Name: Khushi Nitinkumar Patel
PRN: 2020BTECS00037
Batch: T5

Week 1 Assignment

Part: 1

Sorting Algorithm

Q) You are given two sorted array, A and B, where A has a large enough buffer at the end to hold B. Write a method to merge B into A in sorted order.

> Algorithm:

```
mergeArrays( arr1[], arr2[], n1, n2,arr3[])

{

i = 0, j = 0, k = 0;

while(i < n1 or j < n2)

{

If arr1[i] <= arr2[j]

arr3[k] = arr1[i];

i++;

k++;

else

arr3[k] = arr2[j];

j++;
```

k++;

}

while(i < n1)//Add remaining elements of arr1 if any

{

arr3[k] = arr1[i];

i++;

k++;

}

while(j < n2) //Add remaining elements of arr1 if any

{

arr3[k] = arr2[j];

j++;

k++;

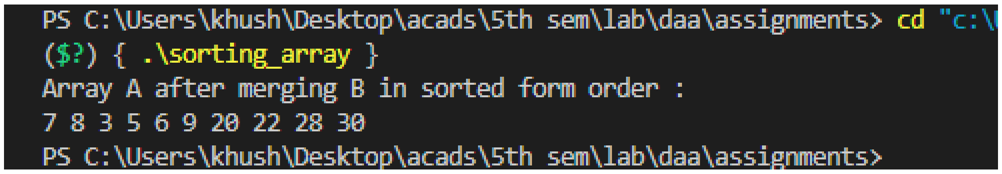
}

Code:

```
#include <iostream>

#include <bits/stdc++.h>

#define NA -1



using namespace std;



void sortedArray(int a[], int b[], int n, int m){
    int i = n - 1;
    int j = m - 1;

    int lastIndex = n + m - 1;

    while(j>=0){

        if(i>=0 && a[i] > b[j]){

            a[lastIndex] = a[i];
            i--;

        }

        else {
            a[lastIndex] = b[j];
            j--;
        }

        lastIndex--;
    }
}
```

```
void printArray(int *arr, int n){

    cout<< "Array A after merging B in sorted form"

    " order : " <<endl;



    for(int i=0; i<n; i++)

    cout<<arr[i]<<" ";

}




int main(){

    int a[]={6,20,22,28,30,NA,NA,NA,NA};

    int n=5;

    int size_a=10;

    int b[] = {7,8,3,5,9};

    int m = 5;

    sortedArray(a,b,n,m);

    printArray(a, size_a);

    return 0;

}
```

OUTPUT:



```
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assignments> cd "c:\
($?) { .\sorting_array }
Array A after merging B in sorted form order :
7 8 3 5 6 9 20 22 28 30
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assignments>
```

Time Complexity: O(n1+n2)

Space Complexity: O(n1+n2)

Q) Write a method to sort an array of string so that all the anagrams are next to each other.

>Algorithm:

1. Store all strings and its index in the vector of the pair of string and int.

2. Sort the strings in all vectors.

3. Now sort the entire vector such that all anagrams will come next to each other.

4. Now store the answer in another array with the help of an index from vector pair and original array.

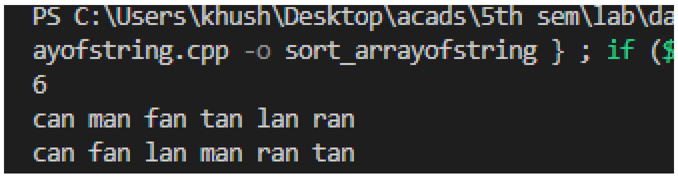
5. Print answer array.

