

# プログラミングコンテストに おける

## 計算幾何入門

秀 郁未 (東京大学農学部 3 年)

# 強い人へ

- ・ N点(多め)与えられる。N点をうまくつないで次のような多角形を作れ。
  - 与えられたN点は全てこの多角形の周上にある
  - この多角形の面積は, N点の凸包の面積の半分以上
- ・ GCJ2013 Round 3 の Rural Planning なので暇なら通しておいてください
- ・ あとで解説します

# はじめに

- ・ 幾何問題？

# 幾何問題とは？

- ・ プログラミングコンテストで度々、平面上 / 空間上で図形を処理する問題というのが登場します。
- ・ 例: 三角形の座標が与えられるので、内心の座標を求めよ。
- ・ 明示的に登場しなくても、ある値を平面上にプロットすると解けるというものも多いです。

# どこで出るの？

- ・ 有名な例: ICPC
  - ・ 国内では毎年頻出 (しかも重くて難しい)
  - ・ World Finals 2016 でも出題された
- ・ JOI? IOI?
  - ・ 時々出る
  - ・ Constellation 1,2, / IOI2007 Flood とか…

# IOI シラバス

- ・ 浮動小数は範囲外？
- ・ 全部有理数で処理できる問題なら出るかもしれない

# 目次

- ・ 1. 平面幾何の構成要素の基本部分
- ・ 2. 基本的な計算幾何のアルゴリズム
- ・ 3. 応用的な使い方・実践例
- ・ 4. 注意点

# 目次

- ・ 1. 平面幾何の構成要素の基本部分
- ・ 2. 基本的な計算幾何のアルゴリズム
- ・ 3. 応用的な使い方・実践例
- ・ 4. 注意点



# 数

- ・ 実数(doubleやlong double)を使う時は、計算誤差がある
- ・ `const double EPS = (小さい値);`
- ・ という値 EPS を活用しよう (この小さい値は、大きすぎないようにちゃんと見積もる必要がある)
- ・ 数の比較は、以下のような関数を用意するとよい
- ・ `int sgn(double a){return (a<-EPS)?-1:(a>EPS)?1:0;}`

# 数

- ・ 直線だけしか登場しない時など、(円が絡まない時は)有理数を使うという手もある
- ・ 分母と分子を用意する
- ・ 最近では有理数で計算しないと誤差が厳しい問題も
- ・ IOI側のシラバス回避対策…?

# 構成要素: 点

- ・  $x$ 座標と $y$ 座標を持っている
- ・ 位置ベクトルとして扱いやすいと便利
- ・ 実装方針？
  - ・ `<complex>` ?
  - ・ 構造体 ?

# 構成要素: 点

- ・ これらを踏まえて用意したいもの
- ・ `double x, y`
- ・ ベクトルの加減、定数倍(\*と/), -1倍(-a)、回転
- ・ ベクトル長、内積、外積、単位円上での角度
- ・ ソート基準 (例: xの昇順、同点時はyの昇順)

# Pt の基本的演算

- $\text{Pt}(x,y) \pm \text{Pt}(u,v) = \text{Pt}(x \pm u, y \pm v)$
- $\text{Pt}(x,y) * k = \text{Pt}(x*k, y*k)$   $\text{Pt}(x,y) / k = \text{Pt}(x/k, y/k)$
- $-\text{Pt}(x,y) = \text{Pt}(-x, -y)$
- $\text{Pt}(x,y) * \text{Pt}(u,v) = \text{Pt}(x*u - y*v, x*v + y*u)$   
( $u = \cos \theta$ ,  $v = \sin \theta$  を入れてみよう)
- $\text{Pt}(x,y).\text{abs}() = \text{sqrt}(x*x + y*y)$
- $\text{Pt}(x,y).\text{dot}(\text{Pt}(u,v)) = x*u + y*v$
- $\text{Pt}(x,y).\text{det}(\text{Pt}(u,v)) = x*v - y*u$
- $\text{Pt}(x,y).\text{arg}() = \text{atan2}(y,x)$

# Pt のソート

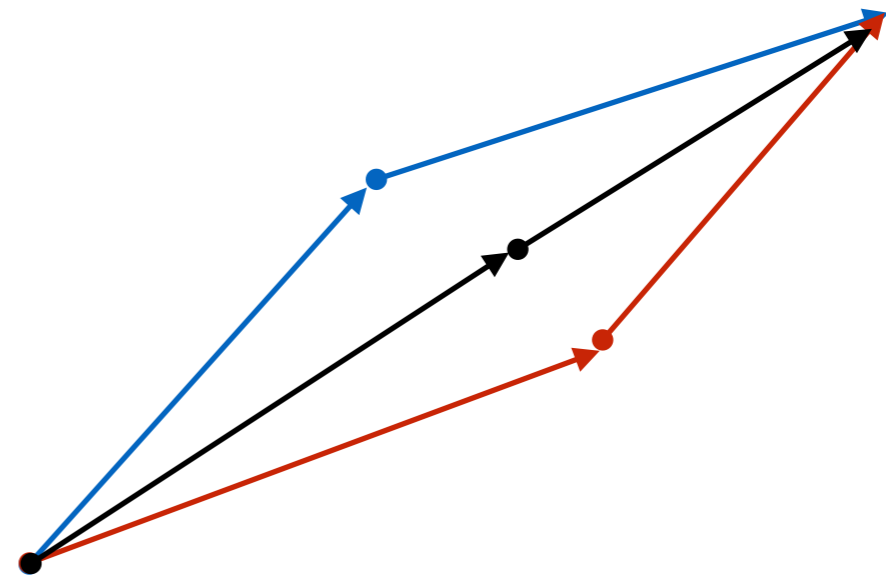
- ```
inline bool operator<(const Pt &a, const Pt &b){  
    if(sgn(a.x-b.x))return sgn(a.x-b.x)<0;  
    return sgn(a.y-b.y)<0;  
}
```

