**CHUYÊN ĐỀ DUYÊN HẢI**

**HÌNH HỌC PHẲNG TRONG TIN HỌC**

*GV: Nguyễn Như Thắng – THPT Chuyên tỉnh Lào Cai*

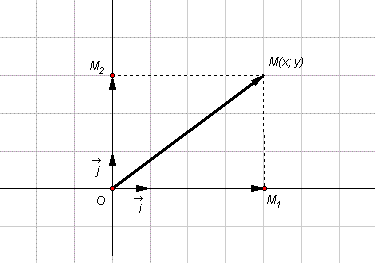
**1. Mở đầu**

Có nhiều bài toán hình học phẳng, quan sát bằng mắt thường thì lời giải là hiển nhiên, tuy nhiên muốn lập trình bằng máy tính để giải nó lại khá khó khăn, lại là một vấn đề hoàn toàn khác. Nhưng nếu có kiến thức toán về hình học phẳng tốt, học sinh hoàn toàn có thể làm chủ các bài toán hình học phẳng đó và các lời giải nhiều khi lại là điều khá đơn giản.

Trong Chuyên đề này, tôi muốn chia sẻ một số kiến thức toán về hình học phẳng, một số vấn đề thường quan tâm trong hình học phẳng.

Có một số vấn đề trình bày ở đây có nhiều thuật toán, song tôi chỉ trình bày thuật toán được cho là tối ưu. Toàn bộ code trong chuyên đề được được thể hiện bằng ngôn ngữ C++.

**2. Một số đối tượng cơ bản của hình học phẳng**

2.1. Hệ trục tọa độ Đề Các *Oxy*: Xem SGK Hình học 10, tiết 4.

2.2. Điểm trên hệ trục tọa độ *Oxy*:

Cho điểm *M(x,y)* như hình vẽ, *x* được gọi là hoành độ, *y* được gọi là tung độ của điểm M trong hệ trục tọa độ Đề Các *Oxy*, khi đó:



- Giả sử *x,y* là số nguyên, khi đó biểu diễn điểm M trong máy tính có thể dùng một trong hai cách sau:

|  |  |
| --- | --- |
| typedef pair<int,int> Point; | struct Point {  int x;  int y;  }; |

- Khoảng cách giữa 2 điểm  và , hoặc độ dài của vecctor  được tính bằng: 

|  |
| --- |
| double dist(Point A,Point B) {  return sqrt((B.x-A.x)\*(B.x-A.x)+(B.y-A.y)\*(B.y-A.y));  } |

2.3. Đường thẳng

Đường thẳng trong mặt phẳng xác định khi biết được 2 điểm *A, B* phân biệt nằm trên đường thẳng đó. Khi đó đường thẳng được xác định là tập hợp các điểm *M(x,y)* sao cho: 



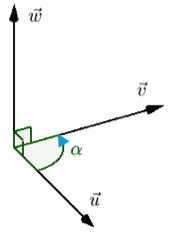
Nếu *t<0* thì *M* nằm ngoài *AB* về phía *A*

Nếu *0<t<1* thì M nằm giữa *A, B*

Nếu *t>1* thì *M* nằm ngoài *AB* về phía *B*.

2.4. Tích chéo, tích vô hướng của 2 vector

2.4.1. Tích chéo của hai vector:



Quay phải, tích chéo dương

Tích chéo của 2 vector  và (tích chéo là khái niệm được suy ra từ khái niệm tích có hướng trong không gian vector Ơclit nhiều chiều):

Giá trị của nó bằng định thức của ma trận  hoặc tính bằng . Trong đó góc là góc định hướng, có số đo từ  tới  🡺Giá trị lượng giác *sin* của góc định hướng = là:



Tích chéo có tác dụng để kiểm tra chiều quay từ  đến  là chiều quay phải, hay quay trái, ví dụ trong hình vẽ trên là quay trái khi đó ; chiều quay từ  đến  là quay phải nếu tích chéo có giá trị âm; và ,  thẳng hàng nếu tích chéo của chúng bằng 0.

|  |
| --- |
| int ccw(Point A, Point B, Point C) {  double t=(B.x-A.x)\*(C.y-A.y)-(B.y-A.y)\*(C.x-A.x);  if (t>0)  return 1; //quay trai  if (t<0)  return -1; //quay phai  return 0; //thang hang  } |

2.4.2. Tích vô hướng của 2 vector (hay còn gọi là tích chấm): Tích vô hướng của 2 vector là một số có giá trị là: 

Cài đặt tích vô hướng, tích có hướng bằng kĩ thuật chồng toán tử như sau:

|  |  |
| --- | --- |
| Tích vô hướng (tích chấm) | Tích chéo |
| int operator \*(Point u, Point v) {  return (u.x\*v.x+u.y\*v.y);  } | int operator ^(Point u, Point v) {  return (u.x\*v.y-u.y\*v.x);  } |

2.5. Góc

- Góc tạo bởi hai vector ,  có giá trị lượng giác .

