UCS415 – Design and Analysis of Algorithms

Lab Assignment 1
Name: Ojaswi Kumar
Roll no:102296001
Group: 2CS11

Q1 Write a program to implement iterative as well as recursive versions of the following algorithms:

1) Binary search (https://www.geeksforgeeks.org/binary-search/),
   i. recursive

CODE:

```
//recursive approach
#include<iostream>
using namespace std;
int binarySearch(int *arr,int k,int low,int high){
    int mid;
    if(low>high){
        return false;
    }
    else{
        mid=(low+high)/2;
        if(k==arr[mid]){
            return mid;
        }
        else if(k>arr[mid]){
            return binarySearch(arr,k,mid+1,high);
        }
        else{
            return binarySearch(arr,k,low,mid-1);
        }
    }

}
int main(){
    int n;
    cout<<"enter the size of the array"<<endl;
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    int k;
    cout<<"enter the element to be searched"<<endl;
    cin>>k;
    int low=0,high=n-1;
```

```
    if(!binarySearch(arr,k,low,high)){
        cout<<"not found"<<endl;
    }
    else{
        cout<<"found"<<endl;
    }

}
```

ii. iterative

```
        #include <bits/stdc++.h>
#include <iostream>
using namespace std;

int binarySearch(vector<int> v, int To_Find)
{
    int lo = 0, hi = v.size() - 1;
    int mid;
    // This below check covers all cases , so need to check
    // for mid=lo-(hi-lo)/2
    while (hi - lo > 1) {
        int mid = (hi + lo) / 2;
        if (v[mid] < To_Find) {
            lo = mid + 1;
        }
        else {
            hi = mid;
        }
    }
    if (v[lo] == To_Find) {
        cout << "Found"
             << " At Index " << lo << endl;
    }
    else if (v[hi] == To_Find) {
        cout << "Found"
             << " At Index " << hi << endl;
    }
    else {
        cout << "Not Found" << endl;
    }
}

int main()
{
    vector<int> v = { 1, 3, 4, 5, 6 };
    int To_Find = 1;
    binarySearch(v, To_Find);
    To_Find = 6;
    binarySearch(v, To_Find);
    To_Find = 10;
    binarySearch(v, To_Find);
```

```
    return 0;
}
```

2)Merge sort (https://www.geeksforgeeks.org/iterative-merge-sort/), and

i. iterative

```
#include<iostream>
using namespace std;
void merge(int *arr,int l,int m,int h);
int min(int x,int y){
    return (x<y)?x:y;
}
void mergeSort(int *arr,int n){
    int curr_size;
    int low;
    for(curr_size=1;curr_size<=n-1;curr_size=2*curr_size){
        for(low=0;low<n-1;low+=2*curr_size){
            int mid=min(low+curr_size-1,n-1);
            int high=min(low+2*curr_size-1,n-1);
            merge(arr,low,mid,high);
        }
    }
}
void merge(int *arr,int l,int m,int h){
    int i,j,k;
    int n1=m-l+1;
    int n2=h-m;
    int L[n1],R[n2];
    for(i=0;i<n1;i++){
        L[i]=arr[l+i];
    }
    for(j=0;j<n2;j++){
        R[j]=arr[m+1+j];
    }
    i=0;
    j=0;
    k=1;
    while(i<n1 && j<n2){
        if(L[i]<=R[j]){
            arr[k]=L[i];
            i++;
        }
        else{
            arr[k]=R[j];
            j++;
        }
        k++;
    }
    while(i<n1){
        arr[k]=L[i];
        i++;
```

```
        k++;
    }
    while(j<n2){
        arr[k]=R[j];
        j++;
        k++;
    }
}
void printArray(int *arr,int size){
    for(int i=0;i<size;i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}
int main(){
    int size;
    cout<<"enter the size of the array"<<endl;
    cin>>size;
    int arr[size];
    for(int i=0;i<size;i++){
        cin>>arr[i];
    }
    cout<<"given array is"<<endl;
    printArray(arr,size);
    mergeSort(arr,size);
    cout<<"sorted array is"<<endl;
    printArray(arr,size);
    return 0;
}
```

ii. recursive

```
#include<iostream>
using namespace std;
void printArray(int *A,int n){
    for(int i=0;i<n;i++){
        cout<<A[i];
    }
    cout<<endl;
}
void merge(int *A,int mid,int low,int high){
    int i,j,k,B[100];
    i=low;
    j=mid+1;
    k=low;

    while(i<=mid && j<=high){
        if(A[i]<A[j]){
            B[k]=A[i];
            i++;
```

```
            k++;
        }
        else{
            B[k]=A[j];
            j++;
            k++;
        }
    }
    while(i<=mid){
        B[k]=A[i];
        k++;
        i++;
    }
    while(j<=high){
        B[k]=A[j];
        k++;
        j++;
    }
    for(int i=low;i<=high;i++){
        A[i]=B[i];
    }
}
void mergeSort(int A[],int low,int high){
    int mid;
    if(low<high){
        mid=(low+high)/2;
        mergeSort(A,low,mid);
        mergeSort(A,mid+1,high);
        merge(A,mid,low,high);
    }
}
int main(){
    int n;
    cout<<"enter the size of array"<<endl;
    cin>>n;
    int A[n];
    for(int i=0;i<n;i++){
        cin>>A[i];
    }
    printArray(A,n);
    mergeSort(A,0,6);
    printArray(A,n);
    return 0;
}
```

3) Quick sort (https://www.geeksforgeeks.org/iterative-quick-sort/)
   i. Iterative

