Московский авиационный институт (национальный исследовательский университет)

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

Лабораторная работа по предмету "Дискретный анализ" №8

Студент: Кострюков Е.С.

Преподаватель: Макаров Н.К.

Группа: М8О-307Б-22

Дата:

Оценка:

Подпись:

Москва 2024 г.

Оглавление

Цель работы

Заданы длины N отрезков, необходимо выбрать три таких отрезка, которые образовывали бы треугольник с максимальной площадью.

Постановка задачи

Формат ввода

На первой строке находится число N, за которым следует N строк с целыми числами-длинами отрезков.

Формат вывода

Если никакого треугольника из заданных отрезков составить нельзя – 0, в противном случае на первой строке площадь треугольника с тремя знаками после запятой, на второй строке – длины трёх отрезков, составляющих этот треугольник. Длины должны быть отсортированы.

Общий алгоритм решения

1. Ввод данных

- Читается количество отрезков *N* и их длины.
- Одновременно находится максимальная длина отрезка *maxLength*, чтобы оптимизировать сортировку.

2. Сортировка длин

- Используется сортировка подсчётом для упорядочивания длин отрезков по убыванию.
- Сортировка подсчётом подходит, так как длины — целые числа, и диапазон значений известен (от *1* до *maxLength*).

3. Поиск треугольника с максимальной площадью

- Проверяются все подряд идущие тройки длин в отсортированном массиве.
- Для каждой тройки отрезков проверяется неравенство треугольника: $a < b + c$, где *a, b, c* - длины сторон.
- Если тройка удовлетворяет неравенству, вычисляется площадь треугольника по формуле Герона:

$S = \sqrt{p(p-a)(p-b)(p-c)}$ где p - полупериметр треугольника: $p = \frac{a+b+c}{2}$

- Если площадь больше текущего максимума, тройка сохраняется как "лучший треугольник", а её площадь — как максимальная.

4. Вывод результата

- Если найден хотя бы один треугольник, выводятся его максимальная площадь с точностью до трёх знаков и длины его сторон в порядке возрастания.
- Если треугольник составить невозможно, выводится *0*.

Сложность алгоритма

1. Сортировка подсчётом: O(*N* + *maxLength*).
2. Поиск треугольника: O(*N*).

Всего O(*N* + *maxLength*), что эффективно, если *maxLength* умеренно велико.

Реализация

```
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>
#include <algorithm>

using namespace std;

// Функция для вычисления площади треугольника по формуле Герона
double calculateArea(double a, double b, double c) {
    double p = (a + b + c) / 2; // полупериметр
    return sqrt(p * (p - a) * (p - b) * (p - c));
}

// Функция для выполнения сортировки подсчётом
vector<int> countingSort(const vector<int>& lengths, int maxLength) {
    vector<int> freq(maxLength + 1, 0);

    // Подсчёт частот
    for (int length : lengths) {
        freq[length]++;
    }

    // Формирование отсортированного массива
    vector<int> sortedLengths;
    for (int i = maxLength; i >= 1; i--) {
        while (freq[i] > 0) {
            sortedLengths.push_back(i);
            freq[i]--;
        }
    }

    return sortedLengths;
}

// Функция для поиска треугольника с максимальной площадью
pair<double, vector<int>> findMaxTriangle(const vector<int>&
sortedLengths) {
    double maxArea = 0;
    vector<int> bestTriangle;
```

```
    for (int i = 0; i < (int)sortedLengths.size() - 2; i++) {
        int a = sortedLengths[i];
        int b = sortedLengths[i + 1];
        int c = sortedLengths[i + 2];
        if (a < b + c) {
            double area = calculateArea(a, b, c);
            if (area > maxArea) {
                maxArea = area;
                bestTriangle = {a, b, c};
            }
        }
    }

    return {maxArea, bestTriangle};
}

// Функция для вывода результата
void printResult(double maxArea, const vector<int>& bestTriangle) {
    if (maxArea > 0) {
        vector<int> sortedTriangle = bestTriangle;
        sort(sortedTriangle.begin(), sortedTriangle.end());
        cout << fixed << setprecision(3) << maxArea << endl;
        for (int side : sortedTriangle) {
            cout << side << " ";
        }
        cout << endl;
    } else {
        cout << 0 << endl; // Невозможно составить треугольник
    }
}

int main() {
    int N;
    cin >> N;

    vector<int> lengths(N);

        int maxLength = 0;
    for (int i = 0; i < N; i++) {
        cin >> lengths[i];
        maxLength = max(maxLength, lengths[i]);
    }

    vector<int> sortedLengths = countingSort(lengths, maxLength);

    auto [maxArea, bestTriangle] = findMaxTriangle(sortedLengths);
    //pair<double, vector<int>> result = findMaxTriangle(sortedLengths);
    //double maxArea = result.first;
    //vector<int> bestTriangle = result.second;

    printResult(maxArea, bestTriangle);
}
```

Пример работы

Ввод:	Вывод:	Ввод:	Вывод:	Ввод:	Вывод:
4 1 2 3 5	0	5 4 2 6 8 1	11.619 4 6 8	6 10 15 3 6 8 20	72.618 10 15 20

Вывод

В результате выполнения данной лабораторной работы я освоил применение жадных алгоритмов для решения задач, связанных с поиском оптимальных решений.

Главное, что я усвоил – это принцип работы жадного подхода, основанный на локальном выборе, который гарантирует глобально оптимальное решение при правильной формулировке задачи. В данной задаче я научился эффективно определять, образуют ли три отрезка треугольник, и находить тройку, обеспечивающую максимальную площадь.

Я понял, как упорядочение входных данных (сортировка) позволяет существенно упростить поиск решения, а также почему достаточно проверять только подряд идущие тройки длин в отсортированном массиве. Это помогает избежать проверки всех возможных комбинаций, что уменьшает временную сложность алгоритма.

Эта лабораторная работа наглядно продемонстрировала силу жадных алгоритмов в задачах оптимизации, их эффективность и простоту реализации при правильной постановке проблемы.