**Chủ đề: CHUYÊN ĐỀ HÌNH HỌC**

**Môn: TIN HỌC**

**MỞ ĐẦU**

I. LÝ DO CHỌN ĐỀ TÀI

Trong toán học khi gặp các bài toán hình học ta thường giải quyết nó bằng cách nhìn trực quan, tư duy lôgic còn trong Tin học thì đòi hỏi nhiều vấn đề khó khăn hơn, đầu tiên là phải biết đưa về mô hình toán học, biểu diễn các đối tượng hình học, áp dụng các thuật giải trong Tin học kết hợp với các công thức trong Toán học để xây dựng thành chương trình. Hơn nữa trong Tin học khi giải bài toán hình học cần phải xử lý nhiều đối tượng như là: lựa chọn cấu trúc dữ liệu, biểu diễn các công thức toán học, lựa chọn phương pháp tính toán, cách xử lý sai số trong quá trình tính toán là một trong vấn đề cần hệ thống lại để giúp cho học sinh có cách nhìn nhận và giải quyết bài toán Hình học một cách dễ dàng.

* Các bài toán Hình học trong đề thi học sinh giỏi thường yêu cầu tính toán với độ chính xác rất cao, vì vậy học sinh cần vận dụng nhiều kiến thức mới đáp ứng yêu cầu.
* Tài liệu bồi dưỡng học sinh giỏi về chuyên đề này khá ít và bài tập hướng dẫn giải cụ thể, học sinh khó tìm được nguồn tài liệu tham khảo.

Trong quá trình giảng dạy và bồi dưỡng học sinh giỏi tôi nhận thấy rằng học sinh cần được hệ thống các kiến thức về Hình học, biết cách đưa ra các mô hình toán học từ các yêu cầu bài tập phức tạp. Cần hướng cho học sinh phát hiện ra những mối quan hệ của các bài toán với các quan hệ của các đối tượng Hình học, từ đó có thể áp dụng các kiến thức thông dụng trong Toán học mà các em có thể đã biết để giải.

Với mong muốn phần nào giúp học sinh và giáo viên tìm ra các cách đơn giản để giải một số bài toán Hình học thông dụng trong quá trình học chuyên Tin và Bồi dưỡng học sinh giỏi dự thi các cấp. Tôi lựa chọn và viết chuyên đề: “**Vận dụng các đối tượng trong Hình học phẳng để giải bài toán trong Tin học”.**

**II. MỤC ĐÍCH NGHIÊN CỨU**

Thông qua việc nghiên cứu đề tài giúp tôi và học sinh có thể hiểu sâu sắc hơn các đối tượng hình học, các phương pháp hình học, vận dụng các đối tượng này để giải các bài toán trong Tin học các liên quan đến hình học phẳng như điểm, đoạn thẳng, đường thẳng, đa giác… nhằm bồi dưỡng cho học sinh tham dự học sinh giỏi các cấp.

Đưa ra các bài tập vận dụng cụ thể từng phương pháp trong hình học phẳng, xây dựng các ý tưởng thuật giải chi tiết hướng dẫn học sinh cài đặt chương trình.

**III. ĐỐI TƯỢNG NGHIÊN CỨU**

Các đối tượng Hình học phẳng để giải các bài toán trong Tin học, các cấu trúc dữ liệu biểu diễn các đối tượng, các phương pháp Hình học, thuật toán vận dụng.

**IV. PHƯƠNG PHÁP NGHIÊN CỨU**

**4.1 Phương pháp nghiên cứu lý luận:**

Nghiên cứu tài liệu sách giáo khoa chuyên Tin quyển 3 và các tài liệu về toán học có liên quan đối các đối tượng trong hình học phẳng, các tài liệu xử lý sai số trong quá trình tính toán, Các tài liệu về ngôn ngữ lập trình….

**4.2 Phương pháp nghiên cứu thực tiễn:**

Sử dụng ngôn ngữ lập trình C++, công thức toán học được soạn thảo bằng phần mềm MathType.

Biên soạn các bài tập từ cơ bản đến nâng cao và có hướng dẫn chi tiết hệ thống lại các kiến thức để học sinh dễ theo dõi và tiếp cận.

**V. Ý NGHĨA THỰC TIỄN CỦA ĐỀ TÀI**

Qua nghiên cứu lý luận và thực tiễn dạy học ở trường tôi nhận thấy việc vận dụng các đối tượng trong Hình học phẳng là rất cần thiết để giải các bài toán trong Tin học có liên qua, giúp học sinh bổ sung các kiến thức cần thiết qua đó có thể vận dụng để giải các lớp bài toán về Hình học phẳng, tham dự kì thi học sinh giỏi các cấp.

NỘI DUNG

I. CƠ SỞ LÝ THUYẾT

1. Biểu diễn hình học trên máy tính.

Các đối tượng điểm, đoạn thẳng, đường thẳng là những đối tượng cơ bản nhất khi biểu diễn hình học trong máy tính. Để biểu diễn các đối tượng cơ bản đó thường sử dụng các cách biểu diễn sau đây

### 1. 1. Điểm:

struct point

{ double x, y;};

### 1.2. Đường thẳng:

struct line

{double d1,d2;}

### 1.3. Đa giác:

point Polygon[n];

Để thuận lợi thì khi biểu diễn đa giác ta nên thêm hai đỉnh ở đầu và cuối: đỉnh 0 bằng đỉnh n và đỉnh n + 1 bằng đỉnh 1.

## 2. Một số khái niệm cơ bản khác:

### 2.1. Đường gấp khúc:

Một đường gấp khúc trên mặt phẳng gồm một dãy liên tiếp các đoạn thẳng [p1,p2]; [p2, p3],….[pn-1, pn], mỗi đoạn thẳng được gọi cạnh, các đầu mút của các đoạn thẳng gọi là đỉnh.

### 2.2. Đa giác:

Một đa giác là một đường gấp khúc khép kín điểm pn trùng với điểm p1

### 2.3. Đa giác tự cắt

Một đa giác được gọi là tự cắt nếu có hai cạnh không liên tiếp có điểm chung.

### 2.4. Đa giác lồi:

Một đa giác gọi là đa giác lồi nếu như đa giác luôn nằm cùng một phía đối với đường thẳng đi qua một cạnh bất kì của đa giác. Đa giác lồi là đa giác không tự cắt.

## 3. Một số phương pháp hình học:

Trong các bài toán hình học phần lớn các đối tượng đều được thể hiện trên hệ trục tọa độ Descartes, việc biểu diễn các thành phần tọa độ có thể sử dụng cả kiểu số thực và số nguyên.

Một trong những khó khăn gặp phải khi giải quyết các bài toán hình học là ta phải làm việc với số thực. Khi làm việc với số thực phải chấp nhận những sai số nhất định, vì vậy khi so sánh hai giá trị thì không được dùng dấu “==” mà phải xét trị tuyệt đối hiệu của hai giá trị nhỏ hơn một giá trị Epsilon nào đó. Epsilon là một số tương đối bé.

### 3.1. Tính khoảng cách giữa 2 điểm:

Cho 2 điểm A(xA, yA) và B(xB, yB) khi đó khoảng cách giữa 2 điểm A, B được tính: 

|  |
| --- |
| double Dist(point a, point b)  { return sqrt((b.x-a.x)\*(b.x-a.x)+(b.y-a.y)\*(b.y-a.y));} |

### 3.2. Vị trí tương đối giữa 3 điểm.

Cho 3 điểm A(xA, yA) và B(xB, yB) , C(xC, yC) khi đó có 3 khả năng xảy ra:

A

B

C

A

B

C

*Rẽ phải Rẽ trái Thẳng hàng*

k= 

Nếu k<0 thì rẽ phải

Nếu k>0 thì rẽ trái

Nếu k==0 thì 3 điểm thẳng hàng.

**Cài đặt:**

|  |
| --- |
| int CCW(P p1, P p2, P p3 )  {  float k=(p2.x-p1.x)\*(p3.y-p2.y)-(p3.x-p2.x)\*(p2.y-p1.y);  if (abs(k)<=Eps) return 0;  else  if (k>0) return 1;  else return -1;} |

### 3.3. Phương trình đường thẳng đi qua hai điểm:

Cho 2 điểm A(xA, yA) và B(xB, yB) khi đó phương trình đường thẳng đi qua 2 điểm có dạng: 

Khi cài đặt chương trình để dễ dàng ta có thể biểu diễn dưới dạng tổng quát như sau:

|  |
| --- |
| void line(point p1,point p2)  { float a=p2.y-p1.y;  float b=p1.x-p2.x;  float c=-(a\*p1.x+b\*p1.y);  } |

### 3.4. Diện tích đa giác.

Trong mặt phẳng với tọa độ Descartes cho đa giác p=p1p2…pn là đa giác không tự cắt (điểm pi có tọa độ (xi, yi). Ta bổ sung thêm điểm p0 trùng với pn và điểm pn+1 trùng với điểm p1, khi đó diện tích đa giác được tính bằng công thức sau:



**Cài đặt:**

|  |
| --- |
| void dientich()  { float s;  p[0]=p[n];  p[n+1]=p[1];  for (int i=1; i<=n; i++)  s=s+(p[i+1].x-p[i].x)\*(p[i+1].y+p[i].y);  cout<<setprecision(2)<<fixed<<abs(s/2);} |

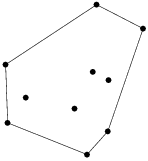
Việc ta lấy trị tuyệt đối của S là có lý do của nó. Nếu đi theo thứ tự p1, p2,…, pn là ngược chiều kim đồng hồ thì ta sẽ có S<0, còn ngược lại ta sẽ nhận được S >0.

Lưu ý: Công thức trên chỉ đúng với đường biên của đa giac là đường gấp khúc khép kín, không tự cắt. Nếu không thoải mãn điều kiện này thì công thức sẽ sai.

Ví dụ: Đa giác gồm 4 các đỉnh lần lượt như sau: (0,0) (1, 1), (0, 1) (1, 0) nếu tính theo công thức trên thì diện tích là 0.00 nhưng thực tế đa giác này có diện tích là 0.5.

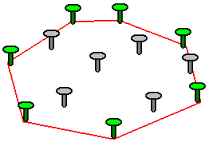
### 3.5. Bao lồi:

Trong **hình học tính toán (computational geometry), bao lồi (convex hull)** của một tập điểm là tập lồi nhỏ nhất (theo diện tích, thể tích, …) mà tất cả các điểm đều nằm trong tập đó.



# Giải thích trực quan về bao lồi trên mặt phẳng

Một cách trực quan, nếu ta coi các điểm trong một tập hợp là các cái đinh đóng trên một tấm gỗ, bao lồi của tập điểm đó có viền ngoài tạo bởi sợi dây chun mắc vào các cái đinh sau khi bị kéo căng về các phía như hình vẽ sau:



# Thuật toán tìm bao lồi trên mặt phẳng

Bài toán tìm bao lồi của một tập điểm trên mặt phẳng là một trong những bài toán được nghiên cứu nhiều nhất trong hình học tính toán và có rất nhiều thuật toán để giải bài toán này. Sau đây là hai thuật toán phổ biến nhất:

## Thuật toán bọc gói

**Thuật toán bọc gói (Gift wrapping algorithm)** hay còn gọi là thuật toán **Jarvis march** là một trong những thuật toán tìm bao lồi đơn giản và dễ hiểu nhất. Tên thuật toán xuất phát từ sự tương tự của thuật toán với việc đi bộ xung quanh các điểm và cầm theo một dải băng gói quà.

