Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2
по “Алгоритмам и структурам данных”
Тимус

Выполнил:
Студент группы P3206
Сорокин А.Н.
Преподаватели:
Косяков М.С.
Тараканов Д.С.

Санкт-Петербург
2024

Задача №3 “Медиана на плоскости”
Решение за O(n).

Необходимо найти 2 точки такие, что между ними будут находится равное количество точек. Очевидно, что прямую, удовлетворяющую заданному условию можно провести ИЗ любой точки в какую-то конкретную точку. Таких прямых будет $\frac{n}{2}$ из $\frac{n(n-1)}{2}$ всех существующих прямых. Для удобства (и адекватных вычислений угла) мы берем крайнюю нижнюю (а из крайних нижних – самую левую) точку как начальную и ищем точку, в которую из начальной надо провести прямую.

Прошлое решение сортировало точки по полярному углу и выводило серединную точку. Однако, информация о положении других точек нас не интересует. Необходимо более быстрое решение, которое направлено на нахождение *конкретной точки*.

Уберем нулевую точку, получим массив нечетной длины (из условия становится очевидным) из разных точек (тоже очевидно, смотреть условие), и задача сводится к нахождению k-ой порядковой статистики.

Quickselect – алгоритм, разделяющий массив на 2 части – в одной находятся элементы, меньшие опорного, в другой – большие. Определяется опорный элемент $pivot$ (каким-то образом, обычно берется первый/последний элемент) и далее за линию массив проходится до этого элемента, все меньшие элементы свапая (partition). Затем в зависимости от k выбираются элементы, слева или справа от $pivot$. Если $k = индекс\, от\, pivot$, то алгоритм останавливается.

Quickselect работает в среднем за линию.
Каждую итерацию от массива откусывается примерно половина, поэтому время можно рассчитать как:
$T(n) = n + T\left(\frac{n}{2}\right) = \sum_{i=0}^{\infty} \frac{n}{2^i} = 2n = O(n)?$, по свойству УГП.

Однако, если на вход подается уже отсортированный массив, то Quickselect будет “откусывать” по 1 элементу каждую итерацию, что приведет к:

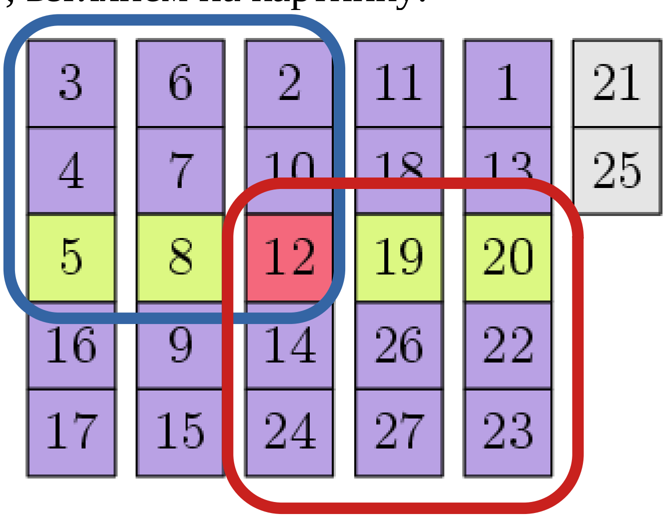
$$T(n)=n+T(n-1)=\sum_{i=0}^{n-1}(n-i)=\frac{n(n+1)}{2}=O(n^2)$$ - сумма АП.

Неудачный выбор опорного элемента может привести к увеличению времени до $O(n^2)$. Следовательно, необходимо на каждой итерации выбирать элемент удачно, чтобы можно было отделять как можно большие участки массива.

Решением данной проблемы является алгоритм **BFPRT** (или **Median of Medians**), позволяющий гарантированно находить удачный опорный элемент.

Массив разделяется на участки по 5 элементов, в каждом из которых находится медиана (например, сортировкой – поскольку размер участка фиксированно меньше либо равен 5, то время нахождения медианы для каждого $O(1)$). Затем, если кол-во медиан больше 1, медиана находится уже среди них. Таким образом, полученная медиана медиан является наиболее удачным опорным элементом.