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

```
void swap(int *a,int *b){
    int temp=*a;
    *a=*b;
    *b=temp;
}
int partition(int *arr,int low,int high){
    int x=arr[high];
    int i=(low-1);
    for(int j=2;j<=high-1;j++){
        if(arr[j]<=x){i++;
        swap(&arr[i],&arr[j]);
        }
    }
    swap(&arr[i+1],&arr[high]);
    return(i+1);
}
void quickSort(int *arr,int low,int high){
    int stack[high-low+1];
    int top=-1;
    stack[++top]=low;
    stack[++top]=high;
    while(top>=0){
        high=stack[top--];
        low=stack[top--];
        int p=partition(arr,low,high);
        if(p-1>1){
            stack[++top]=1;
            stack[++top]=p-1;
        }
        if(p+1<high){
            stack[++top]=p+1;
            stack[++top]=high;
        }
    }
}
void printArray(int *arr,int n){
    for(int i=0;i<n;i++){
        cout<<arr[i]<<" ";
    }
}
int main(){
    int n;
    cout<<"enter the size of the array"<<endl;
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    quickSort(arr,0,n-1);
    printArray(arr,n);
    return 0;
```

```
}
```

ii. recursive

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

void printArray(int *A, int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d ", A[i]);
    }
    printf("\n");
}

int partition(int A[], int low, int high)
{
    int pivot = A[low];
    int i = low + 1;
    int j = high;
    int temp;

    do
    {
        while (A[i] <= pivot)
        {
            i++;
        }

        while (A[j] > pivot)
        {
            j--;
        }

        if (i < j)
        {
            temp = A[i];
            A[i] = A[j];
            A[j] = temp;
        }
    } while (i < j);

    // Swap A[low] and A[j]
    temp = A[low];
    A[low] = A[j];
    A[j] = temp;
    return j;
}

void quickSort(int A[], int low, int high)
```

```
{
    int partitionIndex; // Index of pivot after partition

    if (low < high)
    {
        partitionIndex = partition(A, low, high);
        quickSort(A, low, partitionIndex - 1);  // sort left subarray
        quickSort(A, partitionIndex + 1, high); // sort right subarray
    }
}

int main()
{
    int n;
    cout<<"enter the size of the array"<<endl;
    cin>>n;
    int A[n];
    for(int i=0;i<n;i++){
        cin>>A[i];
    }

    printArray(A, n);
    quickSort(A, 0, n - 1);
    printArray(A, n);
    return 0;
}
```

Additional Questions:

1) https://www.hackerearth.com/practice/algorithms/searching/binary-search/tutorial/

```
#include<iostream>
using namespace std;
int binarySearch(int *arr,int low,int high,int key){
    while(low<=high){
        int mid=(high+low)/2;
        if(arr[mid]<key){
            low=mid+1;
        }
        else if(arr[mid]>key){
            high=mid-1;
        }
        else{
            return mid+1;
        }
    }
    return -1;
}
int main(){
    int n;
    cout<<"enter the size of the array: ";
```

```
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    int low=0;
    int high=n-1;
    int q;
    cout<<"enter the number of queries: ";
    cin>>q;
    while(q--){
        int x;
        cin>>x;
        cout<<binarySearch(arr,0,n-1,x)<<endl;

    }
}
```

2) https://www.hackerearth.com/practice/algorithms/sorting/merge-sort/tutorial/

```
#include <bits/stdc++.h>
using namespace std;
void merge(int arr[], int low, int mid, int high)
{
    int i = low, j = mid + 1, k = 0;
    int tempArr[high - low + 1];
    while (i <= mid && j <= high)
    {
        if (arr[i] >= arr[j])
        {
            tempArr[k] = arr[j];
            j++;
            k++;
        }
        else
        {
            tempArr[k] = arr[i];
            i++;
            k++;
        }
    }
    while (i <= mid)
    {
        tempArr[k] = arr[i];
        k++;
        i++;
    }
    while (j <= high)
    {
```

```
        tempArr[k] = arr[j];
        k++;
        j++;
    }
    k = 0;
    for (i = low; i < high + 1; i++)
    {
        arr[i] = tempArr[k];
        k++;
    }
}
void merge_sort(int a[], int low, int high)
{
    if (low < high)
    {
        int mid = low + (high - low) / 2;
        merge_sort(a, low, mid);
        merge_sort(a, mid + 1, high);
        merge(a, low, mid, high);
    }
}
int main()
{
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    // merge_sort(arr, 0, n - 1);
    int count = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if ((i < j) && (arr[i] > arr[j]))
            {
                count++;
            }
        }
    }
    cout<<count<<endl;
    return 0;
}
```

3) https://www.codechef.com/problems/MYSITM

```
#include<bits/stdc++.h>
```

```
using namespace std;
 #define type unsigned long long int

int main() {

    // your code goes here
    int t;
    cin>>t;
    while(t--){
       type n,h,w;
        cin>>n>>h>>w;
       type min_s=1,max_s=max(n*h,n*w);
        while(min_s<max_s){
          type mid=(min_s+max_s)/2;
         type ans=(mid/w)*(mid/h);
           if(ans<n){
               min_s=mid+1;
           }
           else{
               max_s=mid;
           }
       }
        cout<<max_s<<endl;

    }
    return 0;
}
```

4) https://www.hackerearth.com/practice/algorithms/sorting/merge-sort/practice-problems/algorithm/median-game-june-easy-19-3722be60/

```
for i in range(int(input())):
    n=int(input())
    l=list(map(int,input().split()))
    ans=min(l)+max(l)
    print(ans)
```

5) https://www.hackerearth.com/practice/algorithms/sorting/quick-sort/practice-problems/algorithm/lex-finds-beauty-0d0bc1b6/

```
n,k=map(int,input().split())
A=list(map(int,input().split()))
#k count of numbers greater than a[l]
sum=0
A.sort()
for i in range(n-k):
    sum=sum+A[i]
print(sum)
```