# Pt から作れるもの

- ・ 線分、直線  
Pt を 2 つ使えばよい
- ・ 円  
中心 Pt と半径 double
- ・ 多角形  
n角形はn個の Pt の列

# 点の進行方向 (iSP)

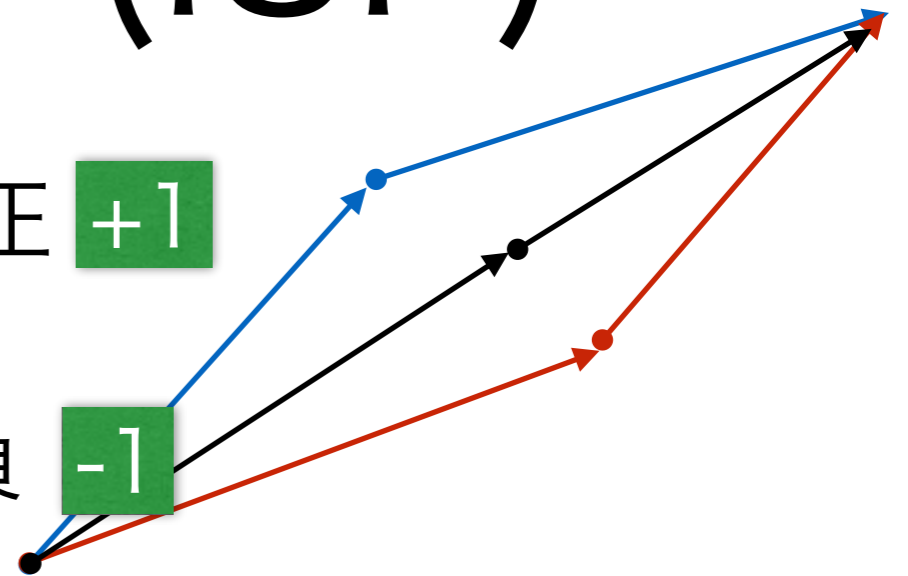
- ・ 3点  $a, b, c$  があたえられた時に、 $a \rightarrow b \rightarrow c$  の移動でどう曲がるかを判定したい
- ・ よく使うので関数にしよう





# 点の進行方向 (iSP) 戻り値

- ・ 左に曲がる場合:  $(b-a).det(c-a)$  が正 +1
- ・ 右に曲がる場合:  $(b-a).det(c-a)$  が負 -1
- ・ まっすぐの場合:  $(b-a).det(c-a)$  が0
- ・ c-a-bの順に並ぶ:  $(b-a).dot(c-a)$  が負 -2
- ・ a-b-cの順に並ぶ:  $(a-b).dot(c-b)$  が負 +2
- ・ a-c-bの順に並ぶ: それ以外 0



# まとめ

- ・ 点:  $P_t$   
加減乗除や $\det, \text{dot}$ , 長さなどをもたせておく
- ・  $P_t$  から直線、円、多角形が簡単に表せる
- ・ 正負を判定する関数 $\text{sgn}$
- ・ 点の進行方向を判定する関数 $iSP$

# 目次

- ・ 1. 平面幾何の構成要素の基本部分
- ・ 2. 基本的な計算幾何のアルゴリズム
- ・ 3. 応用的な使い方・実践例
- ・ 4. 注意点

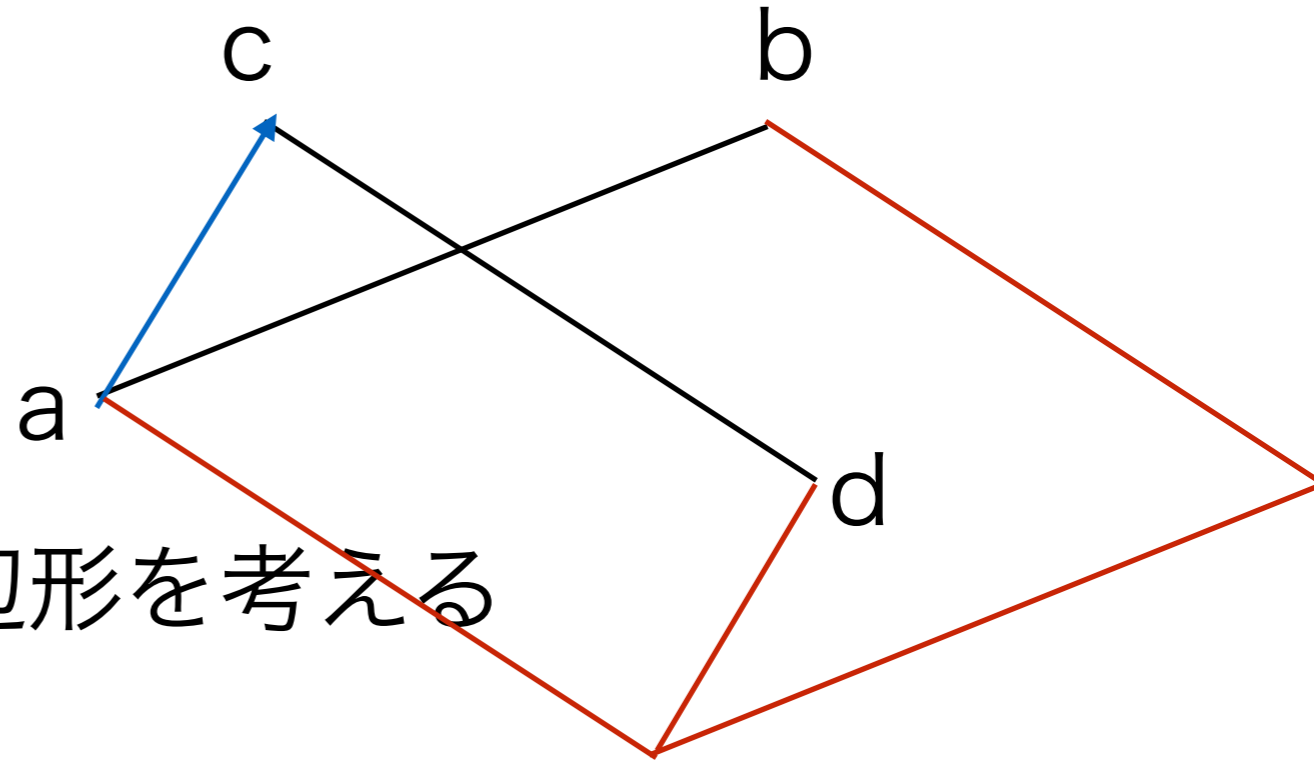
# 本編

- ・ 計算が絡んでくるものを取り扱っていきます
- ・ ここに挙げているものの他にも便利なものはたくさんあると思いますが、ここに挙げた既存のものを使えば大体は作れると思います
- ・ ライブラリ整備がんばってください

# 直線/線分交差判定

- ・ ほとんど iSP を使えばできる
- ・ 直線の交差判定(直線abと直線cd)  
交差する:  $(b-a).det(c-d)$  が0でない  
平行: ↑が0だが  $(b-a).det(c-a)$  が0でない  
同一直線: それ以外
- ・ 線分の交差判定(線分abと線分cd)  
「直線abに関してc,dが逆側にある vice versa」  
 $iSP(a,b,c) * iSP(a,b,d) \leq 0 \ \&\& \ iSP(c,d,a) * iSP(c,d,b) \leq 0$   
線分が重なってたり端点で交わっているときもtrueなことに注意

# 2直線の交点



- ・ 直線 $ab$ , 直線 $cd$  の交点
- ・ 右のような2つの平行四辺形を考える
- ・ 面積比と高さの比が等しいので、これを利用すると交点が求められる
- ・ 平行四辺形の(符号つき)面積 $\rightarrow$ これは $\det$ そのもの

