

Name: Chandranshu Bhardwaj

Roll No: 102203797

Group: ZCO-17

DAA Lab Assignment 1

Binary Search – Iterative

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int binarySearch(int arr[], int l, int r, int x)
5  {
6      while (l <= r) {
7          int m = l + (r - l) / 2;
8
9          if (arr[m] == x)
10             return m;
11
12         if (arr[m] < x)
13             l = m + 1;
14
15         else
16             r = m - 1;
17     }
18
19     return -1;
20 }
21
22 int main()
23 {
24     int arr[] = {2, 5, 9, 15, 46, 74, 85, 91, 127};
25     int x = 91;
26     int n = sizeof(arr) / sizeof(arr[0]);
27     int result = binarySearch(arr, 0, n - 1, x);
28     if (result == -1)
29         cout << "Element is not present in array";
30     else
31         cout << "Element is present at index " << result;
32
33     return 0;
34 }
```

```
PS D:\DAA Assignments\Assignment 1> g++ BinarySearch-Iterative.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Element is present at index 7
PS D:\DAA Assignments\Assignment 1> 
```

Binary search – Recursive

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int binarySearch(int arr[], int l, int r, int x)
5  {
6      if (r >= l) {
7          int mid = l + (r - l) / 2;
8
9          if (arr[mid] == x)
10             return mid;
11
12         if (arr[mid] > x)
13             return binarySearch(arr, l, mid - 1, x);
14
15         return binarySearch(arr, mid + 1, r, x);
16     }
17
18     return -1;
19 }
20
21 int main()
22 {
23     int arr[] = {2, 5, 9, 15, 46, 74, 85, 91, 127};
24     int x = 91;
25     int n = sizeof(arr) / sizeof(arr[0]);
26     int result = binarySearch(arr, 0, n - 1, x);
27     if (result == -1)
28         cout << "Element is not present in array";
29     else
30         cout << "Element is present at index " << result;
31
32     return 0;
33 }
```

```
PS D:\DAA Assignments\Assignment 1> g++ BinarySearch-Recursive.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Element is present at index 7
PS D:\DAA Assignments\Assignment 1> █
```

Merge Sort – Iterative

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  void merge(int arr[], int l, int m, int r);
5
6  int min(int x, int y) { return (x<y)? x :y; }
7
8
9  void mergeSort(int arr[], int n)
10 {
11     int curr_size;
12     int left_start;
13
14     for (curr_size=1; curr_size<=n-1; curr_size = 2*curr_size)
15     {
16         for (left_start=0; left_start<n-1; left_start += 2*curr_size)
17         {
18             int mid = min(left_start + curr_size - 1, n-1);
19
20             int right_end = min(left_start + 2*curr_size - 1, n-1);
21
22             merge(arr, left_start, mid, right_end);
23         }
24     }
25 }
26
27 void merge(int arr[], int l, int m, int r)
28 {
29     int i, j, k;
30     int n1 = m - l + 1;
31     int n2 = r - m;
32
33     int L[n1], R[n2];
34
35     for (i = 0; i < n1; i++)
36         L[i] = arr[l + i];
37     for (j = 0; j < n2; j++)
38         R[j] = arr[m + 1+ j];
39
40     i = 0;
41     j = 0;
42     k = l;
43     while (i < n1 && j < n2)
44     {
45         if (L[i] <= R[j])
46         {
47             arr[k] = L[i];
```

