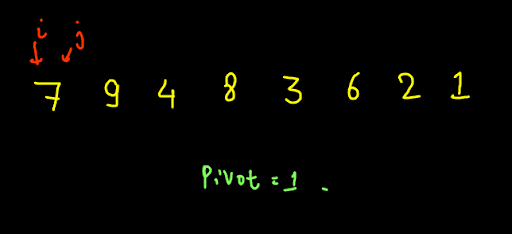
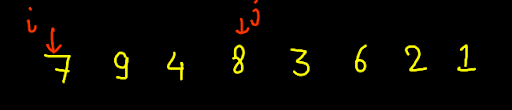
**1. PROBLEM DISCUSSION**

You are given an array(arr) of integers and a pivot. You have to re-arrange the given array in such a way that all elements smaller or equal to pivot lie on the left side of pivot and all elements greater than pivot lie on its right side. You have to achieve this in linear time.

**2. APPROACH**



Let's take the above example pivot element in this case is 1. So we have the arrange elements such that all the elements less than 1 are on the left and elements greater than 1 are on the right. So we will maintain two pointers i and j,using this we will divide the array into 3 parts.



Like in the above example we have traversed some part of the array. Elements less than or equal to pivot will be in from (0 to i-1),there are no such elements yet. Elements greater the pivot will be in from (i to j) ie. from(0 to 2) Elements in the range (j+1 to n) are unexplored ie. from(3 to 7). Now in the unexplored region we will traverse the array and one by one will visit a new element and move it to its correct region . we analyse arr[i]. If arr[i]>pivot then j is increased by 1 which means the unknown area is decreased by one and the area for greater elements is increased. Else if arr[i]<=pivot, then we swap the elements at i and j and increment both i and j by one. This decreases the unknown area by one element, less than or equal to area increases and the size of the greater than area remains the same but the area gets shifted forward.

ConsoleCpp

#include<bits/stdc++.h>

using namespace std;

void swap(int arr[], int i, int j) {

cout << "Swapping " << arr[i] << " and " << arr[j] << endl;

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

void partition(int arr[], int n, int pivot) {

int p1 = -1;

int itr = 0;

while (itr < n) {

if (arr[itr] <= pivot) {

swap(arr, itr, ++p1);

}

itr++;

}

}

void print(int arr[], int n) {

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

}

int main() {

int n;

cin >> n;

int arr[n];

for (int i = 0; i < n; i++) {

cin >> arr[i];

}

int pivot;

cin >> pivot;

partition(arr, n, pivot);

print(arr, n);

}

**3. ANALYSIS**

Time complexity O( N ). Space complexity s=O(1)