Name - Dhruv Saini
Section - ML

Roll No. - 2015241, 24
Subject - Design & Analysis of Algo.
TCS 505

Tutorial

1-6 Done in Assignment 1

7

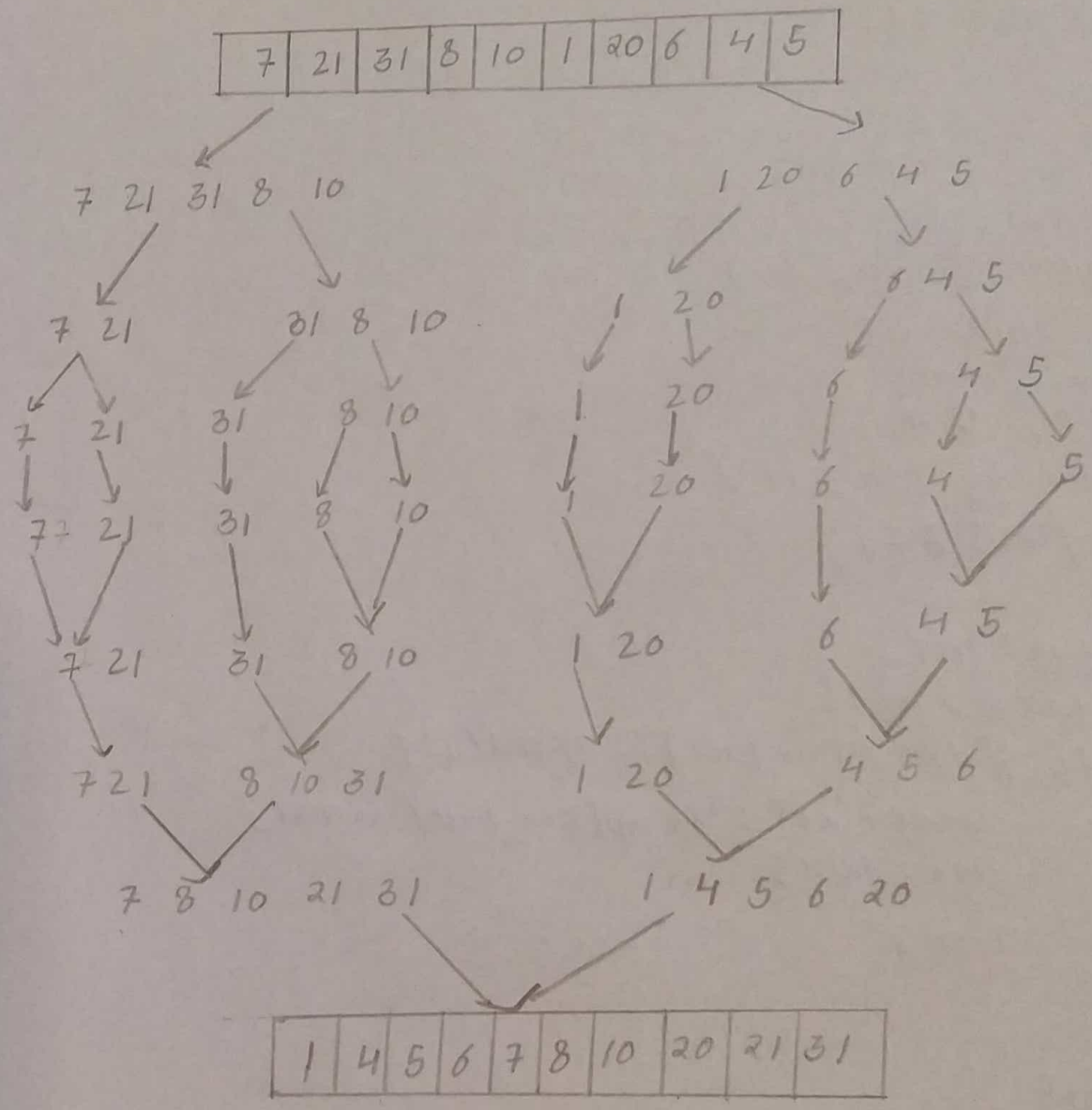
```
#include <iostream>
#include <vector>
#include <map>

using namespace std;

int main() {
    int n;          cin>>n;
    vector<int> v(n);
    for(int i=0; i<n; ++i)
        cin >> v[i];
    int k;   cin>>k;
    map<int, int> mp;
    for(int i=0; i<n; ++i)
      mp[v[i]] = i;

    bool flag = true;
    for( auto x : mp) {
        if(mp.find(k-x.first) != mp.end()) {
            cout << i << ":" << mp[k-x.first] << endl;
            flag = false;
            break;
        }
    }
    if(flag) {
        cout<<"No such pair found";
    }
    return 0;
}
```

8

Quicksort is the fastest general-purpose sort.
In most practical situations, Quicksort is the method of choice.
If stability is important & space is available,
Mergesort might be best.

9

Inversion count for an array indicates how far (or close) the array is from being sorted. If the array is already sorted, then the inversion count is zero, but if the array is sorted in the reverse order, the inversion count is maximum.



Total no. of inversions = 31

10

The best case for quick sort will be when the middle element is picked as a pivot.

The worst case for quick sort is when array is sorted in either increasing or decreasing order.

11

Recurrence Relation

- Best case

  Quick sort - $T(n) = 2T(n/2) + n$

  Merge sort - $T(n) = 2T(n/2) + n$

- Worst case

  Quick sort - $T(n) = T(n-1) + n$

  Merge sort - $T(n) = 2T(n/2) + n$

Similarities

1) Both the methods follow Divide & conquer approach.
2) Both have best case time complexities as $O(n \log n)$

Differences

1) Merge sort is a stable algorithm while quick sort is unstable algorithm.
2) Worst case time complexity of Merge sort is $O(n \log n)$ while that of quick sort is $O(n^2)$
3) Merge sort is external sorting algorithm where data is sorted in main memory.

12

```
// Selection Sort - Stable version
void selectionSortAlgo(int arr[], int n){
    for(int i=0; i<n-1; ++i){
        int min = i;
        for(int j=i+1; j<n; ++j){
            if(arr[min] > arr[j])
                min = j;
    }
    int key = arr[min];
    while(min > i){
        arr[min] = arr[min-1];
        --min;
    }
    arr[i] = key;
}
```

13

```
// Bubble sort
void bubbleSort(int arr[], int n){
    bool flag = true;
    for(int i=0; i<n-1; ++i){
        flag = true;
        for(int j=0; j<n-i-1; ++j){
            if(arr[j] > arr[j+1]){
                int t = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = t;
                flag = false;
            }
        }
```

```
        if (flag) {
            break;
        }
    }
```

14

For this purpose we will use external sorting technique i.e. Merge sort should be used.

Internal sorting - In internal sorting, all the data is stored in main memory all the time while sorting.

External sorting - In external sorting, data is stored in the slower external memory (usually Hard drive).

In the sorting phase, chunks of data small enough to fit in main memory are read, sorted & written out to a temporary file.