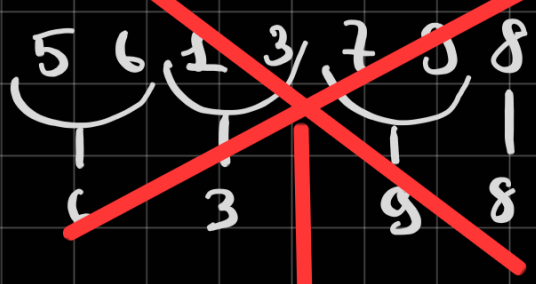
Merge insertion sort

Binary tree containers ?

input X of n elements

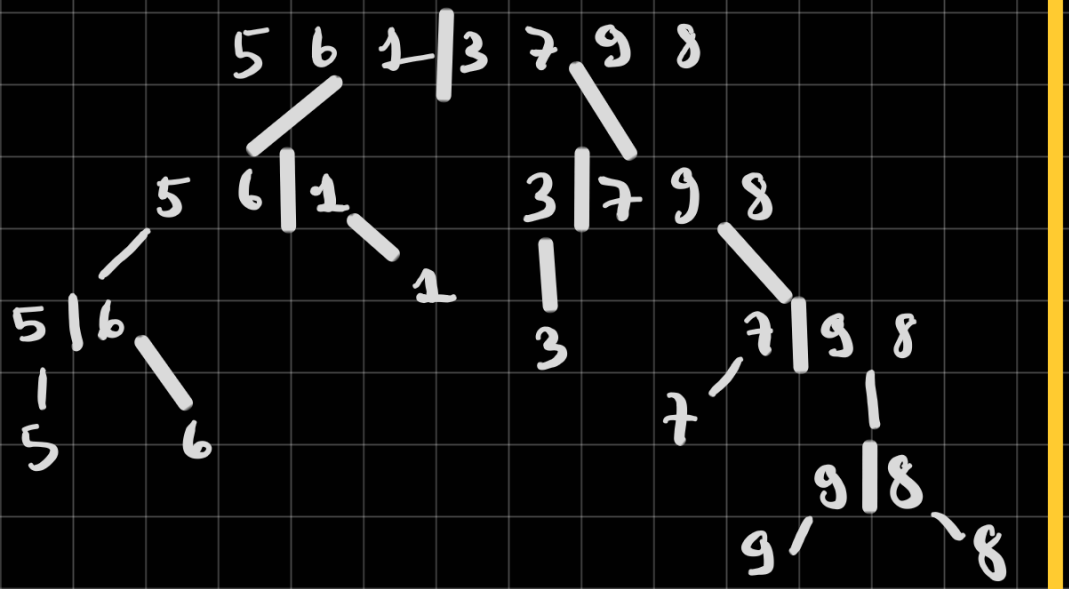
Grouping X in $\frac{n}{2}$ pair of elements

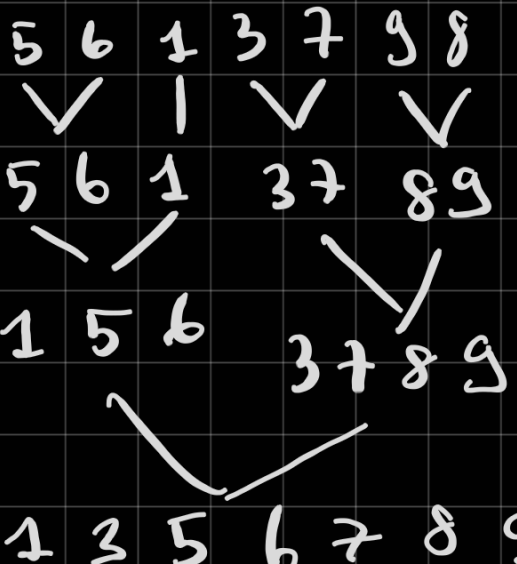


Determine the largest of the pair

Recursively sort n/2 larger elements

```
? int r_sort(vector<int> tab)
  {   if (tab[i
```





MERGE SORT O(N*log(N))