* Bước đầu tiên của thuật toán là chọn một điểm chắc chắn nằm trong bao lồi, ví dụ, điểm có tung độ lớn nhất (nếu có nhiều điểm cùng có tung độ lớn nhất thì có thể chọn điểm tung độ lớn nhất và hoành độ lớn nhất).
* Xuất phát từ điểm này, mục tiêu của ta là sẽ lần lượt đi đến các điểm khác cho đến khi quay trở lại điểm ta chọn lúc đầu.
* Ban đầu, ta nhìn về phía bên phải. Khi đi đến các điểm khác, ta sẽ lưu lại:

+ Điểm P mà ta đang chọn.

+ Vector v⃗chỉ hướng ta đang nhìn.

* Tiếp theo, thuật toán sẽ lặp lại liên tục các bước sau cho đến khi tìm được bao lồi.

+Ta quay mặt theo chiều kim đồng hồ cho đến khi ta nhìn thấy một điểm, gọi điểm đó là Q.

+ Rồi ta cầm theo dải băng và đi đến điểm Q. Khi ta đến điểm đấy, ta thay: v⃗ thành PQ và P thành Q.

* Thuật toán kết thúc, khi ta quay trở về điểm ban đầu. Lúc này ta đã đi đến tất cả các đỉnh của bao lồi theo chiều kim đồng hồ.

Để xác định điểm ta nhìn thấy đầu tiên khi ta quay mặt theo chiều kim đồng hồ, ta duyệt tất cả các điểm R trong tập, ngoại trừ điểm P. Với mỗi điểm, ta xét vector  =;  u⃗ tạo với v⃗một góc θ nhỏ nhất sẽ tương ứng với điểm Q. Để tìm θ nhỏ nhất, ta tìm cosθ lớn nhất: 

Với mỗi lần tìm điểm tiếp theo, ta cần duyệt qua tất cả các điểm trong tập, vì vậy độ phức tạp của mỗi lần tìm điểm là O(n) với n là số lượng điểm trong tập. Số lần tìm điểm tiếp theo phụ thuộc vào số lượng điểm là đỉnh của bao lồi, gọi số lượng điểm đó là h, khi đó độ phức tạp của cả thuật toán là O(n\*h). Trong trường hợp xấu nhất, h=n hay tất cả các điểm trong dữ liệu vào tạo thành một đa giác lồi, độ phức tạp của thuật toán là O(n2), không đủ nhanh khi n>5000.

## Thuật toán Graham

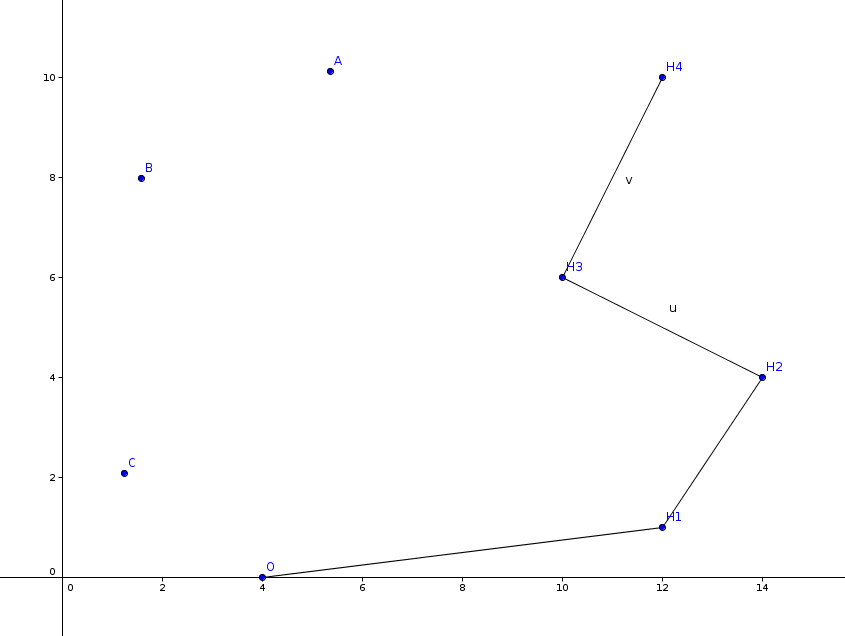
Thuật toán Graham có độ phức tạp trong trường hợp xấu nhất nhỏ hơn thuật toán bọc gói, song thuật toán Graham lại phức tạp hơn.

* Đầu tiên, ta xác định một điểm mà chắc chắn thuộc bao lồi. Thông thường, khi cài đặt người ta chọn điểm có tung độ nhỏ nhất (nếu có nhiều điểm như vậy thì chọn điểm trái nhất). Gọi điểm này là điểm O.
* Chọn hệ trục tọa độ có gốc là điểm vừa chọn, đổi các tọa độ các điểm còn lại theo hệ trục tọa độ mới (chú ý lúc cài đặt thường ta không đổi trục tọa độ, nhưng khi tính góc hoặc sắp xếp ở bước tiếp theo cần chú ý tránh nhầm lẫn).
* Tiếp theo, ta sắp xếp các điểm còn lại theo thứ tự tăng dần của góc tạo bởi trục hoành theo chiều dương và  với I trong các điểm còn lại.
* Ta xét các điểm theo thứ tự ta vừa sắp xếp, với mỗi điểm ta sửa lại bao lồi H. Gọi điểm đầu tiên được cho vào bao lồi H là H1, điểm cuối cùng là Hh(ban đầu h=0). Khi xét mỗi điểm ta làm như sau:

1. Thêm điểm P vào cuối bao lồi H. Tức là ta tăng h lên 1 và đặt Hh=P.

2. Nếu h<3 xét tiếp điểm tiếp theo, ngược lại làm bước 3.

3. Xét 3 điểm Hh,Hh−1, và H h−2. Có thể sau khi cho thêm điểm Hh, ta biết được điểm Hh−-1 chắc chắn không nằm trong bao. Gọi  và. Nếu khi đi theo hướng  rồi đi theo hướng  là ta đã bẻ góc ngược chiều kim đồng hồ, hay u⃗ ×v⃗ >0, thì cả ba điểm đều tạm thuộc bao, và ta xét tiếp điểm tiếp theo. Nhưng nếu u⃗ ×v⃗ <0 thì góc tạo bởi   và   sẽ tạo ra đa giác lõm và điểm Hh−1phải bị loại bỏ, có nghĩa là Hh−1được đặt là Hh và h giảm đi 1. Sau đó quay lại bước 2 cho đến khi xét hết các điểm.

 Hình minh họa:

* Ta đang xây dựng bao lồi, đến vị trí h=4
* Góc tạo bởi  và  lõm, nên ta cần bỏ điểm H3 khỏi bao lồi.

Sau quá trình trên, ta đã có một bao lồi H1,H2,…,Hh sắp xếp ngược chiều kim đồng hồ.

Để đảm bảo ta loại bỏ điểm và thêm điểm với độ phức tạp O(1) ta có thể dùng cấu trúc dữ liệu [stack](https://vnoi.info/wiki/algo/data-structures/data-structures-overview#1-ctdl-l%C6%B0u-tr%E1%BB%AF_1-2-stack-queue-deque).

Về độ phức tạp của thuật toán, ta thấy bước sắp xếp các điểm có độ phức tạp O(nlogn). Mỗi điểm được cho vào bao nhiều nhất một lần nên tổng độ phức tạp của các bước thêm điểm là O(n), và mỗi điểm bị loại ra khỏi bao nhiều nhất một lần nên tổng độ phức tạp của các bước xóa điểm là O(n), do đó độ phức tạp của bước xét các điểm là O(n). Vậy, độ phức tạp của thuật toán Graham là O(nlogn) phù hợp cho hầu hết các bài toán.

**Cài đặt:**

|  |
| --- |
| struct point  { int x,y;};  point a[10005],h[10005];  int ccw(point a, point b, point c  { int k=(b.x-a.x)\*(c.y-b.y)-(b.y-a.y)\*(c.x-b.x);  return k;  }  bool cmp(point a,point b)  {  if (a.y==b.y) return a.x<b.x;  return a.y<b.y;  }  bool cmp2(point d, point b)  { int t=ccw(a[1],d,b);  if (t==0) return d.x<b.x;  return (t>0);  }  void graham()  { sort(a+1,a+n+1,cmp);  sort(a+2,a+n+1,cmp2);  h[1]=a[1] ; h[2]=a[2] ;  m=2 ;  a[n+1]=a[1];  for (int i=3 ; i<=n+1 ; i++)  { while(m>=2 and ccw(h[m-1],h[m],a[i])<=0) m--;  m++;h[m]=a[i]; }  m--;} |

3.6. Tìm khoảng cách cặp điểm gần nhất:

Cho tập điểm, tìm cặp điểm có khoảng cách nhỏ nhất trong tập điểm trên.

Khoảng cách giữa hai điểm được tính theo công thức như sau:



***Hướng dẫn:***

Một cách làm đơn giản ta có thể xét tất cả các cặp điểm và lưu lại cặp điểm có khoảng cách nhỏ nhất. Nhưng như vậy thì chi phí thuật toán sẽ là O(N2). Chúng ta hoàn toàn có thể giải quyết bài toán này với chi phí là O(NlogN) với việc áp dụng tư tưởng “chia để trị”.

Bước 1: Chia N điểm cần xử lý thành 2 tập hợp bằng một đường thẳng đứng.

Bước 2: Gọi đệ quy tìm cặp điểm gần nhất cho tập bên trái và tập bên phải.

Bước 3: Tìm khoảng cách ngắn nhất giữa một điểm thuộc tập bên trái và một điểm thuộc tập bên phải.

Bước 3 là bước khó nhất trong các bước trên.

Giả sử trong bước 1 ta chia N điểm thành 2 tập bằng đường thẳng có phương trình x=a và trong bước 2, sau khi gọi đệ quy có được khoảng cách giữa cặp điểm gần nhất là D. Một cách tự nhiên, ta thấy trong bước 3 chỉ cần xét những điểm có ∣xi−a∣<D vì các điểm bên ngoài khoảng đó chắc chắn sẽ cho khoảng cách lớn hơn D.

Ngoài ra, với mỗi điểm pi thỏa điều kiện ở trên, ta chỉ cần so nó với những điểm pj cũng thỏa điều kiện ở trên và pj có yi−D<yj≤yi (\*).

Để thực hiện nhanh bước 1 ta cần sắp xếp các điểm theo x, còn bước 3 cần phải sắp các điểm theo y. Để giải quyết vấn đề này, ta sắp các điểm theo x ngay từ đầu, rồi lồng việc sắp xếp theo y vào đệ quy bằng sắp xếp trộn.

Phương pháp này có độ phức tạp O(NlogN).

**Ví dụ minh họa:**

**Đề bài:**

Cho N (2 <= N <= 105) điểm trên mặt phẳng, hãy tìm cặp điểm có khoảng cách nhỏ nhất.

**Dữ liệu vào: Từ tệp NEAR.INP**

- Dòng đầu tiên chứa số N.

- N dòng tiếp theo mỗi dòng chứa một cặp số thực (giá trị tuyệt đối không lớn hơn 107) biểu diễn tọa độ một điểm.

