федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский УНИВЕРСИТЕТ информационных технологий, механики и оптики

**Факультет программной инженерии и компьютерной техники**

**Кафедра «BT»**

**ЛАБОРАТОРНАЯ РАБОТА № \_2\_**

**ПО ДИСЦИПЛИНЕ «*Алгоритмы и структуры данных*»**

Выполнил(а): Чан Чунг Дык

Группа: P3202

Санкт-Петербург

2019

1207. Median on the Plane

Algorithm :

+ Because we don’t have 3 points in a straight line, so that to divide all points into 2 groups, we can choose any point to make line from him. So i choose the first point. And the second point, I will loop all of other point and check if 2 points were choose accept.

+ To check 2 point accept, we check can they divide others into 2 groups, in which has the same point. With each other point, we check 3 points were choose make a clock wise or not. If it is, we increase count 1. And check count == (n-2)/2 , two point were choose is right.

|  |
| --- |
| #include <iostream>  using namespace std;  const int64\_t maxn = 10000 + 10;  int64\_t n;  struct point64\_t {  int64\_t x;  int64\_t y;  };  point64\_t arr[maxn];  void input() {  cin >> n;  for (int64\_t i = 0; i < n; i++)  cin >> arr[i].x >> arr[i].y;  }  bool clockwise(int64\_t x1, int64\_t y1, int64\_t x2, int64\_t y2, int64\_t x3,  int64\_t y3) {  return ((x2 - x1) \* (y3 - y1) - (y2 - y1) \* (x3 - x1) < 0);  }  void solve() {  int64\_t count = 0;  int64\_t mid = (n - 2) / 2;  for (int64\_t i = 1; i < n; i++) {  count = 0;  for (int64\_t j = 1; j < n && count <= mid; j++)  if (j != i) {  if (clockwise(arr[0].x, arr[0].y, arr[i].x, arr[i].y, arr[j].x,  arr[j].y))  count++;  }  if (count == mid) {  cout << "1 " << i + 1;  return;  }  }  }  int main() {  input();  solve();  } |

## 1604. Country of Fools

Algorithm : Choose in array 2 sign a,b , which has arr[a] + arr[b] maximum. Write down a,b to output following this way : write the bigger first. Example arr[a] > arr[b] => output : a b. then decrease 1 from them and put them again arrays.

Using heap max

|  |
| --- |
| #include <iostream>  #include <vector>  #include <algorithm>  using namespace std;  const int maxn = 10000 + 10;  int k;  struct sign {  int number;  int pos;  };  sign arr[maxn];  struct compare {  bool operator()(sign &a, sign &b) const { return a.number < b.number; }  };  void input() {  cin >> k;  for (int i = 0; i < k; i++) {  cin >> arr[i].number;  arr[i].pos = i + 1;  }  }  int main() {  input();  vector<sign> v(arr, arr + k);  make\_heap(v.begin(), v.end(), compare());  while (v.size() >= 2) {  pop\_heap(v.begin(), v.end(), compare());  sign tmp1 = v.back();  v.pop\_back();  pop\_heap(v.begin(), v.end(), compare());  sign tmp2 = v.back();  v.pop\_back();  cout << tmp1.pos << " " << tmp2.pos << " ";  tmp1.number--;  tmp2.number--;  if (tmp1.number > 0)  v.push\_back(tmp1);  push\_heap(v.begin(), v.end(), compare());  if (tmp2.number > 0)  v.push\_back(tmp2);  push\_heap(v.begin(), v.end(), compare());  }  if (v.size() != 0) {  pop\_heap(v.begin(), v.end(), compare());  sign tmp1 = v.back();  v.pop\_back();  for (int i = 0; i < tmp1.number; i++)  cout << tmp1.pos << " ";  }  } |

## 1726. Visits

## Sort by X. Find how many times we use the edge (arr\_x[i] vs arr\_x[i-1]) => Pre\_result1 ;

## Sort by Y. Find how many times we use the edge (arr\_y[i] vs arr\_y[i-1]) => Pre\_result2;

## Result = 2\*(Pre\_result1 + Pre\_result2) / (n\*(n-1));

|  |
| --- |
| #include <iostream>  #include <cmath>  #include <stdlib.h>  #include <cstdint>  using namespace std;  const int64\_t maxn = 100000 + 10;  int64\_t n;  int64\_t arr\_x[maxn];  int64\_t arr\_y[maxn];  void input() {  cin >> n;  for (int64\_t i = 0; i < n; i++)  cin >> arr\_x[i] >> arr\_y[i];  }  int compareMyType(const void \*a, const void \*b) {  if (\*(int64\_t \*)a < \*(int64\_t \*)b)  return -1;  if (\*(int64\_t \*)a == \*(int64\_t \*)b)  return 0;  if (\*(int64\_t \*)a > \*(int64\_t \*)b)  return 1;  }  void solve() {  int64\_t res = 0;  qsort(arr\_x, n, sizeof(int64\_t), compareMyType);  qsort(arr\_y, n, sizeof(int64\_t), compareMyType);  int64\_t res1 = 0, res2 = 0;  for (int64\_t i = 1; i < n; i++)  res1 += (arr\_x[i] - arr\_x[i - 1]) \* i \* (n - i);  // cout << res1 << endl;  for (int64\_t i = 1; i < n; i++)  res2 += (arr\_y[i] - arr\_y[i - 1]) \* i \* (n - i);  // cout << res2 << endl;  res = 2 \* (res1 + res2) / (n \* (n - 1));  cout << res;  }  int main() {  input();  solve();  } |