# 2直線の交点

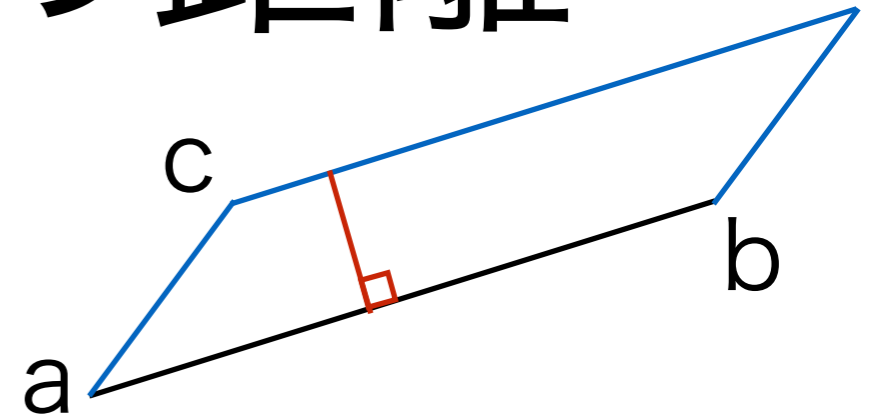
- ・ 前のスライドでやることをコードにすると、
- ・  $a + (b-a) * (c-a).det(d-c) / (b-a).det(d-c)$
- ・ (ただし、2直線が平行や一致してないことを事前に確認しないとイケない)

# 直線/線分と点の距離

- 直線abと点cの距離

先ほどと同様に平行四辺形の面積(det)を活用

$$\text{abs}( (c-a).\text{det}(b-a) / (b-a).\text{abs}() )$$

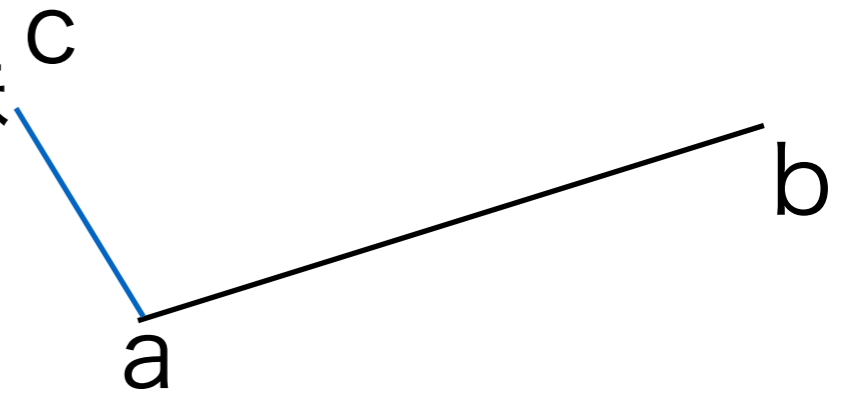


- 線分abと点cの距離

$\angle cab$ や $\angle cba$ が鈍角のときだけ特殊

$\angle cab$ が鈍角 $\Leftrightarrow (b-a).\text{dot}(c-a)$ が負

それ以外のときは直線版と同じ





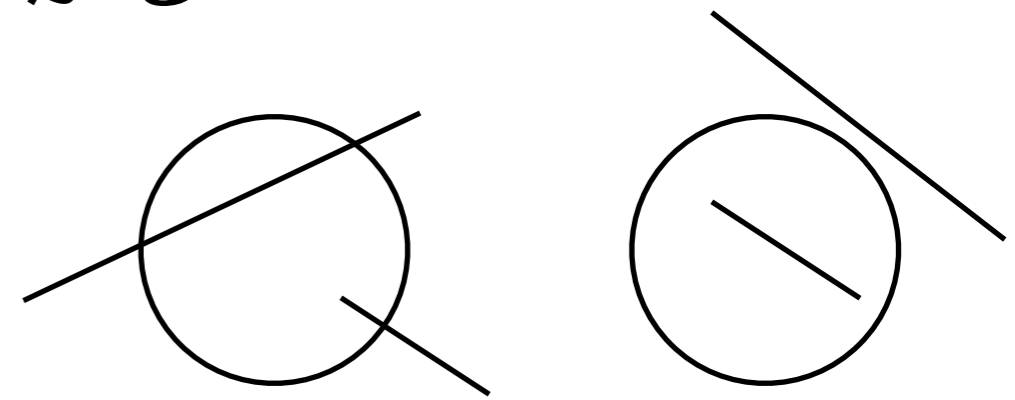
# 直線/線分同士の距離

- ・ 直線と直線の距離: 平行なら1つの直線とのこりの直線上の任意の1点との距離、それ以外なら0
- ・ 直線と線分の距離: 交差していたら0、それ以外なら線分の両端と直線との距離のうち小さいほう
- ・ 線分と線分の距離: 交差していたら0、それ以外なら4通りの線分と点の距離のmin

# 円と直線/線分の交差判定

- 円と直線の交差判定は、円の中心と直線との距離が半径 $r$ より大きいか小さいかで分かる

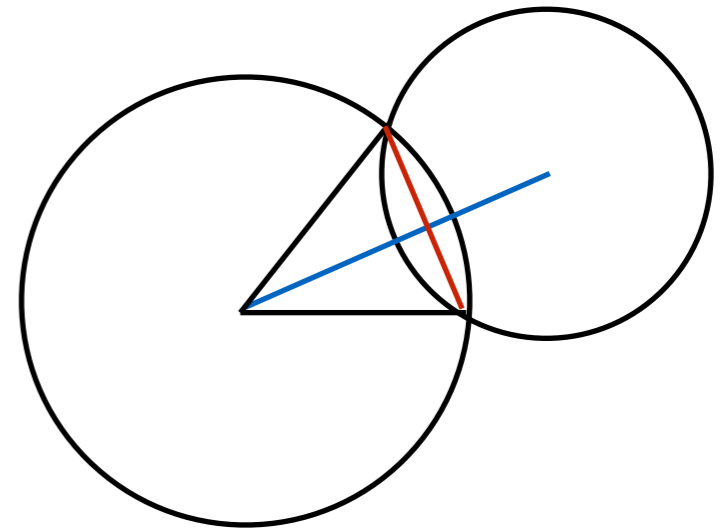
- 円 $(a,r)$ と線分 $bc$ の交差判定  
次の2つを満たしていればよい



- 点 $a$ と線分 $bc$ の距離が $r$ 以下 (円の内側を通る)
- $\max((a-b).abs(), (a-c).abs())$ が $r$ 以上 (円の外側を通る)