**Dữ liệu ra: ghi vào tệp NEAR.OUT**

Một số duy nhất (ghi chính xác đến 3 chữ số thập phân sau dấu phẩy) là khoảng cách nhỏ nhất tìm được.

|  |  |
| --- | --- |
| **NEAR.INP** | **NEAR.OUT** |
| 5  1 1  2 2  3 3  4 4  5 5 | 1.414 |

|  |
| --- |
| #include <bits/stdc++.h>  #define MAXN 100000  using namespace std;  struct point {  double x, y;};  point a[MAXN];  double mindist;  point t[MAXN];  bool cmp1( point a, point b) {  return a.x < b.x;}  bool cmp2( point a, point b) {  return a.y < b.y;}  void kcmin( point a, point b) {  double dist = sqrt((a.x-b.x)\*(a.x-b.x) + (a.y-b.y)\*(a.y-b.y));  if (dist < mindist) mindist = dist;  }  void find(int l, int r) {  if (r <= l) return;  if (r == l + 1) {  kcmin(a[l], a[r]);  if (!cmp2(a[l], a[r])) swap(a[l], a[r]);  return;}  int m = (l + r) / 2;  double midx = a[m].x;  find(l, m);  find(m+1, r);  merge(a+l, a+m+1, a+m+1, a+r+1, t, cmp2);  copy(t, t+r-l+1, a+l);  int tm = 0;  for (int i=l; i<=r; i++) if (abs(a[i].x-midx) < mindist)  {  for (int j=tm-1; j>=0 && t[j].y > a[i].y-mindist; j--)  kcmin(a[i], t[j]);  t[tm++] = a[i];  }  }  int main() {  ios::sync\_with\_stdio(false); cin.tie(0);  //freopen("near.inp", "r", stdin);  ///freopen("near.out", "w", stdout);  int n; cin >> n;  for (int i=0; i<n; i++) cin >> a[i].x >> a[i].y;  mindist = 1e20;  sort(a, a+n, cmp1);  find(0, n-1);  cout<<setprecision(3)<<fixed<<mindist;  return 0;  } |

***Nguồn đề bài+test****:* [*https://vn.spoj.com/problems/NEAREST/*](https://vn.spoj.com/problems/NEAREST/)

***Code****:*[*https://drive.google.com/file/d/1l7Gpe68iPRAfqfhvRDL1rTXLMGvt29NJ/view?usp=sharing*](https://drive.google.com/file/d/1l7Gpe68iPRAfqfhvRDL1rTXLMGvt29NJ/view?usp=sharing)

**II. MỘT SỐ BÀI TẬP VẬN DỤNG**

**Bài 1. Du lịch**

Trên một tỉnh XYZ có N địa điểm du lịch. Có thể coi là N điểm trên mặt phẳng tọa độ, mỗi điểm du lịch có tọa độ xi, yi. Chủ tịch hội đồng quản trị công ty ABC cần đặt bến tại một trong N điểm du lịch sao cho tổng khoảng cách từ điểm đó đến các điểm du lịch còn lại là nhỏ nhất. Coi khoảng cách giữa hai điểm du lịch chính bằng khoảng cách giữa hai điểm trên mặt phẳng tọa độ. Nếu có nhiều điểm như vậy bạn hãy chọn điểm có số hiệu nhỏ nhất.

**Dữ liệu vào: từ tệp** DULICH.INP

* Dòng 1 ghi n (n ≤100)
* N dòng tiếp theo, dòng thứ i ghi hai số nguyên xi, yi thể hiện tọa độ của một

điểm.

**Dữ liệu ra: Ghi vào tệp DULICH.OUT**

Dòng duy nhất ghi hai số, số đầu là số hiệu của điểm tìm được và số thứ hai là số thực thể hiện tổng khoảng cách đến các điểm còn lại (3 chữ số thập phân)

|  |  |
| --- | --- |
| **DULICH.INP** | **DULICH.OUT** |
| 1  6 29 | 1 0.000 |

# *Hướng dẫn:*

Bài này ta vận dụng công thức tính khoảng cách giữa 2 điểm trên mặt phẳng. Vì n<100 Nên ta có thể tỉnh khoảng cách của tất cả điểm đề cho.

Với mỗi điểm du lịch ta tính tổng khoảng cách đến n-1 điểm còn lại. (Dùng công thức tính khoảng cách giữa 2 điểm). Trong các tổng khoảng cách tính được đưa ra tổng khoảng cách nhỏ nhất và vị trí của điểm có tổng khoảng cách nhỏ nhất.

Công thức áp dụng:

Khoảng cách giữa hai điểm được tính theo công thức như sau:



Độ phức tạp thuật toán: O(N2)

**Cài đặt:**

|  |
| --- |
| #include <bits/stdc++.h>  # define ll long long  using namespace std;  struct point  {  ll x,y;  };  point a[1001]; double res; ll n, vt;  double dist(point a, point b)  { return sqrt( (a.x - b.x)\*(a.x - b.x) + (a.y - b.y) \* (a.y - b.y) );}  int main()  {  //freopen("DULICH.INP","r",stdin);  //freopen("DULICH.OUT","w", stdout);  cin >> n ;  for(int i=1;i<=n;i++)  cin >> a[i].x >> a[i].y; res=1e12 ;vt=0;  for(int i=1;i<=n;i++) // xet cac diem i  { double s=0;  for(int j=1;j<=n;j++) // duyet tat ca cac diem  if (i != j) s+= dist(a[i],a[j]);  if (s < res) res= s, vt = i;  }  cout << vt << " ";  cout << fixed<< setprecision(3) << res;  return 0;} |

*Test:* [*https://drive.google.com/file/d/1\_WFBX3fa2mqH5iYBr00qeh-dtermXExq/view?usp=sharing*](https://drive.google.com/file/d/1_WFBX3fa2mqH5iYBr00qeh-dtermXExq/view?usp=sharing)

*Code:*[*https://drive.google.com/file/d/1AH5Wfy0V9VWnEU20s7PCM0XdBw5ugTCU/view?usp=sharing*](https://drive.google.com/file/d/1AH5Wfy0V9VWnEU20s7PCM0XdBw5ugTCU/view?usp=sharing)

**Bài 2: Đa giác:**

Có N điểm trên mặt phẳng với tọa độ là các số tự nhiên. Một đa giác lồi nhiều đỉnh nhất là một đa giác lồi có các đỉnh là gốc tọa độ và một số đỉnh trong các điểm đã cho, và có số đỉnh là nhiều nhất. Điểm gốc, nghĩa là điểm có tọa độ O(0, 0) phải là một trong các đỉnh của đa giác lồi nhiều đỉnh nhất .

Yêu cầu: Viết chương trình xác định số đỉnh của đa giác này.

Một đa giác là lồi nếu mọi đoạn thẳng có đầu mút nằm trong đa giác đều nằm hoàn toàn trong đa giác đó.

Các cạnh liên tiếp của một đa giác không được phép song song với nhau.

#### Dữ liệu vào: Từ dagiac.inp

* Dòng đầu tiên chứa số tự nhiên N (2≤N≤100) là số điểm được cho.
* Mỗi dòng trong số N dòng tiếp theo chứa 2 số tự nhiên X, Y

(1≤X≤100, 1≤Y≤100) cách nhau bởi khoảng trắng, cho biết tọa độ của một điểm. Các điểm đều phân biệt nhau.

#### Dữ liệu ra: Ghi vào tệp dagiac.out

* In ra một số nguyên duy nhất là số đỉnh của đa giác lồi nhiều đỉnh nhất .

Lưu ý: kết quả luôn không nhỏ hơn 3.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  4 2  2 2  2 3  3 2  3 1 | 4 |
| 8  10 8  3 9  2 8  2 3  9 2  9 10  10 3  8 10 | 8 |
| 10  9 6  1 7  2 2  3 9  8 7  3 2  9 4  3 1  9 7  6 9 | 7 |

***Hướng dẫn:***

Ta dựa vào công thức tính diện tích đa giác lồi khi các đỉnh được đánh thứ tự ngược chiều kim đồng hồ thì s<0.

Sắp xếp các đỉnh theo thứ tự tăng dần tỷ số (x là hoành độ, y là tung độ).

Gọi F[i][j] là số đỉnh của đa giác khi xét đến điểm thứ i, j. (j<i)

Xét theo chiều ngược kim đồng hồ nếu 3 điểm i, j và k tạo thành đa giác có s<0 thì cập nhật số đỉnh tối đa của đa giác:

Công thức quy hoạch động: F[i][j]=max(F[i][j], F[j][k]+1).

Kết quả bài toán số đỉnh tối đa của đa giác là max(F[i][j]).

Độ phức tạp của thuật toán: O(N3)

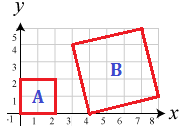
|  |
| --- |
| #include <bits/stdc++.h>  # define ll long long  using namespace std;  ll n, f[105][105];  struct point {  int x, y;  };  point a[105];  void nhap()  { cin>>n;  for (int i=1; i<=n; i++)  {  cin>>a[i].x>>a[i].y;  }  }  bool check(point i, point j, point k)  {  int z= (i.x-j.x)\*(i.y+j.y)+(j.x-k.x)\*(j.y+k.y)+(k.x-i.x)\*(k.y+i.y);  if (z<0) return true;  return false;  }  void xuly()  { for (int i=1; i<=n-1; i++)  for (int j=i+1; j<=n; j++)  {  if (a[j].x\*a[i].y>a[i].x\*a[j].y)  {  swap(a[i].x, a[j].x);  swap(a[i].y, a[j].y);  }  }  for (int i=2; i<=n; i++)  for (int j=1; j<=i-1; j++)  { f[i][j]=3;  for (int k=1; k<=j-1; k++)  if (check(a[i], a[j], a[k]))  f[i][j]=max(f[i][j], f[j][k]+1);  }  ll kq=3;  for (int i=2; i<=n;i++)  for (int j=1; j<=i-1; j++)  kq=max(kq, f[i][j]);  cout<<kq;  }  int main(){  //freopen("dagiac.inp", "r", stdin);  // freopen("dagiac.out", "w", stdout);  nhap();  xuly();  return 0;  } |

***Test:*** [*https://vn.spoj.com/problems/NKPOLI/*](https://vn.spoj.com/problems/NKPOLI/)

***Code:***<https://drive.google.com/file/d/1FFFJjEBWn6KBEDhVvQN5pLm0U2cywBK/view?usp=sharing>

**Bài 3**: **Đếm hình vuông**

Trong mặt phẳng tọa độ, cho một lưới các điểm nguyên có kích thước n+1 dòng và m+1 cột. Điều đó có nghĩa là nếu (x, y) là một điểm nguyên trong lưới thì  
 0 ≤ x ≤ m và 0 ≤ y ≤ n. Người ta có thể chọn 4 điểm nào đó trong lưới để tạo thành các hình vuông như hình bên dưới (n=5 và m=8).



Trong số các hình vuông này, diện tích của một số hình vuông là lẻ (diện tích của B là 17), hình vuông còn lại có diện tích là chẵn (diện tích của A là 4). Hãy đếm xem tổng cộng có bao nhiêu hình vuông có diện tích lẻ.

**Dữ liệu vào: Cho bởi file văn bản HV.INP**

- Là hai số nguyên n và m cách nhau một khoảng trắng (1 ≤ n, m ≤ 105)

**Dữ liệu ra: Ghi vào tập văn bản HV.OUT**

- Là số lượng số hình vuông có diện tích lẻ. Kết quả đảm bảo số lượng này là số nguyên 64 bít.

**Ví dụ:**

|  |  |
| --- | --- |
| **HV.INP** | **HV.OUT** |
| 1 2 | 2 |
| 3 3 | 12 |

***Hướng dẫn thuật toán:***

- *Trường hợp 1:* Xét hình vuông có cạnh song song đường chéo chính:

Gọi i là độ dài hình vuông🡺số điểm trên lưới có độ dài i là: L=i+1 (điểm).

