ICPC勉強会 ~幾何入門~

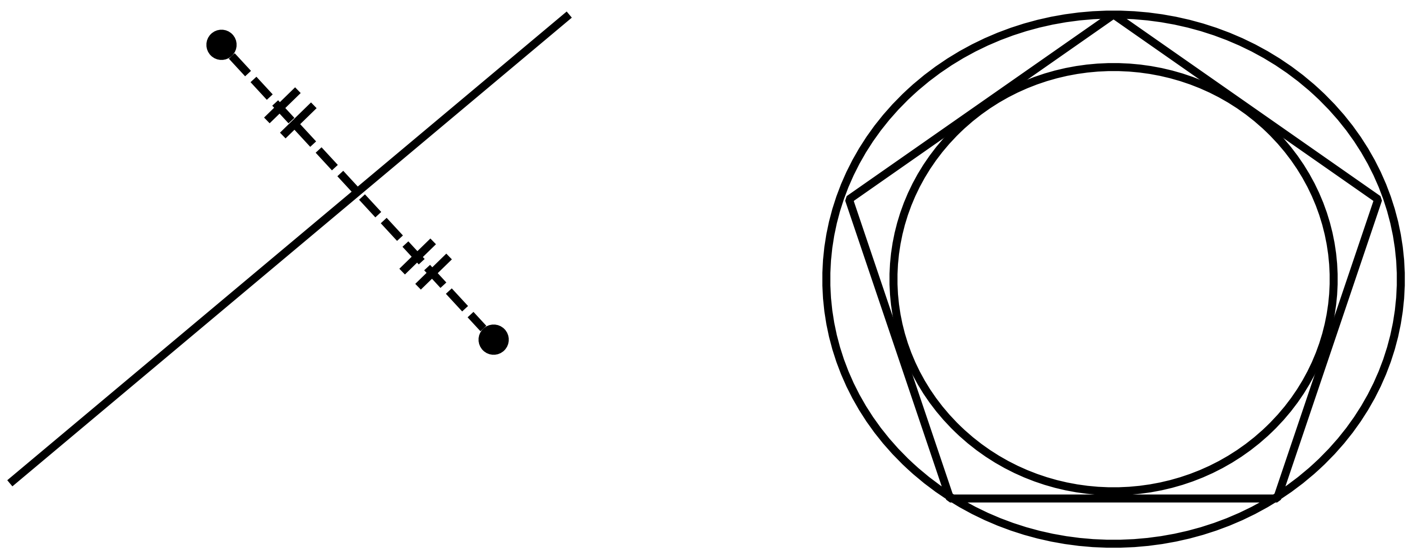
アルゴリズム研究室　井上祐馬

内容

- 計算幾何
  - 座標・ベクトルの表現
    - 点間の距離
    - 内積・外積
    - 点の位置関係
  - 線分・直線の表現
    - 距離・交差判定

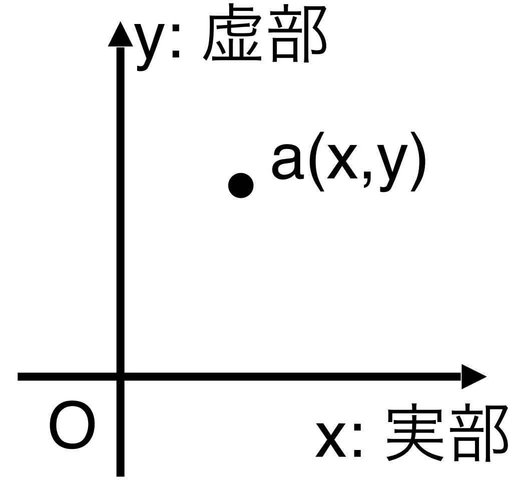
計算幾何

- x,y座標系上での点や図形などを考え、位置関係や指定動作の結果を答える
- 今回は2次元の場合のみを扱う



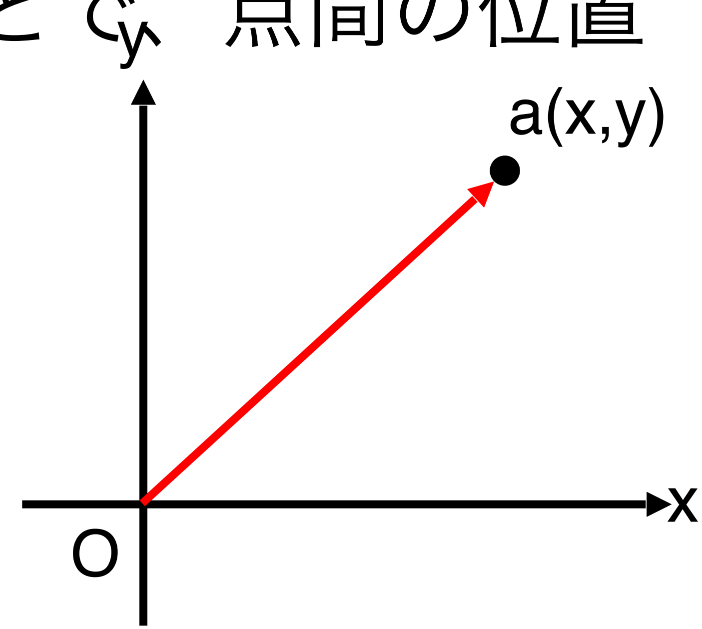
座標の表現

- 点はC++ STLのcomplex型を用いる
  - タイプ量削減、直感的理解のため、
    - typedef complex<double> Point;
  - などとする
- x座標はreal()、y座標はimag()で参照
  - 点a(x,y)は、
    - Point a;
    - a.real() = x; a.imag() = y;
  - と定義すればよい
- 複素平面と考える



