Minimum Platforms Required at a Railway Station

Divyansh Joshi (RA2011003011138)
Kankana Majumder(RA2011003011142)
Bhanu Pratap Singh (RA2011003011149)

Submitted to Ms. Revathi .M
On 17-06-2022

Problem Statement

Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits.

Explanation

- If we observe the question, we need to find out how many trains will have an overlapping schedule . It means, if there is a train, Train A, and we need to find if there are any trains arriving before the arrival of a train . So we need to sort both of the arrays combined. Then we need to check if there are any trains whose departures are after the arrival of a train . After sorting, we will keep track of the platforms needed. Hence, here we can use Greedy Approach algorithm.

Algorithm

1. Sort the arrival and departure times of trains.
2. Create two pointers i=0, and j=0, and a variable to store ans and current count plat
3. Run a loop while i<n and j<n and compare the ith element of arrival array and jth element of departure array.
4. If the arrival time is less than or equal to departure then one more platform is needed so increase the count, i.e., plat++ and increment I
5. Else if the arrival time is greater than departure then one less platform is needed to decrease the count, i.e., plat– and increment j
6. Update the ans, i.e. ans = max(ans, plat).

CODE:

```
#include <algorithm>
#include <iostream>
using namespace std
int findPlatform(int arr[], int dep[], int n){
sort(arr, arr + n);
        sort(dep, dep + n);
        int plat_needed = 1, result = 1;
        int i = 1, j = 0;
        while (i < n && j < n) {
        if (arr[i] <= dep[j]) {
        plat_needed++;
        i++;
        }
```

CODE(cont.)

```
else if (arr[i] > dep[j]) {
plat_needed--;
j++;
if (plat_needed > result)
        result = plat_needed;
}       return result;
}
int main()
{
int arr[]= { 900, 940, 950, 1100, 1500, 1800 };
int dep[] = { 910, 1200, 1120, 1130, 1900, 2000 };
int n = sizeof(arr) / sizeof(arr[0]);
cout << findPlatform(arr, dep, n);
return 0;
}
```

Output

Output

- Input: arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}
- dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}
- Output: 3

Explanation

- There are at-most three trains at a time (time between 9:40 to 12:00)
- Input: arr[] = {9:00, 9:40}
- dep[] = {9:10, 12:00}
- Output: 1 Explanation: Only one platform is needed.

Time Complexity

- Time Complexity: O(nlogn) Sorting takes O(nlogn) and traversal of arrays takes O(n) so overall time complexity is O(nlogn)

Conclusion

- For solving the minimum number of platform required we use a greedy approach to find the optimal solution the current choice is made such that it is the best choice at that moment, without worrying about the future consequences of the said choice. Once made, this choice is never altered . Though intuitive, it is very hard to prove the correctness of the algorithm . Greedy algorithm refers to a class of algorithms that to some optimization problem. It does not always lead to the global optimal solution . A correct greedy algorithm is very fast and efficient as compared to other algorithms, such as dynamic programming.

Thank You