Hình vuông có độ dài i có thể di chuyển trên lướt theo (n+1-i)(m+1-i) vị trí khác nhau. 🡺 Số lượng hình vuông có cạnh song song trục tọa độ: (n+1-i)(m+1-i).

* *Trường hợp 2*: Xét hình vuông có cạnh song song với đường chéo chính.

Nếu lướt vuông ban đầu có L điểm mỗi cạnh thì sẽ có L-2 hình vuông🡺 Số lượng hình vuông có độ dài i là: i-1 (vì L=i+1). Các hình vuông này có thể di chuyển trên lưới theo (n+1-i)(m+1-i) vị trí.

🡺Số lượng hình vuông có cạnh song song đường chéo chinh là: (i-1)(n+1-i)(m+1-i)



- Vì diện tích hình vuông cần tìm là 1 số lẻ🡺Hình vuông được xét phải có độ dài là một số lẻ🡺 i lẻ:

Vậy số hình vuông có diện tích lẻ là:

**

** *Với i lẻ*

Độ phức tạp thuật toán: O(min(n,m))

***Cài đặt:***

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  long long n, m;  void xuly()  { long long kq=0;  int k=min(n, m);  for (int i=1; i<=k; i++)  if (i%2==1) kq=kq+i\*(m-i+1)\*(n-i+1);  cout<<kq;  }  int main(){  cin>>n>>m; xuly();  return 0;} |

***Test*:**https://drive.google.com/file/d/1JOgLcnzz2EvLMqaNCfyNBL6i35kssVOp/view?usp=sharing

***Code:***<https://drive.google.com/file/d/1oS470cVSZPlN8w1bJiFQQi2o_QC14sTz/view?usp=sharing>

## Bài 4. Đường cao tốc

Trong thành phố X có N khu chung cư, khu chung cư thứ i có tọa độ xi, yi. Lãnh đạo thành phố X rất quan tâm đến N khu chung cư này và có ý định xây một đường cao tốc song song với trục hoành. Khi đó chắc chắn từ N khu chung cư sẽ phải làm thêm đường từ chung cư của mình đến đường cao tốc song song với trục tung. Mỗi khu chung cư làm một đường (không chung nhau). Hỏi tổng độ dài các đường cần làm nhỏ nhất là bao nhiêu (Hai đường có thể trùng nhau trên mặt phẳng tọa độ - khi đó một cái nằm trên và cái kia nằm dưới)

**Dữ liệu vào: Từ tệp CHUNGCU.INP**

+ Dòng 1 ghi N (N ≤100)

+ N dòng tiêp theo, dòng thứ i ghi hai số nguyên xi, yi thể hiện tọa độ của một điểm

**Dữ liệu ra Ghi vào tệp CHUNGCU.OUT**

Ghi một số nguyên duy nhất là đáp số tìm được.

|  |  |
| --- | --- |
| **CHUNGCU.INP** | **CHUNGCU.OUT** |
| 5  -2282 2142  -2886 -228  436 -2782  991 -2468  1378 -4259 | 8955 |

***Hướng dẫn:***

Để tổng khoảng cách từ các chung cư đến đường cao tốc là nhỏ nhất thì đường cao tốc cần xây dựng ở vị trí giữa tung độ các chung cư.

* Cần phải sắp xếp các chung cư theo thứ tự tăng dần tung độ.
* Đường cao tốc cần đặt chính là tung độ giữa các điểm vừa sắp xếp
* Tính tổng quãng đường từ khu chung cư đến đường cao tốc.

ĐPT: O(N\*logN)

Cài đặt:

|  |
| --- |
| #include <bits/stdc++.h>  # define ll long long  using namespace std;  struct point  {  ll x, y;  };  ll n; point a[1001];  bool sx(point a, point b)  {  return a.y<b.y;  }  void xuly()  { ll ct;  sort (a+1, a+1+n, sx);  if (n % 2 ==0) ct= a[n/2].y;  else ct = a[(n+1)/2].y;  ll res=0;  for ( int i=1; i <=n ;i++)  res += abs( a[i].y -ct);  cout << res;  }  int main()  { //freopen("chungcu.inp" , "r", stdin);  //freopen("chungcu.out" , "w", stdout);  cin >> n;  for ( int i=1 ; i <=n ; i++)  cin >> a[i].x >> a[i].y;  xuly();  return 0;  } |

***Test***:<https://drive.google.com/file/d/170k8men3SpbHxGWGq_zsboqLAB5Kq9FS/view?usp=sharing>

***Code***:<https://drive.google.com/file/d/1PiTqVlMJRPLMvNUHBtjYb15OQjA2imm/view?usp=sharing>

**Bài 5: Rào cây**

Người ta muốn rào các cây ở một khu vườn để bảo vệ. Hàng rào được tạo bởi một đường gấp khúc. Khép kín có các đỉnh là một số cây làm cột mốc. sao cho các cây khác phải nằm trong hàng rào. (một số cây có thể nằm trên biên)

Hãy xác định một phương án rào cây sao cho số cây phải làm cột mốc là ít nhất.

**Input: Cho bởi file văn bản raocay.out**

* Dòng đầu ghi số n ≤10000 là số cây.
* Các dòng tiếp theo mỗi dòng là 1 cặp số nguyên (ghi cách nhau ít nhất một khoảng trắng) mô tả hoành độ, tung độ của một cây.

Các cây được đánh số từ 1 đến n theo trình tự xuất hiện trong file.

**Output: ghi ra file văn bản raocay.inp**

* Dòng đầu là số cây làm cột mốc. Dòng sau là số hiệu các cây này theo đúng thứ tự tạo hàng rào (ghi cách nhau ít nhất một khoảng trắng).

|  |  |
| --- | --- |
| raocay.inp | raocay.out |
| 7  2 1  4 2  2 3  4 4  -2 4  -2 3  -4 2 | 5  1 2 4 5 7 |

***Hướng dẫn:***

Sử dụng thuật toán Graham để tìm bao lồi của tập đỉnh trong n đỉnh cho trước. Đưa ra màn hình số lượng đỉnh thuộc bao lồi và thứ tự các đỉnh đó.

Độ phức tạp của thuật toán O(n\*logn) với n là số lượng điểm đề cho.

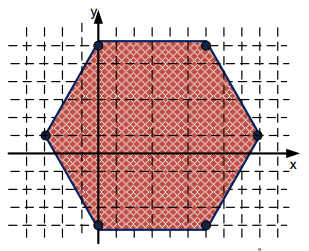
|  |
| --- |
| #include<bits/stdc++.h>  #define faster ios\_base :: sync\_with\_stdio(false) ; cin.tie(0) ; cout.tie(0)  using namespace std;  int n,m;  struct point  { int x,y,i;};  point a[10005],h[10005];  int ccw(point a, point b, point c)  {  int k=(b.x-a.x)\*(c.y-b.y)-(b.y-a.y)\*(c.x-b.x);  return k;}  bool cmp(point a,point b)  {  if (a.y==b.y) return a.x<b.x;  return a.y<b.y;  }  bool cmp2(point d, point b)  { int t=ccw(a[1],d,b);  if (t==0) return d.x<b.x;  return (t>0);  }  void Graham()  { sort(a+1,a+n+1,cmp);  sort(a+2,a+n+1,cmp2);  h[1]=a[1] ; h[2]=a[2] ; m=2 ;  a[n+1]=a[1];  for (int i=3 ; i<=n+1 ; i++)  { while(m>=2 and ccw(h[m-1],h[m],a[i])<=0) m--;  m++;h[m]=a[i];  }  cout<<m-1<<endl;  for (int i=1 ; i<m ; i++) cout<<h[i].i<<" ";  }  int main()  { //freopen("raocay.inp","r",stdin);  //freopen("raocay.out","w",stdout);  faster;  cin>>n;  for (int i=1 ; i<=n ; i++)  {cin>>a[i].x>>a[i].y;  a[i].i=i;}  Graham();  return 0;} |

*Test:*[*https://drive.google.com/file/d/1vJupRCuG2CIDBpmATicp\_MY\_dQ1J2Ont/view?usp=sharing*](https://drive.google.com/file/d/1vJupRCuG2CIDBpmATicp_MY_dQ1J2Ont/view?usp=sharing)

*Code:*[*https://drive.google.com/file/d/1vaJzToi2sYAY4UIO1B5LmnVqg4qG7TXM/view?usp=sharing*](https://drive.google.com/file/d/1vaJzToi2sYAY4UIO1B5LmnVqg4qG7TXM/view?usp=sharing)

## Bài 6: Lục giác đều (OLP 2008)

Lục giác đều là một dạng cấu trúc đặc biệt trong thiên nhiên. Bạn có thể gặp lục giác đều khi quan sát cách bố trí cánh của nhiều loại hoa, khi quan sát cấu trúc của tổ ong, khi nghiên cứu sơ đồ liên kết giữa Các bon và Ôxy trong các hợp chất hữu cơ và vô cơ. Mũ đinh ốc cũng tạo thành một lục giác đều. Lục giác đều là một trong số hiếm hoi các loại đa giác đều có thể phủ kín mặt phẳng.



Một bạn sinh viên quyết định chọn “Vai trò và vị trí của lục giác đều trong thiên nhiên” làm đề tài báo cáo trong một buổi sinh hoạt ngoại khóa. Để chuẩn bị số liệu cho bản thuyết trình của mình bạn đó đã khảo sát rất nhiều dữ liệu về cấu trúc lục giác gặp trong thiên nhiên và cuộc sống. Mỗi dữ liệu khảo sát là một dãy tọa độ 6 đỉnh trong mặt phẳng của lục giác. Bạn sinh viên muốn biết 6 điểm này có thể là đỉnh của một lục giác đều hay không. Ví dụ, nếu tọa độ của 6 điểm nhận được là (-3,1), (6,6.19615), (0,6.19615), (9,1), (0, -4.19615), (6, -4.19615) thì câu trả lời là có. Với dữ liệu phong phú thu thập được, việc kiểm tra trở thành một công việc nặng nề và tẻ nhạt nếu không sử dụng máy tính.

**Yêu cầu:** Cho tọa độ 06 đỉnh (xi, yi). Hãy kiểm tra xem 06 đỉnh trên có tạo thành một hình lục giác đều hay không.

**Input: Cho bởi file văn bản LG.INP**

- Là 06 cặp số thực (xi, yi), mỗi số cách nhau một khoảng trắng (-103 ≤ xi, yi ≤ 103).

**Output: Ghi vào file văn bản LG.OUT**

- Nếu 06 đỉnh trên tạo thành hình lục giác đều, in ra YES.

- Nếu không, in ra NO.

Lưu ý: Các giá trị thực được so sánh với độ chính xác 10-4.

|  |  |
| --- | --- |
| **LG.INP** | **LG.OUT** |
| -3 1 6 6.19615 0 6.19615 9 1 0 -4.19615 6 -4.19615 | YES |
| 0 6 0 -4 6 6 6 -4 -1 1 9 1 | NO |

***Hướng dẫn:***

- Với mỗi đỉnh i, tính khoảng cách từ nó đến các đỉnh còn lại lưu vào mảng kc. Chú ý: khoảng cách (p[i], p[j]) = khoảng cách (p[j], p[i]) được tính là một.

- Sắp xếp mảng kc theo chiều tăng dần.

- Điều kiện để 6 đỉnh tạo thành lục giác đều là: *(kc[1]=kc[6]) và (kc[7]=kc[12) và (kc[13]=kc[15]).*.