Чтобы понять это, взглянем на картинку:



Здесь желтым обозначены медианы, а красным – медиана медиан. По картинке можно понять, что если индекс искомого элемента меньше индекса 12, то из массива гарантированно исчезает весь красный квадрат, а если больше – весь синий квадрат. Во входном массиве 27 элементов. Значит, такой выбор опорного элемента гарантированно отсекает чуть более 30% элементов на каждой итерации.

Почему это работает за линию:
Медиана на каждом участке ищется за константу, участков $\frac{n}{5}$. Если медиан больше 1, поиск производится рекурсивно - $T(\frac{n}{5})=T(\frac{2n}{10})$.
Поскольку (гарантированно) удачный выбор опорного элемента уменьшает вектор в худшем случае на 30%, то $T(\frac{7n}{10})$. Так же некоторые операции происходят за линию (например, partition) - Cn .
Тогда имеем:

$$T(n)=T(\frac{2n}{10})+T(\frac{7n}{10})+Cn=Cn+(\frac{2}{10}+\frac{7}{10})Cn+(\frac{2}{10}+\frac{7}{10})(\frac{2}{10}+\frac{7}{10})Cn+...=Cn*\sum_{i=0}^{\infty}(\frac{9}{10})^{i}=10Cn=O(n)$$

Код решения:

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

using namespace std;

struct Point {
  long long x, y, i;
  double a = 0;
};
bool compare(const Point a, const Point b) { return a.a < b.a; }
Point findKMin(vector<Point> v, int l, int r, int k) {
  // BFPRT
  int p1 = l, p2 = r;
  while (p1 < p2) {
    if (p1 >= p2) {
      break;
    }
    int n = p2 - p1 + 1;
```

```
    if (n < 5) {
      sort(v.begin() + p1, v.begin() + p1 + n, compare);
      int m = (p1 * 2 + n) / 2;
      swap(v[p1], v[m]);
      break;
    }
    int p = p1;
    for (int i = p1; i < n / 5; ++i) {
      int p_left = p1 + i * 5;
      int p_right = p1 + i * 5 + 5;
      sort(v.begin() + p_left, v.begin() + p_right, compare);
      int m = (p_left + p_right) / 2;
      swap(v[p], v[m]);
      ++p;
    }
    p2 = p - 1;
  }

  // partition
  int i = r;
  for (int j = r; j >= 1; --j) {
    if (v[j].a < v[l].a) {
      swap(v[j], v[i]);
      --i;
    }
  }
  swap(v[l], v[i]);
  if (i == k) {
    return v[i];
  } else if (i > k) {
    return findKMin(v, l, i - 1, k);
  }
  return findKMin(v, i + 1, r, k);
}

int main() {
```

```
  int n;
  cin >> n;
  long long x, y;
  vector<Point> vp;
  Point downmostPoint = {(long long)10e7, (long long)10e7, 0};
  for (int i = 1; i <= n; ++i) {
    cin >> x >> y;
    vp.push_back({x, y, i});
    if (y < downmostPoint.y) {
      downmostPoint = {x, y, i};
    } else if (y == downmostPoint.y) {
      if (x < downmostPoint.x) {
        downmostPoint = {x, y, i};
      }
    }
  }
  swap(vp.back(), vp[downmostPoint.i - 1]);
  vp.pop_back();
  for (int i = 0; i < vp.size(); ++i) {
    vp[i].x -= downmostPoint.x;
    vp[i].y -= downmostPoint.y;
    vp[i].a = atan2(vp[i].y, vp[i].x);
  }
  Point nextPoint = findKMin(vp, 0, vp.size() - 1, vp.size() / 2);
  cout << downmostPoint.i << " " << nextPoint.i;

