
:L1(I+N)→A
:L1(I)→L1(I+N)
:A→L1(I)
:1→P
:End
:End
:EffDess
:For(J,1,20)
:Ligne(J,0,J,L1(J))
:End
:End
```

Variables : I, N, P, A, J : entiers L_1 : liste
Entrées et initialisation
Effacer dessin
Effacer liste L_1
pour I de 1 à 20 **faire**
AléaEnt$(1,20) \to L_1(I)$
Tracer segment $(I\ ,\ 0\ ,\ I\ ,\ L_1(I))$
fin
$20 \to N$
Traitement et sorties
tant que $P = 1$ ou $N > 1$ **faire**
$0 \to P$ *permutation = faux*
$\text{Ent}\left(\frac{N}{1,3}\right) \to N$ *on calcule le pas*
si $N < 1$ **alors**
$1 \to N$ *pas minimum de 1*
fin
pour I de 1 à $20 - N$ **faire**
si $L_1(I) > L_1(I+N)$ **alors**
$L_1(I+N) \to A$
$L_1(I) \to L_1(I+N)$
$A \to L_1(I)$
$1 \to P$ *permutation = vrai*
fin
fin
pour J de 1 à 20 **faire**
Tracer segment $(J\ ,\ 0\ ,\ J\ ,\ L_1(J))$
fin
fin

On obtient la première et la quatrième visualisation suivantes :





5 Tri rapide

On trie cette fois-ci par partition ou segmentation (*quicksort* en anglais). On détermine un pivot dans la liste qui permet de diviser celle-ci en deux partie : une dont les éléments sont inférieurs ou égaux au pivot et une autre dont les éléments sont supérieurs au pivot.

On réitère ce processus avec la partie inférieure et la partie supérieure.

Le principe est donc de "*diviser pour régner*".

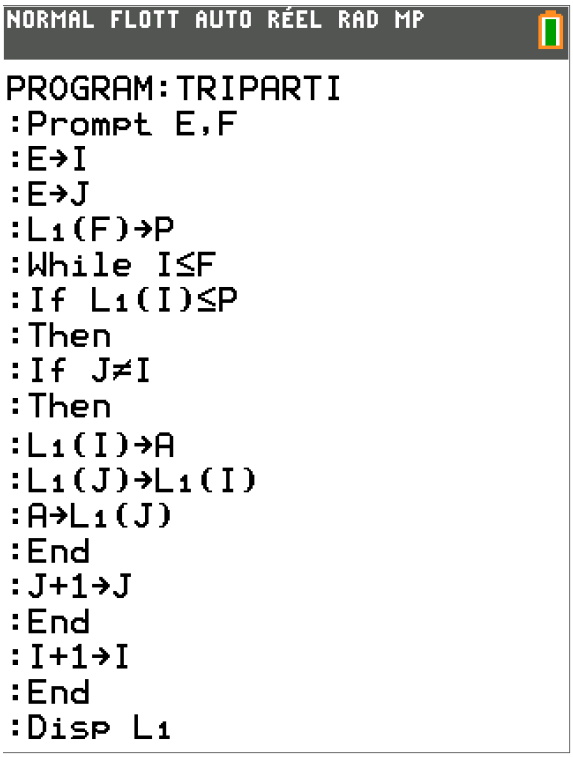
Le tri rapide est l'un des tris les plus rapide et donc l'un des plus utilisé. Cependant ce tri ne peut pas tirer avantage du fait que l'entrée est déjà presque triée. Dans ce cas particulier, il est plus avantageux d'utiliser le tri par insertion.

5.1 Exemple

Dans l'exemple suivant, on prendra comme pivot l'élément le plus à droite.

Soit la liste suivante : 6 - 10 - 23 - 37 - 16 - 5 - 66 - 8 - 90 - 23

On prend comme pivot 23 et l'on cherche un algorithme qui permette de partitionner la série entre les éléments plus petits ou égaux à 23 et ceux plus grand.
On peut proposer l'algorithme de partition suivant entre le E-ième et le F-ième élément de la liste L_1 préalablement rentrée :



Variables : I, J, P, E, F, A : entiers L_1 : liste
Entrées et initialisation
Lire E, F
$E \to I$
$E \to J$