Độ phức tạp thuật toán: O(n2)

|  |
| --- |
| #include <bits/stdc++.h>  # define Eps 0.00001  using namespace std;  struct point  { double x, y;};  point d[100]; double b[100]; long long n;  void nhap()  {for (int i=1; i<=6; i++)  {cin>>d[i].x>>d[i].y;}  }  double kc(point a, point b)  { return sqrt((b.x-a.x)\*(b.x-a.x)+(b.y-a.y)\*(b.y-a.y));}  void tinh()  { for (int i=1; i<=5; i++)  for (int j=i+1; j<=6; j++)  { n++;  b[n]=kc(d[i], d[j]);}  sort (b+1,b+1+n);  if((abs(b[1]-b[6])<Eps) && (abs(b[7]-b[12])<Eps) && (abs(b[13]-b[15])<Eps))  cout<<"YES"; else cout<<"NO";}  int main(){  //freopen("LG.INP", "r", stdin);  //freopen("LG.OUT", "w", stdout);  nhap(); tinh();  return 0;} |

*Nguồn đề và test::* [*http://ntucoder.net/Problem/Details/3371*](http://ntucoder.net/Problem/Details/3371)

*Code:*[*https://drive.google.com/file/d/1XbUhom2C4lBfbDGpNs\_TUWwqFaaf20mn/view?usp=sharing*](https://drive.google.com/file/d/1XbUhom2C4lBfbDGpNs_TUWwqFaaf20mn/view?usp=sharing)

## Bài 7: Khoảng cách

Trước cửa nhà Mr Bill có một cái hồ rất rộng. Giữa hồ có một hòn đảo nhỏ. Một lần Mr Bill nảy ra ý định bắc một cái cầu từ cửa nhà mình đến đảo giữa hồ để kinh doanh du lịch. Một vấn đề khá hóc búa đối với Mr Bill là làm thế nào xác định được khoảng cách từ nhà mình đển đảo giữa hồ?.

Có thể mô tả đảo giữa hồ như là một đa giác lồi còn nhà của Mr Bill như là một điểm nằm ngoài đa giác đó trên mặt phẳng toạ độ. Bạn hãy lập trình giúp Mr Bill tính khoảng cách nhỏ nhất từ nhà mình đến đảo.

**Input: Vào từ file văn bản BILL.INP**

* Dòng đầu tiên ghi N là số đỉnh của đa giác (N≤1000)
* Dòng thứ hai ghi toạ độ của điểm được xem như là nhà của Mr Bill
* Tiếp theo là N dòng, mỗi dòng liệt kê toạ độ của một đỉnh của đa giác. Các đỉnh của đa giác được liệt kê ngược theo chiều kim đồng hồ.

**Output: Ghi ra file BILL.OUT**

Ghi ra một số thực duy nhất là khoảng cách từ nhà của Mr Bill đến đảo giữa hồ (giữ lại 4 chữ số phần thập phân).

|  |  |
| --- | --- |
| BILL.INP | BILL.OUT |
| 3  0 0  2 0  0 2  2 2 | 1.4142 |

***Hướng dẫn:***

Bài này ta cần đi kiểm tra 1 điểm có thuộc đa giác lồi đã cho hay không?

* Gọi Q là nhà của Bill, p[i] là mảng chứa tọa độ đa giác, res là kết quả bài toán.
* Đầu tiên ta tính khoảng cách từ Q đến các đỉnh đa giác. Cập nhật giá trị khoảng cách nhỏ nhất. res:=min(res,khoangcach(Q,p[i]));
* Tuy nhiên khoảng cách từ đỉnh đa giác tới nhà bill chưa phải là khoảng cách ngắn nhất từ bill đến đa giác, vì thế ta phải tìm khoảng cách ngắn nhất từ Bill (điểm Q) đến từng cạnh của đa giác cụ thể như sau:

+ Gọi (d1) là đường thẳng đi qua 2 đỉnh p[i] và p[i+1] của đa giác.

(d1) có dạng: a1x+b1y+c=0

+Gọi (d2) là đường thẳng đi qua điểm Q và vuông góc với (d1).

(d2) nhận vector (a1, b1) làm vector chỉ phương.

(d2): a2x+b2y+c=0 với a2=b1, b2=-a1, c=-b1\*Q.x+a1\*Q.y;

+Tìm điểm M là tọa độ giao điểm của d1 và d2.

Nếu điểm M thuộc cạnh ( p[i], p [i+1]) đang xét thì ta tính khoảng cách từ Q đến M cập nhật lại res.

Xét vị trí tương đối giữa 2 đường thẳng (d1) và (d2)







Nếu D ≠0 thì (d1) và (d2) cắt nhau tại điểm M.x=Dx/D; M.y=Dy/D;

Nếu (D=0) và (Dx≠0) thì (d1) song song (d2)

Nếu (D=Dx=Dy) thì (d1) trùng (d2).

Độ phức tạp của thuật toán: O(n)

**Cài đặt:**

|  |
| --- |
| #include <bits/stdc++.h>  # define ll long long  # define maxn 10000  # define Eps 0.0000001  using namespace std;  ll n;  struct point  { double x,y;};  point p[maxn], Q;  void nhap()  { cin>>n;  cin>>Q.x>>Q.y;  for (int i=1; i<=n;i++)  cin>>p[i].x>>p[i].y;  }  double dist(point p1, point p2)  {  double k=(p2.x-p1.x)\*(p2.x-p1.x)+(p2.y-p1.y)\*(p2.y-p1.y);  return sqrt(k);  }  bool equal (double x, double y)  {  double k=abs(x-y);  if (k<=Eps) return true;  return false;  }  void line(point p1,point p2, double &a, double &b, double &c)  { a=p2.y-p1.y;  b=p1.x-p2.x;  c=-(a\*p1.x+b\*p1.y);  }  int ccw(point a, point b, point c)  { double k=(b.x-a.x)\*(c.y-b.y)-(b.y-a.y)\*(c.x-b.x);  if (abs(k)<=Eps) return 0;  else if (k>0) return 1;  else return -1;  }  bool giaodiem(double a1,double b1, double c1, double a2, double b2, double c2, point &M)  { double D=a1\*b2-a2\*b1;  double Dx=b1\*c2-b2\*c1;  double Dy=a1\*c2-a2\*c1;  if (equal (D,0)==true) return false ;  M.x=Dx/D;  M.y=-Dy/D;  return true;  }  bool kt(point M, point p1, point p2)  { if ((ccw(p1, M, p2)==0) && (M.x>=min(p1.x, p2.x ))&& (M.x<=max(p1.x, p2.x)) && (M.y>=min(p1.y, p2.y )) && (M.y<=max(p1.y, p2.y)))  return true; else return false;  }  void xuly()  { double a1, b1, c1, a2, b2, c2; point M;  double res=dist(Q, p[1]);  for (int i=2; i<=n; i++) res=min(res, dist(Q, p[i]));  p[n+1]=p[1];  for (int i=1; i<=n; i++)  {  line (p[i], p[i+1], a1, b1, c1);  a2=b1; b2=-a1; c2=-b1\*Q.x+a1\*Q.y;//xay dung d2;  if (giaodiem(a1, b1, c1, a2, b2, c2, M)==true)  if (kt(M, p[i], p[i+1])==true) res=min(res, dist(Q,M));  }  cout << setprecision(4)<<fixed << res;  }  int main(){  //freopen("BILL.INP", "r", stdin);  //freopen("BILL.OUT", "w", stdout);  nhap();  xuly();  return 0;} |

***Test****:* [*https://www.spoj.com/THPTCBT/problems/MTKC/*](https://www.spoj.com/THPTCBT/problems/MTKC/)

***Code****:* [*https://drive.google.com/file/d/1g2J-Ac8oeBnjsn-Bxz9enxil\_j7vOQEO/view?usp=sharing*](https://drive.google.com/file/d/1g2J-Ac8oeBnjsn-Bxz9enxil_j7vOQEO/view?usp=sharing)

**Bài 8: Bắn máy bay:**

Vương quốc C11 sắp chuẩn bị gặp hiểm họa cực lớn từ vương quốc XYZ. Theo thông tin mật báo của điệp viên "Không Không Thấy", rằng XYZ sẽ gửi một Đại đội máy bay B52 ra thả bom tại HN (thủ đô của C11), và quyết đem HN về thời kỳ đồ đá. Và thật may mắn, điệp viên của ta đã rất nhanh tay lấy được sơ đồ chiến thuật tấn công của địch.

Đại đội máy bay bao gồm tổng cộng N chiếc, được đánh số từ 1→N. Mỗi chiếc sẽ có một vị trí xác định trong đội hình, và được cụ thể bằng một cặp số (x,y) cho biết tọa độ tương đối của nó. Theo nghiên cứu, cứ mỗi 3 chiếc trong đội hình tạo thành một hình tam giác sẽ tăng chỉ số chắc chắn của đội hình lên một đơn vị. Và độ an toàn của một chiếc máy bay trong đội hình là số lượng đội hình tam giác mà nó tham gia.

Vì thế ta cũng đã đề ra một chiến thuật, mỗi lần ta sẽ bắn rơi chiếc máy bay có độ an toàn nhỏ nhất trong đội hình của chúng.

#### Dữ liệu vào: Từ tệp MB.INP

* Dòng đầu tiên là số nguyên dương N (N≤200)
* N dòng tiếp theo là các cặp số nguyên (x,y) cho biết tọa độ của các chiếc máy bay. (|x|,|y|≤1000000000)(|x|,|y|≤1000000000)

#### Dữ liệu ra: Ghi vào tệp MB.OUT

* Cho biết chỉ số chắc chắn của đội hình lúc đầu và số thứ tự của chiếc máy bay được bắn rơi đầu tiên (nếu có nhiều máy bay cùng độ an toàn thì chọn chiếc có số thứ tự nhỏ nhất).

|  |  |
| --- | --- |
| MB.INP | MB.OUT |
| 4  1 2  0 0  1 0  2 0 | 3 2 |

***Hướng dẫn:***

* Gọi p[i] là tọa độ các máy bay.
* Ta cần đi kiểm tra 3 điểm p[i], p[j], p[k] với i, j, k=1..n. có tạo thành tam giác hay không? bằng cách kiểm tra 3 điểm p[i], p[j], p[k] thẳng hàng hay không. Sử dụng hàm CCW đểm kiểm tra 3 điểm thẳng hàng.
* Tìm chỉ số chắc chắn đội hình lúc đầu là số lượng tam giác được tạo được bởi n điểm.
* Tìm thứ tự chiếc máy bay bị bắn rơi đầu tiên: Gọi s[i] là số lượng tam giác có chứa điểm i thứ tự chiếc máy bay bị bắn rơi đầu tiên chính là giá trị i nhỏ nhất mà có s[i]min.

