Rehnoor Aulakh

102317137

2Q16

Q1 Binary Search Program

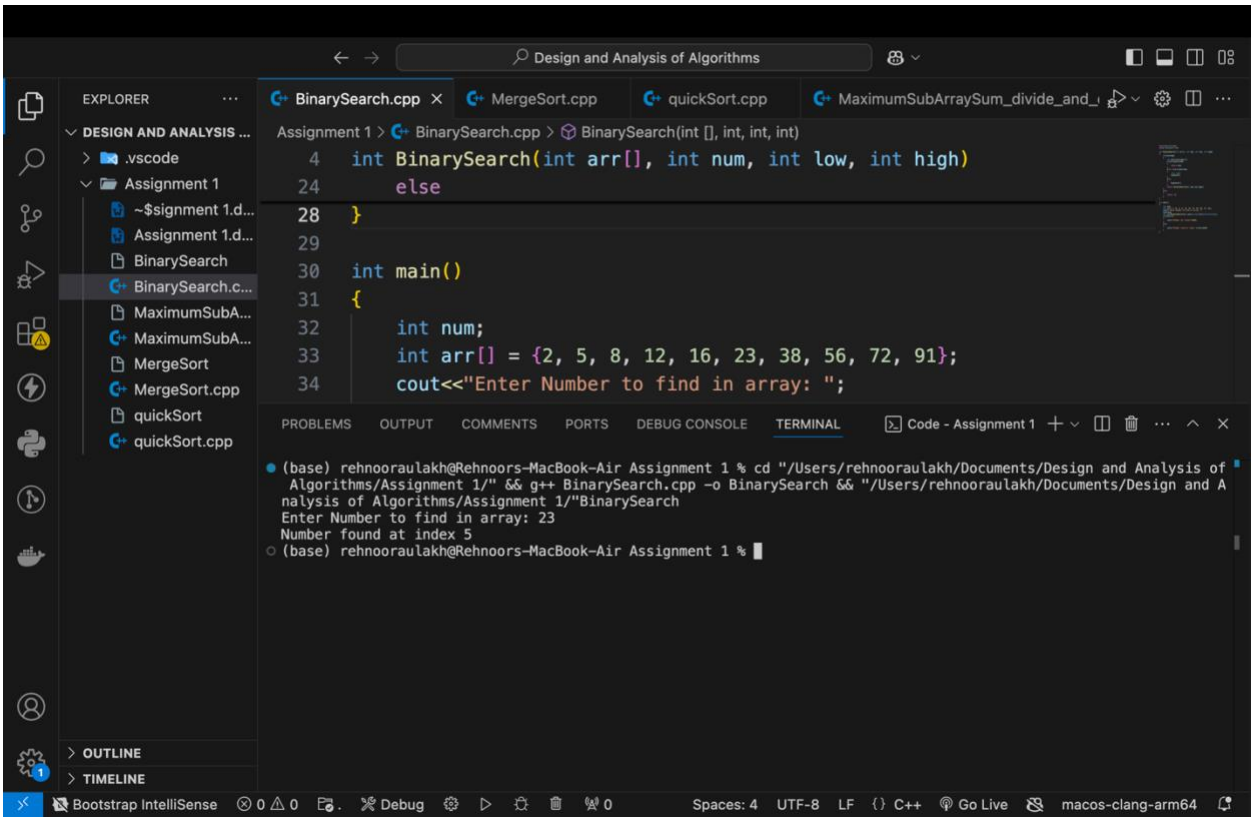
```
#include<iostream>
using namespace std;

int BinarySearch(int arr[], int num, int low, int high)
{
   if(low<high)
   {
      int mid=(low+high)/2;
      if(arr[mid]==num)
      {
         return mid;
      }
      else if(arr[mid]<num)
      {
         //go right
         low=mid+1;
      }
      else
      {
         high=mid-1;
      }
      return BinarySearch(arr,num,low,high);
```

```
    }
    else
    {
        return -1;
    }
}

int main()
{
    int num;
    int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
    cout<<"Enter Number to find in array: ";
    cin>>num;
    int res=BinarySearch(arr,num,0,sizeof(arr)/sizeof(int));
    if(res==-1)
    {
        cout<<"Number not found"<<endl;
    }
    else
    {
        cout<<"Number found at index "<<res<<endl;
    }
}
```

