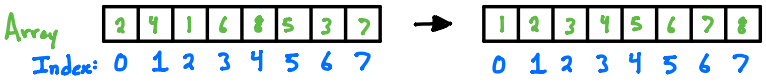
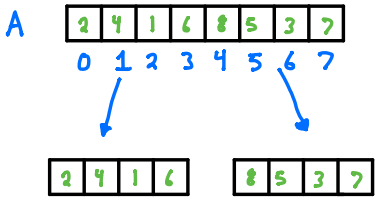
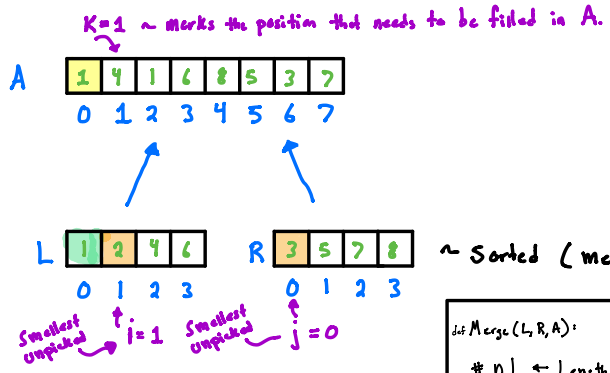
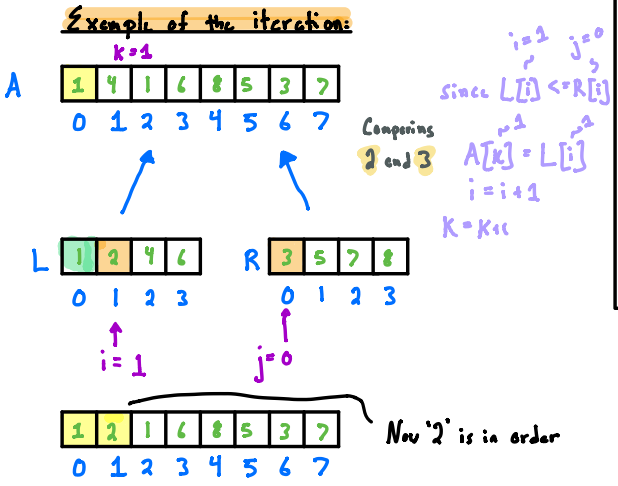
Merge Sort





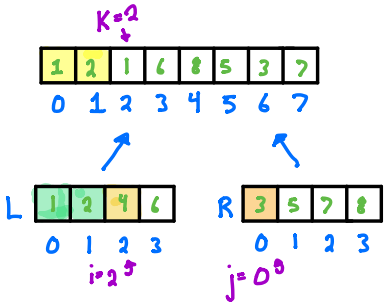




Sudo Code:

```
def Merge(L,R,A):
  # n L ← Length(L)   List
  # n R ← Length(R)   List
  i = j = k = 0
  while(i < #nL  and  j < #R):
    if(L[i] <= R[j]):        ~ Comparing smallest unpicked in L with the smallest unpicked in R
      A[k] = L[i]
      i = i+1
    else:
      A[k] = R[j]
      j = j+1
    K = K+1   ← moving to next index in our sorted list
```

next iteration:

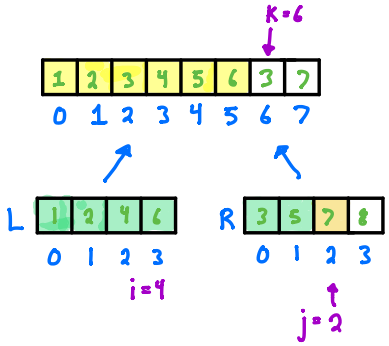


~ no we compare at i=2 and j=0 for position at K=2 index.