ĐPT O(n3)

**Cài đặt:**

|  |
| --- |
| #include <bits/stdc++.h>  # define ll long long  using namespace std;  const float Eps=0.000001;  ll n, kq, res;  struct point {  ll x, y;};  point a[200]; ll s[200];  void nhap()  { cin>>n;  for (int i=0; i<n; i++)  cin>>a[i].x>>a[i].y;  }  int ccw(point p1, point p2, point p3 )  {  if ((p2.x-p1.x)\*(p3.y-p1.y)==(p2.y-p1.y)\*(p3.x-p1.x)) return 1;  return 0;  }  void xuly()  {  kq= n\*(n-1)\*(n-2)/6;  for (int i=0; i<n-2; i++)  for ( int j=i+1; j<n-1;j++)  for (int k=j+1; k<n; k++)  if (ccw(a[i], a[j], a[k])==1)kq--;  else s[i]++, s[j]++, s[k]++;  res=1;  for (int i=1; i<n; i++)  if (s[i]<s[res]) res=i;  cout<<kq<<" "<<res+1;  }  int main(){  //freopen("MB.INP", "r", stdin);  //freopen("MB.OUT", "w", stdout);  nhap();  xuly();  return 0;  } |

*Test:* [*https://vn.spoj.com/problems/C11TRCNT/*](https://vn.spoj.com/problems/C11TRCNT/)

*Code:*[*https://drive.google.com/file/d/1Fw90DPHrurcilg23L2OFtHiPEaNuTZUw/view?usp=sharing*](https://drive.google.com/file/d/1Fw90DPHrurcilg23L2OFtHiPEaNuTZUw/view?usp=sharing)

## Bài 9: Mưa thiên thạch

Phú ông nhận được thông tin về một trận mưa thiên thạch sắp ập xuống trái đất. Không những thế, Phú ông còn biết tọa độ của vị trí điểm rơi của mỗi một thiên thạch. Phú ông nhờ Cuội xác định xem có bao nhiêu thiên thạch có thể rơi xuống cánh đồng của ông ta. Cánh đồng của Phú ông có dạng một hình đa giác lồi được xác định bởi danh sách các đỉnh được liệt kê theo thứ tự ngược chiều kim đồng hồ.

**Yêu cầu:** Xác định xem trong tập cho trước các điểm rơi của thiên thạch, có bao nhiêu điểm nằm trong cánh đồng của Phú ông. Các điểm nằm trên biên của cánh đồng không được tính là điểm nằm trong cánh đồng.

### Dữ liệu vào: Cho từ tệp TT.INP

- Dòng đầu tiên là số nguyên n (3 <= n <= 5000) là số đỉnh của đa giác lồi mô tả cánh đồng của Phú ông.

- Mỗi dòng trong n dòng tiếp theo chứa cặp tọa độ của một đỉnh của đa giác lồi.

- Dòng tiếp theo là số nguyên m (2 <= m <= 5000) - số thiên thạch rơi xuống.  
- Mỗi dòng trong số m dòng cuối cùng chứa 2 số là tọa độ điểm rơi của một thiên thạch.  
Các tọa độ là các số nguyên có trị tuyệt đối không quá 106.

### Dữ liệu ra: Ghi vào tệp TT.OUT

Ghi ra m dòng, mỗi dòng tương ứng với 1 điểm rơi của thiên thạch. Ghi "YES" nếu điểm rơi của thiên thạch nằm trong cánh đồng và ghi "NO" nếu trái lại.

|  |  |
| --- | --- |
| TT.INP | TT.OUT |
| 4  2 4  8 4  6 8  4 6  4  3 5  4 7  5 5  6 7 | NO  NO  YES  YES |

***Hướng dẫn:***

Ta gọi S là diện tích đa giác ban đầu. Ta tính diện tích đa giác bằng cách tính tổng diện tích các tam giác tạo bởi 3 đỉnh.(P[1], P[i], P[i+1]) i=2,…n.

Ta nhận thấy nếu 1 điểm thuộc đa giác thì tổng diện tích tạo bởi điểm đó với 2 đỉnh còn lại của đa giác sẽ bằng diện tích đa giác ban đầu.

Ví dụ: cần điểm tra điểm M có thuộc đa giác hay không?. Ta tính tổng diện tích các tam giác tạo bởi đỉnh M với 2 đỉnh còn lại của đa giác. S1+=SMP[i]P[i+1] i=1…N.

Nếu S==S1 thì điểm M thuộc đa giác ngược lại thì M không thuộc đa giác.

Lưu ý: Khi 3 điểm thẳng hàng thì không tồn tại tam giác tạo bởi 3 điểm đó.

Dùng công thức tính diện tích đa giác để tính diện tích tam giác

Độ phức tạp của thuật toán: O(n\*m) với n là số đỉnh đa giác, m số lượng thiên thạch.

***Cài đặt:***

|  |
| --- |
| #include <bits/stdc++.h>  #define ll long long  using namespace std;  struct Point{  ll x,y;  };  const int N=5000+10;  int n,m;  ll S;  Point P[N],T[N];  void nhap(){  //freopen("TT.INP","r",stdin);  //freopen("TT.OUT","w",stdout);  cin>>n;  for(int i=1;i<=n;i++) cin>>P[i].x>>P[i].y;  cin>>m;  for (int i=1; i<=m; i++)  cin>>T[i].x>>T[i].y;  }  ll tinh(Point A,Point B,Point C){  return abs((B.x-A.x)\*(C.y-A.y)-(B.y-A.y)\*(C.x-A.x));  }  bool check(int k){  ll Stg=0;  for(int i=1;i<=n;i++){  ll t=tinh(T[k],P[i],P[i+1]);  if (t==0) return false;  else Stg+=t;  }  if (2\*Stg==2\*S) return true;  return false;}  void xuly(){  P[n+1]=P[1]; P[0]=P[n]; S=0;  for(int i=2;i<=n-1;i++)  S+=tinh(P[1],P[i],P[i+1]);  for(int i=1;i<=m;i++)  if (check(i)) cout<<"YES"<<'\n';  else cout<<"NO"<<'\n';  }  int main(){  nhap(); xuly();  return 0;  } |

Ta có thể tối ưu thuật toán bằng cách kết hợp kĩ thuật chặt nhị phân như sau:

* Giả sử cần kiểm tra điểm pm thuộc đa giác (P1, P2….Pn) hay không?
* Gọi 2 điểm d, c là 2 điểm thuộc đa giác chứa cạnh (P1, Pd) và (P1, Pc) sao cho cạnh (P1, Pm) nằm giữa 2 cạnh này. Dùng chặt nhị phân trên đoạn [2, n] để tìm 2 điểm d và c. Khi đó ta tính diện tích các tam giác sau:

S=S1dc (là tam giác tạo bởi 3 điểm (p1, pd, pc))

S1= S1dm

S2= S1cm;

S3= Smcd

Nếu: S1=0 hoặc S2=0 hoặc S3=0 thì điểm pm nằm trên cạnh đa giác 🡺cout<<”NO”;

Nếu S== S1, +S2+S3thì điểm M thuộc đa giác🡺cout<<”YES”;

Ngược lai điểm M không thuộc đa giác🡺cout<<”NO”

Độ phức tạp thuật toán: O(m\*logn) với n là số đỉnh đa giác, m số lượng thiên thạch.

**Cài đặt:**

|  |
| --- |
| #include <bits/stdc++.h>  # define ll long long  using namespace std;  struct point  {ll x, y;};  int n, m;  point p[5005], a[5005], pm[5005];  int ccw(point p1, point p2, point p3)  {  ll k=(p2.x-p1.x)\*(p3.y-p2.y)-(p3.x-p2.x)\*(p2.y-p1.y);  if (abs(k)==0) return 0;  else  if (k>0) return 1;  else return -1;  }  ll dt(point p1, point p2, point p3)  {  a[1]=p1; a[2]=p2;a[3]=p3; a[4]=a[1];  ll dt=0;  for (int i=1; i<=3; i++)  dt=dt+(a[i+1].x-a[i].x)\*(a[i+1].y+a[i].y);  return abs(dt);  }  int kt(point pm)  { ll s1,s2, s3, s;  int d=2, c=n;  while (d+1<c)  {  int mid=(d+c)/2;  if (ccw(p[1], p[mid], pm)>0) d=mid;  else c=mid;  }  s1=dt(p[1], p[d], pm);// dien tich da giac p1, pd, pm;  s2=dt(p[1], p[c], pm);  s3=dt(pm, p[d], p[c]);  s=dt(p[1], p[d], p[c]);  if (s3==0 || s1==0 || s2==0) return 0;  if (s==(s1+s2+s3)) return 1;  else return 0;  }  int main(){  //freopen("TT.INP","r",stdin);  //freopen("TT.OUT","w",stdout);  cin>>n;  for (int i=1; i<=n; i++) cin>>p[i].x>>p[i].y;  cin>>m;  for (int i=1; i<=m; i++)  { cin>>pm[i].x>>pm[i].y;  if (kt(pm[i])==1) cout<<"YES"<<endl;  else cout<<"NO"<<endl;}  return 0;} |

Test: <https://vn.spoj.com/problems/METERAIN/>

Code:<https://drive.google.com/file/d/1dA0OVmpcY7AADEM7kAopp1_25uNX54TX/view?usp=sharing>

## Bài 10: Ruộng bậc thang (Đề thi HSG quốc gia 2015)

Ở các vùng cao hiếm đất cùng mặt bằng để canh tác, khi tiến hành trồng trọt trên các sườn đồi núi có đất màu, người ta phải bạt tam cấp để tạo thành những vạt đất bằng. Khu vực đất dốc dùng để canh tác như vậy gọi là ruộng bậc thang. Hình ảnh các khu ruộng bậc thang vẫn luôn là một hình ảnh đẹp ở các vùng cao khiến du khách và các nhà nhiếp ảnh đam mê và tốn không ít phim ảnh. Gia đình Hoàng có một khu ruộng bậc thang bao quanh một ngọn đồi được chia thành các khoang bậc thang, mỗi khoang trồng một loại cây. Khi nhìn từ trên cao xuống, ta thấy các khoang bậc thang này có hình dạng của các đa giác lồi lồng nhau. Ngoại trừ khoang chứa đỉnh đồi có biên là một đa giác lồi chứa đỉnh đồi, mỗi khoang còn lại được xác định bởi hai đa giác lồng nhau: đa giác có diện tích lớn hơn được gọi là biên ngoài của khoang còn đa giác có diện tích nhỏ hơn được gọi là biên trong của khoang. Mỗi khoang có màu đặc trưng của loại cây được trồng ở khoang đó. Vốn là một người say mê chụp ảnh, muốn có một bức ảnh đẹp, Hoàng tìm cách thay đổi không quá k loại cây được trồng ở k khoang để khi nhìn từ trên cao xuống sẽ thấy một vùng cùng màu có diện tích lớn nhất. Hoàng đã ghi nhận được danh sách m đa giác lồi mô tả biên ngoài của m khoang và màu tương ứng của chúng. Do sơ xuất, Hoàng đã để các thông tin về các khoang trong danh sách bị xáo trộn, không còn được liệt kê theo đúng trình tự từ khoang trong đến khoang ngoài.

**Yêu cầu:**Cho biết thông tin về danh sách mà Hoàng đã ghi nhận và số nguyên k, hãy tìm cách thay đổi không quá k loại cây được trồng ở k khoang để khi nhìn từ trên cao xuống sẽ thấy một vùng cùng màu có diện tích lớn nhất.

**Dữ liệu vào: Cho từ tệp TFIELD.INP**

* Dòng đầu chứa hai số nguyên dương m, k (k ≤ m);
* Dòng thứ i trong số m dòng tiếp theo chứa thông tin về khoang thứ i trong danh sách mà Hoàng ghi nhận bao gồm:
* Đầu tiên là số nguyên ni là số đỉnh của đa giác lồi mô tả biên ngoài của khoang;
* Tiếp theo là số nguyên cithể hiện màu của khoang (1 ≤ ci ≤ m);
* Cuối cùng là ni cặp số nguyên, mỗi số có trị tuyệt đối không quá 109, là tọa độ của một đỉnh của đa giác. Các đỉnh của đa giác được liệt kê theo thứ tự ngược chiều kim đồng hồ.