Output

Enter Number to find in array: 23

Number found at index 5



Q2 Merge Sort

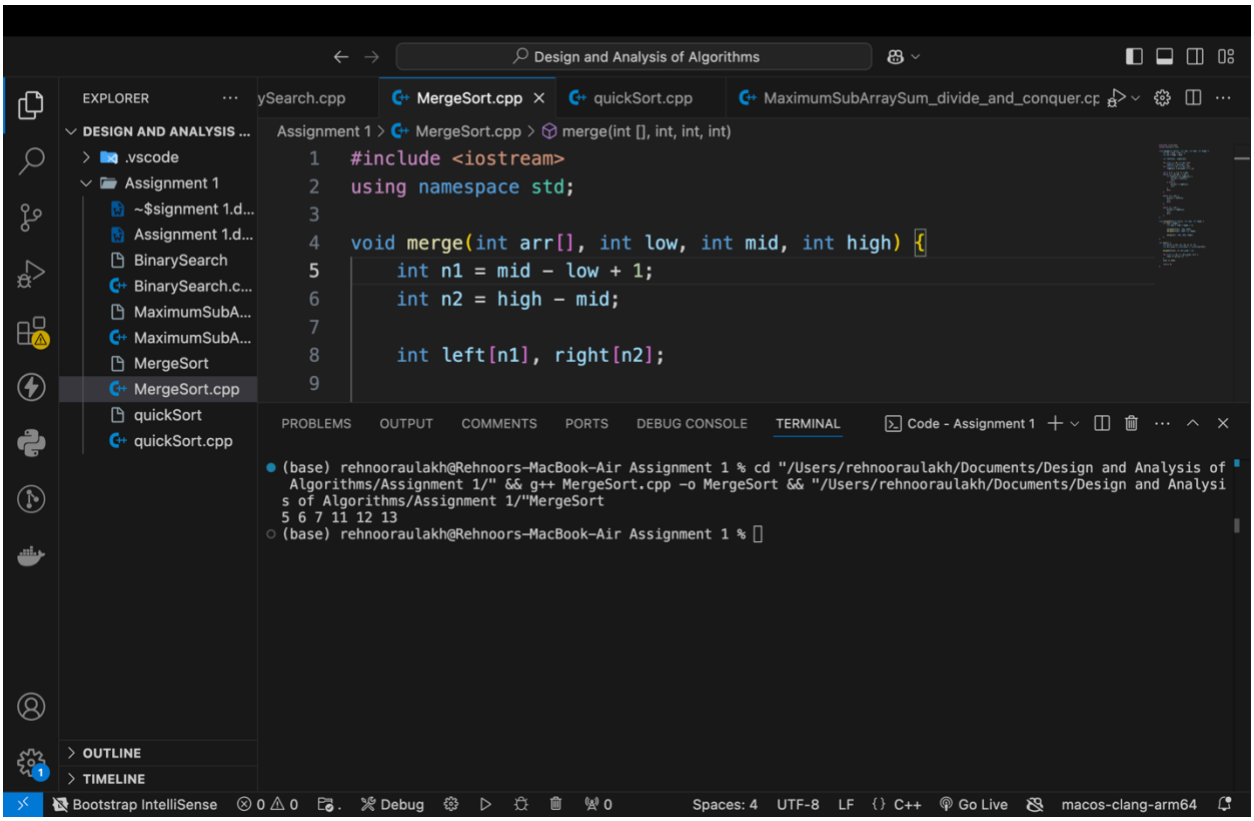
```cpp
#include <iostream>
using namespace std;

void merge(int arr[], int low, int mid, int high) {
    int n1 = mid - low + 1;
    int n2 = high - mid;

    int left[n1], right[n2];

    for (int i = 0; i < n1; i++)
```

```
    left[i] = arr[low + i];
for (int j = 0; j < n2; j++)
    right[j] = arr[mid + 1 + j];

int i = 0, j = 0, k = low;
while (i < n1 && j < n2) {
    if (left[i] <= right[j]) {
        arr[k] = left[i];
        i++;
    } else {
        arr[k] = right[j];
        j++;
    }
    k++;
}