- Góc tạo bởi tia *OA* và trục *Ox* có giá trị lượng giác **

|  |
| --- |
| double goc(Point A) {  double t=atan2(A.y,A.x);  if (t<0)  t=t+2\*acos(-1);  return t;  } |

2.6. Tam giác: Được xác định bởi 3 điểm A, B, C, có độ dài 3 cạnh thỏa mãn tất các điều kiện: *a+b>c; b+c>a; a+c>b.*

Diện tích tam giác được tính thông qua tích có hướng của vector như sau:



|  |
| --- |
| double sTriangle(Point A, Point B, Point C){  double s=(B.x-A.x)\*(C.y-A.y)-(B.y-A.y)\*(C.x-A.x);  return abs(s/2);  } |

Dựa vào diện tích tam giác, ta có thể tính khoảng cách từ điểm *C* đến đường thẳng *d* đi qua điểm *A, B* như sau:



|  |
| --- |
| double dist2(Point A, Point B, Point C){  return 2\*sTriangle(A,B,C)/dist(A,B);  } |

2.7. Đa giác: Đa giác là một đường gấp khúc khép kín. Trong lập trình, một đa giác được lưu bởi một dãy các đỉnh liên tiếp nhau A1, A2, … , AN. Khi đó **diện tích đại số** của một đa giác không tự cắt được định bởi công thức sau:



Nếu *S>0* thì ta đã liệt kê các đỉnh theo chiều ngược chiều kim đồng hồ.

Nếu *S<0* thì ta đã liệt kê các đỉnh theo chiều ngược chiều kim đồng hồ.

Còn *|S|* chính là diện tích của đa giác.

Công thức trên dễ dàng chứng minh bằng cách đi lần lượt theo các đỉnh biên của đa giác, tại mỗi đỉnh kẻ đường thẳng đứng xuống trục *Ox*, chia đa giác thành các hình thang vuông với hai đáy song song với trục tung *Oy* để tính diện tích.

2.8. Đường tròn: Tập hợp các điểm cách đều một điểm cho trước (gọi là tâm). Đường tròn biểu diễn thông qua tọa độ tâm và bán kính đường tròn. Đường tròn tâm *A* bán kính *r* kí hiệu toán học là: *(A,r).* Tương tự điểm thì có 2 cách biểu diễn với đường tròn như sau:

|  |  |
| --- | --- |
| struct circle {  Point A;  double r;  }; | typedef pair<pair<int,int>,double> circle; |

Một điểm nằm trong đường tròn khi khoảng cách của của điểm đó đến tâm đường tròn nhỏ hơn hoặc bằng bán kính. Ngược lại, khoảng cách tới tâm lớn hơn bán kính thì nó nằm ngoài đường tròn.

Hai đường tròn có điểm chung nếu khoảng cách giữa 2 tâm nhỏ hơn tổng hai bán kính và ngược lại.

Diện tích hình tròn: 

**3. Một số bài toán cơ bản của hình học phẳng**

3.1. Biểu diễn tuyến tính

Cho ba vectơ  hãy tìm hai số *p, q* để: 

Khi đó hai số *p, q* được tính bằng công thức sau:





+ Nếu *D* =  thì có duy nhất một cách biểu diễn tuyến tính vectơ qua vectơ 

+ Nếu *D = 0* thì hai vectơ song song với nhau, khi đó

*(p,q)* = *(nan; nan)* nếu  song song với 

*(p,q)* = *( inf, inf)* nếu  không song song với 

3.2. Giao điểm của hai đường thẳng

Trên mặt phẳng tọa độ Đề-các cho hai đường thẳng với phương trình tổng quát:





Hãy tìm giao điểm của hai đường thẳng đã cho

Đặt  ,  và  bài toán trở thành biểu diễn vectơ  qua tổ hợp tuyến tính của hai vectơ  và :  và biện luận cho giá trị giao điểm tìm được

3.3. Tìm giao điểm của hai đoạn thẳng

Cho bốn điểm *A, B, C, D* trên *Oxy*. Hãy cho biết hai đoạn thẳng AB và CD có giao nhau không, nếu có cho biết tọa độ giao điểm

Thuật toán 1:

+ Viết phương trình đường thẳng đi qua AB và CD

+ Tìm giao điểm M

+ Kiểm tra xem M có nằm trên AB hay không

Với  và  thì phương trình đường thẳng đi qua *A, B* là:







Tương tự như vậy ta tìm phương trình đường thẳng qua *CD* sau đó tìm giao điểm *M* bằng cách giải hệ hai phương trình bậc nhất hai ẩn sử dụng định thức, sau đó kiểm tra xem *M* có nằm giữa *A, B* hay không theo **(2.3).**

Nhận xét: Cách làm trên cần thực hiện nhiều phép tính nên không hiệu quả, dẫn đến tăng sai số.

Thuật toán 2:

Nếu M là giao điểm duy nhất của AB và CD thì  để: 



Như vậy, ta chỉ cần biểu diễn tuyến tính  qua  và , sau khi tìm được *p* và *q* (theo **3.1**) ta kiểm tra  và tìm tọa độ điểm *M* như sau:



Chương trình tìm giao điểm của 2 đoạn thẳng:

|  |
| --- |
| double p=(double)((C-A)^(C-D))/((B-A)^(C-D));  double q=(double)((C-A)^(B-A))/((C-D)^(B-A));  if (0<p & p<1 & 0<q & q<1) {  Point M=A+p\*(B-A); //Giao cua AB, CD là diem M  } |

Tương tự như trên ta có thể tìm giao điểm của 1 đường thẳng và 1 tia tương tự như tìm giao điểm của 2 đoạn thẳng như trên, chỉ khác điều kiện bây giờ là:



|  |
| --- |
| double p=(double)((C-A)^(C-D))/((B-A)^(C-D));  double q=(double)((C-A)^(B-A))/((C-D)^(B-A));  if (0<p & 0<q & q<1) {  Point M=A+p\*(B-A); //Giao cua tia AB voi doan CD là diem M  } |

3.4. Tìm hai điểm gần nhau nhất

Cho tập hợp Q gồm *n* điểm. Tìm cặp điểm trong Q có khoảng cách gần nhất.

