ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Жадные алгоритмы. Задача 2931 “Maximum Spending After Buying Items”»

Выполнил работу

Морозов Дмитрий

Академическая группа №J3110

Принято

Ментор, Вершинин Владислав

Санкт-Петербург

2024

1. Введение

Цель работы: решить задачу, используя жадный алгоритм

Задачи:

­­­­­­­­­­— Провести анализ задачи

—Реализовать жадный алгоритм

— Проанализировать затрачиваемую память и время работы программ

1. Реализация
   1. Анализ задачи, почему именно жадный алгоритм

Условие задачи приведено на изображении №1. Согласно нему, у нас есть m магазинов, в каждом из которых есть n товаров, которые передаются в порядке не возрастания цены. Цена за товар вычисляется, как стоимость \* номер дня. Купить можно только самый дешевый товар, из ранее не купленных в этом магазине. Требуется определить, сколько денег максимум можно потратить. Чтобы прийти к такой сумме, требуется сначала покупать дешевые товары, чтобы дорогие остались на самые последние дни. Поэтому эту задачу можно решить с помощью жадного алгоритма, так как на каждом шаге мы будем делать оптимальный выбор по определенному критерию.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок №1 – условие задачи

* 1. Написание программы
     1. Реализация solve

На данном этапе была реализована функция maxSpending, которая определяет максимальную сумму, которую можно потратить. Так как покупать можно самые дешевые товары, создается вектор из указателей right\_ind, каждое i-ое значение которого хранит индекс самого дешевого из не купленных товаров i-го магазина. Далее работает цикл, который выбирает минимальный по стоимости товар для покупки для каждого дня. Чтобы определить товар для покупки, надо найти самый дешевый товар из возможных, здесь и нужен right\_ind. Если right\_ind[i] ≥ 0, значит в i-ом магазине еще есть товар для покупки. После выявления оптимального товара его вес прибавляется к результату, а i-ый указатель в right\_ind уменьшается на единицу, так как мы купили товар в этом магазине. Реализация maxSpending представлена на изображении №2.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок №2 – реализация функции maxSpending

* 1. Экспериментальная часть работы

На данном этапе производилась оценка затрачиваемой памяти и асимптотики. Также была осуществлена проверка встроенными тестами

* 1. Анализ проделанной работы

На данном этапе был выполнен анализ проделанной работы, удалось ли достичь необходимую цель и выполнить задачи.

1. Экспериментальная часть

На данном этапе производилась оценка затрачиваемой памяти и асимптотики. Также была осуществлена проверка встроенными тестами

Подсчёт по памяти:

Переменные res и min\_value занимают в сумме 16 байт, как объекты long long. На остальные переменные типа int выделяется еще 4 \* 4 = 16 байт. Вектор right\_ind занимает m \* 4 байт.

Итог: 16 + 16 + m \* 4 = 32 + m \* 4 байт, где m – количество магазинов.

Подсчёт асимптотики:

Функция maxSpending состоит из вложенного цикла, который для каждого из m \* n дней ищет минимальный по стоимости из n товаров, поэтому работает за O(n \* n \* m) = O(m \* n^2), где m – количество магазинов, n – количество товаров в каждом магазине. Так как m не более 10, можно сказать, что программа имеет сложность O(n^2).

Проверка тестами:  
Написанная программа успешно прошла все встроенные тесты. Результаты представлены на изображении №3.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рисунок №3 – Доказательство решения задачи

1. Заключение

В ходе лабораторной работы мною была решена задача с использованием жадного алгоритма. Данный подход был здесь уместен, это подтверждают успешные результаты работы.

ПРИЛОЖЕНИЕ А

Листинг кода файла lab\_7.cpp

#include <iostream>

#include <vector>

class Solution {

public:

    long long maxSpending(vector<vector<int>>& values) {

        long long res = 0; //результат

        int m = values.size(); //кол-во магазинов

        int n = values[0].size(); //кол-во товаров

        std::vector<int> right\_ind(m, n - 1); //указатели на доступные товары

        for (int day = 1; day <= m \* n; day++) {

            long long min\_value = 1000001; //минимальный правый товар

            int min\_i = 0;

            int min\_j = 0;

            for (int i = 0; i < m; i++) { //поиск самого дешевого товара

                int j = right\_ind[i];

                if (j >= 0) {

                    int value = values[i][j];

                    if (value < min\_value) {

                        min\_value = value;

                        min\_i = i;

                        min\_j = j;

                    }

                }

            }

            res += day \* min\_value;

            right\_ind[min\_i] -= 1; //меняем указатель, т.к. купили товар

        }

        return res;

    }

};