Sorting Algorithms & Convex Hull

Elias Moons (traduction par Guillaume Derval)

March 7, 2016

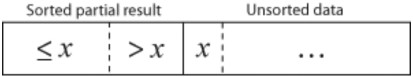
Table of Contents

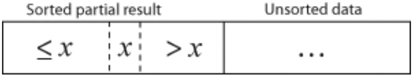
Table of Contents

Table of Contents

Tri par insertion

Idea

- A chaque itération, prendre une élement dans la partie non triée du tableau et le mettre au bon endroit dans la partie triée
- 

- deviens
- 

Tri par insertion

6	5	3	1	8	7	2	4

Tri par insertion

6	5	3	1	8	7	2	4

6	5	3	1	8	7	2	4

Tri par insertion

6	5	3	1	8	7	2	4

5	6	3	1	8	7	2	4

5	6	3	1	8	7	2	4

Tri par insertion

5	6	3	1	8	7	2	4

5	3	6	1	8	7	2	4

3	5	6	1	8	7	2	4

3	5	6	1	8	7	2	4

Tri par insertion

3	5	6	1	8	7	2	4

3	5	1	6	8	7	2	4

3	1	5	6	8	7	2	4

1	3	5	6	8	7	2	4

1	3	5	6	8	7	2	4

Tri par insertion

1	3	5	6	8	7	2	4

1	3	5	6	8	7	2	4

Tri par insertion

1	3	5	6	8	7	2	4

1	3	5	6	7	8	2	4

1	3	5	6	7	8	2	4

Tri par insertion

1	3	5	6	7	8	2	4

1	3	5	6	7	2	8	4

1	3	5	6	2	7	8	4

1	3	5	2	6	7	8	4

1	3	2	5	6	7	8	4

1	2	3	5	6	7	8	4

1	2	3	5	6	7	8	4

Tri par insertion

1	2	3	5	6	7	8	4

1	2	3	5	6	7	4	8

1	2	3	5	6	4	7	8

1	2	3	5	4	6	7	8

1	2	3	4	5	6	7	8

1	2	3	4	5	6	7	8

Tri par insertion

Complexité

Moyenne	Meilleure	Pire	Mémoire
$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$

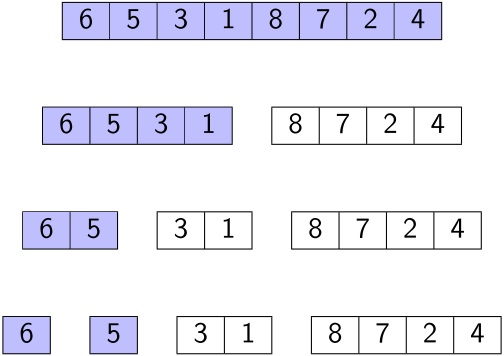
Table of Contents

Merge Sort

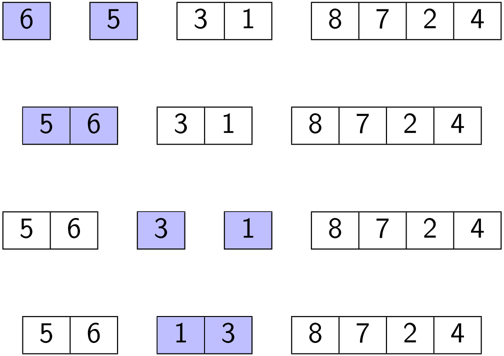
Idée

- Division récursive du tableau en "moitiés", jusqu'à atteindre des tableaux de taille 1
- Fusion des sous-tableaux pour produire des sous-tableau plus grand et triés

Merge Sort



Merge Sort



Merge Sort

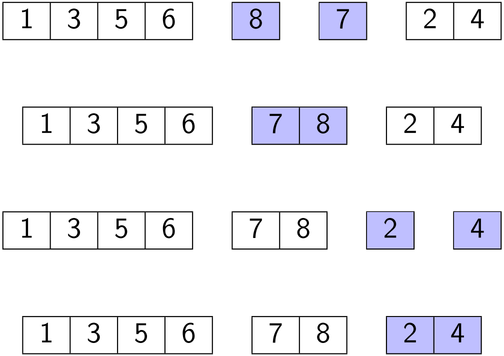
5	6		1	3		8	7	2	4

1	3	5	6		8	7	2	4

1	3	5	6		8	7		2	4

1	3	5	6		8		7		2	4

Merge Sort



Merge Sort

1	3	5	6

7	8

2	4

1	3	5	6

2	4	7	8

1	2	3	4	5	6	7	8

Moyenne	Meilleure	Pire	Mémoire
O(n log(n))	O(n log(n))	O(n log(n))	O(n)