Hai số liên tiếp trên cùng dòng được ghi cách nhau bởi dấu cách.

**Dữ liệu ra: Ghi ra tệp TFIELD.OUT**

* Ghi ra một số thực là diện tích vùng cùng màu lớn nhất sau khi thay đổi không quá k  loại cây được trồng ở k  khoang (kết quả đưa ra với độ chính xác 1 chữ số sau dấu chấm thập phân).

**Ràng buộc:**

* Có 40% số test ứng với 40% số điểm của bài thỏa mãn điều kiện: m ≤ 10; k = 1; các đa giác mô tả biên ngoài của các khoang là hình chữ nhật;
* Có 40% số test khác ứng với 40% số điểm của bài thỏa mãn điều kiện: m ≤ 10; các đa giác mô tả biên ngoài của các khoang là tam giác;
* Có 20% số test còn lại ứng với 20% số điểm của bài thỏa mãn điều kiện: m, ni ≤ 1000.

|  |  |  |
| --- | --- | --- |
| TFIELD.INP | TFIELD.OUT |  |
| 3 1  4 1 0 0 1 0 1 1 0 1  4 1 -2 -3 5 -3 5 5 -2 5  3 2 -1 -1 4 -1 -1 4 | 56.0 | https://lh6.googleusercontent.com/-71uLPzo0grg/VLdt_zbb8jI/AAAAAAAAAJ0/oLS4N1-0sLs/w362-h345-no/TFIELD.png |

***Hướng dẫn:***

Cho các mảnh ruộng là các đa giác lồi bao nhau, mỗi mảnh một màu, hỏi có thể đổi màu cho không quá *k* mảnh để chúng đồng màu và có diện tích lớn nhất.

1. Tính diện tích mỗi đa giác gọi là s[i],

2. Do các mảnh để không theo thứ tự nên ta cần sắp lại các mảnh theo thứ tự tăng dần s[i].

3. Với 2 mảnh i và j bất kỳ, ta sẽ thống kê được màu có số lượng lớn nhất *maxc*, như trong test đề bài thì với *i=1, j=3*, *maxc* = 2 vì có số lượng màu 1 lớn nhất là 2. Việc thống kê này nên dùng mảng *ccnt*[] với chỉ số là mã màu và được tính đồng thời cùng vòng lặp *i* và *j*. Nếu *j-i+1-maxk<=k* thì từ *i* đến *j* có thể biến đổi về cùng một màu, ta cập nhật diện tích bằng *s[j]-s[i-1]* vào biến kq nếu chúng lớn hơn *kq*. Chi phí sẽ bằng O(N2).

***Cài đặt***

|  |
| --- |
| #include <bits/stdc++.h>  #define maxn 1000  #define maxm 1000  using namespace std;  struct point {  int x, y;  };  struct poly{  point p[maxn];  int n;  int s;  int c;  };  poly d[maxm]; int dem[maxm], kq, m,k, maxc;  void nhap()  {  cin>>m>>k;  for (int i=1; i<=m;i++)  {  cin>>d[i].n>>d[i].c;  for (int j=1; j<=d[i].n; j++)  cin>>d[i].p[j].x>>d[i].p[j].y;  d[i].p[d[i].n+1]=d[i].p[1];  }  }  long long tinhs(poly d)  { long long s=0;  for (int i=1; i<=d.n; i++)  s=s+(d.p[i+1].x-d.p[i].x)\*(d.p[i+1].y+d.p[i].y);  return abs(s);  }  bool sx(poly a, poly b)  {return a.s<b.s;}  void xuly()  { for (int i=1; i<=m; i++)  d[i].s=tinhs(d[i]);  sort (d+1, d+1+m, sx);  for (int i=1; i<=m; i++)  { for (int i=1; i<=m; i++) dem[i]=0;  maxc=0;  for (int j=i; j<=m; j++)  { dem[d[j].c]++;  if (maxc<dem[d[j].c]) maxc=dem[d[j].c];  if (j-i+1-maxc<=k)  if (d[j].s-d[i-1].s>kq)  kq=d[j].s-d[i-1].s;  }  }  }  void ghi()  { if (kq%2==0) cout<<kq/2<<"."<<"0";  else cout<<kq/2<<"."<<"0";  }  int main(){  //freopen("TFIELD .INP", "r", stdin);  //freopen("TFIELD .OUT", "w", stdout);  nhap(); xuly();ghi();  return 0;} |

***Test****:* [*https://vn.spoj.com/problems/TFIELD/*](https://vn.spoj.com/problems/TFIELD/)

***Code****:*[*https://drive.google.com/file/d/1HJSUWoFugjIV3TRM2jrST39QT8luHtb/view?usp=sharing*](https://drive.google.com/file/d/1HJSUWoFugjIV3TRM2jrST39QT8luHtb/view?usp=sharing)

**III. MỘT SỐ BÀI TẬP TỰ GIẢI**

## Bài 1 : Bảo vệ thành phố

Ở vương quốc Xa Rất Rất Xa, có N thành phố được đánh số từ 1 đến N. Thành phố thứ i nằm ở vị trí có tọa độ (xi, yi). Tuy thế giới vẫn đang trong thời gian hòa bình, vị vua của đất nước tin rằng họ phải luôn sẵn sàng bảo vệ đất nước. Quốc vương quyết định xây một tường thành khép kín có dạng một đa giác để bảo vệ một số thành phố của vương quốc. Các thành phố nằm trong và trên biên giới của tường thành này sẽ có thể yên tâm phát triển trong yên bình. Để đảm bảo cho sự vững chắc của nền kinh tế, phải có ít nhất K thành phố được bảo vệ. Tuy nhiên, việc xây dựng một tường thành kiên cố là vô cùng tốn kém, vì vậy, Quốc vương muốn chọn phương án xây tương thành có chu vi nhỏ nhất.

Cho tọa độ của N thành phố, và số nguyên dương K, nhiệm vụ của bạn là tìm một tường thành bao quanh ít nhất K thành phố với chu vi nhỏ nhất.

**Input**

* Dòng thứ nhất gồm 2 số nguyên dương N, K.
* N dòng tiếp theo: mỗi dòng chứa 2 số nguyên xi, yi là tọa độ của một thành phố trong vương quốc.

**Ouput**

* Ghi ra một số nguyên duy nhất là độ dài nhỏ nhất của tường thành, với 4 chữ số sau dấu phẩy.

### Giới hạn

* 1 ≤ K ≤ N ≤ 50. Trong 35% số test có 1 ≤ K ≤ N ≤ 25.
* Các tọa độ có giá trị tuyệt đối không vượt quá 106.

|  |  |  |
| --- | --- | --- |
| **Input** | **Ouput** |  |
| 5 4  1 1  2 5  3 2  3 3  4 1 | 8.0645 | http://i37.photobucket.com/albums/e81/beo_chay_so/voj_sample_zps6cadef14.png |

## Bài 2: Vườn cây

Bạn Minh Đức là chủ một vườn cây ăn quả lớn ở miền Nam. Nếu nhìn từ trên cao xuống, các gốc cây giống như các điểm trên mặt phẳng tọa độ là mặt đất. Đã nhiều năm rồi, bạn Minh Đức không được bội thu do nạn đạo tặc. Do vậy, năm ngoái bạn Minh Đức quyết tâm rào khu vườn của mình lại. Để làm được điều này, bạn Minh Đức chăng đường rào theo các gốc cây để tạo thành một đa giác bao kín vườn cây. Do tính keo kiệt và chi phí của đường rào là rất đắt, bạn Minh Đức đã tính toán chi li để đường rào có chu vi nhỏ nhất có thể. Chắc các bạn cũng biết đây là bài toán tin cơ bản : tìm bao lồi nhỏ nhất của một tập điểm. Tuy nhiên, năm nay bạn Minh Đức cũng không thu hoạch được thêm nhiều. Lý do là có một số cây ở đường biên của hàng rào vẫn không thoát khỏi bàn tay của đạo tặc. Năm nay bạn Minh Đức quyết xây dựng lại hàng rào để cho không còn cây nào nằm ở đường biên nữa. Để làm được điều này, thay vì chăng đường rào theo các gốc cây, bạn Minh Đức sẽ chăng đường rào theo các cột sắt có sẵn trong vườn. Vị trí của các cây và cột sắt đã rõ, nhưng xây dựng làm sao để hàng rào có chu vi nhỏ nhất vẫn là vấn đề nan giải. Bạn hãy giúp bạn Minh Đức giải quyết bài toán khó trên và cùng chia sẻ một vụ mùa bội thu.

### Input

* Dòng đầu là số N ( N <= 100 ). Là số cây trong vườn.
* N dòng sau, mỗi dòng ghi 2 số là tọa độ của một cây trong vườn.
* Dòng tiếp theo là số M ( M <= 100 ). Là số cột sắt trong vườn.  
   - M dòng sau, mỗi dòng ghi 2 số là tọa độ của một cột sắt.  
   Các tọa độ đều là số nguyên trong khoảng -10000..10000.

### Output

* In ra một số duy nhất là độ dài nhỏ nhất của hàng rào với đúng 2 chữ số sau dấu chấm thập phân ( có làm tròn ). Dữ liệu luôn đảm bảo có ít nhất 1 cách xây hàng rào thỏa mãn.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1  0 2  3  -2 0  2 0  0 4 | 12.94 |

**Bài 3: Người yêu hòa bình**

Ngày xửa ngày xửa, quần đảo của Pirate bị chia cắt bởi các thế lực cát cứ. Cuối cùng, có 3 thế lực mạnh nhất đã thâu tóm hết các nước lân bang để tạo thành thế Tam Đảo: đó là nước Rỉ, nước Quá và nước Đanh. Giữa ba nước này cứ diễn ra chọi dừa liên miên khiến dân chúng u đầu mẻ trán như cơm bữa, cuộc sống lầm than không sao kể xiết.

Pirate sinh ra trong thời loạn, xuất thân từ một gia đình trồng chuối, từ nhỏ đã nối tiếng với tài vừa xem đá bóng vừa... làm thơ. Thấy con trai mình có tài văn chương thiên bẩm, phụ thân Pirate mời cả "Bắc Cương, Nam Vũ" đến dạy chữ cho anh, những mong sau này con trai có thể trở thành một nhà thuyết khách, đi khắp Tam Đảo giảng hòa ba nước, tạo phúc cho bá tánh.

Sao bao nhiêu sóng gió, cuối cùng, bằng việc hứa sẽ truyền cho các chúa đảo về bí quyết trồng chuối to, Pirate đã đạt được một thỏa thuận hòa bình cho Tam Đảo. Đó là sự hình thành một "vùng ngừng chọi". Pirate kéo tấm bản đồ các hòn đảo trong quần đảo ra. Ở trên đó, mỗi đảo được kí hiệu bởi một điểm nguyên trên mặt phẳng tọa độ và bị duy nhất một trong ba nước chiếm giữ. "Vùng ngừng chọi" sẽ có hình dạng một tam giác với ba đỉnh là ba hòn đảo trên tấm bản đồ này mà trong đó không có hai hòn đảo nào bị chiếm giữ bởi cùng một nước. Ba nước cam đoan sẽ không chọi dừa trong phần diện tích che phủ bởi "vùng ngừng chọi".

Pirate chỉ muốn "vùng ngừng chọi" có diện tích lớn nhất để bá tánh bớt lầm than và có thể chăm lo trồng chuối. Hãy giúp anh ấy xác định diện tích lớn nhất có thể của "vùng ngừng chọi"!