$L_1(F) \to P$ *pivot élément le + à droite*
Traitement
tant que $I \leqslant F$ **faire**
si $L_1(I) \leqslant P$ **alors**
si $J \neq I$ **alors**
$L_1(I) \to A$
$L_1(J) \to L_1(I)$ *On permute*
$A \to L_1(J)$ *les éléments*
fin
$J + 1 \to J$
fin
$I + 1 \to I$
fin
Sorties : Afficher L_1

Si l'on détaille l'algorithme à la série, en prenant $E = 1$ et $F = 10$. En rouge les éléments qui ont permutés.

I	J	Début du tant que									
1	1	6	10	23	37	16	5	66	8	90	23
2	2	6	10	23	37	16	5	66	8	90	23
3	3	6	10	23	37	16	5	66	8	90	23
4	4	6	10	23	37	16	5	66	8	90	23
5	4	6	10	23	16	37	5	66	8	90	23
6	5	6	10	23	16	5	37	66	8	90	23
7	6	6	10	23	16	5	37	66	8	90	23
8	6	6	10	23	16	5	8	66	37	90	23
9	7	6	10	23	16	5	8	66	37	90	23
10	7	6	10	23	16	5	8	23	37	90	66
11	8	Fin du tant que									

On sépare la série en deux et on réitère le processus :

$E=1$ et $F=8-2=6$								
I	J	Début du tant que						
1	1	6	10	23	16	5	8	
2	2	6	10	23	16	5	8	
3	2	6	10	23	16	5	8	
4	2	6	10	23	16	5	8	
5	2	6	5	23	16	10	8	
6	3	6	5	8	16	10	23	
7	4	Fin du tant que						

$E=7$ et $F=10$					
I	J	Début du tant que			
7	7	23	37	90	66
8	8	23	37	90	66
9	9	23	37	90	66
10	9	23	37	66	90
11	10	Fin du tant que			

On sépare de nouveau la série

$E=1, F=4-2=2$			
I	J	D. tq	
1	1	6	5
2	1	5	6
3	2	F. tq	

$E=3$ et $F=8-2=6$					
I	J	Début tant que			
3	3	8	16	10	23
4	4	8	16	10	23
5	5	8	16	10	23
6	6	8	16	10	23
7	7	Fin tant que			

$E=3$ et $F=7-2=5$				
I	J	Début tq		
3	3	8	16	10
4	4	8	16	10
5	4	8	10	16
6	5	Fin tq		

Le programme passe ensuite toute la série en revue et ne fait aucune permutation. Il affiche alors : 5 - 6 - 8 - 10 - 16 - 23 - 23 - 37 - 66 - 90

5.2 L'algorithme

L'algorithme du tri rapide est récursif c'est à dire qu'il s'appelle lui-même. Il sépare les éléments inférieurs ou égaux au pivot et les éléments supérieurs. Il procède ensuite de même avec les deux séries ainsi formées et ainsi de suite comme les tableaux précédents.

Malheureusement, on ne peut faire de programme récursif avec la Ti82 ou Ti83. Voici le programme en pseudo-code et celui avec xcas.

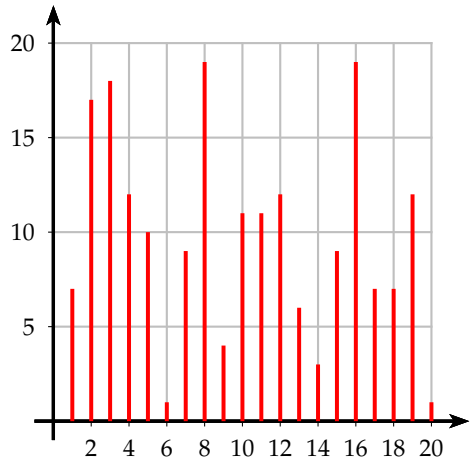
Avant d'appeler le programme, on crée une liste de 20 nombres aléatoire entre 1 et 20.