while (i < n1) {
    arr[k] = left[i];
    i++;
    k++;
}

while (j < n2) {
    arr[k] = right[j];
    j++;
    k++;
}
```

```
}

void mergeSort(int arr[], int low, int high) {
    if (low < high) {
        int mid = (low + high) / 2;

        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);

        merge(arr, low, mid, high);
    }
}

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    mergeSort(arr, 0, arr_size - 1);

    for (int i = 0; i < arr_size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Output

5 6 7 11 12 13



Q3 Quick Sort

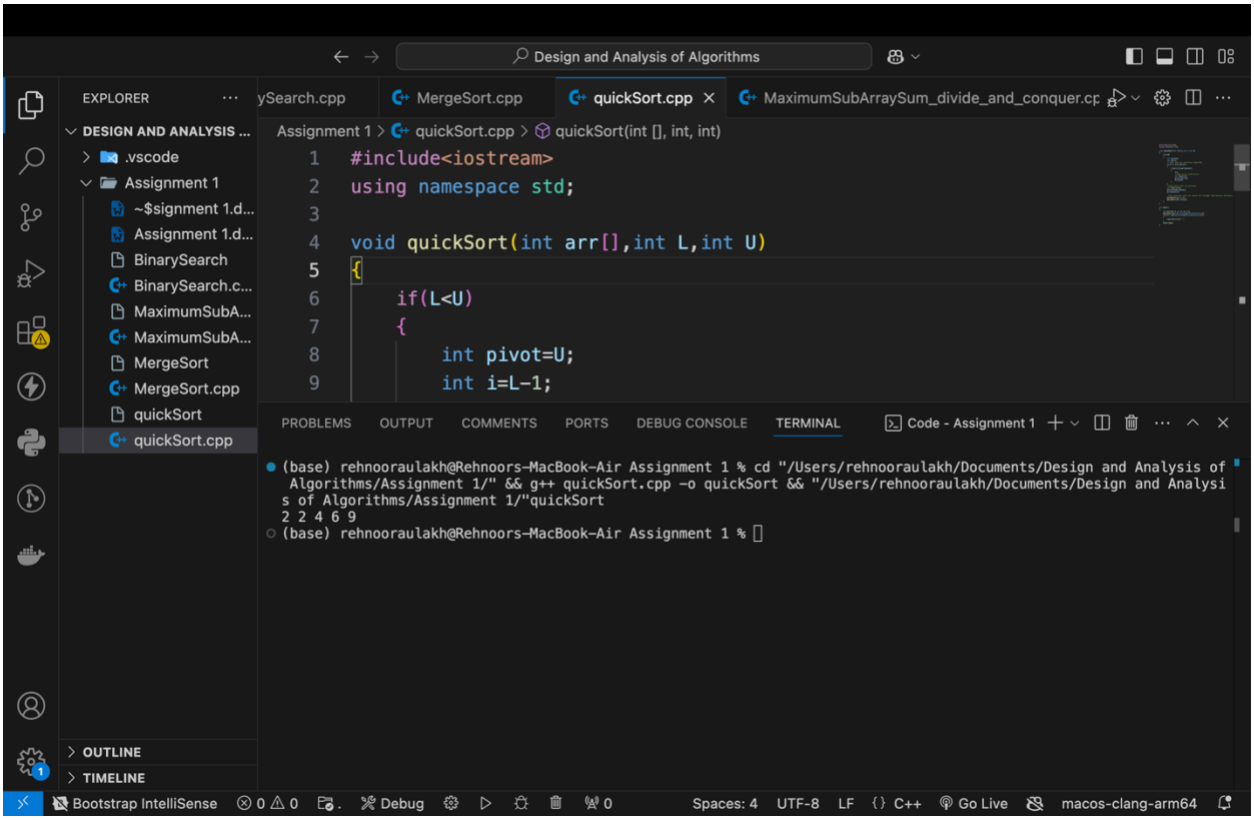
```cpp
#include<iostream>
using namespace std;


