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: Е. С. Пищик
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №8

Задача: Вариант №3.

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

Данный жадный алгоритм основан на расположении длин сторон в порядке убывания и проверке начиная сверху, берется три самых больших стороны считается площадь, если такой треугольник возможен, делаем проверку сравнивая с текущей наибольшей площадью, если значение больше, то запоминаем. Из-за сортировки требует O(n log n) времени.

2 Исходный код

```
main.cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <iomanip>

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

bool ValidTriangle(int const& s_1, int const& s_2, int const& s_3)
{
    if((s_1 < (s_2 + s_3)) && (s_2 < (s_1 + s_3)) && (s_3 < (s_1 + s_2)))
        return true;
    else
        return false;
}

double Area(int const& s_1, int const& s_2, int const& s_3)
{
    double p = 0.5 * (s_1 + s_2 + s_3);
    return sqrt(p) * sqrt(p - s_1) * sqrt(p - s_2) * sqrt(p - s_3);
}

int main()
{
    std::vector<int> data;
    int n = 0, s = 0, s_1 = 0, s_2 = 0, s_3 = 0;
    double max_area = 0.0, cur_area = 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(), CompareFunc);

    for(int i = 1; i < int(data.size() - 1); ++i)
    {
        if(data.size() < 3)
            break;
        if(ValidTriangle(data[i - 1], data[i], data[i + 1]))
        {
            cur_area = Area(data[i - 1], data[i], data[i + 1]);
            if(cur_area > max_area)
            {
```

```
48                  max_area = cur_area;
49                  s_1 = data[i + 1];
50                  s_2 = data[i];
51                  s_3 = data[i - 1];
52              }
53          }
54      }
55
56      if(max_area == 0)
57          std::cout << 0 << '\n';
58      else
59      {
60          printf("%.3f\n", max_area);
61          std::cout << s_1 << ' ' << s_2 << ' ' << s_3 << '\n';
62      }
63      return 0;
64  }
```

3 Консоль

```
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution$ cat test.txt
4
1
2
3
5
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution$ ./solution<test.txt
0
```

4 Тест производительности

Тест состоит из нахождения наибольшей площади для 50000 и 100000 сторон.

```
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution$ ./benchmark
Time for algo with 50000 sides: 0.12 seconds
Time for algo with 100000 sides: 0.29 seconds
```

Алгоритм работает, явно лучше чем наивный алгоритм за O(n**2).

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я познакомился с жадными алгоритмами, посмотрел набор задач, которые можно решать данным видом алгоритмов, написал простой жадный алгоритм по определению наибольшей площади треугольника.

Список литературы

[1] Жадные алгоритмы