### Input:

* Dòng thứ nhất là số nguyên N, số đảo mà mỗi nước chiếm giữ (cả ba nước đều chiếm được cùng một số lượng đảo).
* Tiếp theo là một dòng trống.
* N dòng tiếp theo, mỗi dòng là hai số nguyên x y, thể hiện vị trí của một hòn đảo thuộc về nước Rỉ.
* Tiếp theo là một dòng trống.
* N dòng tiếp theo, mỗi dòng là hai số nguyên x y, thể hiện vị trí của một hòn đảo thuộc về nước Quá.
* Tiếp theo là một dòng trống.
* N dòng tiếp theo, mỗi dòng là hai số nguyên x y, thể hiện vị trí của một hòn đảo thuộc về nước Đanh.

### Output:

Output gồm một dòng duy nhất là diện tích lớn nhất của "vùng ngừng chọi" tìm được (in ra một chữ số sau dấu phẩy).

|  |  |
| --- | --- |
| Input | Output |
| 5  0 0  -8 0  3 0  6 2  -3 8  -5 -7  2 0  -1 0  8 0  4 -2  0 6  -1 -2  4 -4  -5 -4  2 -5 | 74.0 |

Giải thích: "vùng ngừng chọi" có các đỉnh tại (-3, 8), (8, 0), (-5, -4).

### Giới hạn:

* 1 ≤ N ≤ 105
* Trong 50% số test có 1 ≤ N ≤ 200
* Các điểm có tọa độ là số nguyên có trị tuyệt đối không quá 109.

## Bài 4: Trụ sở xa nhất

Microhoo và Googloo là hai công ty Tin học cạnh tranh nhau ở cùng một thành phố. Mỗi công ty có một số văn phòng nằm ở một số điểm trong thành phố. Để bảo vệ thông tin quan trọng khỏi lọt đến đối thủ, cả hai công ty đã cam kết sẽ đặt trụ sở chính càng xa nhau càng tốt.

Cho trước vị trí của các văn phòng của Microhoo và Googloo, nhiệm vụ của bạn là viết một chương trình để giúp hai công ty chọn từ các văn phòng đã có để đặt trụ sở chính sao cho khoảng cách giữa hai trụ sở chính của hai công ty là lớn nhất.

**Input**

Dữ liệu vào gồm nhiều bộ dữ liệu tương ứng với nhiều test. Dòng đầu tiên chứa một số nguyên dương không lớn hơn 20 là số lượng các bộ dữ liệu. Các dòng tiếp theo chứa các bộ dữ liệu.

Với mỗi bộ dữ liệu, dòng đầu tiên chứa số nguyên n (2 ≤ n ≤ 30000) là tổng số lượng văn phòng của cả hai công ty. Dòng thứ i trong n dòng tiếp theo chứa ba số nguyên xi, yi, ci (0 ≤ |xi|, |yi| ≤ 108, 0 ≤ ci ≤ 1) cách nhau bởi dấu trống, với (xi, yi) là tọa độ của văn phòng thứ i và nó là văn phòng của Microhoo nếu ci = 0 và Googloo nếu ci = 1.

Cho biết mỗi công ty có ít nhất một văn phòng.

**Output**

Với mỗi bộ dữ liệu, ghi ra trên một dòng phần nguyên của khoảng cách lớn nhất giữa trụ sở chính của Microhoo và Googloo.

Ví dụ

|  |  |
| --- | --- |
| Input | Output |
| 2  2  0 0 0  3 -2 1  5  1 5 1  -5 2 0  3 7 1  6 -2 0  5 1 0 | 3  9 |

## Bài 5: Trò chơi với những viên bi (Đề thi HSG quốc gia 2014)

Trong một hội thi Ballgame, ban tổ chức chuẩn bị một bàn lớn. Trên mặt bàn có n bi xanh đánh số từ 1 đến n và n bi đỏ đánh số từ n + 1 đến 2n. Mỗi trận đấu, các vận động viên sẽ chơi luân phiên nhau. Đến lượt chơi của mình, Hùng cần tìm 3 bi mà vị trí của chúng là thằng hàng hanu và sao cho trong số đó có hai bi đỏ và 1 bi xanh (khi đó ăn được một bi đỏ), hoặc là có hai bi xanh và 1 bi đỏ (khi đó được ăn 1 bi xanh).

### Yêu cầu

Cho biết tọa độ trên mặt phẳng tọa độ Đề-các của vị trí và màu của các bi hiện tại trên bàn, bạn hãy giúp Hùng chọn 3 bi để chơi.

### Dữ liệu vào: Từ tệp GAME.INP

* Dòng đầu ghi số nguên dương n.
* Dòng thứ i trong số n dòng tiếp theo ghi hai số nguyên là hoành độ và tung độ trên mặt phẳng tọa độ Đề-các của vị trí đặt bi xanh với chỉ sô i.
* Dòng thứ i trong số n dòng cuối cùng ghi hai số nguyên là hoàng độ và tung độ trên mặt phẳng tọa độ Đề-các của vị trí đặt bi đỏ với chỉ số n + i.
* Hoàng độ và tung độ không vượt quá 10^6, vị trí các bi là đôi một phân biệt.

### Giới hạn

* 30% số test có n <= 2;
* 30% số test khác có n <= 100.
* 40% số test còn lại có n <= 1000.

### Dữ liệu ra: Ghi dữ liệu vào tệp GAME.OUT

Ghi ra 3 chỉ số của các viên bi mà Hùng cần chọn, nếu không thể chọn được 3 bi nào, ghi ra -1. Nếu có nhiều đáp án, ghi ra một đáp án bất kì.

Ví dụ:

|  |  |
| --- | --- |
| GAME.INP | GAME.OUT |
| 3  1 1  2 2  4 9  3 3  6 20  8 100 | 1 2 4 |

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN:

## Kết luận:

Chuyên đề đã trình bày một số khái niệm cơ bản về Hình học phẳng, các phương pháp Hình học phổ biến, một bài toán Hình học cơ bản trong tin học và áp dụng giải quyết những bài tập tiêu biểu liên quan. Những bài tập trong chuyên đề được tôi sưu tầm và lựa chọn ở nhiều nguồn tại liệu khác nhau. Chuyên đề đã được tôi sử dụng trong quá trình giảng dạy học sinh đội tuyển. Kết quả thu được là học sinh có cái nhìn tổng quan và nắm vững những bài toán Hình học cơ bản trong tin học từ đó làm nền tảng cho việc giải quyết những bài toán khó hơn.

## Hướng phát triển:

Mặc dù đã rất cố gắng nhưng do thời gian có hạn cùng với sự hạn chế về chuyên môn nên số lượng bài tập, cùng hướng dẫn giải và test kèm theo vẫn còn hạn chế. Bên cạnh đó còn vấn đề cần phải có một phương pháp truyền đạt như thế nào để học sinh tích cực, chủ động lĩnh hội được các kiến thức chuyên sâu. Trong thời gian tới tôi sẽ tiếp tục phát triển chuyên đề này theo hướng hệ thống bài tập theo từng chủ đề liên quan đến các bài toán Hình học với mức độ khó cao hơn. Tôi hy vọng nhận được sự giúp đỡ, góp ý của đồng nghiệp để bài viết được hoàn thiện hơn, góp phần nhỏ cho công tác bồi dưỡng học sinh giỏi.

Tôi xin chân thành cảm ơn!

**TÀI LIỆU THAM KHẢO**

[1] https://en.wikipedia.org/wiki/Graham\_scan

[2] http://vnoi.info/wiki/translate/wcipeg/Convex-Hull

[3] Đỗ Mạnh Dũng, ***Bài viết về thuật toán hình học***, http://vnoi.info/forum/5/4970/

[4] Hồ Sĩ Đàm (chủ biên), (2009), ***Tài liệu giáo khoa chuyên tin quyển 1, 2, 3***, NXB GD

[5] http://codeforces.com/problemset

[6] <http://www.spoj.com/>

[7] <http://ntucoder.net/Problem/List?ThemeID=30>

MỤC LỤC

[I. LÝ DO CHỌN ĐỀ TÀI 2](#_Toc81430778)

[II. MỤC ĐÍCH NGHIÊN CỨU 2](#_Toc81430782)

[III. ĐỐI TƯỢNG NGHIÊN CỨU 2](#_Toc81430783)

[IV. PHƯƠNG PHÁP NGHIÊN CỨU 2](#_Toc81430784)

[4.1 Phương pháp nghiên cứu lý luận: 2](#_Toc81430785)

[4.2 Phương pháp nghiên cứu thực tiễn: 2](#_Toc81430786)

[V. Ý NGHĨA THỰC TIỄN CỦA ĐỀ TÀI 3](#_Toc81430787)

[NỘI DUNG 4](#_Toc81430788)

[I. CƠ SỞ LÝ THUYẾT 4](#_Toc81430789)

[1. Biểu diễn hình học trên máy tính. 4](#_Toc81430790)

[1. 1 Điểm: 4](#_Toc81430791)

[1.2 Đường thẳng: 4](#_Toc81430792)

[1.3 Đa giác: 4](#_Toc81430793)

[2. Một số khái niệm cơ bản khác: 4](#_Toc81430794)

[2.1 Đường gấp khúc: 4](#_Toc81430795)

[2.2 Đa giác: 4](#_Toc81430796)

[2.3 Đa giác tự cắt 4](#_Toc81430797)

[2.4 Đa giác lồi: 4](#_Toc81430798)

[3. Một số phương pháp hình học: 4](#_Toc81430799)

[3.1 Tính khoảng cách giữa 2 điểm: 5](#_Toc81430800)

[3.2 Vị trí tương đối giữa 3 điểm. 5](#_Toc81430801)

[3.3 Phương trình đường thẳng đi qua hai điểm: 5](#_Toc81430802)

[3.4 Diện tích đa giác. 6](#_Toc81430803)

[3.5 Bao lồi: 6](#_Toc81430804)

[3.6 Tìm khoảng cách cặp điểm gần nhất: 10](#_Toc81430809)

[II. MỘT SỐ BÀI TẬP VẬN DỤNG: 12](#_Toc81430810)

[Bài 1. Du lịch 12](#_Toc81430811)

[Bài 2: Đa giác: 14](#_Toc81430813)

[Bài 3: Đếm hình vuông 17](#_Toc81430814)

[Bài 4. Đường cao tốc 19](#_Toc81430815)

[Bài 5: Rào cây 21](#_Toc81430816)

[Bài 6: Lục giác đều (OLP 2008) 23](#_Toc81430866)

[Bài 7: Khoảng cách 25](#_Toc81430867)

[Bài 8: Bắn máy bay: 28](#_Toc81430868)

[Bài 9: Mưa thiên thạch 30](#_Toc81430869)

[Bài 10: Ruộng bậc thang (Đề thi HSG quốc gia 2015) 35](#_Toc81430872)

[III. MỘT SỐ BÀI TẬP TỰ GIẢI 38](#_Toc81430941)

[Bài 1 : Bảo vệ thành phố 38](#_Toc81430942)

[Bài 2: Vườn cây 39](#_Toc81430944)

[Bài 3: Người yêu hòa bình 40](#_Toc81430947)

[Bài 4: Trụ sở xa nhất 41](#_Toc81430951)

[Bài 5: Trò chơi với những viên bi (Đề thi HSG quốc gia 2014) 42](#_Toc81430952)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN: 44](#_Toc81430957)

[Kết luận: 44](#_Toc81430958)

[Hướng phát triển: 44](#_Toc81430959)