**Họ và tên:** Nguyễn Hồ Trường An

**MSSV:** 47.01.103.024

**MÔ TẢ:**

**Đầu vào:**

vector<Job>& jobs: một vector các công việc (Job), mỗi công việc được mô tả bằng ba thuộc tính là thời điểm bắt đầu công việc (start), thời điểm kết thúc công việc (end) và lợi nhuận (profit).

**Mô tả đầu ra:** Lợi nhuận tối đa có thể đạt được khi xếp lịch các công việc.

**Cách xử lí:**

* Sắp xếp các công việc theo thời gian kết thúc tăng dần bằng hàm sort.
* Khởi tạo một mảng dp với kích thước bằng số lượng công việc trong danh sách, dp[i] lưu lợi nhuận tối đa có thể đạt được khi chỉ xét các công việc từ 0 đến i.
* Với mỗi công việc, tìm công việc trước đó có thời điểm kết thúc nhỏ hơn hoặc bằng thời điểm bắt đầu công việc hiện tại. Nếu tìm thấy, thêm lợi nhuận của công việc đó vào lợi nhuận của công việc hiện tại. Lấy max giữa lợi nhuận của công việc hiện tại và tổng lợi nhuận của công việc.

**CÀI ĐẶT**

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

struct Job {

int start, end, profit;

};

bool cmp(Job a, Job b) {

return a.end < b.end;

}

int jobScheduling(vector<Job>& jobs) {

sort(jobs.begin(), jobs.end(), cmp);

int n = jobs.size();

vector<int> dp(n, 0);

dp[0] = jobs[0].profit;

for (int i = 1; i < n; i++) {

int currProfit = jobs[i].profit;

int latestJobIdx = -1;

for (int j = i - 1; j >= 0; j--) {

if (jobs[j].end <= jobs[i].start) {

latestJobIdx = j;

break;

}

}

if (latestJobIdx != -1) {

currProfit += dp[latestJobIdx];

}

dp[i] = max(currProfit, dp[i - 1]);

}

return dp[n - 1];

}

int main() {

vector<Job> jobs = { {1, 3, 5}, {2, 5, 6}, {4, 6, 5}, {6, 7, 4}, {5, 8, 11}, {7, 9, 2} };

int maxProfit = jobScheduling(jobs);

cout << "Max profit: " << maxProfit << endl;

return 0;

}