Code:

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

using namespace std;

int main()

{

  int n;
```

```
    cin >> n;

    string arr[n];

    for (int i = 0; i < n; i++)

        cin >> arr[i];

    vector<pair<string, int>> v;

    for (int i = 0; i < n; i++)

        v.push_back({arr[i], i});

    for (int i = 0; i < v.size(); i++)

        sort(v[i].first.begin(), v[i].first.end());

    sort(v.begin(), v.end());

    string ans[n];

    for (int i = 0; i < n; i++)

        ans[i] = arr[v[i].second];

    for (int i = 0; i < n; i++)

        cout << ans[i] << " ";

    cout << endl;

}
```

OUTPUT



```
PS C:\Users\khush\Desktop\acads\5th sem\lab\da
ayofstring.cpp -o sort_arrayofstring } ; if ($
6
can man fan tan lan ran
can fan lan man ran tan
```

Time complexity analysis:

Let there be n-words and each word has a maximum of m characters.

O(nmLogm + mnLogn).

Q) Given a sorted array of n integers that has been rotated an unknown number of times, write code to find an element in the array. You may assume that the array was originally sorted in increasing order.

> **Algorithm:**

EXAMPLE

Input: find 5 in {15, 16, 19, 20, 25, 1, 3, 4, 5, 7, 10, 14}

Output: 8 (the index of 5 in the array)

Algorithm:

The idea is to find the pivot point, divide the array into two sub-arrays and perform a binary search.

The main idea for finding a pivot is –

For a sorted (in increasing order) and rotated array, the pivot element is the only element for which the next element to it is smaller than it.

Using binary search based on the above idea, pivot can be found.

It can be observed that for a search space of indices in the range [l, r] where the middle index is mid,

If rotation has happened in the left half, the element at l will obviously be greater than the one at mid.

Otherwise, the left half will be sorted but the element at mid will be greater than the one at r.

After the pivot is found divide the array into two sub-arrays.

Now the individual sub-arrays are sorted so the element can be searched using Binary Search.

Binary Search

The basic steps to perform Binary Search are:

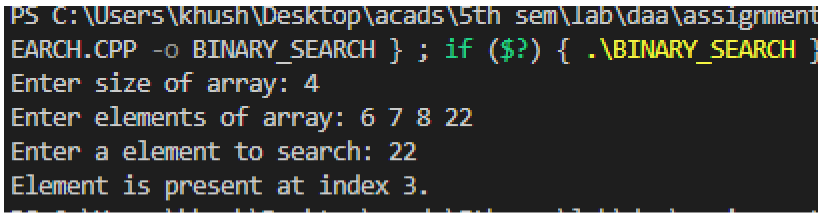
1. Begin with the mid element of the whole array as a search key.
2. If the value of the search key is equal to the item then return an index of the search key.
3. Or if the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
4. Otherwise, narrow it to the upper half.
5. Repeatedly check from the second point until the value is found or the interval is empty.

Code:

```
#include <iostream>

#include <vector>

using namespace std;

int binarySearch(int key, int a[], int l, int h)

{

    int mid = (l + h) / 2;

    if (h < l)

        return -1;

    if (a[mid] == key)

        return mid;

    else if (a[mid] < key)

        return binarySearch(key, a, mid + 1, h);

    else

        return binarySearch(key, a, l, mid - 1);

}

int search(int a[], int n, int target)

{

    int x = 0, ans = 0;

    bool temp = false;

    for (int i = 0; i < n - 1; i++)

    {

        if (a[i] > a[i + 1])
        {
            x = i + 1;
            temp = true;
            break;
        }
    }
    if (temp == true)
    {
        if (target <= a[n - 1])
            ans = binarySearch(target, a, x, n - 1);
        else
            ans = binarySearch(target, a, 0, x - 1);
    }
    else
        ans = binarySearch(target, a, 0, n - 1);
    return ans;
}
int main()
{
    int n, element;
    cout << "Enter size of array: ";
    cin >> n;
    int a[n];
    cout << "Enter elements of array: ";
```

```
  for (int i = 0; i < n; i++)
      cin >> a[i];
  cout << "Enter a element to search: ";
  cin >> element;
  int ans;
  ans = search(a, n, element);
  if (ans == -1)
      cout << "Element is not present in array.";
  else
      cout << "Element is present at index " << ans << ".\n";
  return 0;
}
```

OUTPUT



