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 №6

Тема: Алгоритмические стратегии. Перебор и методы его сокращения

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент Хан А.А.

группа ИКБО-04-20

Москва 2021

Оглавление

Тема

Алгоритмические стратегии. Перебор и методы его сокращения.

Цель

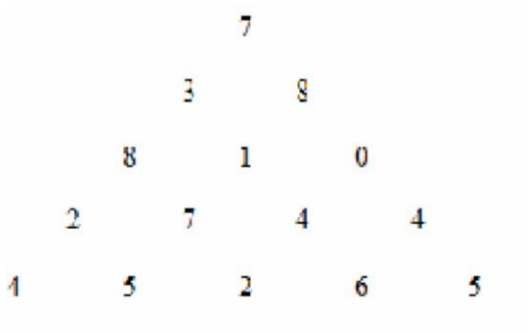
Разработка и программная реализация задач с применение метода сокращения числа переборов.

Персональный вариант

Вариант №9

Постановка задачи

Треугольник имеет вид, представленный на рисунке. Необходимо написать программу, которая вычисляет наибольшую сумму чисел, расположенных на пути, начинающемся в верхней точке треугольника и заканчивающегося на основании треугольника.



Описание алгоритмов и подхода к решению

Для решения задачи применяется динамическое программирование.

Пронумеруем строки в треугольнике сверху вниз от 0 до (n – 1). Далее пронумеруем элементы в каждой строке j от 0 до (j – 1). Будем считать суммы на частичных путях от вершины до основания: $dp[i][j]$ –максимальная сумма среди всех возможных путей от вершины до элемента в строке i с номером j. Заметим, что $dp[0][0]$ равно самому числу в вершине. Далее, для элементов с номером j в строках с номером $i > 1$ верно:

- Если $j = 0$, то $dp[i][0] = dp[i-1][0] + a[i][0]$
- Если $j = i$, то $dp[i][i] = dp[i-1][i-1] + a[i][i]$
- Иначе, $dp[i][j] = \max(dp[i-1][j], dp[i][j]) + a[i][j]$

Так мы пересчитываем суммы на пути от вершины до элемента $a[i][j]$ и берем максимальную, используя данные для элементов с предшествующих рядов.

Для получения ответа необходимо перебрать все суммы на путях, заканчивающихся в элементах последней строк и взять максимум из этих значений.

Время работы алгоритмов

Алгоритм, использующей динамическое программирование, суммарно совершает присваиваний:

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

- по одному на каждый момент в треугольнике.

Алгоритм полного перебора рассматривает все возможные пути от вершины элемента из основания. Легко проверить, что суммарно путей:

$$C_n^1 + C_n^2 + C_n^3 + \cdots + C_n^n = 2^n$$

Код приложения

```
#include <iostream>
#include <vector>
#include <fstream>
#include <ctime>

struct triangle {
    int size;
    std::vector <std::vector <int>> val;

    triangle(int _size) {
        size = _size;
        val.resize(size);

        for (int i = 0; i < size; ++i) {
            val[i].resize(i + 1);
        }
    }
};

std::istream & operator>>(std::istream &in, triangle &a) {
    for (int i = 0; i < a.size; ++i) {
        for (int j = 0; j < a.val[i].size(); ++j) {
            in >> a.val[i][j];
        }
    }
    return in;
}

int dp_cnt = 0;
int bf_cnt = 0;

int get_ans_dp(triangle &a, triangle &dp) {
    dp.val[0][0] = a.val[0][0];
    dp_cnt += 1;

    for (int i = 1; i < a.size; ++i) {
        int cur_sz = a.val[i].size();
        dp.val[i][0] = dp.val[i - 1][0] + a.val[i][0];
        dp.val[i][cur_sz - 1] = dp.val[i - 1][cur_sz - 2] + a.val[i][cur_sz - 1];

        dp_cnt += 2;
```

```
        for (int j = 1; j < cur_sz - 1; ++j) {
            dp.val[i][j] = std::max(dp.val[i - 1][j - 1], dp.val[i - 1][j]) + a.val[i][j];

            dp_cnt += 1;
        }
    }

    int ans = 0;
    for (int i = 0; i < a.size; ++i) {
        ans = std::max(ans, dp.val[a.size - 1][i]);

        dp_cnt += 1;
    }
    return ans;
}

void get_ans_bf(int i, int j, int cur_ans, int &ans, triangle &a) {
    cur_ans += a.val[i][j];
    bf_cnt += 1;

    if (i == a.size - 1) {
        ans = std::max(ans, cur_ans);
        bf_cnt += 1;
        return;
    }

    get_ans_bf(i + 1, j, cur_ans, ans, a);
    get_ans_bf(i + 1, j + 1, cur_ans, ans, a);
}

int main() {
    int n;
    std::cin >> n;

    triangle a(n), dp(n);

    std::cin >> a;

    int dp_ans = get_ans_dp(a, dp);
    std::cout << "dp ans: " << dp_ans << "\n";
    std::cout << "dp cnt: " << dp_cnt << "\n";

    int bf_ans = 0;
    get_ans_bf(0, 0, 0, bf_ans, a);
    std::cout << "bf ans: " << bf_ans << "\n";
```

```
    std::cout << "bf cnt: " << bf_cnt << "\n";

    return 0;
}
```

Результаты тестирования

Тип теста	Результат DP	Результат BF	Кол-во присваиваний DP	Кол-во присваиваний BF
Тест из условия	30	30	20	47
n = 3	1170	1170	9	11
n = 4	1302	1302	14	23
n = 5	1761	1761	20	47
n = 6	1746	1746	27	95
n = 7	2501	2501	35	191
n = 8	2938	2938	44	383

Выводы

В результате проделанной работы, я получила практические навыки по работе с алгоритмическими стратегиями. А также научилась разрабатывать и реализовывать задачи с применение метода сокращения числа переборов.

Список использованной литературы

1. Лекционный материал по структурам и алгоритмам обработки данных Сартакова М. В. (дата обращения 13.11.2021)

2. Дополнительный материал к практическим работам по структурам и алгоритмам обработки данных Сорокина А. В. (дата обращения 13.11.2021)