void quickSort(int arr[],int L,int U)
{
   if(L<U)
   {
      int pivot=U;
```

```
        int i=L-1;
        // O(N) time for partition algorithm
        for(int j=L;j<U;j++)
        {
            if(arr[j]<=arr[pivot])
            {
                i++;
                //swap arr[i] with arr[j]
                int t=arr[i];
                arr[i]=arr[j];
                arr[j]=t;
            }
        }
        //swap pivot with i+1 position
        int t=arr[i+1];
        arr[i+1]=arr[pivot];
        arr[pivot]=t;

        //now recursive calls for quick sort O(logN) time because dividing into half
        quickSort(arr,L,i);
        quickSort(arr,i+2,U);
    }
}

int main()
{
    int arr[]={ 4, 2, 6, 9, 2 };
```

```
    quickSort(arr,0,sizeof(arr)/sizeof(int)-1);
    for(int i=0;i<sizeof(arr)/sizeof(int);i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}
```

Output

2 2 4 6 9



Q4 Maximum Subarray Sum

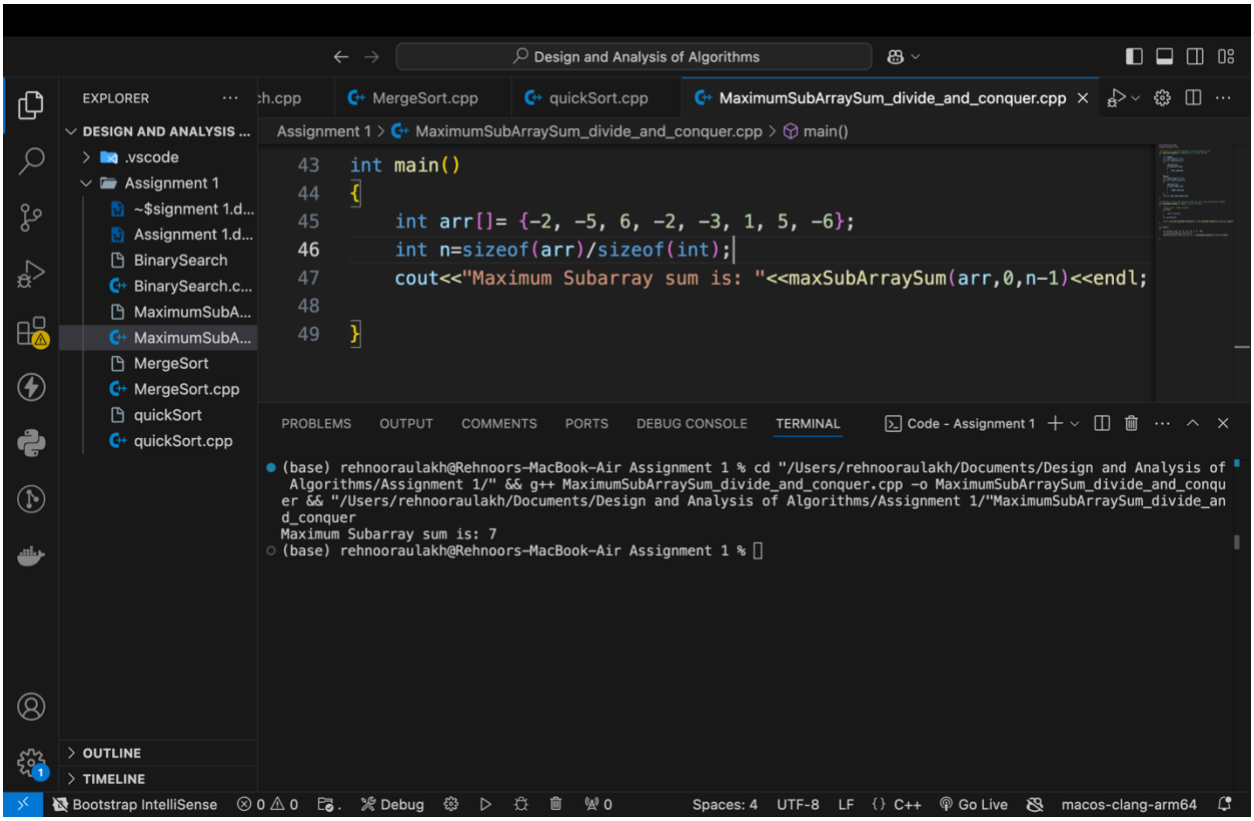
```
#include<iostream>
using namespace std;
```