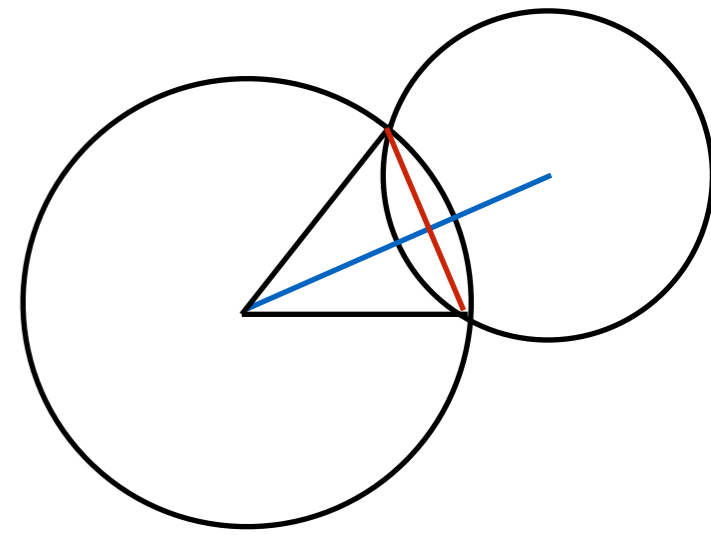
# 円と円の交点

- ・ 円 $(a,r)$ と円 $(b,s)$ の交点を求めたい  
(交点は2つとする)



- ・ 青いベクトルと赤いベクトルの和で求めることにする
- ・ 知りたいもの: 青と赤の交点までの長さ、赤いベクトルの方向と長さ

# 円と円の交点



- 2円の中心の距離を  $d( = (b-a).abs() )$  とする
- 青と赤の交点までの長さ  
連立方程式を解く  $\rightarrow x = (d*d + r*r - s*s)/(2*d)$
- 赤いベクトルの向き  
青を90度回転  $(b-a)*Pt(0,1)$
- 赤いベクトルの長さ  $\text{sqrt}(\text{max}(r*r - x*x, 0.0))$   
 $\text{sqrt}$ の中身が誤差で負になると困る

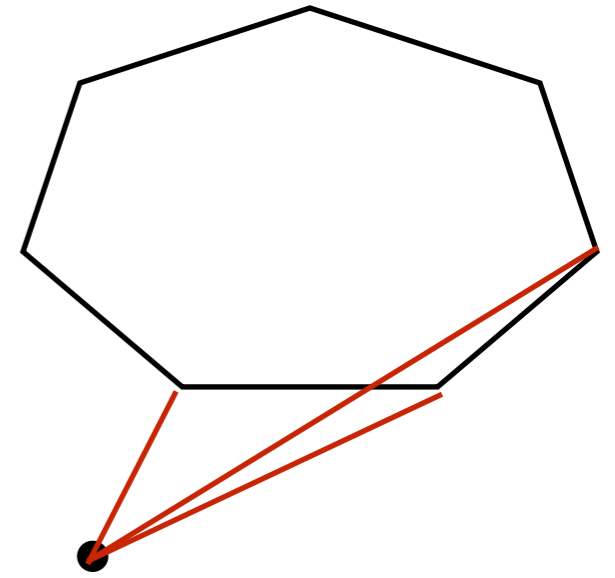
# 読者への課題①

- ・ 垂線の足
- ・ 円と直線の交点 (垂線の足を利用)
- ・ 点 $b$ を通る円 $(a,r)$ への接線
- ・ 円 $(a,r)$ と円 $(b,s)$ の共通部分の面積
- ・ 三角形の内心、外心、垂心

# 面積・重心

- ・ 外積(符号つき面積)を使うと、多角形の面積が簡単に計算できる
- ・ 三角形( $p[0], p[1], p[2]$ )の重心は、 $(p[0]+p[1]+p[2])/3.0$  であることが有名
- ・ 一般の多角形の重心も、(符号つきで)これを使って簡単に実装できる

# 面積



- $(0,0), p[i], p[i+1]$ を3点とする三角形の面積は、 $|p[i].\det(p[i+1])/2|$
- しかし、絶対値をとらず符号を考慮しておくと、後々楽
- 多角形の面積は、↑のことを反時計回りに一周して合計すればよい (打ち消され方を手で確認してみよう)

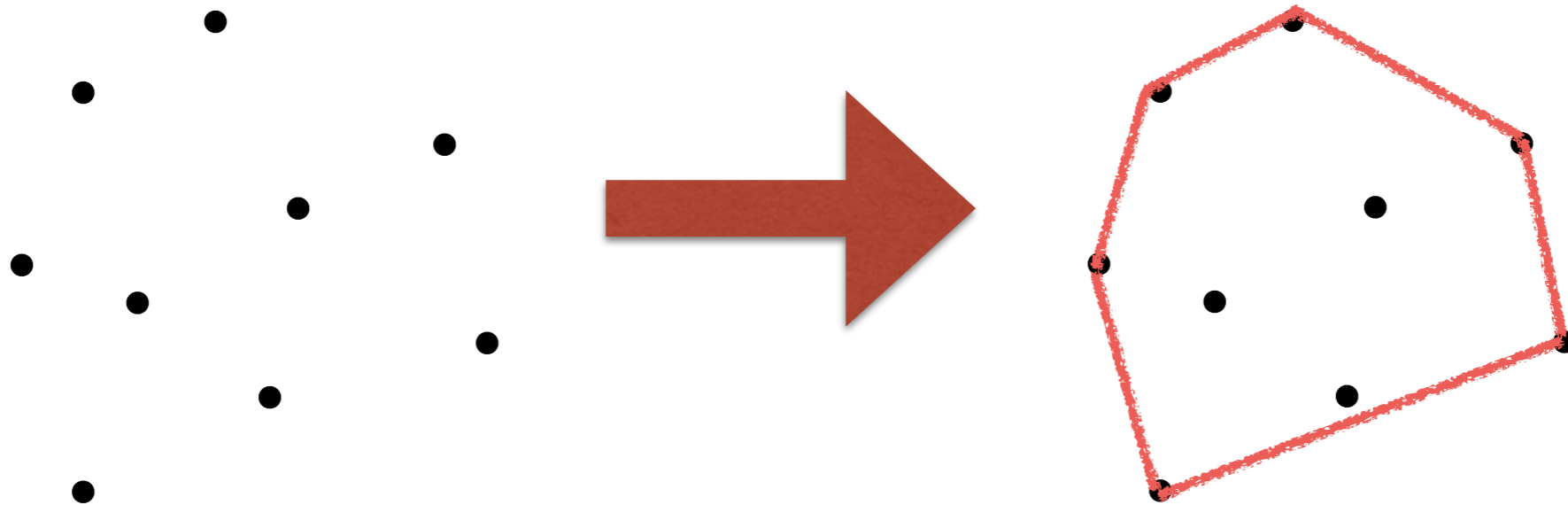
# 重心

- ・ 重心も、先ほどの三角形の符号つき面積を応用すると簡単に求められる
- ・ 座標  $(p[i]+p[i+1])/3$  に重さが  $p[i].\text{det}(p[i+1])$  のおもりがあるときの重心、と捉えられる
- ・ あとは加重平均をとればよい



# 凸包

- ・ 凸包とは？
- ・ N 個の点があります。外から輪ゴムを張ると、輪ゴムはどこを通りますか。



# 凸包

- ・ 頻出テクニックです  
JOIでは、過去の合宿に登場しています  
(Constellation)
- ・ 今回はこれを  $O(N \log N)$  で求めてみましょう

