МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания 8-2

Тема: «Реализация алгоритмов на основе сокращения числа переборов»

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Фамилия И. О.
Группа ААAA-00-00

Москва 2024

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ.....3
1.1 Постановка задачи.....3
1.2 Индивидуальный вариант.....3
2 ХОД РАБОТЫ.....4
2.1 Алгоритм грубой силы.....4
2.2 Жадный алгоритм.....5
2.3 Результаты тестирования.....6
3 ВЫВОД.....7
4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ.....8

1 ВВЕДЕНИЕ

1.1 Постановка задачи

1. Разработать алгоритм решения задачи с применением метода, указанного в варианте и реализовать программу.

2. Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы. Сравнить с числом переборов при применении метода.

1.2 Индивидуальный вариант

Таблица 1 — Индивидуальный вариант

Вариант	Задача	Алгоритм
17	Монетная система некоторого государства состоит из монет достоинством: a1 = 1 < a2 < … < an. Требуется выдать сумму наименьшим возможным количеством монет.	Жадный алгоритм

2 ХОД РАБОТЫ

2.1 Алгоритм грубой силы

Задачу для поиска оптимального количества монет можно решить методом грубой силы, или же полным перебором. Решение состоит в рекурсивном переборе всех вариантов.

Мы берём текущую монету и перебираем такое количество раз, при котором мы сможем использовать эту монету максимально возможное количество раз. Каждую такую монету мы добавляем в текущую комбинацию и рекурсивно вызываем функцию с новыми параметрами. Условиями выхода из рекурсии служат текущая сумма, которая даёт выход из рекурсии если она равна или превышает S — сумму которую необходимо набрать.

Если сумма равна S — то мы нашли новую комбинацию, и проверяем с текущим известным минимальным количеством монет. Если их меньше, то найденная комбинация новая лучшая.

Код функций прямого перебора представлен в листинге 1.

Листинг 1 - Алгоритм грубой силы

```
long long int compare_count = 0;

// Прямой перебор
void find_combinations(const vector<int>& coins, int S, int
current_sum, int current_count, int index, vector<int>&
current_combination, int& min_coins, vector<int>&
best_combination) {

    compare_count += 2;
    if (current_sum == S) {
        if (current_count < min_coins) {
            min_coins = current_count;
            best_combination = current_combination;
        }
        return;
    }

    compare_count += 2;
    if (current_sum > S || index == coins.size()) {
        return;
    }

    int coin = coins[index];
```

```
    int max_use = (S - current_sum) / coin;
    for (int i = 0; i <= max_use; ++i) {
        compare_count += 1;

        for (int j = 0; j < i; ++j) {
            compare_count += 1;
            current_combination.push_back(coin);
        }
          find_combinations(coins, S, current_sum + i * coin,
current_count + i, index + 1, current_combination, min_coins,
best_combination);
        for (int j = 0; j < i; ++j) {
            compare_count += 1;
            current_combination.pop_back();
        }
    }
}

pair<int,      vector<int>>      find_min_coins_bruteforce(const
vector<int>& coins, int S) {
    int min_coins = INT_MAX;
    vector<int> best_combination;
    vector<int> current_combination;
    compare_count = 0;

     find_combinations(coins, S, 0, 0, 0, current_combination,
min_coins, best_combination);

    return {min_coins, best_combination};
}
```

2.2 Жадный алгоритм

Решение жадным алгоритмом оказывается намного проще, однако не всегда даёт оптимальный результат.

Алгоритм идёт от максимальной доступной монеты и смотрит, помещается-ли она в нужную сумму. Если да, то сумма уменьшается на значение этой монеты, и алгоритм повторяется. Если нет, то далее смотрятся монеты номиналом меньше. Код функции жадного алгоритма представлен в листинге 2.

Листинг 2 - Жадный алгоритм

```
// Жадный алгоритм
vector<int> find_min_coins_greedy(const vector<int>& coins, int
S) {
    vector<int> sorted_coins = coins;
    sort(sorted_coins.rbegin(), sorted_coins.rend());
```

```
    compare_count = 0;

    vector<int> combination;

    for (int coin : sorted_coins) {
        while (S >= coin) {
            compare_count += 1;
            S -= coin;
            combination.push_back(coin);
        }
    }

    return  combination;
}
```

2.3 Результаты тестирования

Результаты тестирования представлены на рис. 1, 2. Для тестирования были рассмотрены два случая. В первом случае оба алгоритма дают одинаковый результат. Во втором случае жадный алгоритм выдаёт не оптимальный результат. Сравнительная таблица представлена в таблице 2.

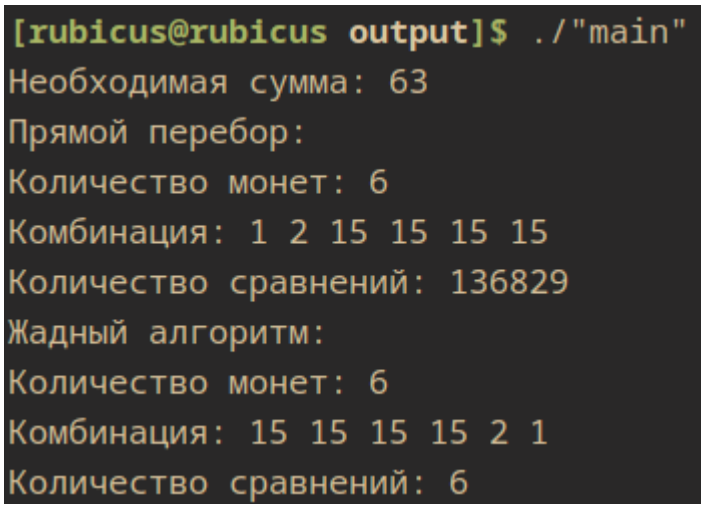

Рисунок 1 - Первый случай

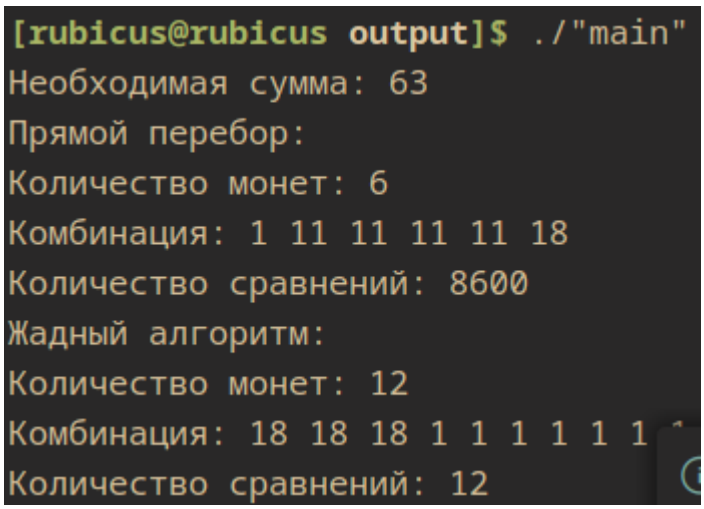

Рисунок 2 - Второй случай

Таблица 1 - Сравнительная таблица количества сравнений

Случай	Алгоритм грубой силы	Жадный алгоритм
Одинаковый	136829 шт.	6 шт.
Жадный неоптимальный	8600 шт.	12 шт.

3 ВЫВОД

Были освоены приём по применению и реализации различных алгоритмов на основе сокращения числа переборов для решения алгоритмических задач.

4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Структуры и алгоритмы обработки данных (часть 2): Лекционные материалы / Рысин М. Л. МИРЭА — Российский технологический университет, 2022/23. – 82 с.