ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 4

«Задача о покрытии множеств»

Выполнил работу

Максудов Кирилл

Академическая группа J3111

Принято

Практик Вершинин Владислав

Санкт-Петербург

2024

Введение

Цель работы: дан массив, необходимо разделить его на две части с равной суммой.

Задачи:

1. Написать алгоритм
2. Измерить время работы алгоритма
3. Теоретическая подготовка

Использовал векторы, неупорядоченные множества, int, string и тд.

1. Реализация

Увидев сложность O(2^N) сразу решил представить перебор всех вариантов множеств как двоичную строку длинны n, где n – количество множеств(1-берем множество, 0 - нет), таким образом я подсчитал количество всех возможных вариантов по формуле n\*n-1, и циклом перебирал все, переводя в двоичную и проверяя получившуюся комбинацию(для проверки создал копию исходного множества и удалял из него подмножества по ходу цикла по строке с дв числом). Если я перекрыл исходное множество полностью, то сравниваю его длину с лучшим прошлым вариантом и в случае необходимости заменяю значение.

Во время выполнения столкнулся с проблемой что вариант 001 отображается как 1, решил это путем создания вектора заданной длинны из нулей => gв цикле иду по этому вектору, а не строке.

1. Экспериментальная часть

Подсчет по памяти(только для циклов и сложных структур):

std::vector<std::unordered\_set<int>> sets – для каждого set: 32\*n бит(исходный вектор с множествами).

std::unordered\_set<int> universe – 32\*n бит (исходное множество)

std::vector<std::unordered\_set<int>> variants - для каждого set: 32\*n бит(ответ).

std::unordered\_set<int> universe2 – 32\*n бит (копия исходного множества)

std::vector<char> comb\_dv(len, '0') – 8\*n бит (двоичная срока в виде вектора)

std::vector<std::unordered\_set<int>> sets2 – для каждого set: 32\*n бит(вектор с текущим вариантом ответа)

Подсчёт асимптотики (только для циклов и сложных структур):

dvoich() - O((log​(a))^2) – a – переводимое число

f() – основная функция поиска вектора подмножеств – содержит 2 вложенных цикла – O(n^3) – n – количество множеств

Изображение №1 – График зависимости времени от числа элементов

Согласно требованиям моего варианта, на вход к моему алгоритму подаётся до 25 множеств. Теоретически заданная сложность задачи составляет O(N^4) и более. Для тестирования алгоритма была собрана статистика, приведенная в таблице №1.

Таблица №1 - Подсчёт сложности реализованного алгоритма

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество множеств | 10 | 15 | 20 | 25 |
| Расчетное время, с | 0,0024235 | 0,0079959 | 0,019384 | 0,0377988 |
| Время, c | 0.0024235 | 0.0054543 | 0.0109504 | 0.0196261 |

1. Заключение

В ходе выполнения работы мною был реализован алгоритм для полного перебора возможных комбинаций множеств. Для улучшения можно попробовать реализовать его рекурсивно или реализовать проход по двоичной строке другими методом(без перевода в вектор).

ПРИЛОЖЕНИЕ А

Листинг кода файла lab-4.cpp

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание