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 группы ИКБО-41-23 Гольд Д.В.

Принял старший преподаватель Рысин М.Л.

Москва 2024

СОДЕРЖАНИЕ

Цель:

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

Задание:

№_	Задача	Метод
9	Треугольник имеет вид, представленный на рисунке. Напишите программу, которая вычисляет наибольшую сумму чисел, расположенных на пути от верхней точки треугольника до его основания. 7 3 8 8 1 0 2 7 4 4 4 5 2 6 5	Динамическое программирование

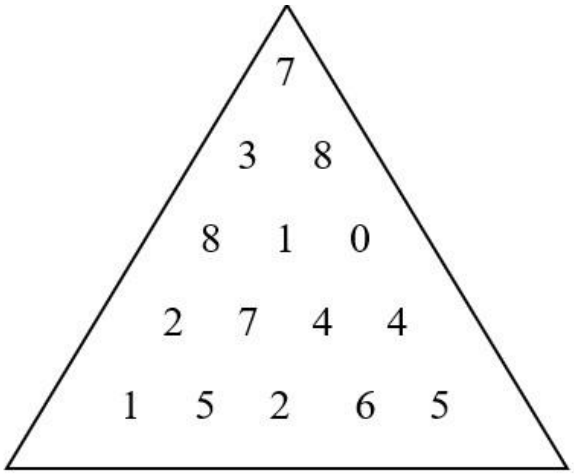
Код программы:

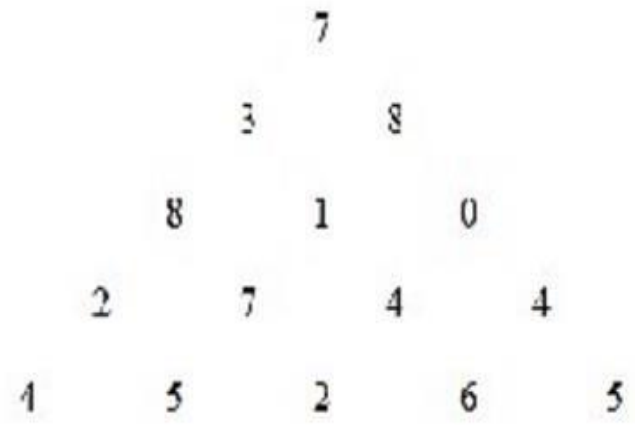
```cpp
#include <iostream>
#include <vector>
#include <algorithm>

int dp_count = 0;
int brute_force_count = 0;
int custom_count = 0;
int top_down_count = 0;

//динамическое программирование (снизу вверх)
int max_path_sum_dp(std::vector<std::vector<int>>& triangle) {
    int n = triangle.size();

    for (int i = n - 2; i >= 0; --i) {
        for (int j = 0; j < triangle[i].size(); ++j) {
            dp_count++;
            int left = triangle[i + 1][j];
            int right = triangle[i + 1][j + 1];
            int chosen = std::max(left, right);
            std::cout << "\ndp: выбор между " << left << " и " << right
                << " для элемента " << triangle[i][j]
                << ", выбираем " << chosen << std::endl;
            triangle[i][j] += chosen;
        }
    }

    return triangle[0][0];
}

//метод грубой силы - рекурсивный brute_force
int max_path_sum_brute_force(std::vector<std::vector<int>>& triangle, int row, int col) {
    brute_force_count++;
    if (row == triangle.size() - 1) {
```

```cpp
//метод грубой силы - рекурсивный brute_force
int max_path_sum_brute_force(std::vector<std::vector<int>>& triangle, int row, int col) {
    brute_force_count++;
    if (row == triangle.size() - 1) {
        std::cout << "\nметод перебора: находимся на базовом уровне с элементом " << triangle[row][col] << std::endl;
        return triangle[row][col];
    }

    int left_recursive_level_element = max_path_sum_brute_force(triangle, row + 1, col);
    int right_recursive_level_element = max_path_sum_brute_force(triangle, row + 1, col + 1);
    int chosen = std::max(left_recursive_level_element, right_recursive_level_element);

    std::cout << "метод перебора: для элемента " << triangle[row][col]
        << ", выбор между " << left_recursive_level_element
        << " (левый) и " << right_recursive_level_element
        << " (правый), выбираем " << chosen << std::endl;

    return triangle[row][col] + chosen;
}


//максимум из четырех возможных чисел (соседей на уровне и следующем уровне)
int max_path_sum_custom(std::vector<std::vector<int>>& triangle) {
    int n = triangle.size();

    for (int i = n - 2; i >= 0; --i) {
        for (int j = 0; j < triangle[i].size(); ++j) {
            custom_count++;
            //custom_count += 4;
            int number_left_same_level = (j > 0) ? triangle[i][j - 1] : 0;
            int number_right_same_level = (j < triangle[i].size() - 1) ? triangle[i][j + 1] : 0;
            int number_level_left_below_next = triangle[i + 1][j];
            int number_level_right_below_next = (j < triangle[i + 1].size() - 1) ? triangle[i + 1][j + 1] : 0;

            int chosen = std::max({ number_left_same_level, number_right_same_level,
                                    number_level_left_below_next, number_level_right_below_next });

            std::cout << "\nдля элемента " << triangle[i][j]
                << ", рассматриваются варианты: "
                << number_left_same_level << " (слева на том же уровне), "
                << number_right_same_level << " (справа на том же уровне), "
                << number_level_left_below_next << " (слева на уровне ниже), "
                << number_level_right_below_next << " (справа на уровне ниже), "
                << "выбираем " << chosen << std::endl;

            triangle[i][j] += chosen;
        }
    }

    return triangle[0][0];
}

//сверху вниз - с вершины треугольника итеративно добавляем значения к двум ближайшим элементам на след уровне
int max_path_sum_top_down(std::vector<std::vector<int>>& triangle) {
    int n = triangle.size();
    std::vector<std::vector<int>> dp = triangle;

    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < triangle[i].size(); ++j) {
            int left_choice = dp[i][j] + triangle[i + 1][j];
```

```
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < triangle[i].size(); ++j) {
            int left_choice = dp[i][j] + triangle[i + 1][j];
            int right_choice = dp[i][j] + triangle[i + 1][j + 1];

            dp[i + 1][j] = std::max(dp[i + 1][j], left_choice);
            dp[i + 1][j + 1] = std::max(dp[i + 1][j + 1], right_choice);

            top_down_count++;

            std::cout << "метод сверху вниз: для узла [" << i << "][" << j << "] (значение: " << triangle[i][j]
                << "), выбор между "
                << left_choice << " (левый узел: [" << i + 1 << "][" << j << "]) и "
                << right_choice << " (правый узел: [" << i + 1 << "][" << j + 1 << "]), ";

            if (left_choice > right_choice) {
                std::cout << "выбираем " << left_choice << std::endl;
            }
            else {
                std::cout << "выбираем " << right_choice << std::endl;
            }
        }
    }

    return *std::max_element(dp[n - 1].begin(), dp[n - 1].end());
}
```

```
    return *std::max_element(dp[n - 1].begin(), dp[n - 1].end());
}

int main() {
    std::vector<std::vector<int>> triangle = {
        {7},
        {3, 8},
        {8, 1, 0},
        {2, 7, 4, 4},
        {1, 5, 2, 6, 5}
    };

    std::vector<std::vector<int>> dp_triangle = triangle;
    int dp_result = max_path_sum_dp(dp_triangle);
    std::cout << "максимальная сумма (1) = " << dp_result << std::endl <<
        "переборы (1): " << dp_count << std::endl << std::endl;

    int brute_force_result = max_path_sum_brute_force(triangle, 0, 0);
    std::cout << "максимальная сумма (2) = " << brute_force_result << std::endl <<
        "переборы (2): " << brute_force_count << std::endl << std::endl;

    std::vector<std::vector<int>> custom_triangle = triangle;
    int custom_result = max_path_sum_custom(custom_triangle);
    std::cout << "максимальная сумма (3) = " << custom_result << std::endl <<
        "переборы (3): " << custom_count << std::endl << std::endl;

    std::vector<std::vector<int>> top_down_triangle = triangle;
    int top_down_result = max_path_sum_top_down(top_down_triangle);
    std::cout << "максимальная сумма (4) = " << top_down_result << std::endl <<
        "переборы (4): " << top_down_count << std::endl << std::endl;

    std::cout << "\n\nрезы: " << dp_result << " " << brute_force_result << " "
        << custom_result << " " << top_down_result << std::endl;

    std::cout << "\nпереборы: " << "динамическое программирование (снизу вверх):" << dp_count
        << " итераций; метод грубой силы - рекурсивный brute_force:" << brute_force_count
        << " итераций;  максимум из четырех возможных чисел (соседей на уровне и следующем уровне)"
        << custom_count << " итераций; " << "сверху вниз - с вершины треугольника итеративно добавляем значения к двум ближайшим элементам на след уровне"
        << top_down_count << " итераций";

    return 0;
}
```





Тестирование

```
dp: выбор между 1 и 5 для элемента 2, выбираем 5

dp: выбор между 5 и 2 для элемента 7, выбираем 5

dp: выбор между 2 и 6 для элемента 4, выбираем 6

dp: выбор между 6 и 5 для элемента 4, выбираем 6

dp: выбор между 7 и 12 для элемента 8, выбираем 12

dp: выбор между 12 и 10 для элемента 1, выбираем 12

dp: выбор между 10 и 10 для элемента 0, выбираем 10

dp: выбор между 20 и 13 для элемента 3, выбираем 20

dp: выбор между 13 и 10 для элемента 8, выбираем 13

dp: выбор между 23 и 21 для элемента 7, выбираем 23
максимальная сумма (1) = 30
переборы (1): 10
```

```
метод перебора: находимся на базовом уровне с элементом 1

метод перебора: находимся на базовом уровне с элементом 5
метод перебора: для элемента 2, выбор между 1 (левый) и 5 (правый), выбираем 5

метод перебора: находимся на базовом уровне с элементом 5

метод перебора: находимся на базовом уровне с элементом 2
метод перебора: для элемента 7, выбор между 5 (левый) и 2 (правый), выбираем 5
метод перебора: для элемента 8, выбор между 7 (левый) и 12 (правый), выбираем 12

метод перебора: находимся на базовом уровне с элементом 5

метод перебора: находимся на базовом уровне с элементом 2
метод перебора: для элемента 7, выбор между 5 (левый) и 2 (правый), выбираем 5

метод перебора: находимся на базовом уровне с элементом 2

метод перебора: находимся на базовом уровне с элементом 6
метод перебора: для элемента 4, выбор между 2 (левый) и 6 (правый), выбираем 6
метод перебора: для элемента 1, выбор между 12 (левый) и 10 (правый), выбираем 12
метод перебора: для элемента 3, выбор между 20 (левый) и 13 (правый), выбираем 20

метод перебора: находимся на базовом уровне с элементом 5

метод перебора: находимся на базовом уровне с элементом 2
метод перебора: для элемента 7, выбор между 5 (левый) и 2 (правый), выбираем 5

метод перебора: находимся на базовом уровне с элементом 2

метод перебора: находимся на базовом уровне с элементом 6
метод перебора: для элемента 4, выбор между 2 (левый) и 6 (правый), выбираем 6
метод перебора: для элемента 1, выбор между 12 (левый) и 10 (правый), выбираем 12

метод перебора: находимся на базовом уровне с элементом 2

метод перебора: находимся на базовом уровне с элементом 6
метод перебора: для элемента 4, выбор между 2 (левый) и 6 (правый), выбираем 6
```

```
метод перебора: находимся на базовом уровне с элементом 6

метод перебора: находимся на базовом уровне с элементом 5
метод перебора: для элемента 4, выбор между 6 (левый) и 5 (правый), выбираем 6
метод перебора: для элемента 0, выбор между 10 (левый) и 10 (правый), выбираем 10
метод перебора: для элемента 8, выбор между 13 (левый) и 10 (правый), выбираем 13
метод перебора: для элемента 7, выбор между 23 (левый) и 21 (правый), выбираем 23
максимальная сумма (2) = 30
переборы (2): 31
```

```
для элемента 2, рассматриваются варианты: 0 (слева на том же уровне), 7 (справа на том же уровне), 1 (слева на уровне ниже), 5 (справа на уровне ниже), выбираем 7
для элемента 7, рассматриваются варианты: 9 (слева на том же уровне), 4 (справа на том же уровне), 5 (слева на уровне ниже), 2 (справа на уровне ниже), выбираем 9
для элемента 4, рассматриваются варианты: 16 (слева на том же уровне), 4 (справа на том же уровне), 2 (слева на уровне ниже), 6 (справа на уровне ниже), выбираем 16
для элемента 4, рассматриваются варианты: 20 (слева на том же уровне), 0 (справа на том же уровне), 6 (слева на уровне ниже), 5 (справа на уровне ниже), выбираем 20
для элемента 8, рассматриваются варианты: 0 (слева на том же уровне), 1 (справа на том же уровне), 9 (слева на уровне ниже), 16 (справа на уровне ниже), выбираем 16
для элемента 1, рассматриваются варианты: 24 (слева на том же уровне), 0 (справа на том же уровне), 16 (слева на уровне ниже), 20 (справа на уровне ниже), выбираем 24
для элемента 0, рассматриваются варианты: 25 (слева на том же уровне), 0 (справа на том же уровне), 20 (слева на уровне ниже), 24 (справа на уровне ниже), выбираем 25
для элемента 3, рассматриваются варианты: 0 (слева на том же уровне), 8 (справа на том же уровне), 24 (слева на уровне ниже), 25 (справа на уровне ниже), выбираем 25
для элемента 8, рассматриваются варианты: 28 (слева на том же уровне), 0 (справа на том же уровне), 25 (слева на уровне ниже), 25 (справа на уровне ниже), выбираем 28
для элемента 7, рассматриваются варианты: 0 (слева на том же уровне), 0 (справа на том же уровне), 28 (слева на уровне ниже), 36 (справа на уровне ниже), выбираем 36
максимальная сумма (3) = 43
переборы (3): 10
```

```
метод сверху вниз: для узла [0][0] (значение: 7), выбор между 10 (левый узел: [1][0]) и 15 (правый узел: [1][1]), выбираем 15
метод сверху вниз: для узла [1][0] (значение: 3), выбор между 18 (левый узел: [2][0]) и 11 (правый узел: [2][1]), выбираем 18
метод сверху вниз: для узла [1][1] (значение: 8), выбор между 16 (левый узел: [2][1]) и 15 (правый узел: [2][2]), выбираем 16
метод сверху вниз: для узла [2][0] (значение: 8), выбор между 20 (левый узел: [3][0]) и 25 (правый узел: [3][1]), выбираем 25
метод сверху вниз: для узла [2][1] (значение: 1), выбор между 23 (левый узел: [3][1]) и 20 (правый узел: [3][2]), выбираем 23
метод сверху вниз: для узла [2][2] (значение: 0), выбор между 19 (левый узел: [3][2]) и 19 (правый узел: [3][3]), выбираем 19
метод сверху вниз: для узла [3][0] (значение: 2), выбор между 21 (левый узел: [4][0]) и 25 (правый узел: [4][1]), выбираем 25
метод сверху вниз: для узла [3][1] (значение: 7), выбор между 30 (левый узел: [4][1]) и 27 (правый узел: [4][2]), выбираем 30
метод сверху вниз: для узла [3][2] (значение: 4), выбор между 22 (левый узел: [4][2]) и 26 (правый узел: [4][3]), выбираем 26
метод сверху вниз: для узла [3][3] (значение: 4), выбор между 25 (левый узел: [4][3]) и 24 (правый узел: [4][4]), выбираем 25
максимальная сумма (4) = 30
переборы (4): 10
```

```
результаты:
динамическое программирование (снизу вверх): сумма чисел = 30
метод грубой силы - рекурсивный brute_force: сумма чисел = 30
максимум из четырех возможных чисел (соседей на уровне и следующем уровне): сумма чисел = 43
сверху вниз - с вершины треугольника итеративно добавляем значения к двум ближайшим элементам на след уровне: сумма чисел = 30


переборы:
динамическое программирование (снизу вверх): 10 итераций
метод грубой силы - рекурсивный brute_force: 31 итераций
максимум из четырех возможных чисел (соседей на уровне и следующем уровне): 10 итераций
сверху вниз - с вершины треугольника итеративно добавляем значения к двум ближайшим элементам на след уровне: 10 итераций
```

ВЫВОД

Разработал алгоритм решения задачи с применением метода, указанного в варианте и реализовал программу.