ベクトルの表現

- 点型はそのままベクトルを表しているとも見れる
  - 原点を始点、点aを終点とするベクトル
- ベクトルと見なすことで、点間の位置関係などが計算可能



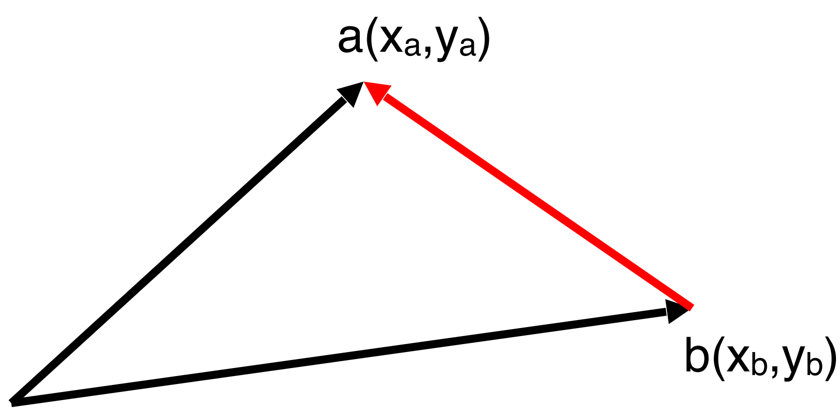
complex型を用いる利点

- 単純にライブラリを写す手間が減る
  - complex型なら加算・乗算・スカラー倍・長さなどが既に実装済み
- 使いやすくするために自分で点を表す構造を定義しても、もちろんよい

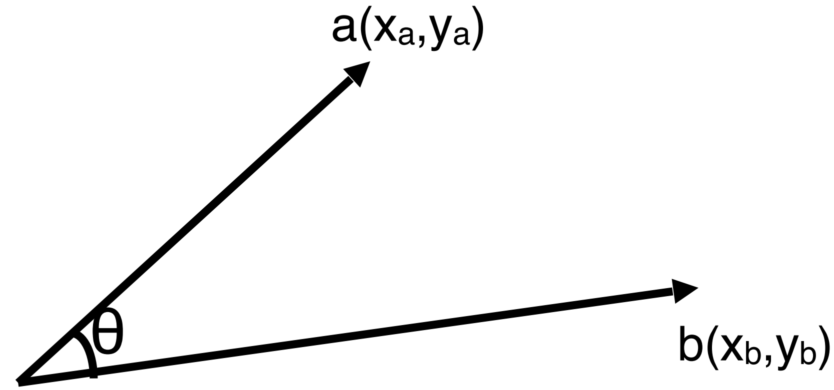
```
struct Point{
  double x,y;
 Point(double a, double b){ x = a; y = b; }
};
```

点と点の距離

- 点a,b間の距離は、ベクトルa-bの長さに等しい
  - double dis(Point a, Point b){ return abs(a-b); }



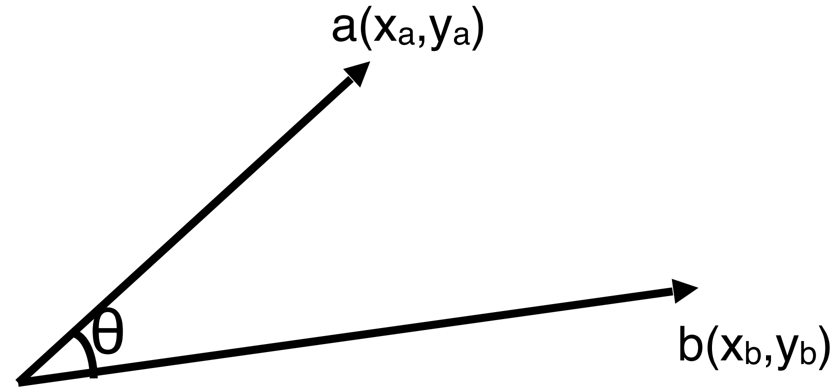
内積



- 2つのベクトル
  a(x_a, y_a)とb(x_b, y_b)の内積dot(a,b)は
  - dot(a,b) = x_a ・ x_b + y_a ・ y_b
- また、a,bのなす角をθとおくと、
  - dot(a,b) = |a| ・ |b| ・ cos θ
- 内積を用いるとcos θが求められる

```
double dot(Point a, Point b){
  retrun a.real()*b.real() + a.imag()*b.imag();
}
```

外積



- 2つのベクトル

  $a(x_a, y_a)$とb(x_b, y_b)の外積cross(a,b)は
  - cross(a,b) = x_a・y_b − y_a・x_b
- また、a,bのなす角をθとおくと、
  - cross(a,b) = |a|・|b|・sinθ
- 外積を用いるとsinθが求められる

```
double dot(P a, P b){
  retrun a.real()*b.imag() − a.imag()*b.real();
}
```

三角形の面積

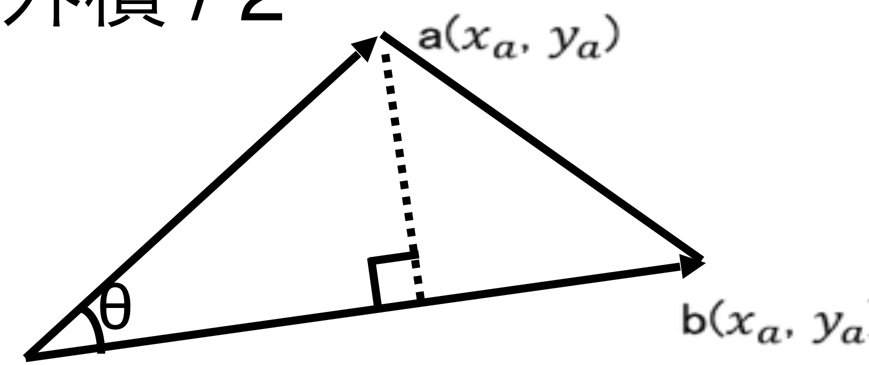
- 外積 = |a| * |b| * sinθ

  = |b| * |bとx_aの距離|

  = 底辺 × 高さ

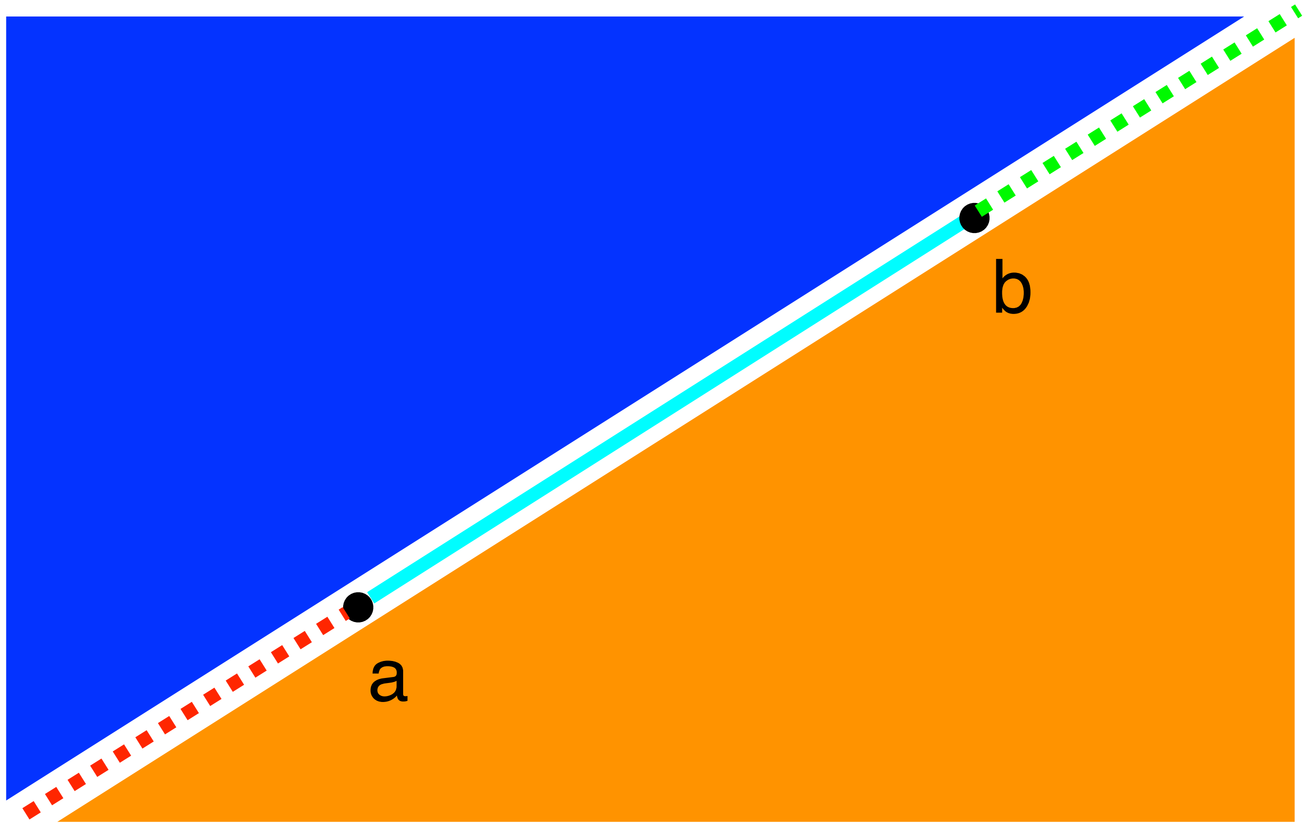
- 三角形の面積 = 底辺 × 高さ / 2

  = 外積 / 2



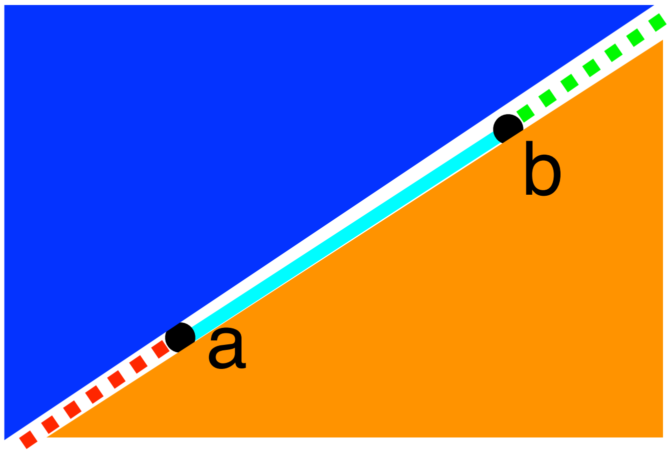
点の位置関係

- 2点a,bから見た点cの位置関係
- ccwと呼ばれる関数がよく用いられる



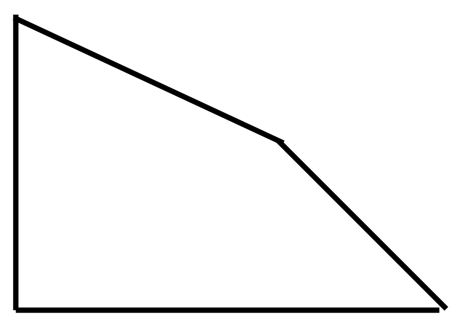
ccw

```
int ccw(Point a, Point b, Point c){
    if(cross(b-a,c-a)>EPS)return 1;
    if(cross(b-a,c-a)<-EPS)return -1;
    if(dot(b-a,c-a)<-EPS)return 2;
    if(abs(b-a)+EPS<abs(c-a)return -2;
    return 0;
}
```

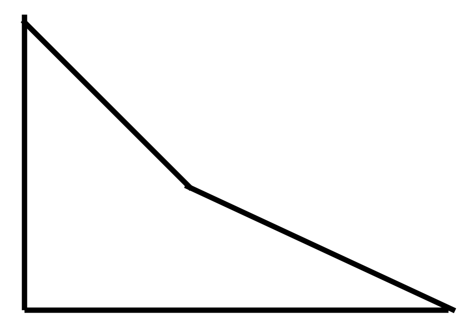


問題: AOJ0035 Is It Convex?