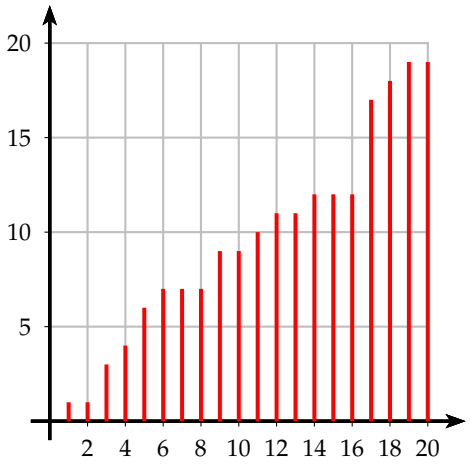
Variables : I : entier L : liste
Entrées et initialisation
  Effacer liste L
Traitement
  pour I de 1 à 20 **faire**
    AléaEnt$(1,20) \to L(I)$
  fin

On programme ensuite un programme tri_rapide qui prend trois arguments : la liste, l'élément le plus à gauche G et l'élément le plus à droite D. On visualise ensuite à chaque étape du tri.

tri_rapide(T,G,D)
Variables : I, C, M, P, A : entiers L : liste
Entrées et initialisation
  $T \to L$
  $G \to M$ *M représente le mur de séparation des deux séries*
  $L(D) \to P$ *On prend comme pivot l'élément le + à droite*
  Effacer dessin
  pour I de 1 à 20 **faire**
    Tracer segment $(I\ ,\ 0\ ,\ I\ ,\ L_1(I))$
  fin
Traitement
  tant que $C \leqslant D$ **faire**
    si $L(C) \leqslant P$ **alors**
      si $M \neq C$ **alors**
        $L(C) \to A$
        $L(M) \to L(C)$ *On permute les éléments $L_1(I)$ et $L_1(J)$*
        $A \to L(M)$
      fin
      $M+1 \to M$
    fin
    $C+1 \to C$ *On passe à l'élément suivant*
  fin
  si $G < M-2$ **alors**
    tri_rapide(L , G ,$M-2$) *tri de la série inf. au pivot*
  fin
  si $M-1 < D$ **alors**
    tri_rapide(L ,$M-1$, D) *tri de la série sup. au pivot*
  fin
Sorties : Afficher L

On obtient la première et la dernière visualisation suivantes à l'aide de xcas :





En programmation xcas, on a :

```
rapide(T,G,D) :={
  local I, C, A, P;
  L :=T;
  M :=G;
  C :=G;
  P :=L[[D]];
  ClrGraph();
  affichage(rouge+line_width_3);

  pour I de 1 jusque length(L) faire
    affichage(segment(point(I , 0) , point(I , L[[I]]) ));
  fpour;

  WAIT(0.1);

  tantque C<=D faire
    si L[[C]]<=P alors
      si M<>C alors
        A :=L[[C]]; L[[C]] :=L[[M]]; L[[M]] :=A; fsi;
      M :=M+1; fsi;
    C :=C+1;
  ftantque;

  si G<M-2 alors rapide(L,G,M-2); fsi
  si M-1<D alors rapide(L,M-1,D); fsi;

  return(L);
};
```

```
L :=makelist(x->alea(20)+1 , 1 , 20);
afficher(L);
rapide(L,1,20);
afficher(L);
```

```
// Succès
L : [15,8,20,19,16,4,20,7,4,14,15,20,7,8,14,10,6,6,12,5]
L : [4,4,5,6,6,7,7,8,8,10,12,14,14,15,15,16,19,20,20,20]
```