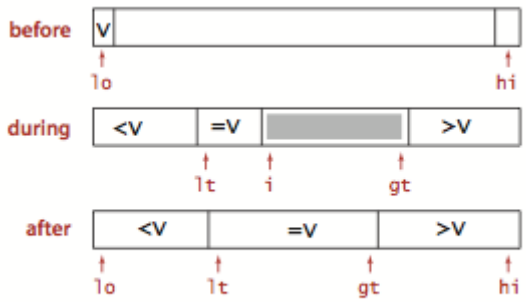
1. Write a program to implement randomized quicksort. Then use standard library function qsort(), that is used for sorting an array. Compare the running time of your program. Then compare the running time of your mergesort program and randomized quicksort program. For this experiment you should generate 1,000 random integers, then store them in an array, and use this array as input for your program.
[Pick a random pivot position with the help of rand() function (defined in stdlib.h header file)
The qsort() function is an implementation of the quicksort algorithm defined in stdlib.h header file]

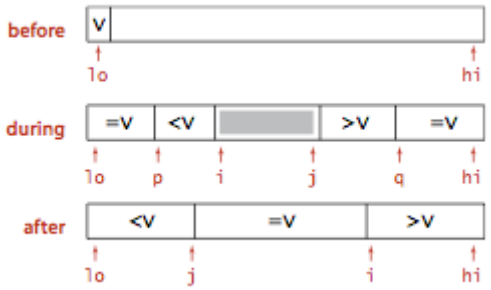
2. Given an array consisting of 0's, 1's and 2's only. Rearrange the array so that place all 0's first, then all 1's and all 2's in last. Write a program to implement an O(n) algorithm. You are not allowed to count the number of 0's, 1's and 2's in the given array and then store all the 0's in the beginning followed by all the 1's then all the 2's.
Input : {2, 0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 1, 0}
Output : {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2}
Input : {2,0,0,1,2,1}
Output : {0,0,1,1,2,2}
[**Hint :** Instead of 2-way partition the array in quicksort, partition the array into three parts, one each for items with keys smaller than pivot, equal to pivot, and larger than the pivot element. Let v be pivot element a[lo]. A single scan i from left-to-right through the array that maintains an index/pointer lt such that a[lo..lt-1] is less than v, an index/pointer gt such that a[gt+1..hi] is greater than v, and an index/pointer i such that a[lt..i-1] are equal to v, and a[i..gt] are not yet examined.
(a[i] < v): exchange a[lt] with a[i]; increment both lt and i
(a[i] > v): exchange a[gt] with a[i]; decrement gt
(a[i] == v): increment i



3-way partitioning overview

Starting with i equal to lo we process a[i] using the 3-way comparison to handle the three possible cases:
- a[i] less than v: exchange a[lt] with a[i] and increment both lt and i
- a[i] greater than v: exchange a[i] with a[gt] and decrement gt
- a[i] equal to v: increment i

Fast three-way partitioning - keep equal keys at both the left and right ends of the subarray. Maintain indices p and q such that a[lo..p-1] that a[q+1..hi] are all equal to a[lo], an index i such that a[p..i-1] are all less than a[lo] and an index j such that a[j+1..q] are all greater than a[lo]. Add to the inner partitioning loop code to swap a[i] with a[p] (and increment p) if it is equal to v and to swap a[j] with a[q] (and decrement q) if it is equal to v before the usual comparisons of a[i] and a[j] with v.



]

3. Given an unsorted array A of n unique elements, and an integer k, where $0 \leq k < n$. Write a program to find the k'th smallest element in the given array. Implement an O(n) in-place algorithm.