- 四角形が与えられる
- 凸な四角形かどうか判定せよ



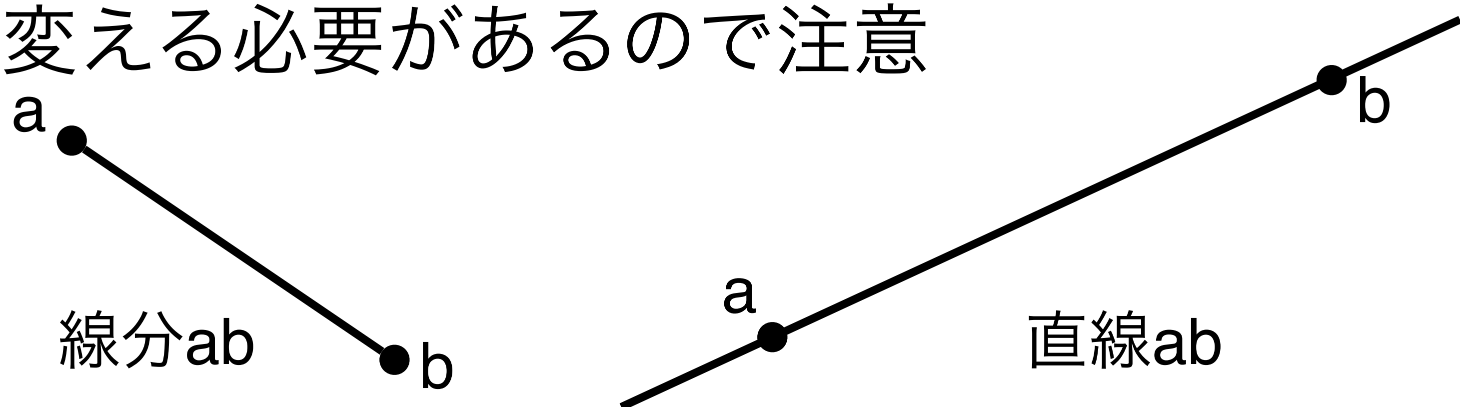
凸多角形



凹多角形

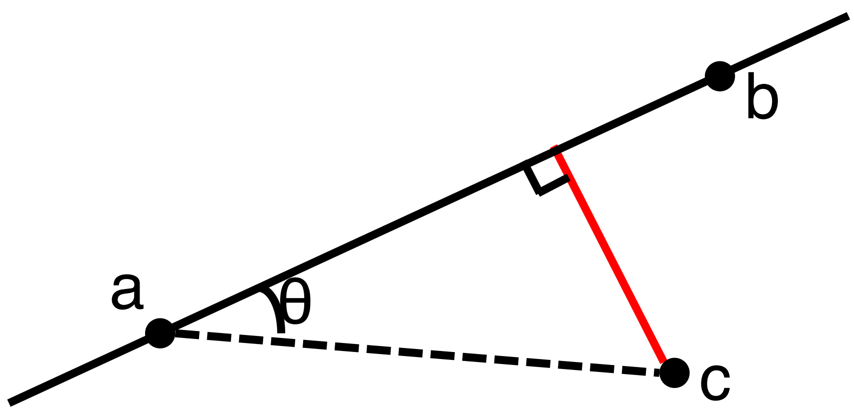
線分・直線の表現

- 線分の2端点を保持しておくことで、線分を表現する
- 直線が通過する任意の2点を保持しておくことで、直線を表現する
    - typedef pair<Point, Point> Line;
- 線分も直線も同じ構造で保持できるが扱いは変える必要があるので注意



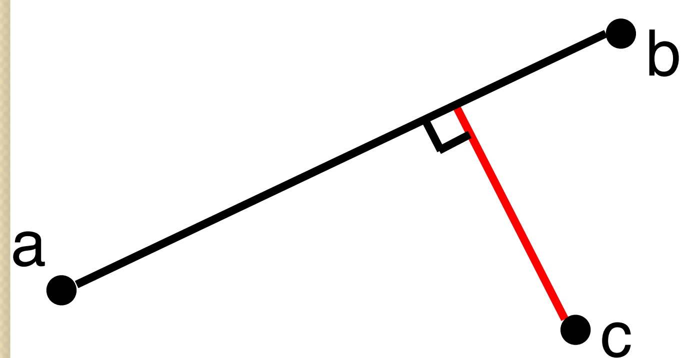
点と直線の距離

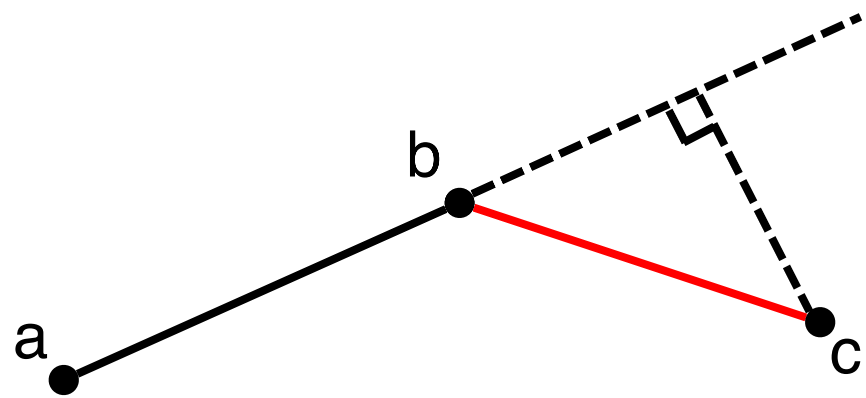
- 直線abと点cとの距離は、cから直線に下ろした垂線の足と、cとの距離
- dis = |c−a|・|sin θ |が成り立つ
- よって、
  - dis = $\dfrac{|cross(c-a,b-a)|}{|b-a|}$



