geeksforgeeks.org

Find maximum meetings in one room - GeeksforGeeks

ShalikramPatel@ShalikramPatel

3 minutes

View Discussion

Improve Article

Saved Article

Like Article

There is one meeting room in a firm. There are **N** meetings in the form of **(S[i], F[i])** where **S[i]** is the start time of meeting **i** and **F[i]** is the finish time of meeting **i**. The task is to find the maximum number of meetings that can be accommodated in the meeting room. Print all meeting numbers

Examples:

> **Input :** s[] = {1, 3, 0, 5, 8, 5}, f[] = {2, 4, 6, 7, 9, 9}
> **Output :** 1 2 4 5
> First meeting [1, 2]
> Second meeting [3, 4]
> Fourth meeting [5, 7]
> Fifth meeting [8, 9]
>
> **Input :** s[] = {75250, 50074, 43659, 8931, 11273, 27545, 50879,

77924},
f[] = {112960, 114515, 81825, 93424, 54316, 35533, 73383,
160252 }
Output : 6 7 1

Approach:
Idea is to solve the problem using the greedy approach which is the same as Activity Selection Problem.

- Sort all pairs(Meetings) in increasing order of second number(Finish time) of each pair.
- Select first meeting of sorted pair as the first Meeting in the room and push it into result vector and set a variable time_limit(say) with the second value(Finishing time) of the first selected meeting.
- Iterate from the second pair to last pair of the array and if the value of the first element(Starting time of meeting) of the current pair is greater then previously selected pair finish time (time_limit) then select the current pair and update the result vector (push selected meeting number into vector) and variable time_limit.
- Print the Order of meeting from vector.

Below is the implementation of the above approach.

- C++
- Java
- Python3
- Javascript

C++

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

```
using namespace std;
void maxMeetings(int s[], int f[], int n)
{
    pair<int, int> a[n + 1];
    int i;
    for (i = 0; i < n; i++) {
        a[i].first = f[i];
        a[i].second = i;
    }
    sort(a, a + n);
    int time_limit = a[0].first;
    vector<int> m;
    m.push_back(a[0].second + 1);
    for (i = 1; i < n; i++) {
        if (s[a[i].second] > time_limit) {
            m.push_back(a[i].second + 1);
            time_limit = a[i].first;
        }
    }
    for (int i = 0; i < m.size(); i++) {
        cout << m[i] << " ";
    }
```

```
}

int main()

{

    int s[] = { 1, 3, 0, 5, 8, 5 };

    int f[] = { 2, 4, 6, 7, 9, 9 };

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

    maxMeetings(s, f, n);

    return 0;

}
```

Java

Python3

Javascript

Output:

1 2 4 5

Time Complexity: O(N log(N))
Auxiliary Space: O(N) for creating a vector of pairs that is approximately equal to O(n)