```
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assignment
EARCH.CPP -o BINARY_SEARCH } ; if ($?) { .\BINARY_SEARCH }
Enter size of array: 4
Enter elements of array: 6 7 8 22
Enter a element to search: 22
Element is present at index 3.
```

Time Complexity: O(log N) Binary Search requires log n comparisons to find the element.

Space Complexity: O(1).

Q) Imagine you have a 20GB file with one string per line. Explain how you would sort the file.

> This is accomplished by external sort. Bring only part of the data into memory at x megabytes each. Sort each chunk separately and saved back to the file system. Once all chunks are sorted, merge the chunks, one by one.

Q) Given a sorted array of string which is interspersed with empty string,

write a method to find the location of a given string.

EXAMPLE

Input: find "ball" in {"at", "", "", "ball", "", "", "car", "", "", "dad", "",""}

Output: 3

Algorithm:

1. Like normal binary search, we compare the given str with the middle string.

2. If middle string is empty, we find the closest non-empty string x (by

linearly searching on both sides).

3. Once we find x, we do standard binary search, i.e., we compare given str

with x.

4. If str is the same as x, we return an index of x.

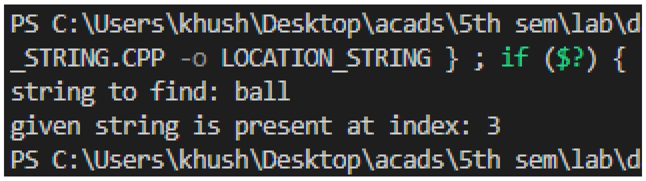
5. If str is greater, we recur for the right half, else we recur for the left half.

Code:

```cpp
#include <iostream>

using namespace std;

int compareStr(string s1, string s2)

{

    int i = 0;

    while (s1[i] == s2[i] && s1[i] != '\0')

    {

        i++;

    }

    if (s1[i] > s2[i])

        return -1;

    return (s1[i] < s2[i]);

}

int searchStr(string s[], string str, int first, int last)

{

    if (first > last)

        return -1;

    int mid = (first + last) / 2;

    if (s[mid].empty())

    {

        int left = mid - 1;

        int right = mid + 1;
        while (true)

        {

            if (left < right && right > last)

                return -1;

            if (right <= last && !s[right].empty())

            {

                mid = right;

                break;

            }

            if (left >= first && !s[left].empty())

            {

                mid = left;

                break;

            }

            right++;

            left--;

        }

    }

    if (compareStr(str, s[mid]) == 0)

        return mid;

    else if (compareStr(str, s[mid]) < 0)

        return searchStr(s, str, mid + 1, last);

    return searchStr(s, str, first, mid - 1);

}
```

```
int main()

{

    string s[] = {"at", "", "", "ball", "", "", "car", "",

                  "", "dad", "", ""};

    string str = "ball";

    int n = sizeof(s) / sizeof(s[0]);

    cout << "string to find: " << str << endl;

    if (searchStr(s, str, 0, n - 1) == -1)

        cout << "given string is not present.";

    else

        cout << "given string is present at index: " << searchStr(s, str, 0, n - 1);

    return 0;

}
```

OUTPUT



```
PS C:\Users\khush\Desktop\acads\5th sem\lab\d
_STRING.CPP -o LOCATION_STRING } ; if ($?) {
string to find: ball
given string is present at index: 3
PS C:\Users\khush\Desktop\acads\5th sem\lab\d
```

Time complexity O(L(logN)).

Q) Given an M*N matrix in which each row and each column is sorted in ascending order, write a method to find an element.

> Algorithm:

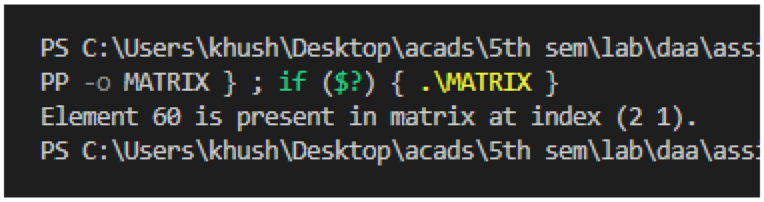
```
   i=0,j=M-1.Tofind x.
   In while loop such that i<N and j>=0
   Compare x with arr[i][j]
   If x > arr[i][j] then it cannot be in this row. i++
   else if x <arr[i][j] it is in same row .j—
   else x==arr[i][j], we found the element.
   Return index.
```

Code:

```cpp
#include <iostream>

using namespace std;

int main()

{

   int v[3][3] = {{3, 30, 38},

                  {44, 52, 54},

                  {57, 60, 69}};

   int N = 3, M = 3, i = 0, j = M - 1, X = 60, ans = 0;

   while (i < N && j >= 0)

   {

      if (X == v[i][j])
```

```
        {
            ans = 1;
            cout << "Element " << v[i][j] << " is present in matrix at index (" << i << " " << j << ").";
            break;
        }
        if (X > v[i][j])
            i++;
        else
            j--;
    }
    if (ans != 1)
        cout << "Element is not present in matrix.";
    return 0;
}
```

OUTPUT



```
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assi
PP -o MATRIX } ; if ($?) { .\MATRIX }
Element 60 is present in matrix at index (2 1).
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assi
```

Time Complexity:O(N)if N>M and O(M) if M>N.

Space Complexity:O(1).

Q) A circus is designing a tower routine consisting of people standing atop

one another’s shoulders. For practical and aesthetic reasons, each person must be both shorter and lighter than the person below him or her. Given the height sand weight of each circus, write a method to compute the largest possible number of people in such tower.

EXAMPLE:

Input(ht,wt): (65, 100) (70, 150) (56, 90) (75,190) (60, 95) (68, 110).

Output: The longest tower is length 6 and includes from top to bottom:

(56, 90) (60, 95) (65, 100) (68, 110) (70, 150) (75, 190)

> Algorithm:

Sort vector of vectors v by heights. O(P*logP).

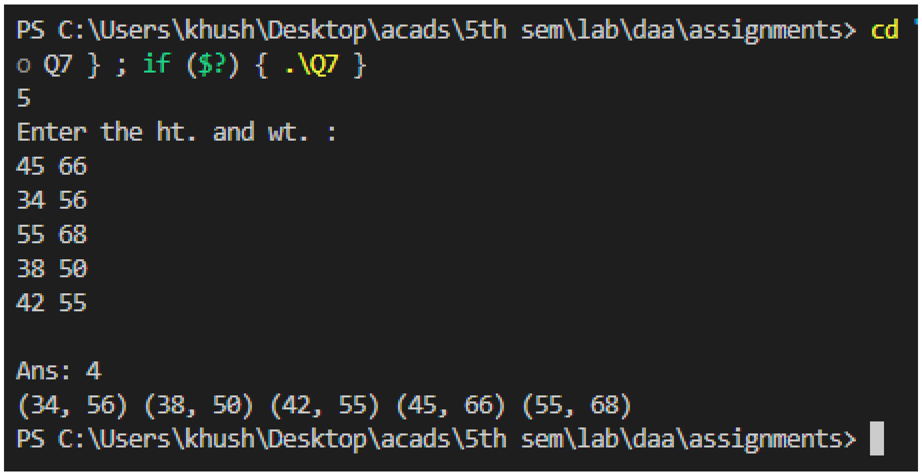
```
start = 0

end = 0

max_people = 0

for i in range(1, len(v)): # O(P).

if v[i].second > v[i - 1].second Weight check.