点と線分の距離

- 線分abと点cとの距離は、基本的に直線a,bと点cの距離に等しい
- ただし垂線の足が線分上にないときはより近い端点との距離になる

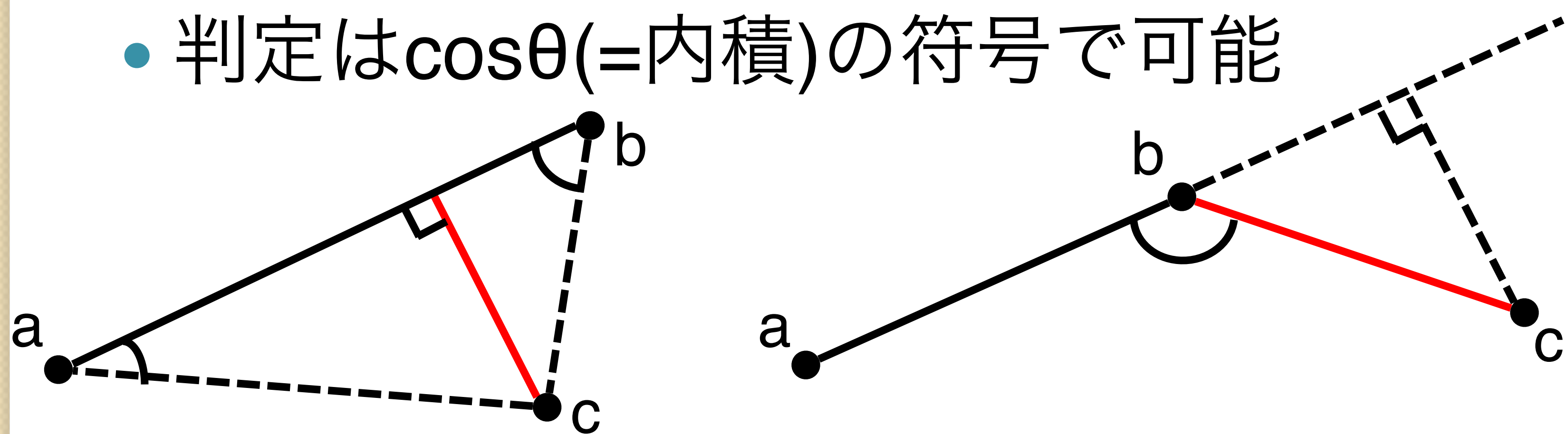




点と線分の距離

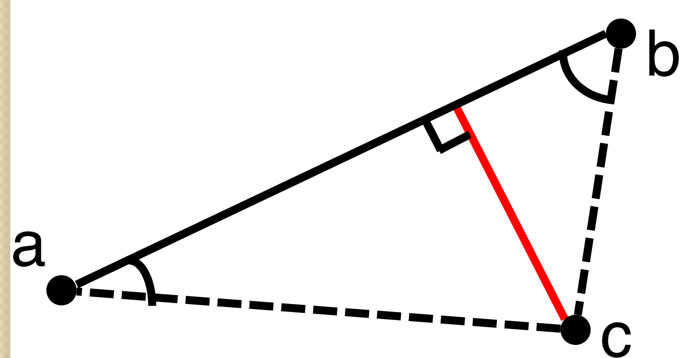
- 垂線の足がa側にある場合: ∠BAC > 90°
- 垂線の足がb側にある場合: ∠ABC > 90°
- 垂線の足が線分上にある場合:

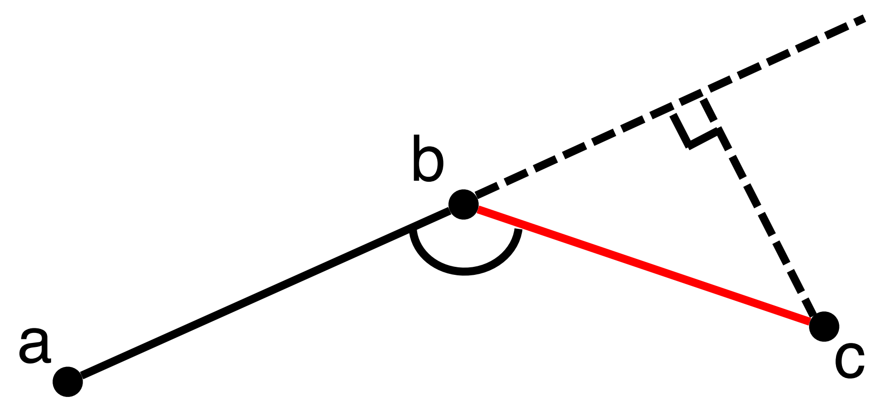
  ∠BAC <= 90°かつ∠ABC <= 90°
- 判定はcosθ(=内積)の符号で可能



点と線分の距離

```
double seg_to_point_dis(Line l, Point p){
  Point a = l.first, b = l.second, c = p;
  if(dot(b-a,c-a)<EPS)return abs(c-a);