  return 0;
}
```

Последние попытки

Автор: [illegible] • Задача: [illegible]

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
[illegible]	04:10:00 8 апр 2024	[illegible]	[illegible]	Clang++ 17 x64	Accepted		0.046	7 144 КБ
[illegible]	21:46:05 7 апр 2024	[illegible]	[illegible]	Clang++ 17 x64	Accepted		0.093	7 032 КБ
[illegible]	21:45:23 7 апр 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	1	0.001	624 КБ
[illegible]	19:31:36 7 апр 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	4	0.015	612 КБ
[illegible]	19:26:39 7 апр 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	4	0.001	620 КБ
[illegible]	19:23:07 7 апр 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	4	0.015	624 КБ
[illegible]	19:22:48 7 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	4	0.015	620 КБ
[illegible]	19:22:22 7 апр 2024	[illegible]	[illegible]	Visual C++ 2022 x64	Wrong answer	4	0.015	112 КБ
[illegible]	19:20:23 7 апр 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	4	0.001	636 КБ
[illegible]	19:13:49 7 апр 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	1	0.001	612 КБ
[illegible]	19:11:48 7 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	1	0.001	620 КБ
[illegible]	19:10:44 7 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	1	0.001	620 КБ
[illegible]	16:26:46 7 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	7	0.001	640 КБ
[illegible]	16:25:52 7 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Runtime error (access violation)	3	0.001	620 КБ
[illegible]	16:20:59 7 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	7	0.015	640 КБ
[illegible]	23:01:42 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Memory limit exceeded	6	0.062	77 364 КБ
[illegible]	22:56:45 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Accepted		0.046	808 КБ
[illegible]	22:52:28 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Runtime error (access violation)	1	0.015	
[illegible]	22:26:28 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Runtime error (access violation)	1	0.001	
[illegible]	22:23:28 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Runtime error (access violation)	6	0.001	388 КБ
[illegible]	22:18:58 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Memory limit exceeded	6	0.078	67 480 КБ
[illegible]	22:16:05 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Runtime error (stack overflow)	6	0.046	65 428 КБ
[illegible]	22:13:32 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Runtime error (stack overflow)	6	0.046	65 440 КБ
[illegible]	21:10:01 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	6	0.001	392 КБ
[illegible]	20:35:18 5 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	26	0.015	400 КБ
[illegible]	15:00:58 4 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Accepted		0.046	692 КБ
[illegible]	14:58:43 4 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Accepted		0.046	692 КБ
[illegible]	14:56:01 4 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Accepted		0.046	680 КБ
[illegible]	13:38:27 4 апр 2024	[illegible]	[illegible]	G++ 13.2 x64	Accepted		0.046	684 КБ
[illegible]	20:55:37 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Accepted		0.031	824 КБ
[illegible]	19:59:30 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Accepted		0.046	808 КБ
[illegible]	19:56:19 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	11	0.001	384 КБ
[illegible]	19:51:41 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	11	0.015	392 КБ
[illegible]	19:29:03 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	3	0.001	392 КБ
[illegible]	19:27:32 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Compilation error			
[illegible]	19:26:41 31 мар 2024	[illegible]	[illegible]	Clang++ 17 x64	Compilation error			
[illegible]	19:13:05 31 мар 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	3	0.001	380 КБ
[illegible]	19:09:05 31 мар 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	3	0.001	380 КБ
[illegible]	19:05:05 31 мар 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	3	0.015	388 КБ
[illegible]	18:38:57 31 мар 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	1	0.001	380 КБ
[illegible]	18:28:23 31 мар 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	2	0.015	384 КБ
[illegible]	18:11:48 31 мар 2024	[illegible]	[illegible]	Clang++ 17 x64	Wrong answer	3	0.001	388 КБ
[illegible]	02:30:09 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	4	0.001	400 КБ