```

48         i++;
49     }
50     else
51     {
52         arr[k] = R[j];
53         j++;
54     }
55     k++;
56 }
57
58 while (i < n1)
59 {
60     arr[k] = L[i];
61     i++;
62     k++;
63 }
64
65 while (j < n2)
66 {
67     arr[k] = R[j];
68     j++;
69     k++;
70 }
71 }
72
73 void printArray(int A[], int size)
74 {
75     int i;
76     for (i=0; i < size; i++)
77         cout <<" "<< A[i];
78     cout <<"\n";
79 }
80
81 int main()
82 {
83     int arr[] = {12, 11, 13, 5, 6, 7};
84     int n = sizeof(arr)/sizeof(arr[0]);
85
86     cout <<"Given array is \n ";
87     printArray(arr, n);
88
89     mergeSort(arr, n);
90
91     cout <<"\nSorted array is \n ";
92     printArray(arr, n);
93     return 0;
94 }

```

PS D:\DAA Assignments\Assignment 1> g++ MergeSort-Iterative.cpp

PS D:\DAA Assignments\Assignment 1> ./a.exe

Given array is

12 11 13 5 6 7

Sorted array is

5 6 7 11 12 13

PS D:\DAA Assignments\Assignment 1> █

Merge Sort – Recursive

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  void merge(int arr[], int l, int m, int r);
5
6  void mergeSort(int arr[], int l, int r)
7  {
8      if (l < r) {
9          int m = l + (r - l) / 2;
10         mergeSort(arr, l, m);
11         mergeSort(arr, m + 1, r);
12         merge(arr, l, m, r);
13     }
14 }
15
16 void merge(int arr[], int l, int m, int r)
17 {
18     int k;
19     int n1 = m - l + 1;
20     int n2 = r - m;
21
22     int L[n1], R[n2];
23
24     for(int i = 0; i < n1; i++)
25         L[i] = arr[l + i];
26     for(int j = 0; j < n2; j++)
27         R[j] = arr[m + 1 + j];
28
29     int i = 0;
30     int j = 0;
31     k = l;
32
33     while (i < n1 && j < n2)
34     {
35         if (L[i] <= R[j]) {
36             arr[k] = L[i];
37             i++;
38         }
39         else {
40             arr[k] = R[j];
41             j++;
42         }
43         k++;
44     }
```

```

46     while (i < n1)
47     {
48         arr[k] = L[i];
49         i++;
50         k++;
51     }
52
53     while (j < n2)
54     {
55         arr[k] = R[j];
56         j++;
57         k++;
58     }
59 }
60
61 void printArray(int A[], int size)
62 {
63     for(int i = 0; i < size; i++)
64         printf("%d ", A[i]);
65
66     cout << "\n";
67 }
68
69 int main()
70 {
71     int arr[] = {2, 51, 9, 125, 46, 74, 37, 91, 127};
72     int arr_size = sizeof(arr) / sizeof(arr[0]);
73
74     cout << "Given array is \n";
75     printArray(arr, arr_size);
76
77     mergeSort(arr, 0, arr_size - 1);
78
79     cout << "\nSorted array is \n";
80     printArray(arr, arr_size);
81     return 0;
82 }

```

```

PS D:\DAA Assignments\Assignment 1> g++ MergeSort-Recursive.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Given array is
2 51 9 125 46 74 37 91 127

Sorted array is
2 9 37 46 51 74 91 125 127

```

Quick Sort – Iterative

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  void swap(int* a, int* b)
5  {
6      int t = *a;
7      *a = *b;
8      *b = t;
9  }
10
11 int partition(int arr[], int l, int h)
12 {
13     int x = arr[h];
14     int i = (l - 1);
15
16     for (int j = l; j <= h - 1; j++) {
17         if (arr[j] <= x) {
18             i++;
19             swap(&arr[i], &arr[j]);
20         }
21     }
22     swap(&arr[i + 1], &arr[h]);
23     return (i + 1);
24 }
25
26 void quickSortIterative(int arr[], int l, int h)
27 {
28     int stack[h - l + 1];
29     int top = -1;
30
31     stack[++top] = l;
32     stack[++top] = h;
33
34     while (top >= 0) {
35
36         h = stack[top--];
37         l = stack[top--];
38     }
```