Một thuật toán sơ đẳng nhất là ta đi tính tất cả các khoảng cách có thể được tạo ra từ 2 điểm trong Q, sau đó tìm khoảng cách lớn nhất trong đó. Độ phức tạp của thuật toán như vậy là O(*n2*).

|  |
| --- |
| double len=dist(a[1],a[2]);  for (int i=1; i<n; i++)  for (int j=i+1; j<=n; j++) {  if (dist(a[i],a[j])<len)  len=dist(a[i],a[j]);  } |

Có cách tiếp cận khác nhằm giảm độ phức tạp thuật toán còn O(*n*log*n*) sử dụng phương pháp chia để trị. Với bài toán kích thước *n* ta chia làm 2 bài toán con kích thước *n*/2 và kết hợp kết quả trong thời gian O(*n*).

**4. Bao lồi**

*Định nghĩa:* Cho tập hợp *n* điểm trên mặt phẳng. Bao lồi của tập này được định nghĩa là đa giác lồi có diện tích nhỏ nhất với các đỉnh thuộc tập đã cho và chứa tất cả *n* điểm ở bên trong hoặc biên của nó.

4.1. Thuật toán tìm bao lồi: Có nhiều thuật toán tìm bao lồi khác nhau như: thuật toán bọc gói, quét Graham,… Ở đây chúng ta xét một cách tìm bao lồi đơn giản hơn bằng Thuật toán **chuỗi đơn điệu** (**Monotone chain**) chỉ sử dụng các phép toán vector đơn giản.

Độ phức tạp chung của thuật toán O(*n*log*n*), trong đó sắp xếp mất O(*n*log*n*), xác định bao lồi mất O(*n*).



Giả sử *n* điểm đã cho là. Không mất tổng quát ta có thể xem các điểm này được xếp theo hoành độ tăng dần. Nếu hoành độ bằng nhau thì chúng được sắp xếp theo tung độ tăng dần. Khi đó ta chắc chắn *A1* (điểm cực trái) và *An* (điểm cực phải) thuộc bao lồi. Các điểm còn lại được chia thành hai phần:

- Các điểm ở nửa trên hay còn gọi là **chuỗi trên** (đường nét đứt trên hình), các điểm này thỏa mãn: 

- Các điểm ở nửa dưới hay còn gọi là **chuỗi dưới** (đường nét liền trên hình), các điểm này thỏa mãn:

|  |
| --- |
| //Point a[maxn]; đa giác ban đầu đã cho  //Point c[maxn]; các đỉnh bao lồi đánh số cùng chiều kim đồng hồ  sort(a+1,a+n+1);  // Tim bao loi  m=2;  c[1]=a[1];  c[2]=a[2];  for(int i=3; i<=n; i++) { ***//tìm chuỗi trên O(n)***  c[++m]=a[i];  while (m>2 && (c[m-1]-c[m-2])^(c[m]-c[m-1])>=0) {  m--;  c[m]=c[m+1];  }  }  int m0=m;  c[++m]=a[n-1];  for(int i=n-2; i>=1; i--) { ***//tìm chuỗi dưới O(n)***  c[++m]=a[i];  while (m-m0>1 && (c[m-1]-c[m-2])^(c[m]-c[m-1])>=0) {  m--;  c[m]=c[m+1];  }  }  --m; |

4.2. Kiểm tra điểm thuộc đa giác

Vấn đề đặt ra: Cho đa giác được xác định bởi *A*[maxn] được liệt kê lần lượt theo một thứ tự nào đó, kiểm tra xem điểm  có thuộc đa giác đó hay không.

Thuật toán 1: Áp dụng được với mọi loại đa giác không tự cắt.

- Nếu điểm *B* nằm trên biên của đa giác, được tính là thuộc đa giác.

- Nếu điểm *B* nằm trong đa giác thì kẻ một tia bất kì xuất phát từ *B* sao cho không đi qua đỉnh hoặc chứa cạnh nào của đa giác. Tia này sẽ cắt đa giác số lẻ lần. Để tìm giao điểm của tia và đoạn thẳng áp dụng phần 3.3.

Độ phức tạp thuật toán O(n).

Thuật toán 2: Chỉ áp dụng với đa giác lồi

- Nếu điểm *B* nằm trên biên của đa giác, được tính là thuộc đa giác.

- Nếu B nằm trong đa giác thì tổng diện tích tạo thành từ tam giác với 1 đỉnh là B và một cạnh là cạnh của đa giác bằng tổng diện tích đa giác.

Độ phức tạp là O(n).

Thuật toán 3: Chỉ áp dụng với đa giác lồi

Có cách khác để kiểm tra điểm có thuộc bao lồi hay không với độ phức tạp *O(logn)* bằng cách chặt nhị phân.

Thuật toán 4: Áp dụng đối với mọi loại đa giác không tự cắt

Dựa theo tổng góc định hướng tạo bởi B với các cặp đỉnh liên tiếp của đa giác. Nếu trị tuyệt đối của tổng đó bằng  thì B nằm trong đa giác.

Độ phức tạp: O(n).

**5. Một số bài tập**

**Bài 1: Hai hình vuông**

Cho hai hình vuông, một hình vuông có các cạnh song song với trục tọa độ, và hình kia có các cạch tạo với trục tọa độ góc 450, kiểm tra xem hai hình vuông có phần chung không. Hai hình được gọi là có phần chung khi tồn tại ít nhất một điểm thuộc cả 2 hình (có thể có 1 điểm chung hoặc vô số điểm chung).

Dữ liệu vào: gồm 2 dòng, mỗi dòng có 4 cặp số nguyên là tọa độ của 2 hình vuông tương ứng. Dòng thứ nhất là đỉnh của hình vuông có cạnh song song với trục tọa độ, dòng thứ hai chứa các đỉnh của tam giác lệch góc 450 với hệ trục tọa độ. Lưu ý các đỉnh được cho theo lần lượt theo thứ tự ngược chiều hoặc thuận chiều kim đồng hồ.

Dữ liệu ra: Nếu 2 hình vuông có phần chung in ra “YES” ngược lại in “NO”

Ví dụ:

|  |  |  |
| --- | --- | --- |
| 2SQUARES.INP | 2SQUARES.OUT | Hình cho ví dụ 1: |
| 0 0 6 0 6 6 0 6 1 3 3 5 5 3 3 1 | YES |
| 0 0 6 0 6 6 0 6 7 3 9 5 11 3 9 1 | NO |
| 6 0 6 6 0 6 0 0 7 4 4 7 7 10 10 7 | YES |
| 0 0 6 0 6 6 0 6  8 4 4 8 8 12 12 8 | YES |

Thuật toán 1: Trong bài trên, có thể quy về bài toán xác định điểm nằm trong đa giác bằng cách kiểm tra xem có điểm nào trong các điểm A,B,C,D có nằm trong hình vuông GHEF và có điểm nào trong các điểm G, H, E, F nằm trong hình vuông ABCD hay không. Nếu có tồn tại điểm như vậy thì 2 hình vuông có phần chung, ngược lại thì không.

Thuật toán 2: Kiểm tra các đoạn thẳng giao nhau, nếu có cạnh nào đó của hình vuông thứ nhất giao với cạnh nào đó của hình vuông thứ 2 thì chúng có điểm chung.

Thuật toán 3: Dựa theo khoảng cách tâm của 2 hình vuông chiếu trên trục Ox và Oy, nếu có 1 khoảng cách lớn hơn tổng của nửa AC và HE thì không giao nhau.

Sau đây là lời giải dựa theo giải thuật số 3 (giải thuật đơn giản nhất, còn giải thuật 1, 2 là bài tập áp dụng cơ bản cho học sinh):

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  int px,py,sx,sy,len,lo,a,b,ans;  int main() {  freopen("2SQUARES.INP","r",stdin);  freopen("2SQUARES.OUT","w",stdout);  for (int i=1; i<=4; i++) {  cin>>a>>b;  px+=a;  py+=b;  }  px/=2,py/=2;  lo=abs(a-px);  for (int i=0; i<4; i++) {  cin>>a>>b;  sx+=a;  sy+=b;  }  sx/=2,sy/=2;  len=abs(sx-a)+abs(sy-b);  ans=max(0,abs(px-sx)-lo)+max(0,abs(py-sy)-lo);  if (ans<=len)  cout<<"YES";  else  cout<<"NO";  return 0;  } |

**Bài 2: Đa giác có các góc bằng nhau**

Poly rất thích đa giác lồi, đặc biệt là các đa giác lồi mà có các góc bằng nhau nhưng độ dài cạnh lại khác nhau. Hãy tìm đa giác lồi có tính chất như trên giúp Poly.

Dữ liệu vào: chứa một số nguyên *n* ( 3 ≤  *n*  ≤ 100) số lượng đỉnh của đa giác.

Dữ liệu ra: In *n* dòng, chứa tọa độ (*xi , yi* ) của các đỉnh của đagiác theo thứ tự theo thứ tự ngược chiều kim đồng hồ. có độ dài cạnh của đa giác nằm trong đoạn [1,1000], độ dài này không nhất thiết là số nguyên. Sai số cho phép là 10-3. Nếu không tìm được đa giá như vậy thì in ra 0. Chỉ cần in ra các đỉnh của 1 đa giác bất kì.

Ví dụ:

|  |  |  |
| --- | --- | --- |
| **POLYGON.INP** | **POLYGON.OUT** | **Giải thích** |
| 8 | 1.000 0.000 7.000 0.000 9.000 2.000 9.000 3.000 5.000 7.000 3.000 7.000 0.000 4.000 0.000 1.000 |  |

Thuật toán: Do đa giác có *n* đỉnh, sẽ có *n* góc, mà các góc bằng nhau, do đó số đo của góc kề bù với cạnh đa giác sẽ lần lượt tăng một lượng so với trục *Ox*. Ví dụ ở trên, ta có  Ta chọn luôn điểm đầu tiên của đa giác là (0;0), các điểm sau sẽ tăng dần độ dài và lệch các góc tương ứng trên.

Chương trình như sau:

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  int n;  int main() {  freopen("POLYGON.INP","r",stdin);  freopen("POLYGON.OUT","w",stdout);  cin>>n;  if (n <= 4)  cout<<0;  else {  double alpha = 2 \* acos(-1) / n, x = 0, y = 0, len = 500;  for (int i = 1; i <= n - 2; i++) {  printf("%.6f %.6f\n", x, y);  len += 0.01;  x += len \* cos(alpha \* i), y += len \* sin(alpha \* i);  }  printf("%.6f %.6f\n%.6f 0.000\n", x, y, x - y / tan(alpha \* (n - 1)));  }  } |

**Bài 3: Trang trí đèn lồng**

Bờm được đánh giá là khéo tay nhất lớp, nhân dịp nhà trường chuẩn bị lễ hội đèn lồng cần treo rất nhiều đèn lồng ngoài sân trường. Cả lớp đã chuẩn bị các cây cọc để căng dây, từ đó sẽ treo đèn lồng lên những dây này. Do không có quá nhiều dây, nên Bờm cần phải xác định cách nối dây **hợp lý nhất** để các bạn có thể treo đèn được ở nhiều vị trí nhất. Qua quan sát, Bờm đưa ra ý tưởng sẽ căng dây từ cọc trên đỉnh cao nhất và đi vòng ngược chiều kim đồng hồ quanh sân qua các vị trí đã có cọc thành đường xoắn ốc, không có đoạn dây nào cắt nhau. Bạn hãy giúp Bờm làm việc đó nhé.

Dữ liệu vào:

- Dòng đầu là số T, phương án mà lớp định căng dây treo đèn lồng.

- Cứ mỗi dòng tiếp theo thể hiện một phương án, với số đầu là *n* điểm có cây cọc, tiếp theo là tọa độ *(x, y)* của các cọc tương ứng của *n* cọc đó trên sân.

Dữ liệu ra: T dòng ứng với T phương án căng dây của lớp, mỗi dòng là tọa độ cây cọc theo thứ tự mà Bờm sẽ căng dây đến khi hết các cọc cần căng dây.

*Ràng buộc:* 

Ví dụ:

|  |  |  |
| --- | --- | --- |
| **LANTERN.INP** | **LANTERN.OUT** |  |
| 2  1  0 0  6  6 3 -8 2 -4 3 -4 -5 -1 7 6 -5 | 0 0  -1 7 -8 2 -4 -5 6 -5 6 3 -4 3 |

Thuật toán: Dựa theo kĩ thuật xác định bao lồi như đã trình bày ở phần 4. Đi vòng theo chiều ngược chiều kim đồng hồ, tuy nhiên không khép kín bao lồi mà lại đi vòng tiếp vào trong để tìm các phần bao lồi bên trong. Những điểm đã được đưa chọn vào bao lồi thì đánh dấu để không kết hợp nó vào bao lồi bên trong. Thuật toán dừng lại khi hết các điểm được đánh dấu thuộc bao lồi.

Độ phức tạp thuật toán: *O(n2)*

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  struct Point {  int x, y;  };  Point p0;  Point nextToTop(stack < Point > &S) {  Point p = S.top();  S.pop();  Point res = S.top();  S.push(p);  return res;  }  int swap(Point &p1, Point &p2) {  Point temp = p1;  p1 = p2;  p2 = temp;  }  int distSq(Point p1, Point p2) {  return (p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y);  }  int CCW(Point p, Point q, Point r) {  int val = (q.y - p.y) \* (r.x - q.x) - (q.x - p.x) \* (r.y - q.y);  if (val == 0)  return 0;  return (val > 0)? 1: 2;  }  int compare(const void \* vp1, const void \* vp2) {  Point \* p1 = (Point \* )vp1;  Point \* p2 = (Point \* )vp2;  int o = CCW(p0, \* p1, \* p2);  if (o == 0)  return (distSq(p0, \* p2) >= distSq(p0, \* p1))? -1: 1;  return (o == 2)? -1: 1;  }  vector< Point > convexHull(vector< Point > points, int n) {  int ymin = points[0].y, min = 0;  for (int i = 1; i < n; i++) {  int y = points[i].y;  if ((y < ymin) || (ymin == y && points[i].x < points[min].x))  ymin = points[i].y, min = i;  }  swap(points[0], points[min]);  p0 = points[0];  qsort(&points[ 1 ], n-1, sizeof(Point), compare);  int m = 1;  for(int i = 1; i < n; i++) {  while(i < n-1 && CCW(p0, points[i], points[i+1]) == 0)  i++;  points[m] = points[i];  m++;  }  if (m < 3)  return points;  stack< Point > S;  S.push(points[0]);  S.push(points[1]);  S.push(points[2]);  for(int i = 3; i < m; i++) {  while (CCW(nextToTop(S), S.top(), points[i]) != 2)  S.pop();  S.push(points[i]);  }  vector< Point > result;  while (!S.empty()) {  Point p = S.top();  result.push\_back( p );  S.pop();  }  return result;  }  void display(vector< Point > v) {  int start = 0;  int maxY = INT\_MIN;  for(int i = 0; i < v.size(); i++) {  if(maxY < v[i].y) {  maxY = v[i].y;  start = i;  }  }  for(int i = 0; i < v.size(); i++) {  int in = start - i;  if(in < 0)  in += v.size();  cout << v[in].x << " " << v[in].y << " ";  }  }  void setDifference(vector< Point > &pts, vector< Point > &result) {  vector< Point > tmp(pts.begin(), pts.end());  pts.clear();  pts.resize(0);  for(vector< Point >::iterator a = tmp.begin(); a != tmp.end(); a++) {  bool flag = false;  for(vector< Point >::iterator it = result.begin(); it != result.end(); it++) {  if(a->x == it->x && a->y == it->y) {  flag = true;  }  }  if( ! flag ) {  pts.push\_back( \* a );  }  }  }  int main() {  int t;  cin >> t;  while(t--) {  int n;  cin >> n;  vector< Point > pts(n);  for (int i = 0; i < n; i++)  cin >> pts[i].x >> pts[i].y;  while (n) {  vector< Point > result = convexHull(pts, n);  display(result);  setDifference(pts, result);  n = pts.size();  }  cout << endl;  }  return 0;  } |