[illegible]	01:56:58 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	4	0.015	392 КБ
[illegible]	01:54:26 31 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	4	0.001	392 КБ
[illegible]	20:35:49 29 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	10	0.015	400 КБ
[illegible]	20:25:22 29 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	10	0.015	404 КБ
[illegible]	18:39:34 18 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	11	0.001	400 КБ
[illegible]	18:27:12 18 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	11	0.015	392 КБ
[illegible]	18:01:15 18 мар 2024	[illegible]	[illegible]	G++ 13.2 x64	Wrong answer	2	0.001	388 КБ

[illegible]

Задача №4 “В Стране Дураков”
Решение за O(n log n).

Задача сводится к построению последовательности такой, что никакие два знака одинакового типа не стоят рядом (если это возможно).
Рассмотрим случаи, когда это невозможно:

- если $k=1$
- если $\left|\left(\sum_{i=1}^{k} n_i - 2*max\left(\{n_i\}_{i=1}^{k}\right)\right)\right|>1$. Чтобы понять данную запись, рассмотрим набор {1, 2, 3} => {4, 2, 2}. Поскольку $8-2*4=0$, их можно расставить как 1 2 1 3 1 2 1 3. Если рассмотреть {1, 2, 3} => {5, 2, 1}, то $8-2*5=-2$ и 1 2 1 2 1 3 1 1.

Как следует расставлять числа.
Очевидно, что массив необходимо отсортировать в порядке убывания и рассматривать максимальные элементы. Пусть $\{n_1, n_2, \ldots n_k\}$, где $n_1 \geq n_2 \geq \ldots \geq n_k$. Тогда необходимо начинать формировать последовательность, чередуя a_1 и a_2 , затем, когда закончится a_2 , выбрать a_3 , затем выбрать a_4 и так далее, чередуя элементы только 2х типов.
Однако данный подход не работает.
Рассмотрим пример:
3
3 5 4
После сортировки имеем: {{2 : 5}, {3 : 4}, {1 : 3}}
Чередуем первые 2 элемента
2 3 2 3 2 3 2 3
Получаем {{2 : 1}, {3 : 0}, {1 : 3}}
Выбираем последний элемент
2 3 2 3 2 3 2 3 2 1
Остается {{2 : 0}, {3 : 0}, {1 : 2}}, и 2 элемента, которые некуда деть.

Данный подход провалился, поскольку в конце реализации всегда будет оставаться $a_k \geq 1$ элемент. Это вызвано тем, что количество знаков разных типо уменьшается неравномерно, т.е на какой-то из итераций $a_i \gg a_{i-2} > a_{i-1}$.

Для решения данной задачи необходимо уменьшать количество знаков разных типов равномерно, то есть на каждой итерации уменьшать на 1 конкретно 2 типа с максимальным количеством знаков (первый и второй максимумы). Для этого на каждой итерации необходимо определять новые 1 и 2 максимумы.
Продемонстрирую алгоритм решения на прошлом примере:
3
3 5 4
После сортировки имеем: {{2 : 5}, {3 : 4}, {1 : 3}}
currentFirstMax = {2 : 5}, currentSecondMax = {3 : 4}
2 3
currentFirstMax = {2 : 4}, currentSecondMax = {1 : 3}
2 3 2 1
currentFirstMax = {2 : 3}, currentSecondMax = {3 : 3}
2 3 2 1 2 3
currentFirstMax = {1 : 2}, currentSecondMax = {2 : 2}
2 3 2 1 2 3 1 2
currentFirstMax = {3 : 2}, currentSecondMax = {1 : 1}
2 3 2 1 2 3 1 2 3 1
currentFirstMax = {2 : 1}, currentSecondMax = {3 : 1}
2 3 2 1 2 3 1 2 3 1 2 3
Получаем {{2 : 0}, {3 : 0}, {1 : 0}}

Теперь встает задача быстрого поиска нового максимума. Поскольку вызывать sort() в векторе на каждой итерации слишком затратно, необходимо использовать другую структуру. priority_queue отлично справляется с задачей: из-за особенностей реализации очередь предоставляет поиск максимума за O(1) (максимум является корнем дерева), и вставку/удаление элементов за O(log n) (бинарный поиск в дереве).

Код решения:
```
#include <iostream>
#include <queue>
#include <string>
#define P pair<int, int>
```

```
using namespace std;
class Compare {
public:
  bool operator()(P a, P b) {
    if (a.second == b.second) {
      return a.first > b.first;
    }
    return a.second < b.second;
  }
};

int main() {
  int k, n;
  cin >> k;
  priority_queue<P, vector<P>, Compare> pq;
  for (int i = 1; i <= k; ++i) {
    cin >> n;
    pq.push({i, n});
  }
  string res = "";
  while (!pq.empty()) {
    P currentFirstMax = pq.top();
    pq.pop();
    res += to_string(currentFirstMax.first) + " ";
    --currentFirstMax.second;
    if (!pq.empty()) {
      P currentSecondMax = pq.top();
      pq.pop();
      res += to_string(currentSecondMax.first) + " ";
      --currentSecondMax.second;
      if (currentSecondMax.second > 0)
        pq.push(currentSecondMax);
    }
    if (currentFirstMax.second > 0)
      pq.push(currentFirstMax);
```

```
  }
  cout << res;
  return 0;
}
```