Walchand College of Engineering, Sangli

Computer Science & Engineering

Third Year

Course: Design and Analysis of Algorithm Lab

Lab course coordinator: Ms N L Mudegol

Batch: T6

**Week 1 Assignment**

Part: 1

**Sorting Algorithm**

Q) You are given two sorted arrays, 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.

Ans:

Algorithm/Pseudocode:

**Initialization:**

* Create two arrays, arr1 and arr2.
* Initialize the lengths of the non-zero parts of arr1 (m) and arr2 (n).
* Set up indices to keep track of positions in arr1, arr2, and the merged array.

**Merge Arrays:**

* Compare elements from the end of arr1's non-zero part and arr2.
* Place the larger element at the end of arr1 and move the respective index.
* Repeat until all elements of one of the arrays are placed.

**Copy Remaining Elements:**

* If arr2 has remaining elements, copy them into arr1.

**Output the Merged Array:**

* Print the elements of the merged arr1.

#include <iostream>

#include <vector>

using namespace std;

void merge(vector<int>& v1, int m, vector<int>& v2, int n) {

int i = m - 1;

int j = n - 1;

int k = m + n - 1;

while (i >= 0 && j >= 0) {

if (v1[i] > v2[j]) {

v1[k--] = v1[i--];

} else {

v1[k--] = v2[j--];

}

}

while (j >= 0) {

v1[k--] = v2[j--];

}

}

int mv1in() {

vector<int> v1 = {1, 2, 3, 4, 5, 0, 0, 0, 0, 0};

vector<int> v2 = {2, 4, 6, 8, 10};

int m = 5;

int n = 5;

merge(v1, m, v2, n);

for (int num : v1){

cout << num << " ";

}

return 0;

}

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

Concept of anagram : words that have specific set of chars but in different order

Approach : Sort all the string but also store their previous indexes and compare each other if they are equal which means they are anagrams .

At the end combine all those indices in the answer vector

#include <bits/stdc++.h>

using namespace std;

int main(){

int n;

cin>>n;

vector<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(auto &[str,idx]:v){

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

}

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

vector<string>ans;

for(auto &[str,idx]:v){

ans.push\_back(arr[idx]);

}

for(auto &str:ans){

cout<<str<<" ";

}

return 0;

}

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.

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)

Ans:

Approach : Use a linear search in the array and find the index

#include <bits/stdc++.h>

using namespace std;

int main(){

int n;

cin>>n;

vector<int>v(n);

for(auto &x:v){

cin>>x;

}

int find\_element;

cin>>find\_element;

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

if(v[i]==find\_element){

cout<<"Element found at index: ";

cout<<i;

return 0;

}

}

return 0;

}

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

To sort a 20GB file that can't fit into memory, use External Merge Sort:

1. Split and Sort Chunks:

- Read the file in manageable chunks (e.g., 500MB).

- Sort each chunk in memory.

- Write each sorted chunk to a temporary file.

2. Merge Sorted Chunks:

- Use a min-heap to merge the sorted chunks.

- Open file pointers for all temporary files.

- Populate the heap with the first element from each file.

- Continuously extract the smallest element from the heap, write it to the output file, and replace it with the next element from the respective file.

- Repeat until all elements are processed.

This method efficiently sorts large files by leveraging external storage and minimizing memory usage.

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: 4

Approach : use linear search and find out the index.

#include <bits/stdc++.h>

using namespace std;

int main(){

int n;

cin>>n;

vector<string>v(n);

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

cin>>v[i];

}

string s;

cin>>s;

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

if(v[i]==s){

cout<<"String found at index: ";

cout<<i;

return 0;

}

}

return 0;

}

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.

Approach: use brute force approach to find the element

#include <iostream>

#include <vector>

using namespace std;

pair<int, int> findElement(const vector<vector<int>>& matrix, int target) {

if (matrix.empty() || matrix[0].empty()) {

return {-1, -1};

}

int rows = matrix.size();

int cols = matrix[0].size();

int row = 0;

int col = cols - 1;

while (row < rows && col >= 0) {

if (matrix[row][col] == target) {

return {row, col};

} else if (matrix[row][col] > target) {

col--;

} else {

row++;

}

}

return {-1, -1};

}

int main() {

int n,m;

cin>>n>>m;

vector<vector<int>> matrix;

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

vector<int>temp(m);

for(int j=0;j<m;j++){

cin>>temp[j];

}

matrix.push\_back(temp);

}

int target;

cin>>target;

pair<int, int> result = findElement(matrix, target);

if (result.first != -1) {

cout << "Element found at position: (" << result.first << ", " << result.second << ")" << endl;

} else {

cout << "Element not found" << endl;

}

return 0;

}

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 heights and weight of each circus, write a method to compute the largest possible number of people in such a 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)

Approach : Sort but custmomised , we call it custom comparator sort . I have use lambda function to implement comparison if the numbers are same

#include <bits/stdc++.h>

using namespace std;

int main(){

int n;

cin>>n;

vector<pair<int,int>>v(n);

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

pair<int,int>temp;

cin>>temp.first>>temp.second;

v[i] = temp;

}

sort(v.begin(),v.end(),[](auto a,auto b){

if(a.first==b.first){

return a.second<b.second;

}

return a.first<b.first;

});

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

cout<<v[i].first<<" "<<v[i].second<<endl;

}

return 0;

}

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*

#include <bits/stdc++.h>

using namespace std;

int main(){

int n;

cin>>n;

vector<int>v(n);

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

cin>>v[i];

}

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

int num;

cin>>num;

int rank=0;

auto it = lower\_bound(v.begin(),v.end(),num);

rank = it-v.begin();

cout<<rank<<endl;

return 0;

}