**Bài 4: Đặt trạm BTS1**

BTS là một cơ sở hạ tầng viễn thông được sử dụng nhằm tạo thông tin liên lạc không dây giữa các thiết bị thuê bao viễn thông tại các khu dân cư. Các thiết bị thuê bao có thể là điện thoại di động, thiết bị internet không dây,… Nhằm nâng cấp dịch vụ mạng điện thoại di động tại đất nước XYZ, người ta mới nghĩ ra phương án hợp nhất tất cả các nhà cung cấp dịch vụ di động lại thành một thể thống nhất lại các vị trí đặt trạm BTS. Là kĩ sư phụ trách kĩ thuật cho quá trình này, Bờm tự hỏi: với mỗi vị trí đặt trạm BTS sau khi hợp nhất có bao nhiêu vị trí khu dân cư sẽ nằm ngoài vùng phủ sóng của trạm mới khi đã biết tọa độ của các vị trí này. Coi vùng phủ sóng di động của trạm BTS dạng hình tròn với tâm là tọa độ đặt trạm BTS.

Dữ liệu vào:

- Dòng đầu tiên ghi tọa độ, bán kính phủ sóng trạm BTS mới.

- Dòng tiếp theo ghi số lượng thuê bao di động *n (n<=106)*

- Các dòng tiếp theo là tọa độ của các điểm dân cư, 

Dữ liệu ra:

- Số lượng thuê bao di động nằm ngoài vùng phủ sóng của trạm BTS mới.

Ví dụ:

|  |  |  |
| --- | --- | --- |
| BTS1.INP | BTS1.OUT | Giải thích |
| 0 0 5  5  2 2 -3 3 4 -3 -6 -6 1 1 | 1 | Vị trí tọa độ (-6;-6) nằm ngoài vùng phủ sóng. |

Thuật toán: Khá đơn giản, chỉ cần kiểm đếm các điểm có khoảng cách đến trạm BTS lớn hơn vùng phủ sóng.

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  const int maxn=1e6;  struct Point {  double x, y;  };  Point a,b;  int dem,n,r;  int distSq(Point A, Point B) { // binh phuong khoang cach  return (A.x-B.x)\*(A.x-B.x)+(A.y-B.y)\*(A.y-B.y);  }  int main() {  freopen("BTS1.INP","r",stdin);  freopen("BTS1.OUT","w",stdout);  cin>>a.x>>a.y>>r>>n;  dem=0;  for (int i=1; i<=n; i++) {  cin>>b.x>>b.y;  if (distSq(a,b)>r\*r)  dem++;  }  cout<<dem;  } |

**Bài 5: Đặt trạm BTS2**

Đặt vấn đề tương tự như bài BTS1, tuy nhiên giờ thì trong tay kĩ sư Bờm có trạm BTS với công nghệ mới, nhưng số lượng quá ít, hiện tại Bờm mới chỉ có trong tay một trạm duy nhất là loại tiên tiến, Bờm cần tính toán để dùng thiết bị công nghệ mới hiệu quả, do vậy việc cần làm là tính toán vị trí đặt trạm BTS mới sao cho phủ sóng được nhiều khu dân cư nhất. Trạm BTS có thể phủ sóng đến vị trí mà có khoảng cách đến trạm sai lệch *10-4* so với bán kính phủ sóng R của trạm.

Dữ liệu vào:

- Dòng đầu tiên là bán kính *R* mà trạm BTS mới có thể phủ sóng.

- Dòng tiếp theo là **, số điểm khu dân cư.

- Các dòng tiếp theo là tọa độ của các điểm dân cư, 

Dữ liệu ra:

- Tọa độ của điểm đặt trạm BTS, có thể có nhiều tọa độ thích hợp, chỉ cần in ra một vị trí duy nhất.

Chú ý vị trí cách trạm BTS khoảng  vẫn thu được sóng.