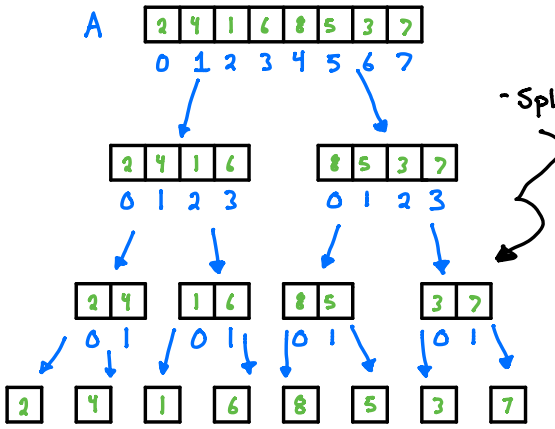
What happens when either i or j index is finished first? (are conditional `while (i < #nL and j < #R):` is false)

```
def Merge(L, R, A):
    # nL ← Length(L)    List
    # nR ← Length(R)    List
    i = j = k = 0

    while (i < #nL and j < #R):
        if (L[i] <= R[j]):
            A[k] = L[i]
            i = i + 1
        else:
            A[k] = R[j]
            j = j + 1
        k = k + 1

    while (i < #nL):
        A[k] = L[i]
        i = i + 1
        k = k + 1

    while (j < #nR):
        A[k] = R[i]
        j = j + 1
        k = k + 1
```

Once were out of this while loop only one while loop at the bottom will execute



Problem Solved

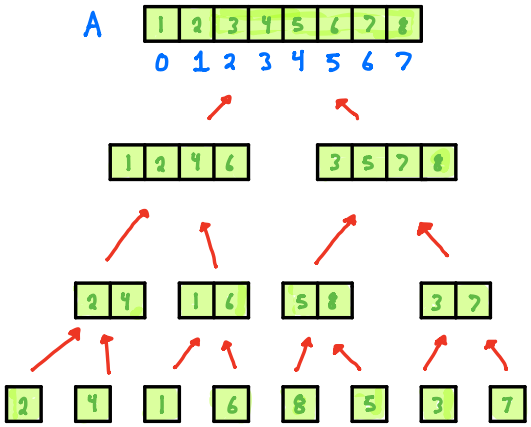


- Split array

Sort each sublist and merge them back

~ Once we reach a list with one element, were good!

Now we can merge them.