```
//Function to find the maximum crossing subarray sum
int maxCrossingSum(int arr[],int l, int m, int h)
{
    int sum=0;
    int left_sum=INT_MIN;
    for(int i=m;i>=l;i--)
    {
        sum+=arr[i];
        if(sum>left_sum)
        {
            left_sum=sum;
        }
    }
    sum=0;
    int right_sum=INT_MIN;
    for(int i=m+1;i<=h;i++)
    {
        sum+=arr[i];
        if(sum>right_sum)
        {
            right_sum=sum;
        }
    }
    return left_sum+right_sum;
}

//function to find the maximum subarray sum using divide and conquer
```

```
int maxSubArraySum(int arr[], int l, int h)
{
    //base case, single element
    if(l==h)
    {
        return arr[l];
    }
    int m=(l+h)/2;

    return
max(max(maxSubArraySum(arr,l,m),maxSubArraySum(arr,m+1,h)),maxCrossingSum(a
rr,l,m,h));
}

int main()
{
    int arr[]= {-2, -5, 6, -2, -3, 1, 5, -6};
    int n=sizeof(arr)/sizeof(int);
    cout<<"Maximum Subarray sum is: "<<maxSubArraySum(arr,0,n-1)<<endl;

}
```

Output

Maximum Subarray sum is: 7


Design and Analysis of Algorithms
EXPLORER
DESIGN AND ANALYSIS ...
.vscode
Assignment 1
~$signment 1.d...
Assignment 1.d...
BinarySearch
BinarySearch.c...
MaximumSubA...
MaximumSubA...
MergeSort
MergeSort.cpp
quickSort
quickSort.cpp
MergeSort.cpp
quickSort.cpp
MaximumSubArraySum_divide_and_conquer.cpp
Assignment 1 > MaximumSubArraySum_divide_and_conquer.cpp > main()
43 int main()
44 {
45 int arr[]= {-2, -5, 6, -2, -3, 1, 5, -6};
46 int n=sizeof(arr)/sizeof(int);
47 cout<<"Maximum Subarray sum is: "<<maxSubArraySum(arr,0,n-1)<<endl;
48
49 }
PROBLEMS
OUTPUT
COMMENTS
PORTS
DEBUG CONSOLE
TERMINAL
Code - Assignment 1
(base) rehnooraulakh@Rehnoors-MacBook-Air Assignment 1 % cd "/Users/rehnooraulakh/Documents/Design and Analysis of Algorithms/Assignment 1/" && g++ MaximumSubArraySum_divide_and_conquer.cpp -o MaximumSubArraySum_divide_and_conquer && "/Users/rehnooraulakh/Documents/Design and Analysis of Algorithms/Assignment 1/"MaximumSubArraySum_divide_and_conquer
Maximum Subarray sum is: 7
(base) rehnooraulakh@Rehnoors-MacBook-Air Assignment 1 %
OUTLINE
TIMELINE
Bootstrap IntelliSense
Debug
Spaces: 4
UTF-8
LF
C++
Go Live
macos-clang-arm64