Ví dụ:

|  |  |  |
| --- | --- | --- |
| BTS2.INP | BTS2.OUT | Giải thích |
| 5  5  2 2 -3 3 4 -3 -6 -6 1 1 | 0 0 | Đặt trạm mới tại (0;0) phục vụ được 4 vị trí. |

Thuật toán:

Chọn ngẫu nhiên 2 điểm, dựng được 2 đường tròn bán kính *R*. Khi đó có 2\*n\*(n+1) đường tròn bán kính R. Đường tròn nào chứa được nhiều điểm nhất chính là đường tròn cần tìm.

Độ phức tạp của thuật toán là O(n3). Trong đó O(n2) là để xác định các đường trong khả dĩ, O(n) để kiểm tra xem có bao nhiêu điểm nằm trong vùng phủ sóng.

Chương trình:

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  #define x first  #define y second  typedef pair<double, double> Point;  double r,n;  Point a[305];  Point operator +(Point A, Point B) {  return Point(A.x+B.x,A.y+B.y);  }  Point operator -(Point A, Point B) {  return Point(A.x-B.x,A.y-B.y);  }  double operator \*(Point A, Point B) {  return A.x\*B.x+A.y\*B.y;  }  Point operator \*(double k, Point A) {  return Point(k\*A.x,k\*A.y);  }  Point operator /(Point A, double k) {  return Point(A.x/k,A.y/k);  }  double distSq(Point A, Point B) {  return (A.x-B.x)\*(A.x-B.x)+(A.y-B.y)\*(A.y-B.y);  }  Point findO(Point A, Point B) { //Tim tam O cua duong tron  double \_x,\_y;  if (4\*r\*r<distSq(A,B))  return Point(200,200);  if (distSq(A,B)<1e-12)  return Point(200,200);  Point M=(A+B)/2;  \_x=M.x+(r\*r-distSq(A,B)/4)\*(A.y-B.y)\*(A.y-B.y)\*distSq(A,B);  if (A.y==B.y)  \_y=M.y+sqrt(r\*r-distSq(A,B)/4);  else  \_y=(M.x-\_x)\*(A.x-B.x)/(A.y-B.y);  return Point(\_x,\_y);  }  int dem(Point O) { //Dem nhung diem nam trong duong tron  int cnt=0;  for (int i=1; i<=n; i++) {  if (distSq(O,a[i])<r\*r+1e-5)  cnt++;  }  return cnt;  }  Point \_max; //Luu toa do diem dat tram BTS de phu song nhieu nhat  int main() {  int cnt;  freopen("BTS2.INP","r",stdin);  freopen("BTS2.OUT","w",stdout);  cin>>r>>n;  for (int i=1; i<=n; i++)  cin>>a[i].x>>a[i].y;  cnt=0;  for (int i=1; i<n; i++)  for (int j=i+1; j<=n; j++) {  Point O=findO(a[i],a[j]);  if (O!=Point(200,200)) {  if (dem(O)>cnt)  \_max=O;  if (dem(a[i]+a[j]-O)>cnt)  \_max=a[i]+a[j]-O;  }  }  cout<<\_max.x<<" "<<\_max.y;  } |

**6. Một số bài khác**

**Bài 6: LOA**

Trong kỳ thi học sinh giỏi môn Tin có N học sinh tại sân trường XYZ. Có thể coi sân trường là một hệ trục tọa độ, mỗi học sinh có một tọa độ xi, yi. Ban tổ chức cần đứng tại một vị trí thật thuận tiện để có thể loa thông báo cho N em học sinh. Cho rằng loa có thể thông báo cho các em học sinh trong phạm vi R nghe thấy. Hãy xác định R nhỏ nhất để có thể thông báo cho N học sinh để có thể đi thi đúng giờ.

Dữ liệu vào: LOA.INP

+Dòng 1 ghi *n* (*n* ≤100)

+*n* dòng tiếp theo ghi hai số nguyên  thể hiện tọa độ của một điểm

Dữ liệu ra: LOA.OUT

Một số thực với 3 chữ số thập phân là kết quả cần tìm.

Ví dụ:

|  |  |
| --- | --- |
| **LOA.INP** | **LOA.OUT** |
| 3  0 0  0 1  1 1 | 0.707 |

Thuật toán: Duyệt 3 điểm khác nhau và tìm bán kính nhỏ nhất chứa 3 điểm đó. Khi đó bán kính R cần tìm chính là bán kính R lớn nhất.

Chương trình: Tham khảo trong thư mục kèm theo chuyên đề này.

**Bài 7:** **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:DULICH.INP

+Dòng 1 ghi *n* (*n* ≤100)

+*n* dòng tiếp theo ghi hai số nguyên  thể hiện tọa độ của một điểm

Dữ liệu ra: 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).

Ví dụ:

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

Thuật toán: Mấu chốt bài này là tính khoảng cách giữa hai điểm du lịch là khoảng cách giữa hai điểm trong mặt phẳng tọa độ Đề-các. Ta thử đặt bến tại các điểm và tính khoảng cách đến các điểm còn lại, lưu lại điểm đặt và tổng khoảng cách ngẵn nhất từ điểm đó đến các điểm còn lại.

Chương trình: Tham khảo trong thư mục kèm theo chuyên đề này.