5 6 1 3 7 9 8

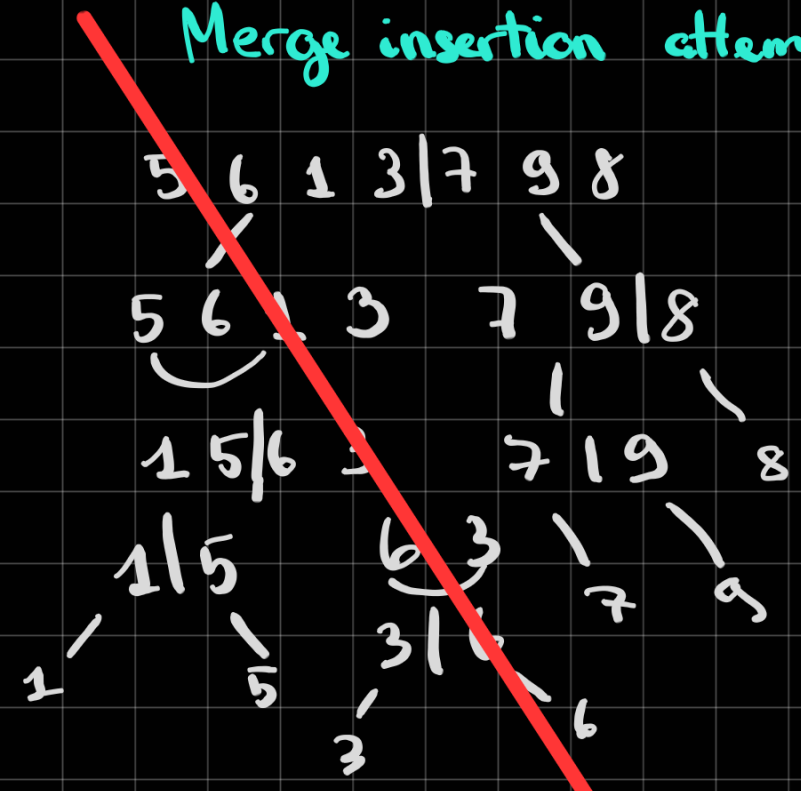
1 5 6 3 7 9 8

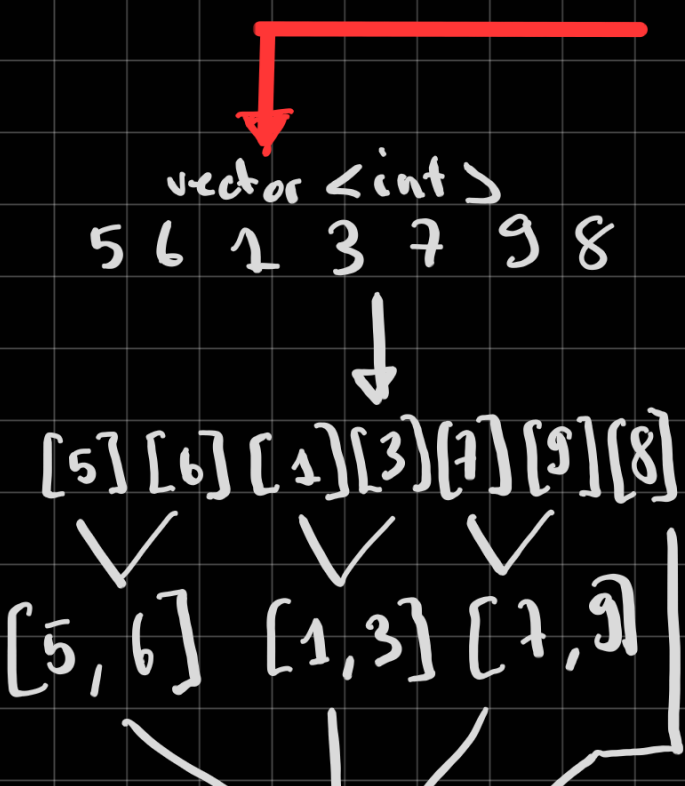
1 3 5 6 7 9 8

1 3 5 6 7 8 9

INSERTION SORT
$O(n^2)$

Merge insertion attempt





```
createpairs ( vector< deque <int>> sub )
for(i=0; i<size-1; i++)
{   if sub[1][0] >= sub[0][0]
```

[2] [1]

[1,5] [3,6] [7,8] [9]

[1,3,5,6,7,9,8]

1,3,5,6,7,8,9

subarrays:

sub array

[1,5] [4,2]

[0][0-1], [1][0,1]

wanted: [1,4,2,5]

j i

tab[0][0-1]

[1][0-1]

[[1, 2]

[4, 8]]

i=0

push(1)

push(4)

1 2 8 4 5 6 70 3 9

[1,2], [4,8], [5,6], [3,70], [9]

1,4,2,8,3,5,6,70,9

remerge(vector<int>& array, vector<deque<int>> sub

[1,2][4,8][5,6][3,70][9]

2

[1,4] [2,8] [3,5] [6,70] [9]

```
void merge_sort(T&arr, int threshold=4)
{ if(arr.size() > 1)
  {  si num sequence > 1
     if (arr.size() <= threshold) sinon merge sort
     {  insertion sort
        for(i=1; i<arr.size(); i++)
        {   for (j=i; j>0 && arr[j-1] > arr[j]
                swap(arr[j-1], arr[j])      j--;)
```

↳ tant que i<arr.size(); i++
j=i et tant que j>0 et arr[j-1] > arr[j]

EXPLICATION

je swap → exemple :

10, 3, 4, 0

i=1, j=1 → swap(arr[0], arr[1])
3, 10, 4, 0

j=0

arr[1] > arr[2] swap
3, 4, 10

i++ → i=2, j=2

i++ → i=3, j=3

3, 4, 10, 0

arr[2] > arr[3]
↳ swap 3, 4, 0, 10
↳ j-- = 2

3, 0, 4, 10

↳ swap
↳ j-- = 1
↳ swap 0, 3, 4, 20
↳ j = 0

NO

}

}

}

else

{

merge_sort

T left
T right
int middle = arr.size()/2;
for(i=0; i < middle; i++)
{ left.push_back (arr[i]); }
for(middle; i < arr.size(); " ") {
right.push_back (arr[i]); }

deux tableau séparé

RÉCURSIVITÉ

merge_sort (left) ⟶ insertion + redivision
merge_sort (right)