ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 4

«NP-полные задачи»

Выполнил работу

Макаров Эльдар

J3114

Принято

Должность, звание Фамилия Имя преподавателя

Санкт-Петербург

2024

1. Введение

Цель работы: изучить и реализовать NP-полную задачу, а также понять, почему высокая асимптотическая сложность — это очень плохо.

Задачи: решить одну из предоставленных задач с указанными ограничениями.

1. Теоретическая подготовка

NP-полная задача — в теории алгоритмов задача с ответом «да» или «нет» из класса NP, к которой можно свести любую другую задачу из этого класса за полиномиальное время (то есть при помощи операций, число которых не превышает некоторого полинома в зависимости от размера исходных данных). Таким образом, NP-полные задачи образуют в некотором смысле подмножество «типовых» задач в классе NP: если для какой-то из них будет найден «полиномиально быстрый» алгоритм решения, то и любую другую задачу из класса NP можно будет решить так же «быстро».

1. Реализация

В этом разделе вам необходимо описать процесс выполнения работы, что вы сделали и какие этапы при этом выполняли, выжимки из кода, библиотеки и особенности реализации. Важно, не бывает 2 этапов выполнения задачи, Начали – закончили.

Этап 1: Выбор задачи

В первую очередь я выбрал подходящую задачу. Я остановился на задаче о рюкзаке, так как она показалась мне довольно простой в решении.

Этап 2: Анализ задачи

Из-за высокого ограничения по асимптотической сложности было очевидно, что задачу нужно решать самым простым и медленным способом, так что я решил проверять каждое возможное подмножество множества.

Этап 3: Разработка алгоритма

Я решил, что проще всего выполнять эту работу будет с помощью рекурсивной функции, которая на каждом шаге рассматривает вариант включить функцию в подмножество или не включить. А когда индекс достигает конца списка, код проверяет не превышает ли общее значение веса предыдущее лучшее значение, и если это так, то обновляет лучший результат.

Этап 4: Написание кода

#include <iostream>

#include <vector>

struct Item {

    int weight;

    int cost;

};

// function dlya perebora

void findsubset(const std::vector<Item>& items, int index, int current\_weight, int current\_cost,

    int weight\_limit, std::vector<int>& current\_selection, std::vector<int>& best\_selection, int& best\_cost) {

    // proverka na luchshee reshenie

    if (index == items.size()) {

        if (current\_weight <= weight\_limit && current\_cost > best\_cost) {

            best\_cost = current\_cost;

            best\_selection = current\_selection;

        }

        return;

    }

    // variant ne vkluchat predmet

    findsubset(items, index + 1, current\_weight, current\_cost, weight\_limit,

        current\_selection, best\_selection, best\_cost);

    // variant vkluchit predmet

    current\_selection.push\_back(items[index].weight);

    findsubset(items, index + 1, current\_weight + items[index].weight, current\_cost + items[index].cost, weight\_limit,

        current\_selection, best\_selection, best\_cost);

    current\_selection.pop\_back();

}

int main() {

    std::vector<Item> items = { {2, 3}, {3, 4}, {4, 5}, {1, 4}, {2, 4} };

    int weight\_limit = 5;

    std::vector<int> current\_selection;

    std::vector<int> best\_selection;

    int best\_cost = 0;

    findsubset(items, 0, 0, 0, weight\_limit, current\_selection, best\_selection, best\_cost);

    // vivod resultata

    std::cout << "[";

    for (int i = 0; i < best\_selection.size(); i++) {

        std::cout << best\_selection[i];

        if (i < best\_selection.size() - 1) {

            std::cout << " ";

        }

    }

    std::cout << "]\n";

    std::cout << "(максимальная стоимость " << best\_cost << ")" << std::endl;

    return 0;

}

1. Экспериментальная часть

Согласно требованиям моего варианта, на вход к моему алгоритму подаётся до 25 элементов. Теоретически заданная сложность задачи составляет O(2^N) и более. Для тестирования алгоритма была собрана статистика, приведенная в таблице №1.

Таблица №1 - Подсчёт сложности реализованного алгоритма

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер входного набора | 1 | 10 | 20 | 25 | 30 | 35 |
| Время выполнения программы, с | 0,0000002 | 0,0001568 | 0,142 | 3,605 | 120,374 | 3794,973 |
| O(2\*N), с | 2 | 1024 | 1048576 | 33554432 | 1073741824 | 34359738368 |

График представляющий визуально удобный формат данных из таблицы №1 представлен на изображении №1.

Изображение №1 - График работы алгоритма

По графику и таблице видно, что даже при малом количестве элементов время выполнения работы стремительно увеличивается. Уже на 30 элементах алгоритм затруднялся, а на 35 он потратил огромное количество времени на выполнение. Дальше время выполнения очевидно только увеличивается и увеличивается.

1. Заключение

В ходе выполнения лабораторной работы мною был реализован алгоритм решения задачи о рюкзаке, которую я решил, используя рекурсивный перебор подмножеств. Цель работы была достигнута: задача решена, программа определяет набор предметов с максимальной стоимостью, удовлетворяющий ограничению по весу. Тестирование на различных входных данных разных размеров подтвердило соответствие полученных результатов теоретическим оценкам экспоненциальной сложности алгоритма (O(2^N)).

В качестве дальнейших исследований можно предложить: использование альтернативных, более эффективных подходов; решение других задач ради закрепления материала.

1. Приложения

- Нет