**Bài 8: CHUNG CƯ**

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: CHUNGCU.INP

+Dòng 1 ghi *n* (*n* ≤100)

+*n* dòng tiếp theo ghi hai số nguyên  thể hiện tọa độ của một điểm

Dữ liệu ra**:** CHUNGCU.OUT

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

Ví dụ:

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

Thuật toán:

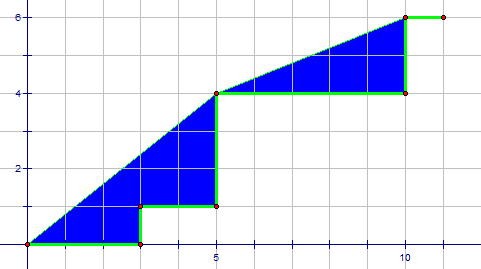
+ Sắp xếp các điểm theo tung độ

+ 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.

Chương trình: Tham khảo trong thư mục kèm theo chuyên đề này.

**Bài 9**. **ĐA GIÁC**

 Trên mặt phẳng lưới toạ độ Đê các, xuất phát từ điểm (0, 0) người ta vẽ một đường gấp khúc có các cạnh song song với trục toạ độ theo quy tắc sau: bút vẽ được điều khiển bằng chương trình là một xâu các ký tự U, R. Gặp lệnh U bút vẽ sẽ chuyển lên trên một đơn vị, còn khi gặp lệnh R bút vẽ sẽ chuyển sang phải một đơn vị. Khi hết chương trình bút vẽ được kéo thẳng về gốc toạ độ. Hình bên tương ứng với chương trình vẽ là RRRURRUUURRRRRUUR.

Yêu cầu: Từ điểm (0,0) nhìn theo đường chéo đến một điểm A mà các điểm còn lại đều nằm phía dưới đường chéo ta được một hình tô đậm, lặp lại như vậy nhưng điểm nhìn là từ điểm A .. Tính tổng diện tích phần tô

Dữ liệu vào: POLY.INP

Gồm nhiều dòng, mỗi dòng chứa một xâu các ký tự R, U, xác định một chương trình vẽ. Bút vẽ luôn chuyển động trong phạm vi lưới kích thước 1000\*1000. Ch­¬ng tr×nh kÕt thóc b»ng lÖnh S.

Dữ liệu ra: POLY.OUT

Các diện tích tìm được, mỗi kết quả trên một dòng, là một số thực với 3 chữ số sau dấu chấm thập phân.

Ví dụ:

|  |  |
| --- | --- |
| POLY.INP | POLY.OUT |
| RRRURRUUURRRRRUURS  RUURS | 13.000  1.000 |

Thuật toán:

+Xác định tọa độ các điểm khi thay đổi U, R và dừng lại khi gặp S

+ Bắt đầu từ (0,0) tìm A điểm mà từ đó mọi điểm đều nằm phía dưới OA. Ta được phần tô thứ nhất và có thể tính diện tích và lặp lại như vậy bắt đầu từ A.

Chương trình: Tham khảo trong thư mục kèm theo chuyên đề này.

**Bài 10.** **GÓC LỚN NHẤT**

Cho N điểm A1,A2,A3,...,AN trên mặt phẳng. Các điểm đều có toạ độ thực và không có 3 điểm bất kỳ trong chúng thẳng hàng. Khi đó ta sẽ xác định một đa giác không tự cắt có đỉnh là một trong số các điểm đã cho và chứa tất cả những điểm còn lại và có chi vi nhỏ nhất. Trong đa giác đó hãy tìm góc lớn nhất tính bằng rad

**Dữ liệu vào:** GOCMAX.INP

Gồm N+1 dòng:

+ Dòng 1: Chứa số N ( N 200)

+ Các dòng tiếp theo mỗi dòng chứa 2 số thực là toạ độ của một điểm.

**Dữ liệu ra:**GOCMAX.OUT

Kết quả là một số thực có 3 chữ số phần thập phân

|  |  |
| --- | --- |
| GOCMAX.INP | GOCMAX.OUT |
| 3  0 0  5 0  0 2.5 | 1.571 |

Thuật toán:

+ Tìm bao lồi chứa N điểm đó

+ Tìm góc lớn nhất của bao lồi đó

Chương trình: Tham khảo trong thư mục kèm theo chuyên đề này.

**Một số bài tự luyện khác:**

- Trên trang Codeforce: <https://codeforces.com/problemset/tags/geometry>

- SPOJ:

<http://vnoi.info/problems/show/MILITARY/>

<http://vnoi.info/problems/show/HEADQRT/>

<http://vnoi.info/problems/show/KMIX/>

<http://vn.spoj.com/problems/VMCXG/>

<http://vn.spoj.com/problems/PRAVO/>

<http://vn.spoj.com/problems/VORAIN/>

<http://vn.spoj.com/problems/BALLGMVN/>

- <https://open.kattis.com/problems/joiningnetwork>

- Sách “**Tài liệu chuyên tin học Bài tập quyển 3”** (có 20 bài)

**7. KẾT LUẬN**

Bài tập hình học có thể có nhiều cách giải sáng tạo nếu học sinh được trang bị kiến thức toán học tốt. Những kiến thức về Hình học hoàn toàn có thể trang bị cho học sinh ngay từ khi học lớp 10, vì khi đó các em đã được học về Vector trong Toán.

Trong giải toán hình học, cần chú ý việc xử lý sai số trong tính toán. Tránh tối đa tính toán với số thực, cố gắng đưa về tính toán với số nguyên lâu nhất có thể trong quá trình giải bài.