

# H1 Design and Analysis of Algorithms

## H2 Assignment 1

H3 By Himanshu Sardana

---

### H4 Question 1

Consider an array `arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}`, using Binary Search Method, find the target 23.

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main() {
6      vector<int> v = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
7      int target = 23;
8      int low = 0;
9      int high = v.size() - 1;
10
11     int idx = 0;
12     bool found = false;
13
14     while(low <= high) {
15         int mid = low + (high - low) / 2;
16         if(v[mid] == target) {
17             found = true;
18             idx = mid;
19             break;
20         }
21
22         else if(v[mid] < target) {
23             low = mid + 1;
24         }
```

```

25
26         else {
27             high = mid - 1;
28         }
29     }
30
31     if(found) cout << "Element found at index " << idx << '\n';
32     else cout << "Element not found" << "\n";
33 }

```

Output:

```

1 Element found at index 5

```

#### H4 Question 2

Implement Merge Sort for the given array `int arr[] = {12, 11, 13, 5, 6, 7}` .

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  void merge(vector<int>& v, int l, int m, int r) {
6      int n1 = m - l + 1;
7      int n2 = r - m;
8
9      vector<int> L(n1);
10     vector<int> R(n2);
11
12     for(int i = 0; i < n1; i++) {
13         L[i] = v[l + i];
14     }
15
16     for(int i = 0; i < n2; i++) {
17         R[i] = v[m + 1 + i];
18     }
19

```

```

20     int i = 0;
21     int j = 0;
22     int k = l;
23
24     while(i < n1 && j < n2) {
25         if(L[i] <= R[j]) {
26             v[k] = L[i];
27             i++;
28         } else {
29             v[k] = R[j];
30             j++;
31         }
32         k++;
33     }
34
35     while(i < n1) {
36         v[k] = L[i];
37         i++;
38         k++;
39     }
40
41     while(j < n2) {
42         v[k] = R[j];
43         j++;
44         k++;
45     }
46 }
47
48 void mergeSort(vector<int>& v, int l, int r) {
49     if(l >= r) {
50         return;
51     }
52
53     int m = l + (r - l) / 2;
54     mergeSort(v, l, m);
55     mergeSort(v, m + 1, r);

```

```

56         merge(v, l, m, r);
57     }
58
59     int main() {
60         vector<int> v = {12, 11, 13, 5, 6, 7};
61         mergeSort(v, 0, v.size() - 1);
62         for(int i : v) {
63             cout << i << " ";
64         }
65     }

```

Output:

```
1  5 6 7 11 12 13
```

#### H4 Question 3

Implement Quick Sort for the given array `int arr[] = {3, 2, 6, 9, 2}`.

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int partition(vector<int> &v, int low, int high) {
6      int pivot = v[high];
7      int i = low - 1;
8      for (int j = low; j < high; j++) {
9          if (v[j] < pivot) {
10             i++;
11             swap(v[i], v[j]);
12         }
13     }
14     swap(v[i + 1], v[high]);
15     return i + 1;
16 }
17
18 void quick_sort(vector<int> &v, int low, int high) {

```

```

19         if (low < high) {
20             int pi = partition(v, low, high);
21             quick_sort(v, low, pi - 1);
22             quick_sort(v, pi + 1, high);
23         }
24     }
25
26     int main() {
27         vector<int> v = {4, 2, 6, 9, 2};
28         quick_sort(v, 0, v.size() - 1);
29
30         for (int i = 0; i < v.size(); i++) {
31             cout << v[i] << " ";
32         }
33     }

```

Output:

```
1 2 2 4 6 9
```

#### H4 Question 4

You are given a one dimensional array that may contain both positive and negative integers, find the sum of contiguous subarray of numbers which has the largest sum.

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main() {
6      vector<int> v = {-2, -5, 6, -2, -3, 1, 5, -6};
7      int currSum = 0;
8      int maxSum = INT_MIN;
9
10     for(int i : v) {
11         currSum += i;
12         if(currSum < 0) {

```

```
13             currSum = 0;
14         }
15         maxSum = max(maxSum, currSum);
16     }
17
18     cout << maxSum;
19 }
```

Output:

```
1 7
```