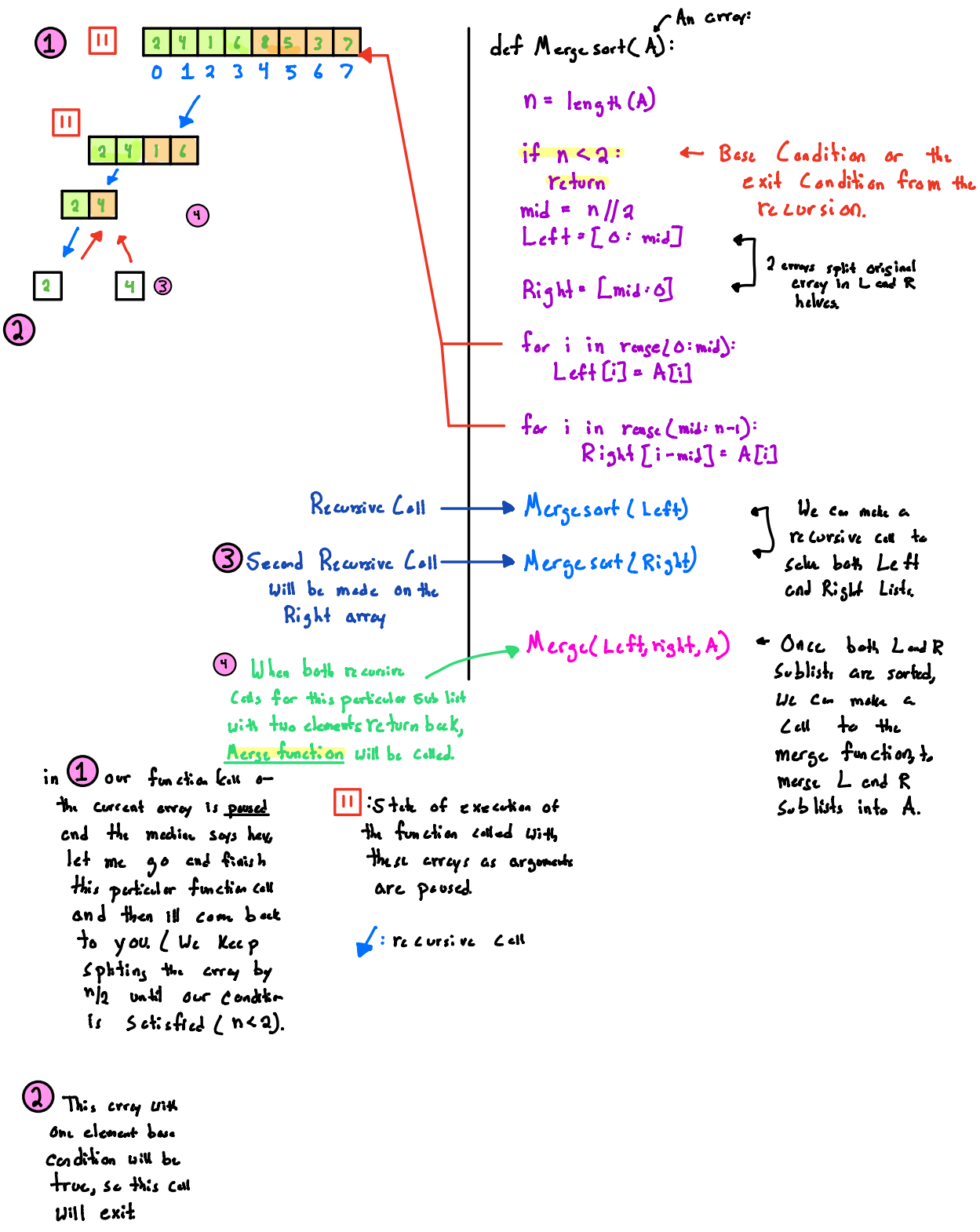



```
def Mergesort(A):          # An array:
    n = length(A)
    if n < 2:
        return
    mid = n//2
    Left = [0 : mid]
    Right = [mid : 0]

    for i in range(0:mid):
        Left[i] = A[i]

    for i in range(mid: n-1):
        Right[i-mid] = A[i]

    Mergesort(Left)
    Mergesort(Right)

    Merge(Left, right, A)
```

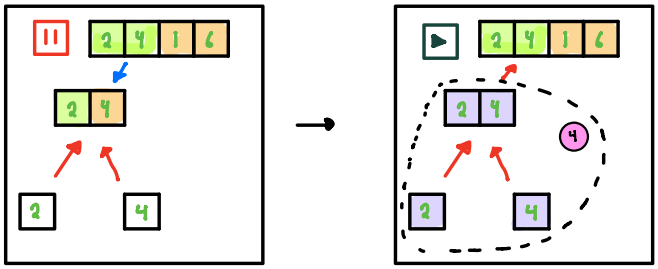
2 arrays split original array in L and R halves.

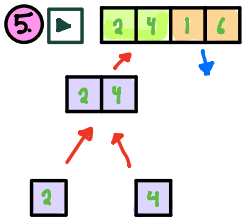
We can make a recursive call to solve both Left and Right lists.

Once both L and R sublists are sorted, we can make a call to the merge function to merge L and R sublists into A.


