Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу «Дискретный анализ»

Студент: Д. А. Тарпанов
Преподаватель: Н. С. Капралов
Группа: М8О-307Б-19
Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №8

Задача:
Разработать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти.

Реализовать программу на языке C или C++, соответсвующую построенному алгоритму. Формат входных и выходных данных описан в варианте задания.

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

Формат входных данных

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

Формат результата
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.

1 Описание

Требуется реализовать жадный алгоритм. Для решения задачи сохраним все длины в массив и отсортируем его по убыванию. Будем брать самые большие отрезки и проверять, можно ли построить из них треугольник.
Временная сложность будет равна $O(nlogn)$, так как используется сортировка.

2 Исходный код

В файле main.cpp приводтся полное решение задачи.

Листинг main.cpp

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

bool ReverseCompare(int &lhs, int &rhs) {
    return lhs > rhs;
}

bool ValidTriangle (int &a, int &b, int &c) {
    if (a < (b + c) && b < (a + c) && c < (a + b)) {
        return true;
    }
    return false;
}

double Area (int &a, int &b, int &c) {
    double p = 0.5 * (a + b + c);
    return sqrt(p) * sqrt(p - a) * sqrt(p - b) * sqrt(p - c);
}

int main() {
    int n, s = 0, a = 0, b = 0, c = 0;
    std::vector<int> data;
    double maxArea = 0.0, currArea = 0.0;
    std::cin >> n;
    for (int i = 0; i < n; ++i) {
        std::cin >> s;
        data.push_back(s);
    }
    std::sort(data.begin(), data.end(), ReverseCompare);
    for (int i = 1; i < data.size() - 1; ++i) {
        if (data.size() < 3) {
            break;
        }
        if (ValidTriangle(data[i - 1], data[i], data[i + 1])) {
            currArea = Area(data[i - 1], data[i], data[i + 1]);
            if (currArea > maxArea) {
                maxArea= currArea;
                a = data[i + 1];
                b = data[i];
                c = data[i - 1];
            }
```

```
        }
    }
    if (maxArea == 0.0) {
        std::cout << "0\n";
    }
    else {
        printf("%.3f\n", maxArea);
        std::cout << a << ' ' << b << ' ' << c << '\n';
    }
}
```

3 Консоль

```
kng@kng-Legion-Y540-15IRH:~/CLionProjects/DA8LAB/cmake-build-debug ./DA8LAB
4
1
2
3
5
0
```

4 Тест производительности

Для изучения производетельности, сравним время работы программы на тестах разных размеров с $n = 10^5, 10^7$

```
1. $10^5$
/home/kng/CLionProjects/DA8LAB/cmake-build-debug/DA8LAB
10ms
Process finished with exit code 0


2.$10^7$
/home/kng/CLionProjects/DA8LAB/cmake-build-debug/DA8LAB
16ms
Process finished with exit code 0
```

5 Выводы

Во время выполнения работы, я познакомился с ;жадными алгоритмами, посмотрел набор задач, которые можно решать данным типом алгоритмов, написал свой простой жадный алгоритм.

Список литературы