Table of Contents

Quicksort

Idée

- Choisir un élement dans le tableau (le pivot)
- Partionnement: mettre les élements du tableau plus grand que le pivot à droite de celui-ci (et donc les éléments plus petits à gauche)
- Appliquer récursivement à gauche et à droite

Quicksort

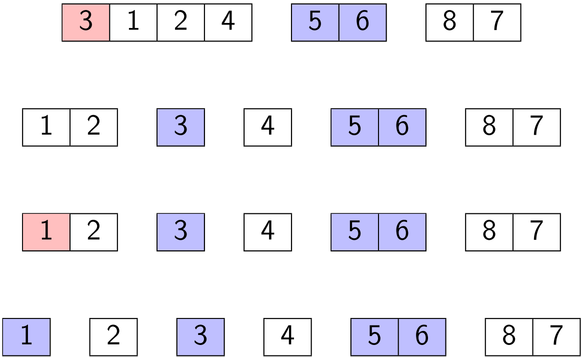
6	5	3	1	8	7	2	4

5	3	1	2	4		6		8	7

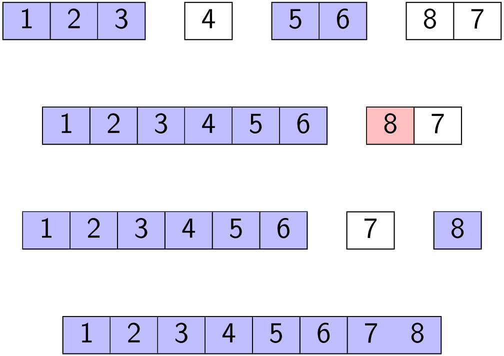
5	3	1	2	4		6		8	7

3	1	2	4		5	6		8	7

Quicksort



Quicksort



Quicksort

Complexité

Moyenne	Meilleure	Pire	Mémoire
$O(n \log(n))$	$O(n \log(n))$	$O(n^2)$	$O(1)$

Table of Contents

Tas (Heap)

Le tas est une structure de donnée qui a deux opérations:

- `push`: ajouter un élément au tas. $O(\log n)$.
- `pop`: retire l'élément le plus grand. $O(\log n)$.

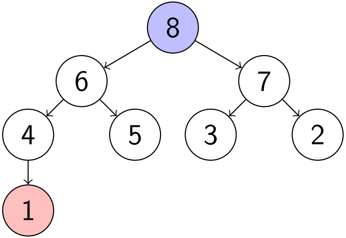
Tri par tas (Heap sort)