1
II
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
II
2 4 1 6
2 4
4
2
4
3
2
def Mergesort(A):
An array:
n = length(A)
if n < 2:
return
Base Condition or the exit Condition from the recursion.
mid = n//2
Left = [0 : mid]
Right = [mid : 0]
2 arrays split original array in L and R halves.
for i in range(0:mid):
Left[i] = A[i]
for i in range(mid: n-1):
Right[i-mid] = A[i]
Recursive Call
Mergesort(Left)
3 Second Recursive Call
Will be made on the Right array
Mergesort(Right)
We can make a recursive call to solve both Left and Right lists.
4 When both recursive Calls for this particular sub list with two elements return back, Merge function will be called.
Merge(Left, right, A)
Once both L and R Sublists are sorted, We can make a Call to the merge function, to merge L and R Sub lists into A.
in 1 our function call of the current array is paused and the machine says hey, let me go and finish this particular function call and then I'll come back to you. (We keep splitting the array by n/2 until our condition is satisfied (n<2).
II : Stack of execution of the function called with these arrays as arguments are paused
: recursive Call
2 This array with one element base condition will be true, so this call will exit

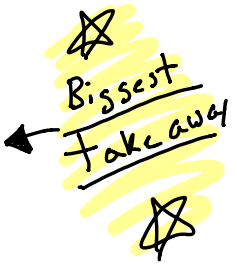
- Once Merge(Left, Right, A) is finished the control will return back to the execution of the previous sublist [2,4,1,6]. (▶)



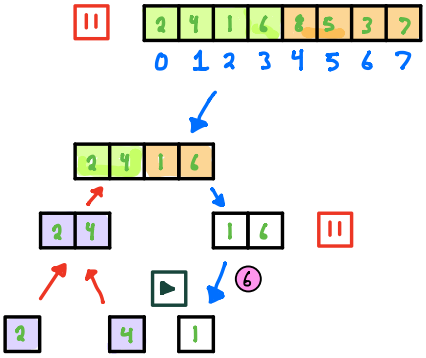


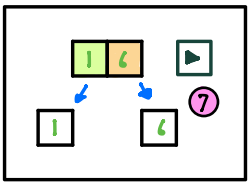
~ Now this sublist will have its control returned back

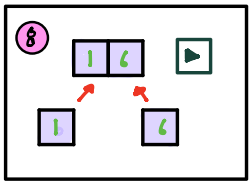
(5) : Now this guy will make the second Merge Sort call. It will call Mergesort(right), which will pass in the right array (which was [1,6]), and then were at the top of our def MergeSort(A) function.



Original Array:



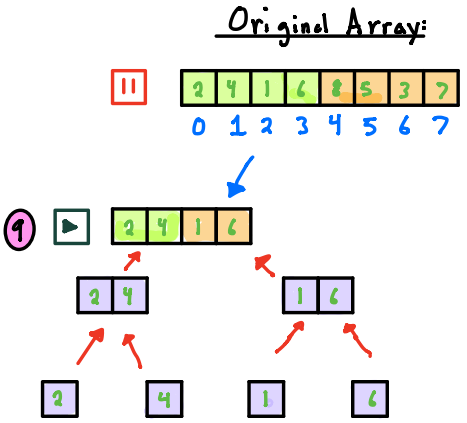




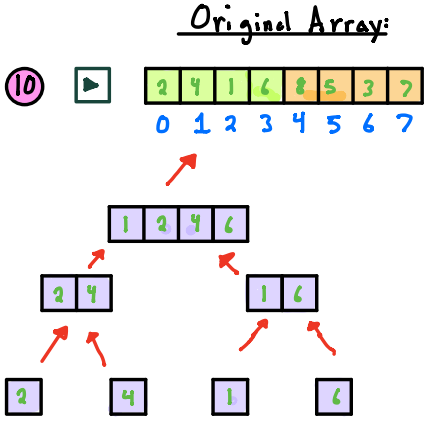
(6) : Now, the array [1,6] is [11]. and we dived the array, by making another recursive call (Mergesort(Left)).

(7) Since the single element is < 2 our Base Condion will move the program to the next line, which will execute the recursive call (Mergesort(Right)) on the right side of the array, which is 6.

(8) When both recursive calls for this particular sub list with two elements return back, Merge(Left, right, A) will be called.