end = i

else:

max_people = end - start + 1

start = i

end = i

ans=max(max_people, end - start + 1)
```

Code:

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

using namespace std;

bool sorted(pair<int, int> &a, pair<int, int> &b)

{

    if (a.first == b.first)

        return a.second < b.second;

    return a.first < b.first;

}



int main()

{

    int n;

    cin >> n;

    int x, y;

    vector<pair<int, int>> v;

    cout<<"Enter the ht. and wt. : "<<endl;



    for (int i = 0; i < n; i++)

    {

        cin >> x;

        cin >> y;

        v.push_back(make_pair(x, y));
```

```
  }
  sort(v.begin(), v.end(), sorted);
  int start = 0, end = 0, maxPeople = 0, ans = INT_MIN;
  for (int i = 1; i < v.size(); i++)
  {
    if (v[i].second > v[i - 1].second)
      end = i;
    else
    {
      maxPeople = end - start + 1;
      start = i;
      end = i;
    }
  }
  ans = max(maxPeople, end - start + 1);
  cout << "\nAns: " << ans << endl;
  for (int i = 0; i < v.size(); i++)
    cout << "(" << v[i].first << ", " << v[i].second
         << ") ";
  return 0;
}
```

OUTPUT



```
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assignments> cd
o Q7 } ; if ($?) { .\Q7 }
5
Enter the ht. and wt. :
45 66
34 56
55 68
38 50
42 55

Ans: 4
(34, 56) (38, 50) (42, 55) (45, 66) (55, 68)
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assignments>
```

Time Complexity O(nlogn)//n=number of people.

Q) Imagine you are reading in stream of integers. Periodically, you wish to be able to look up the rank of number x (the number of values less than or equal to x). Implement the data structures and algorithms to support these operations. That is, Implement the method track (int x), which is called when each number is generated, and the method getRankOfNumber (int x), which return the number of values less than or equal to x (not including x itself).

EXAMPLE

Stream (in order of appearance) : 5, 1, 4, 4, 5, 9, 7, 13, 3

getRankOfNumber(1) = 0

getRankOfNumber(3) = 1

getRankOfNumber(4) =3

Algorithm:

Traverse the array linearly to find rank of x

If arr[i]<=x increment count

If arr[i]==x decrese one from count before returning the answer

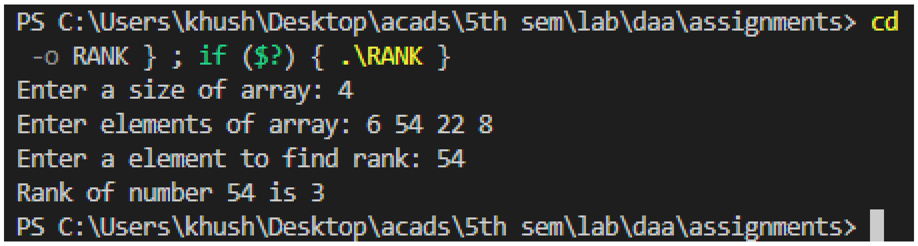
Since we are traversing linearly,one of the element is x itself if it is repeating

Code:

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

using namespace std;

int getRankOfNumber(int arr[], int n, int x)

{

    int count = 0;

    bool temp = false;

    for (int i = 0; i < n; i++)

    {

        if (arr[i] <= x)

            count++;

        if (arr[i] == x)

            temp = true;

    }

    if (temp == true)

        count--;

    return count;

}

int main()

{

    int n;

    cout << "Enter a size of array: ";

    cin >> n;
```

```cpp
  int arr[n];

  cout << "Enter elements of array: ";

  for (int i = 0; i < n; i++)

    cin >> arr[i];

  int x;

  cout << "Enter a element to find rank: ";

  cin >> x;

  cout << "Rank of number " << x << " is " << getRankOfNumber(arr, n, x);

  return 0;

}
```

OUTPUT



```
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assignments> cd
 -o RANK } ; if ($?) { .\RANK }
Enter a size of array: 4
Enter elements of array: 6 54 22 8
Enter a element to find rank: 54
Rank of number 54 is 3
PS C:\Users\khush\Desktop\acads\5th sem\lab\daa\assignments>
```

Time Complexity:O(n)

Space Complexity:O(1)