# 凸包

- ・ 着目ポイント ①

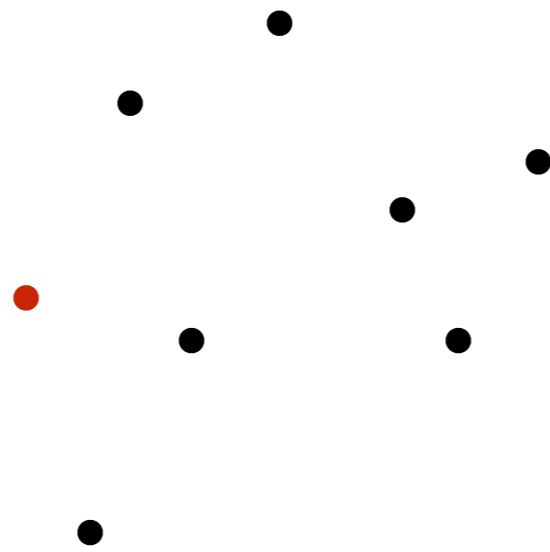
Pt でソートしたときの最小、最大は必ず使われます

- ・ 着目ポイント ②

上方向と下方向は似たような構造で、上下に分けて考えて良さそうです。

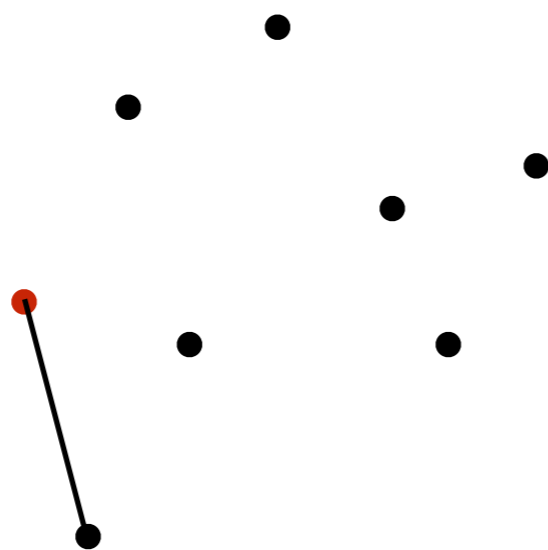
# 凸包

- ・ まず上半分だけ構成してみます
- ・ ソートした後最初の要素は、左端として使われます



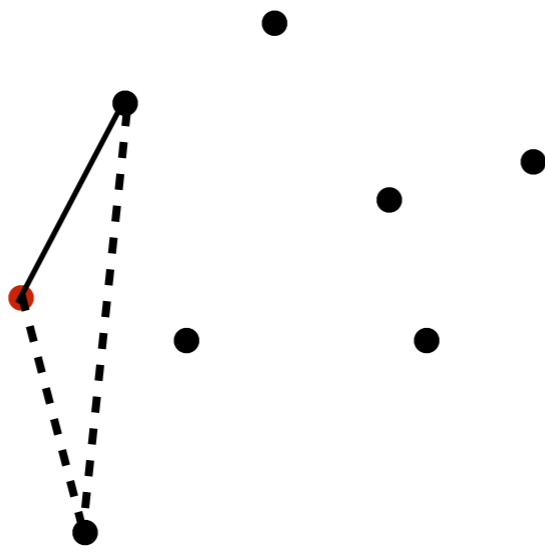
# 凸包

- ・ まず上半分だけ構成してみます
- ・ stackに積んだり下ろしたりする感覚で、左から必要な頂点だけ追加していきます



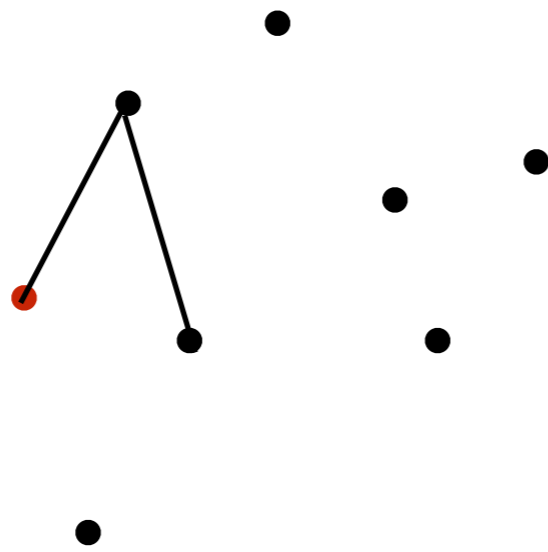
# 凸包

- ・ 頂点を追加するときに、直前の辺から左に曲がる必要があるとき、直前の頂点はいらないので(stackから)消します



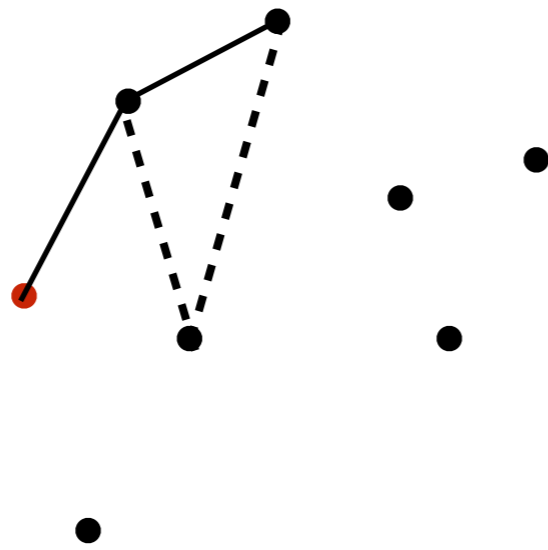
# 凸包

- ・ 頂点を追加するときに、直前の辺から左に曲がる必要があるとき、直前の頂点はいらないので(stackから)消します



# 凸包

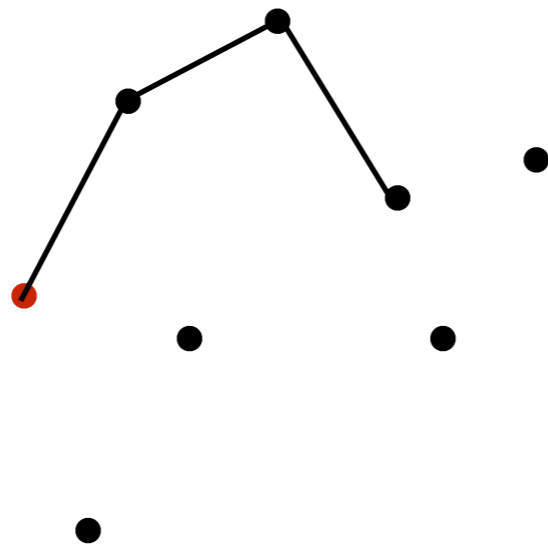
- ・ 頂点を追加するときに、直前の辺から左に曲がる必要があるとき、直前の頂点はいらないので(stackから)消します