  if(dot(a-b,c-b)<EPS)return abs(c-b);
  return line_to_point_dis(l,p);
}
```





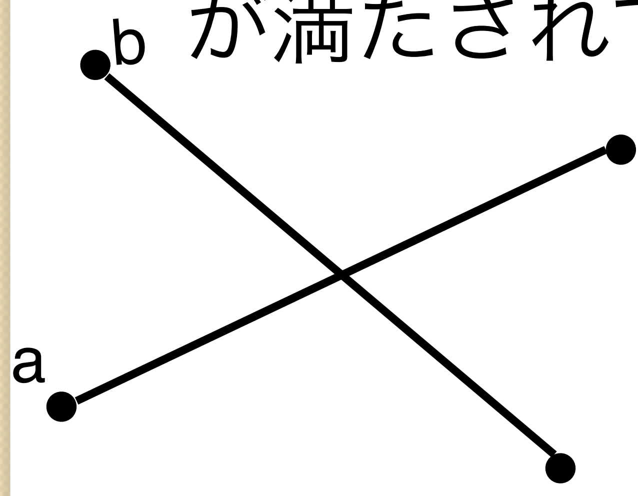
線分と線分の距離

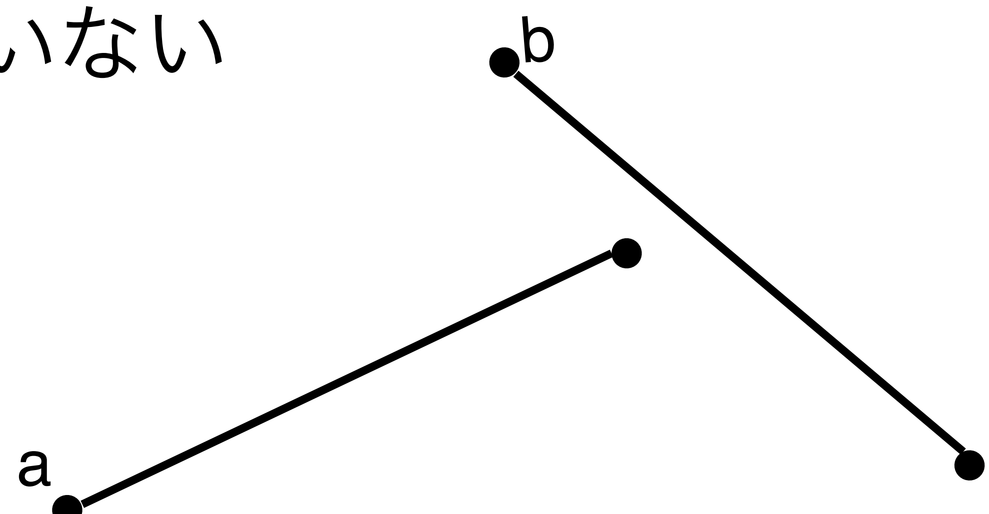
- 片方の端点からもう片方の線分への距離の最小値が線分間の距離

```
double seg_to_seg_dis(Line a, Line b){
  double res = seg_to_point_dis(a, b.first);
  res = min(res, set_to_point_dis(a, b.second) );
  res = min(res, set_to_point_dis(b, a.first) );
  res = min(res, set_to_point_dis(b, a.second) );
  return res;
}
```

