МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8.2

по дисциплине
«Структуры и алгоритмы обработки данных»

Тема: «Реализация алгоритмов на основе сокращения числа переборов»

Выполнил студент: Лазаренко С.А.
Группа: ИКБО-10-23

Москва 2024

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

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

17	Монетная система некоторого государства состоит из монет достоинством $a_1 = 1 < a_2 < ... < a_n$. Требуется выдать сумму наименьшим возможным количеством монет.	Жадный алгоритм

Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы. Сравнить с числом переборов при применении метода. Оформить отчет в соответствии с требованиями документирования.

ХОД РАБОТЫ

Формулировка задачи

Начнем с самой крупной монеты, которая меньше или равна сумме S, и будем использовать её максимально возможное количество раз. Вычитаем номинал монеты из суммы столько раз, сколько это возможно, и уменьшаем S на соответствующее значение. Переходим к следующей по величине монете и повторяем процесс, пока S не станет равным нулю. Жадный алгоритм будет работать корректно, если набор монет является каноническим, то есть для любой суммы оптимальное разбиение всегда достигается с использованием наибольших доступных монет.

Индивидуальный вариант работы – 17.

Описание подхода к решению

Для метода "в лоб" необходимо рассмотреть все возможные комбинации монет, что приводит к экспоненциальному росту числа переборов в зависимости от количества монет. При использовании жадного алгоритма количество переборов будет значительно меньше, поскольку алгоритм просто проходит по списку номиналов монет в убывающем порядке.

Код программы

Реализуем код программы на языке программирования C++ :

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

// Функция для нахождения минимального количества монет для суммы S
int minCoins(vector<int>& coins, int S) {
    int count = 0;
    sort(coins.rbegin(), coins.rend());

    for (int coin : coins) {
        if (S == 0) break;
        int numCoins = S / coin;
        count += numCoins;
        S -= numCoins * coin;
    }

    if (S != 0) {
        cout << "Невозможно набрать сумму этими монетами." << endl;
        return -1;
    }

    return count;
}
```

Рисунок 1 – код функции minCoins

```cpp
int main() {
    int n, S;
    cout << "Введите количество номиналов монет: ";
    cin >> n;

    vector<int> coins(n);
    cout << "Введите номиналы монет: ";
    for (int i = 0; i < n; ++i) {
        cin >> coins[i];
    }

    cout << "Введите сумму, которую нужно набрать: ";
    cin >> S;

    int result = minCoins(coins, S);
    if (result != -1) {
        cout << "Минимальное количество монет: " << result << endl;
    }

    return 0;
}
```

Рисунок 2 – Код функции main

Результаты тестирования

Выполним тестирование программы:

```
gwynbleidd@MacBook-Air-Sergej-3 практика 8.2 % ./main8_2.cpp
Введите количество номиналов монет: 4
Введите номиналы монет: 1 5 10 15
Введите сумму, которую нужно набрать: 89
Минимальное количество монет: 10
gwynbleidd@MacBook-Air-Sergej-3 практика 8.2 % 3
zsh: command not found: 3
gwynbleidd@MacBook-Air-Sergej-3 практика 8.2 % ./main8_2.cpp
Введите количество номиналов монет: 3
Введите номиналы монет: 1 5 10
Введите сумму, которую нужно набрать: 29
Минимальное количество монет: 7
```

Рисунок 3 – Тестирование программы

Тестирование показало, что программа работает корректно.

ВЫВОД

Создание класса для красно-черного дерева позволило применять строгие правила цветовой иерархии, что гарантирует, что высота дерева остается логарифмической по отношению к количеству узлов. Благодаря этому обеспечивается высокая производительность при работе с большими объемами данных. Реализованные методы обхода дерева (симметричный и прямой) позволяют эффективно извлекать данные, в то время как функции для подсчета листьев и определения высоты дерева обеспечивают дополнительный анализ структуры. Визуализация дерева помогает пользователю интуитивно понять его конфигурацию и внутреннюю организацию. Таким образом, проект достигает своей цели, предлагая надежное, быстрое и наглядное решение для работы с динамическими наборами данных.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.

2. Документация по языку C++ [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/cpp/cpp/ (дата обращения 08.09.2024).

3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 04.09.2024)