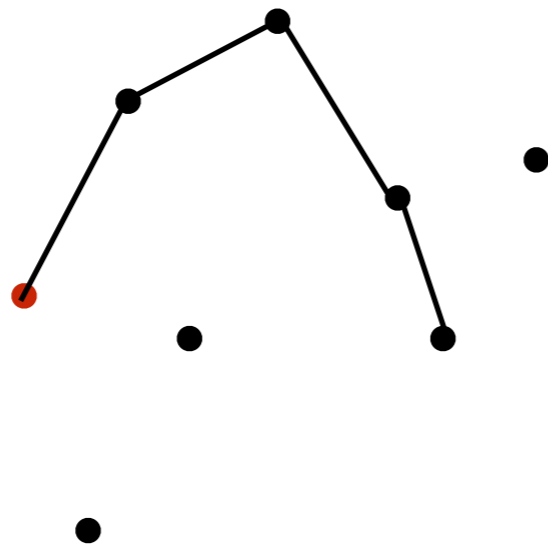
# 凸包

- ・ 頂点を追加するときに、直前の辺から左に曲がる必要があるとき、直前の頂点はいらないので(stackから)消します



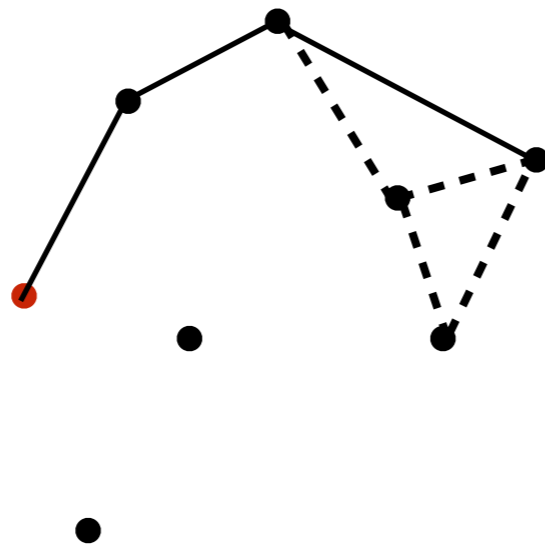
# 凸包

- ・ 頂点を追加するときに、直前の辺から左に曲がる必要があるとき、直前の頂点はいらないので(stackから)消します



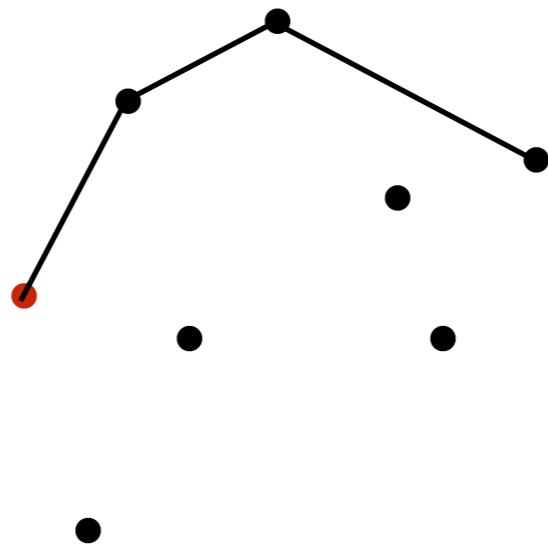
# 凸包

- ・ 頂点を追加するときに、直前の辺から左に曲がる必要があるとき、直前の頂点はいらないので(stackから)消します



# 凸包

- ・ 上凸包完成
- ・ 同様にして下凸包も作れる
- ・ (下凸包)(上凸包を逆順にしたもの)を並べれば完成



# 凸包

- ・ 計算量  $O(N \log N)$  (ソートが一番重い)
- ・ スタックを使って実装すればよい
- ・ 曲がる向きは iSP が使える  
(実は、同一直線上に点があると微妙？ 符号つき面積が正でないときは削除、とするとよい)

# 点の内外判定

- ・ 解決法: 点から変な向きに半直線を引き、それと多角形の辺が交わった回数を数える  
→ 偶数なら外、奇数なら内
- ・ 多角形の頂点と重なるときが面倒
  - ちゃんと対処する ( $\text{sig}(dy) > 0$ と $\text{sig}(dy) \leq 0$ の辺)
  - 絶対重ならなさそうな傾きにする

# 平面走査

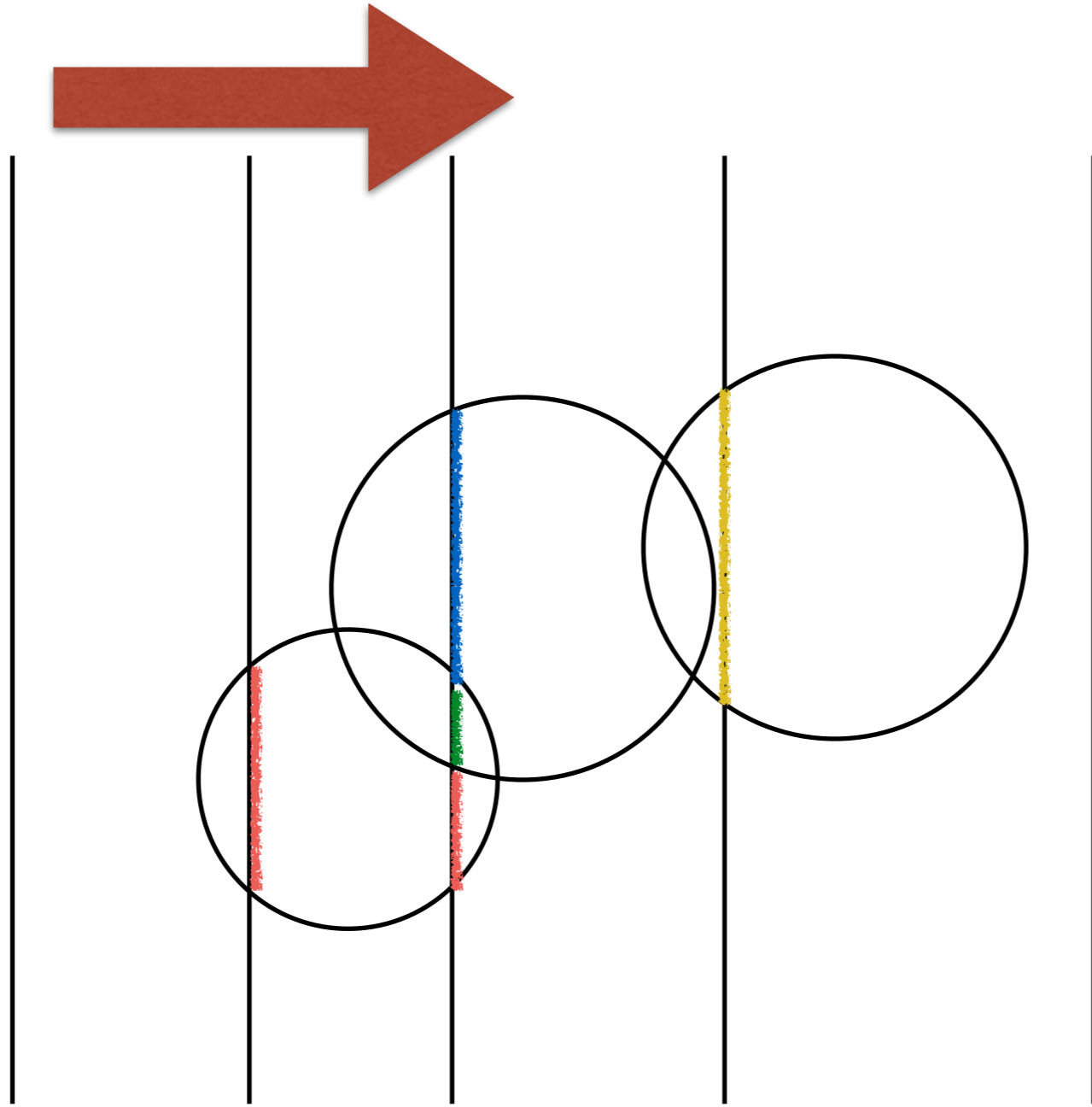
- ・ 平面走査という単語は、データ構造問題でJOlerにもよく知られている(と思う)
- ・ Starry Skyとか
- ・ 幾何にも平面走査がよく使われる
- ・ 代表的なアプローチは2通り

