630. Course Schedule III

Description HintsSubmissionsSolutions

Discuss Editorial Solution Pick One

- Difficulty:**Medium**
- Category:Algorithms
- Acceptance:17.04%
- Contributor:Stomach_ache

There are `n` different online courses numbered from `1` to `n`. Each course has some duration(course length) `t` and closed on d_{th}day. A course should be taken **continuously** for `t` days and must be finished before or on the d_{th} day. You will start at the 1_{st}day.

Given `n` online courses represented by pairs `(t,d)`, your task is to find the maximal number of courses that can be taken.

Example:

```
Input: [[100, 200], [200, 1300], [1000, 1250], [2000, 3200]]

Output: 3

Explanation:

There're totally 4 courses, but you can take 3 courses at most:

First, take the 1st course, it costs 100 days so you will finish it on the 1
00th day, and ready to take the next course on the 101st day.

Second, take the 3rd course, it costs 1000 days so you will finish it on the
 1100th day, and ready to take the next course on the 1101st day.

Third, take the 2nd course, it costs 200 days so you will finish it on the 1
300th day.

The 4th course cannot be taken now, since you will finish it on the 3300th d
ay, which exceeds the closed date.
```

Note:

1. The integer 1 <= d, t, n <= 10,000.
2. You can't take two courses simultaneously.

Seen this question in a real interview before?

Yes

```
#include<iostream>
#include<stdio.h>
#include<algorithm>
#include<unordered_set>
#include<string>
#include<stdlib.h>
#include<queue>
#include<memory.h>
using namespace std;

static bool compare(vector<int> &A, vector<int> &B)
{
    return A.back()<B.back();
}

static int scheduleCourse(vector<vector<int>>& courses)
{
    int t = 0;
    sort(courses.begin(),courses.end(),compare);
    priority_queue<int,vector<int>,less<int>> maxHeap;

    for(auto c:courses)
    {
        if(c[0]+t<=c[1])
        {
            t+=c[0];
            maxHeap.push(c[0]);
        }else
        {
            // MR.BLACK: if we find a better choice
            // substitute the maxHeap with the current one
            // note that this is a greedy choice
            if(c[0]<maxHeap.top() && c[0]+t-maxHeap.top()<=c[1])
            {
                t -=maxHeap.top();
                maxHeap.pop();
                t+=c[0];
                maxHeap.push(c[0]);
            }
        }
    }

    return maxHeap.size();
}

int main(int argc,char *argv[])
{
    //Input: [[100, 200], [200, 1300], [1000, 1250], [2000, 3200]]
    vector<vector<int>>                      input                         =
{{100,200},{200,300},{1000,1250},{2000,3200}};
    int ans = scheduleCourse(input);
    for(auto k:input)
    {
        cout<<k[0]<<"--"<<k[1]<<endl;
    }
    cout<<ans<<endl;
    return 0;
}
```