線分の交差判定

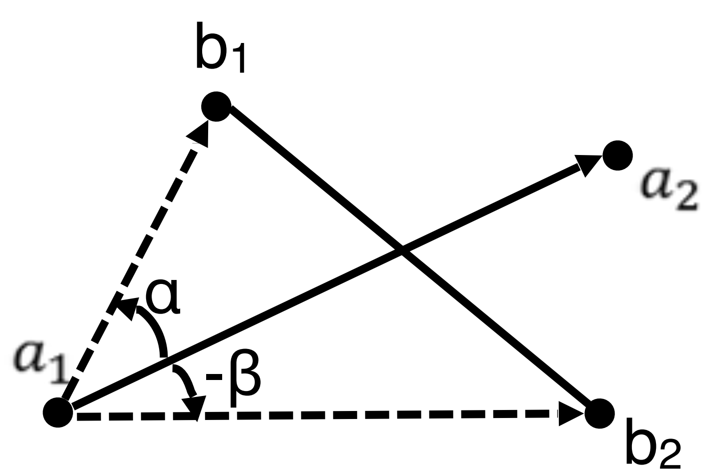
- 線分aとbが交差しているとき、
  - aの2つの端点は直線bの両側にある
  - bの2つの端点は直線aの両側にある
- 逆に、交差していないときはどちらかが満たされていない





線分の交差判定

- 線分a,bが交差する必要十分条件は、
  - bの2つの端点がaの両側にある
  - aの2つの端点がbの両側にある
- 両側にある場合、ccwの積が負になる

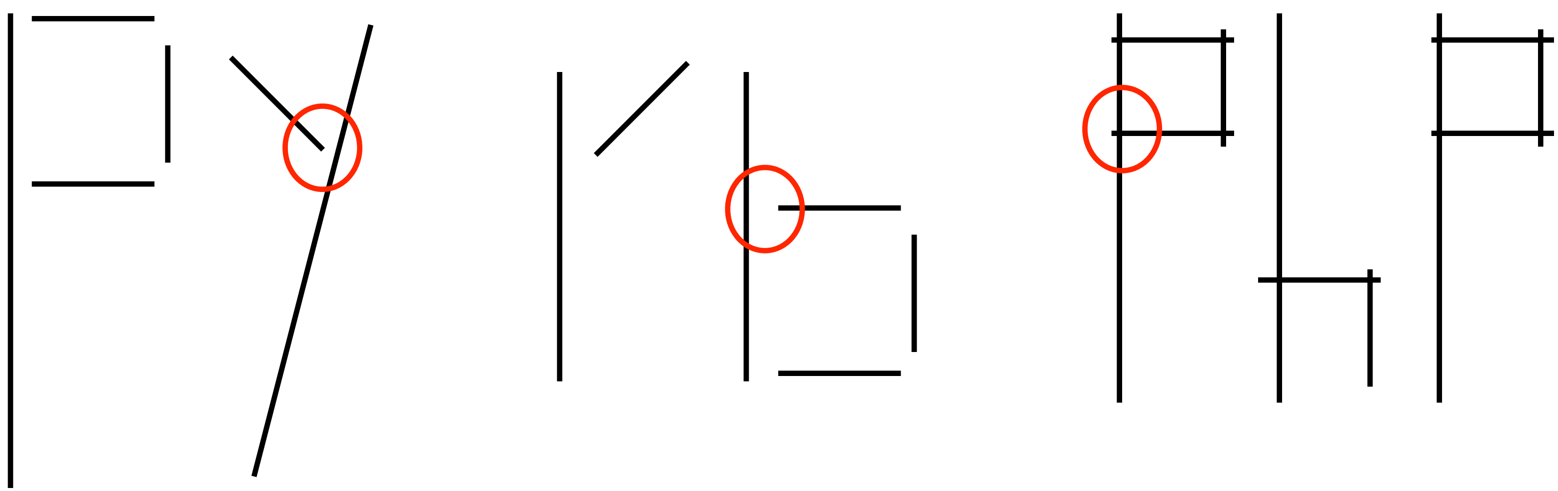


線分の交差判定

```
bool is_cross(Line a, Line b){
  if(ccw(a.first,a.second,b.first)*
     ccw(a.first,a.second,b.second) <= 0 &&
     ccw(b.first,b.second,a.first)*
     ccw(b.first,b.second,a.second) <= 0){
       return true;
  }
  return false;
}
```