```

38
39     int p = partition(arr, l, h);
40
41     if (p - 1 > l) {
42         stack[++top] = l;
43         stack[++top] = p - 1;
44     }
45
46     if (p + 1 < h) {
47         stack[++top] = p + 1;
48         stack[++top] = h;
49     }
50 }
51 }
52
53 void printArr(int arr[], int n)
54 {
55     int i;
56     for (i = 0; i < n; ++i)
57         cout << arr[i] << " ";
58 }
59
60 int main()
61 {
62     int arr[] = {2, 51, 9, 125, 46, 74, 37, 91, 127};
63     int n = sizeof(arr) / sizeof(*arr);
64     quickSortIterative(arr, 0, n - 1);
65     printArr(arr, n);
66     return 0;
67 }

```

```

PS D:\DAA Assignments\Assignment 1> g++ .\QuickSort-Iterative.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
2 9 37 46 51 74 91 125 127
PS D:\DAA Assignments\Assignment 1>

```


Quick Sort – Recursive

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  void swap(int* a, int* b)
5  {
6      int temp = *a;
7      *a = *b;
8      *b = temp;
9  }
10
11 int partition(int arr[], int l, int h)
12 {
13     int x = arr[h];
14     int i = (l - 1);
15
16     for (int j = l; j <= h - 1; j++) {
17         if (arr[j] <= x) {
18             i++;
19             swap(&arr[i], &arr[j]);
20         }
21     }
22     swap(&arr[i + 1], &arr[h]);
23     return (i + 1);
24 }
25
26 void quickSort(int A[], int l, int h)
27 {
28     if (l < h) {
29         int p = partition(A, l, h);
30         quickSort(A, l, p - 1);
31         quickSort(A, p + 1, h);
32     }
33 }
```

```
35 int main()
36 {
37
38     int n = 5;
39     int arr[n] = {2, 51, 9, 125, 46, 74, 37, 91, 127};
40
41     quickSort(arr, 0, n - 1);
42
43     for (int i = 0; i < n; i++) {
44         cout << arr[i] << " ";
45     }
46
47     return 0;
48 }
49
```

```
PS D:\DAA Assignments\Assignment 1> g++ .\QuickSort-Recursive.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
2 9 46 51 125
PS D:\DAA Assignments\Assignment 1> █
```

Maximum Subarray Sum

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int max(int a, int b) {
5      return (a > b) ? a : b;
6  }
7
8  int max(int a, int b, int c) {
9      return max(max(a, b), c);
10 }
11
12 int maxCrossingSum(int arr[], int l, int m, int h) {
13     int sum = 0;
14     int left_sum = INT_MIN;
15     for (int i = m; i >= l; i--) {
16         sum = sum + arr[i];
17         if (sum > left_sum)
18             left_sum = sum;
19     }
20
21     sum = 0;
22     int right_sum = INT_MIN;
23     for (int i = m; i <= h; i++) {
24         sum = sum + arr[i];
25         if (sum > right_sum)
26             right_sum = sum;
27     }
28
29     return max(left_sum + right_sum - arr[m], left_sum, right_sum);
30 }
```

```
32 int maxSubArraySum(int arr[], int l, int h) {
33     if (l > h)
34         return INT_MIN;
35     if (l == h)
36         return arr[l];
37
38     int m = (l + h) / 2;
39
40     return max(maxSubArraySum(arr, l, m - 1),
41               maxSubArraySum(arr, m + 1, h),
42               maxCrossingSum(arr, l, m, h));
43 }
44
45 int main() {
46     int arr[] = {-2, -5, 6, -2, -3, 1, 5, -6};
47     int n = sizeof(arr) / sizeof(arr[0]);
48     int max_sum = maxSubArraySum(arr, 0, n - 1);
49     cout << "Maximum contiguous sum is " << max_sum;
50     return 0;
51 }
```

```
PS D:\DAA Assignments\Assignment 1> g++ MaxSubarraySum.cpp
PS D:\DAA Assignments\Assignment 1> ./a.exe
Maximum contiguous sum is 7
PS D:\DAA Assignments\Assignment 1> █
```