# 平面走査①

- ・ x軸方向にイベント(円の左端が登場、右端が登場、2円が交わる、など)をソートして左から実行
- ・ それぞれのイベントで区間を追加したり消したりできる。



# 平面走査① 例: Nurie

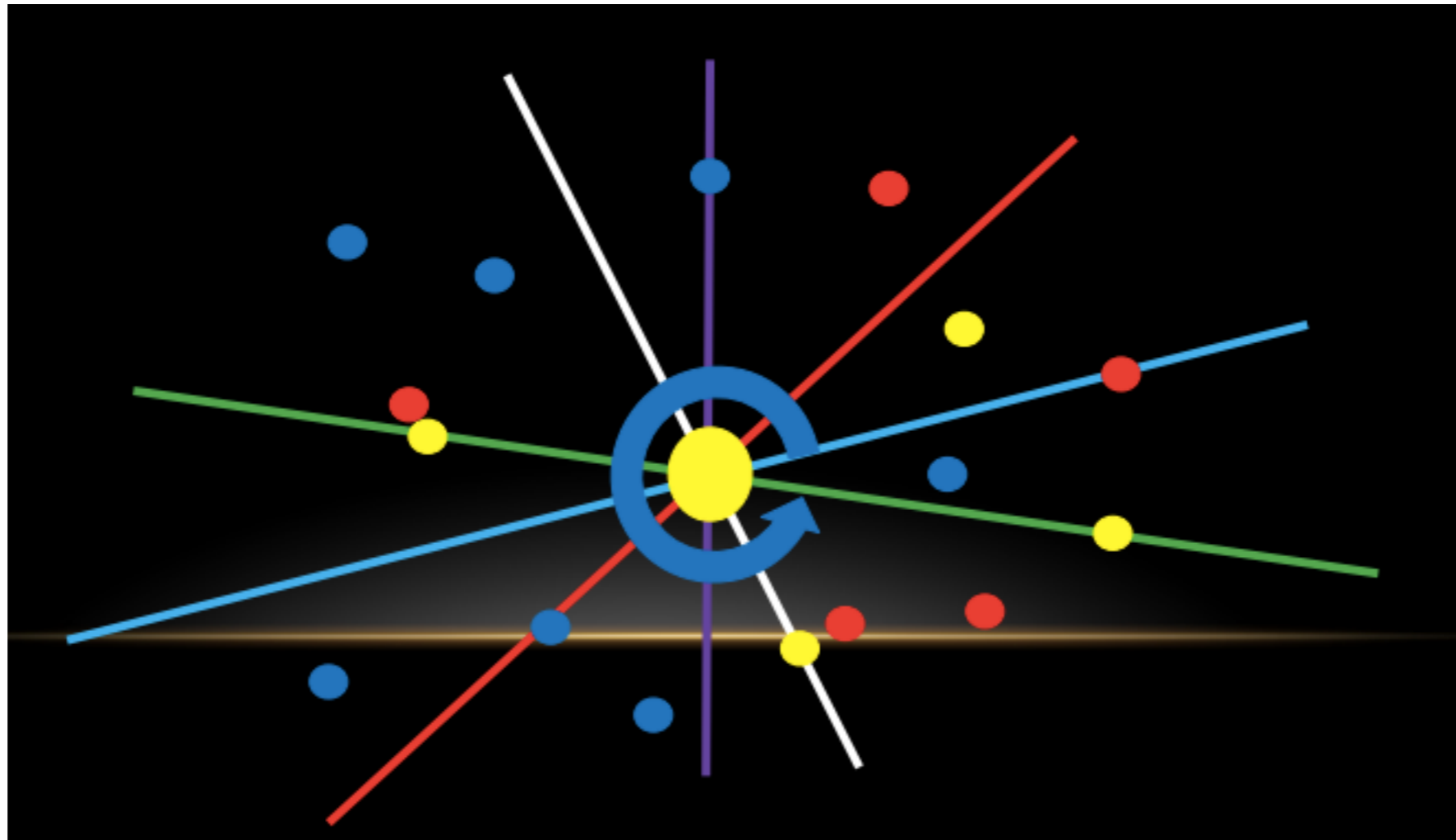


# 平面走査②


- ・ さっきはx軸方向に走査線を平行移動していたが、今回はある点を中心に回す
- ・ 角度が効いてくるときには有効

# 平面走査②

例: Constellation2



# 読者への課題②

- ・ 幾何で数値積分を利用
- ・ 最小包含円 (山登り法  ・ 真面目なやつ)  
真面目: 2012合宿 テレビ放送 参照
- ・ 最近点对 (ずるいやつ ・ 分割統治法(蟻本) )
- ・ 最遠点对 (座標サイズが小さいとき ・ キャリパー法)



# 読者への課題②

- ・ 平面的双対グラフ (ねこまっしぐら2)
- ・ 点 $a$ の凸多角形との内外判定 ( $O(\log N)$ )
- ・ 凸多角形の直線 $ab$ よりも上にあるものを求める
- ・ 点 $a$ から凸多角形への接線 ( $O(\log N)$ )

