ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Решение задач с использованием динамического программирования

на сторонней платформе»

Выполнил работу

Воробьев Андрей

Академическая группа №J3110

Принято

Ментор, Вершинин Владислав

Санкт-Петербург

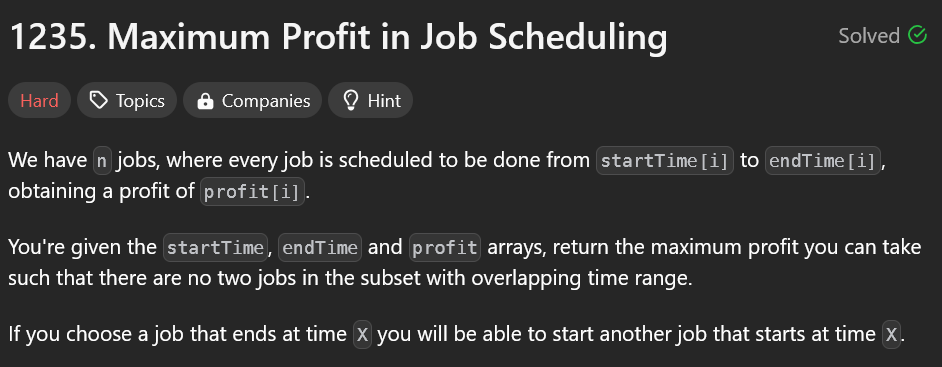
2024

1. Введение

Цель лабораторной работы – научиться уместно применять на практике динамическое программирование.

Задачи:

* найти платформу для решения задач на динамическое программирование
* решить [задачу](https://leetcode.com/problems/maximum-profit-in-job-scheduling/description/?envType=problem-list-v2&envId=dynamic-programming), представленную на изображении 1:



Изображение 1 – Задача на платформе leetcode.com

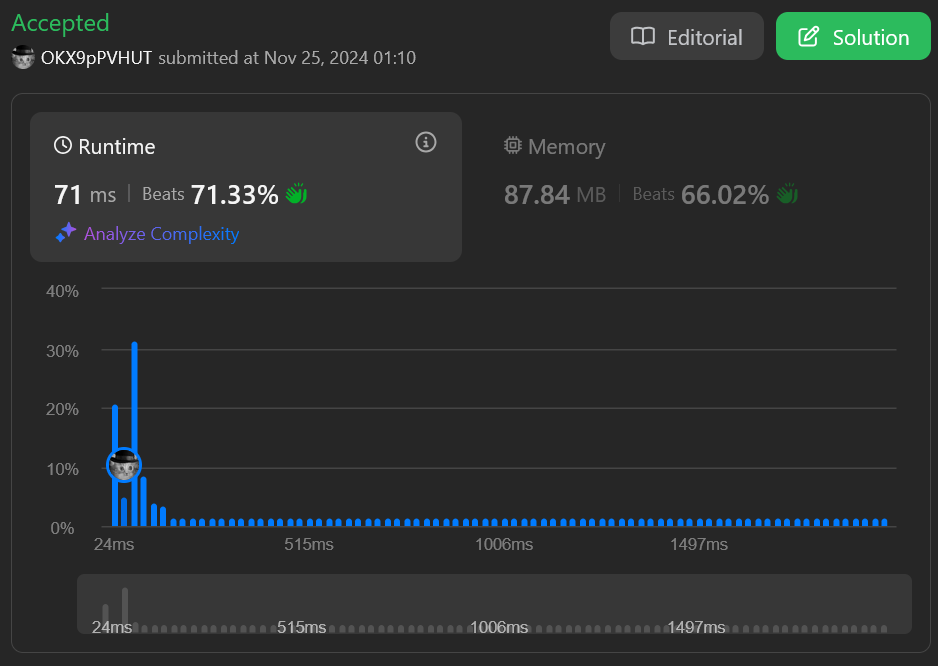
* пройти тесты на платформе
* осознать необходимость использования динамического программирования в решении задачи

1. Реализация

* реализация считывания и сортировки данных
* реализация динамического программирования
* реализация бинарного поиска

1. Экспериментальная часть.

Вся экспериментальная часть с тестами и проверками была реализована на платформе, результаты работы алгоритма представлены на изображении 2.



Изображение 2 – Результаты тестирования алгоритма

По результатам видно, что алгоритм полностью верен и лучше большинства других алгоритмов, написанных пользователями.

1. Заключение.

В ходе выполнения работы мною был реализован алгоритм динамического программирования. Задача действительно, имеет единственное логичное решение через этот подход. Так как в основе задачи лежит решение, вносить ли в расписание дело или не вносить. То есть нужно обраться к уже полученным данным, и выяснить, что будет лучше в итоге. Такая задача решается именно через динамическое программирование.

Алгоритм был реализован на языке C++.

1. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода алгоритма

class Solution {  
public:  
 int jobScheduling(vector<int> &startTime, vector<int> &endTime, vector<int> &profit) {  
 size\_t n = startTime.size();  
  
 vector<vector<int> > jobs;  
 *// дело: {конец, начало, прибыль}* jobs.reserve(n + 1);  
 *// O(n) - память* jobs.push\_back({0, 0, 0});  
 *// нулевая дело* for (size\_t i = 0; i < n; ++i)  
 jobs.push\_back({endTime[i], startTime[i], profit[i]});  
 *// O(n) - время* sort(jobs.begin(), jobs.end());  
 *// O(n \* log n) - время* int res = 0;  
 vector<int> dp;  
 dp.reserve(n + 1);  
 *// O(n) - память* dp.push\_back(0);  
 *// доход от нулевой дела* for (size\_t i = 1; i < n + 1; ++i) {  
 *// O(n) - время* size\_t left = 1, right = i;  
 while (left < right) {  
 int mid = (left + right) / 2;  
  
 if (jobs[mid][0] <= jobs[i][1])  
 left = mid + 1;  
 else  
 right = mid;  
 }  
 *// поиск последнего дела, которое заканчивается до начала текущего дела  
 // O(log (right - left)) - время* dp.push\_back(max(dp[i - 1], dp[left - 1] + jobs[i][2]));  
 *// выбор: брать новое дело и вместе с ним все дела, которые были сделаны до его начала  
 // или оставить прошлый доход* res = max(res, dp[i]);  
 }  
 *// O(n \* log n) - время* return res;  
  
 *// O(n \* log n) - время  
 // O(n) - память* }  
};