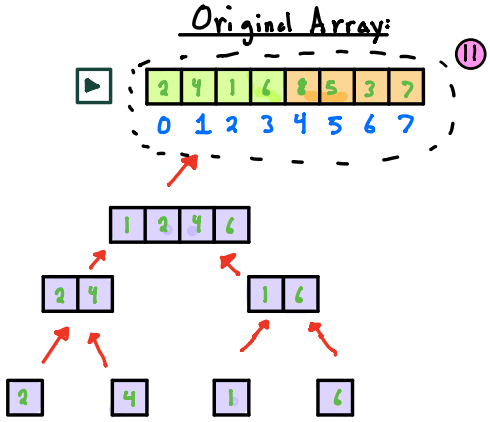


9: Then Control will return back to the array 2,4,1,6 and Merge(Left, right, A) will be called



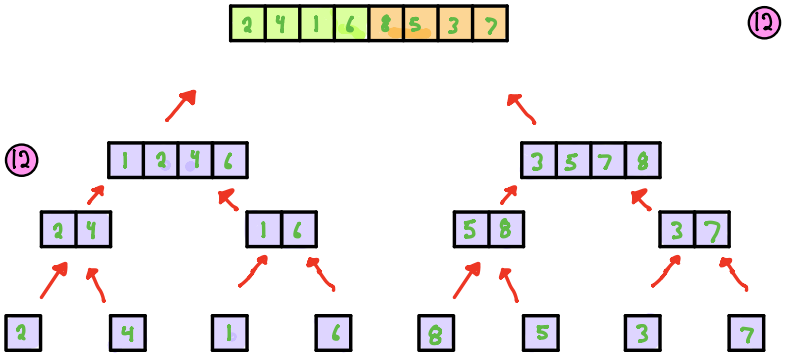
10 Once 1,2,4,6 will finish Merge(Left, right, A) the Control will return back to the function call corresponding to the original array

and then this guy



11 And then this guy will make another recursive call to do the right side of our original array.

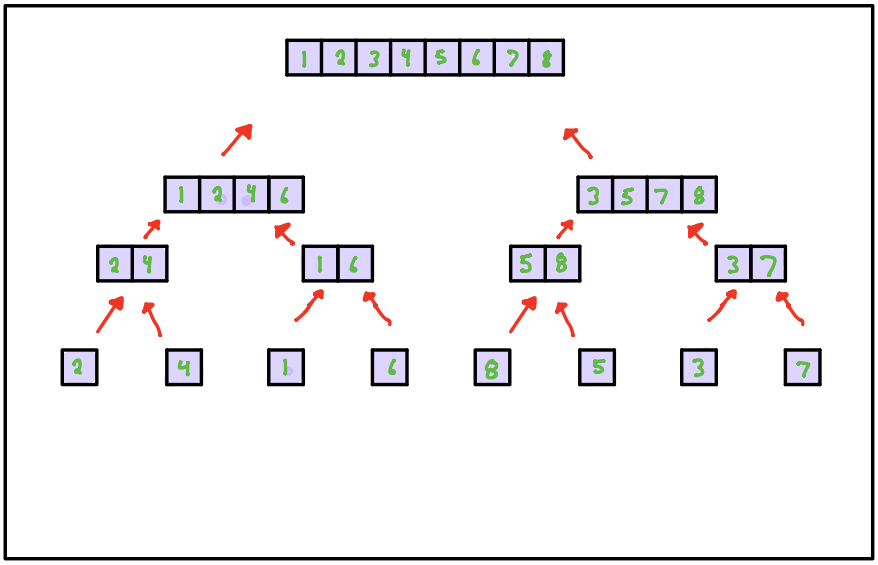
Original Array:



12 Repeat Algorithm on other side!

~ Then Merge(Left, right, A) will be called for 1 2 4 6 and 3 5 7 8

- Boom its merged!



Sorted Array using merge sort