# Quicksort

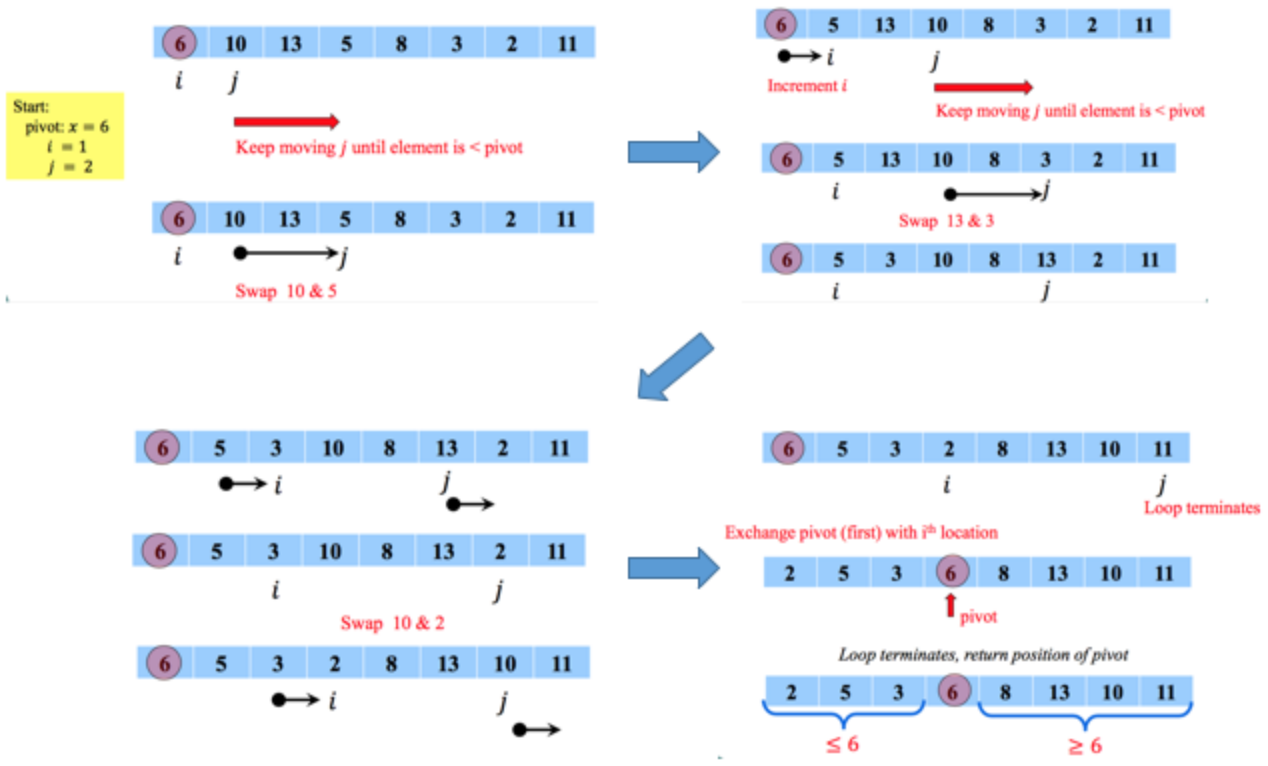
## Quicksort (1%)

### Discription

## PLEASE LISTEN CAREFULLY TO TAs!!!

Quicksort is a divide-and-conquer algorithm. It works by selecting a "pivot" element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. The sub-arrays are then sorted recursively. This can be done in-place, requiring small additional amounts of memory to perform the sorting. The quicksort algorithm consists of three steps:

1. **Divide**: Partition the array into 2 subarrays around a pivot `x`. ( `[...x-1] [...x+1]` )
2. **Conquer**: Sort the 2 subarrays by recursive calls to quicksort.
3. **Combine**: Trivial (subarrays are already sorted)




### `partition` function

The key step in quicksort is pivot selecting. First, pick an element, called a pivot, from the array. Then, reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). This is called Partitioning. After this partitioning, the pivot is in its final position. This is called the partition operation. For simplicity, we will choose the FIRST element of the array as the pivot.

For example, given `A[6] = {2 5 4 1 3 0}`. The pivot of `a` is `2`.

- Before partition `{3, 5, 0, 1, 2, 4}`
- After partition `{1, 0, 2, 4, 3, 5}`

Notice that every element on the left of 2 is smaller than 2, while every element on the right of 2 is greater than 2. The following is the pseudocode of partition:

```
1. Let the first element of the array be the pivot
2. Increase l from the left until A[l] > pivot
3. Decrease r from the right until A[r] < pivot
4. Swap A[l] and A[r]
5. Repeat step 2 and 3 until the halting condition (when does this algorithm stop?)
6. Return the index of the pivot position
```

Please implement the partition function based on the following format.

```
int partition(int* A, int l, int r){
    /*
    your code here
    */
    // return the pivot position, which is an int
}
```

## `quickSort` function

Use the `partition` function to implement the `quickSort` function. The `quickSort` function has three arguments:

1. `int *A`: the array needs to be sorted
2. `int l`: a left index of `A`
3. `int r`: a right index of `A`

The following code snippet is given for you to implement the `quickSort` function.

```
void quickSort(int*A, int l, int r){
    /*
    your code here
    */
}
```

This function will sort every values of `A` between `A[l]` and `A[r]`, both ends are included.