Idée

- Mettre tout les élements dans un tas
- Retirer les élements du tas, un à un

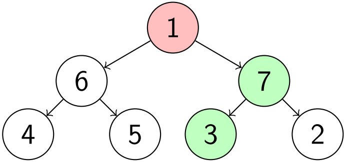
Heapsort

8	6	7	4	5	3	2	1

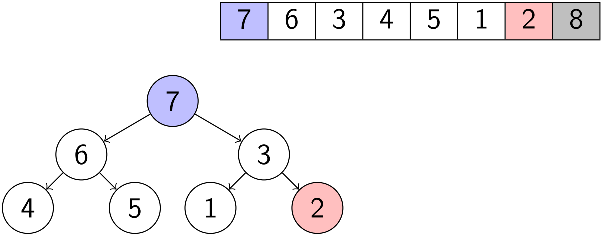


Heapsort

1	6	7	4	5	3	2	8

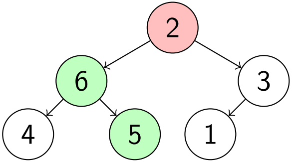


Heapsort



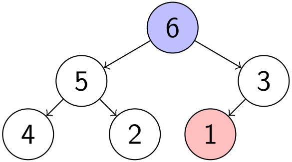
Heapsort

2	6	3	4	5	1	7	8



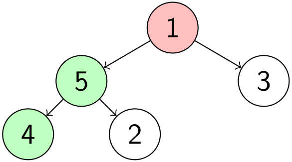
Heapsort

6	5	3	4	2	1	7	8



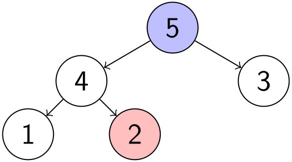
Heapsort

1	5	3	4	2	6	7	8



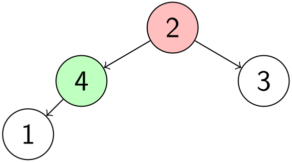
Heapsort

5	4	3	1	2	6	7	8



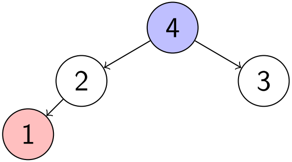
Heapsort

2	4	3	1	5	6	7	8

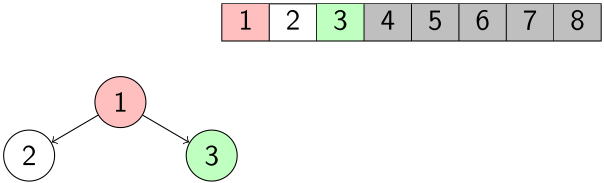


Heapsort

4	2	3	1	5	6	7	8

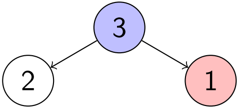


Heapsort



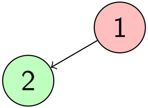
Heapsort

3	2	1	4	5	6	7	8



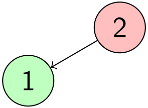
Heapsort

1	2	3	4	5	6	7	8



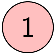
Heapsort

2	1	3	4	5	6	7	8



Heapsort

1	2	3	4	5	6	7	8



Heapsort

1	2	3	4	5	6	7	8

Heapsort

Complexité

Moyenne	Meilleure	Pire	Mémoire
O(n log(n))	O(n log(n))	O(n log(n))	O(1)

Table of Contents

Table of Contents

Convex Hull

Given a set of points in the plane, compute the smallest convex polygon in the plane that contains all the points.

Table of Contents

Graham Scan

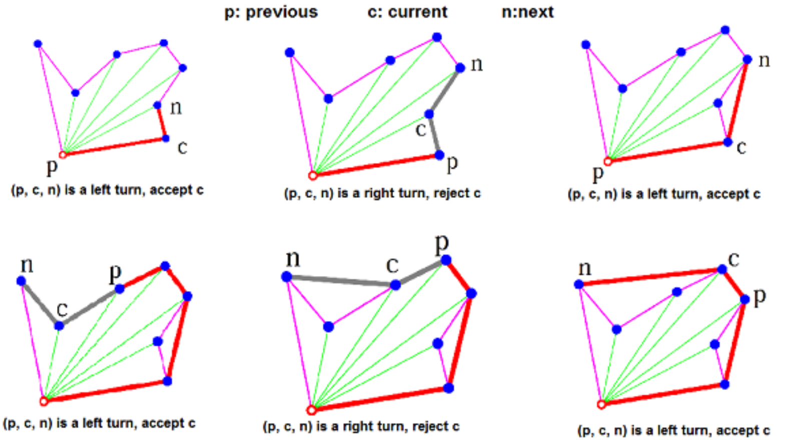
Method of computing the convex hull of a finite set of points in the plane with time complexity O(n log(n)). The algorithm finds all vertices of the convex hull ordered along its boundary.

Graham Scan

Idea

- find the point with the lowest y-coordinate. if there are multiple points with the lowest y-coordinate, the pick that one of them with the lowest x-coordinate. call this point P
- now sort all the points in increasing order of the angle they and point P make with the x-axis
- consider each point in the sorted array in sequence. For each point determine whether coming from the 2 previous points it makes a left of a right turn.
- if it makes a left turn, proceed with the next point
- if it makes a right turn, the second-to-last point is not part of the convex hull and should be removed from the convex hull, continue this removing for as long as the last 3 points make up a right turn

Graham Scan



In the above algorithm and below code, a stack of points is used to store convex hull points. With reference to the code, p is next-to-top in stack, c is top of stack and n is points[i].

Graham Scan

Direction of the turn

To determine whether 3 points constitute a left or a right turn we do not have to compute the actual angles but we can use a cross product.

Consider the 3 points (x_1, y_1), (x_2, y_2) and (x_3, y_3), which we will call P_1, P_2 and P_3.

Now compute the z-component of the cross product of the vectors P_1P_2 and P_1P_3. Which is given by the expression

$(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)$.

If the result is 0, the points are collinear. If the result is positive, the 3 points constitute a left turn. If the result is negative, the 3 points constitute a right turn.