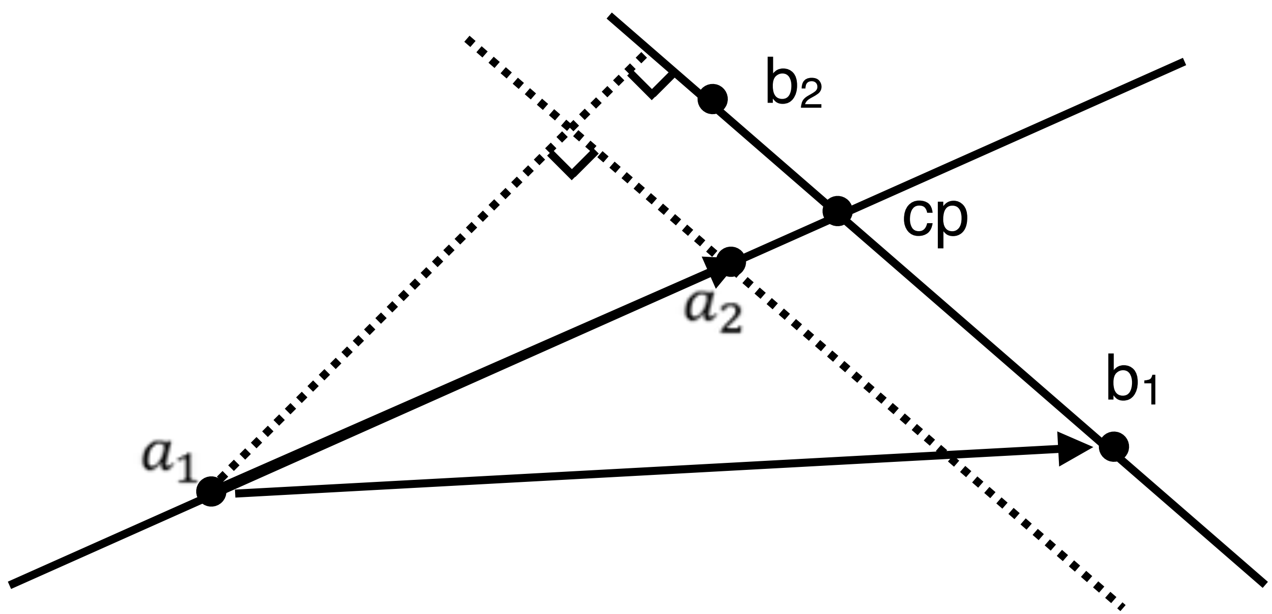
問題: AOJ2351 Closest Segment Pair

- N本の線分が与えられる
- 最も近い線分同士の距離を出力せよ
- $2 \leqq N \leqq 10^5$
- $0 \leqq x_i, y_i \leqq 100$



直線の交点

- aをベクトル, bを直線とみる
- aを交点まで伸ばせば交点がわかる
- 伸ばす比率は垂線までの距離でわかる

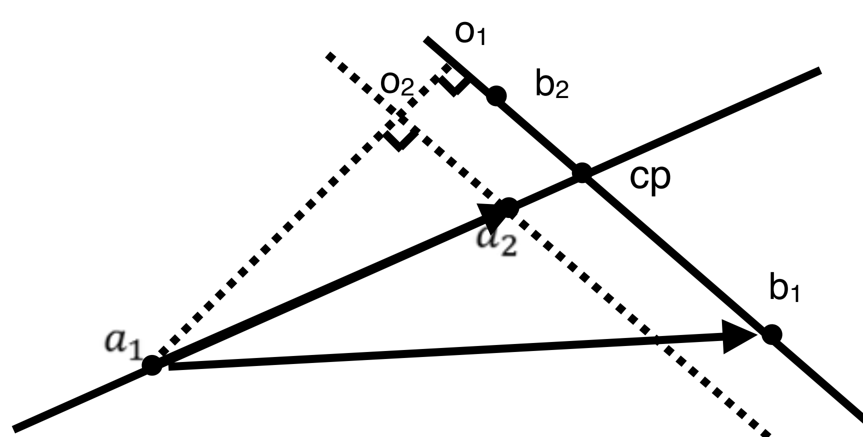


直線の交点

```
Point cross_point(Line a, Line b){
  double d1 = cross(b.second - b.first,
                    b.first - a.first);
  double d2 = cross(b.second - b.first,
                    a.second - a.first);
  return a.first + (a.second - a.first) * d1/d2;
}
```

d1 = |b2-b1| * |b1-a1| * sinx
= |b2-b1| * |o1-a1|
d2 = |b2-b1| * |a1-a1| * siny
= |b2-b1| * |o2-a1|

d1/d2 = |o1-a1| / |o2-a1|
= |cp-a1| / |a1-a1|



問題: AOJ0187 Stoning Fortune

- 3本の線分の交点3つを結んでできる三角形の面積を求めよ
- 同一直線上に同じ線分がある、交差しない線分がある、3点で交わる場合は三角形なしとする
- $-1000 \leqq x_i, y_i \leqq 1000$