# 目次

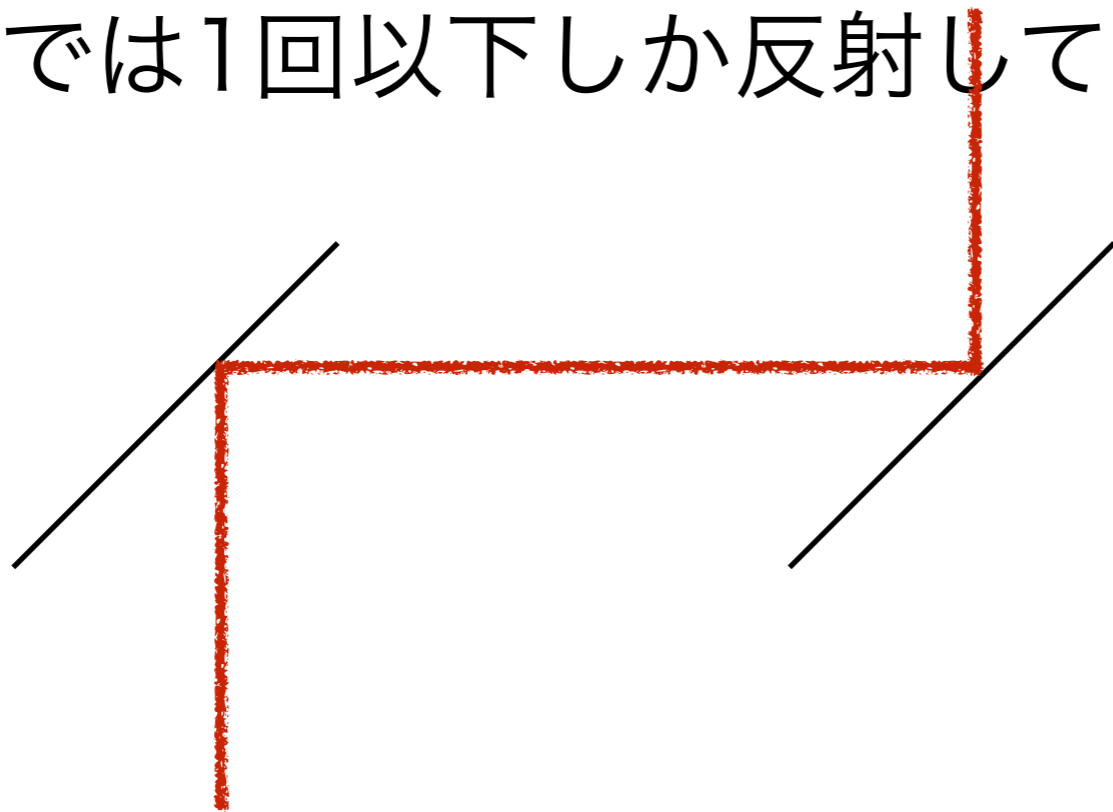
- ・ 1. 平面幾何の構成要素の基本部分
- ・ 2. 基本的な計算幾何のアルゴリズム
- ・ 3. 応用的な使い方・実践例
- ・ 4. 注意点

# 実践例

- ・ 実際にこのような幾何アルゴリズムがコンテストでどのように使われているかを見ていきましょう
- ・ 慣れることが大切
- ・ ライブラリ化しておこう

# 例1. レーザー光の反射

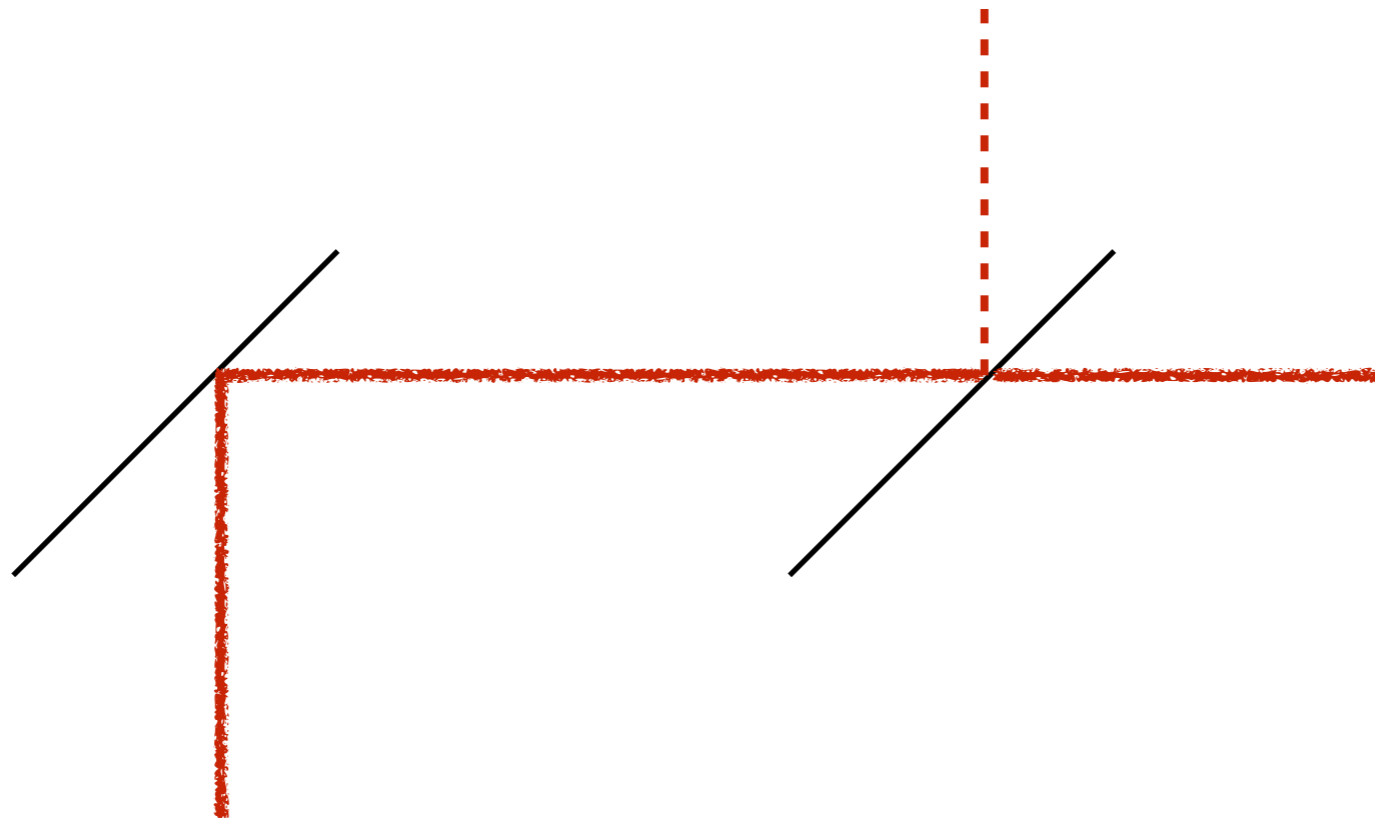
- ・ 鏡が幾つかある場所で光源から目的までの光の進む最短路は？
- ・ 同じ鏡では1回以下しか反射してはいけない





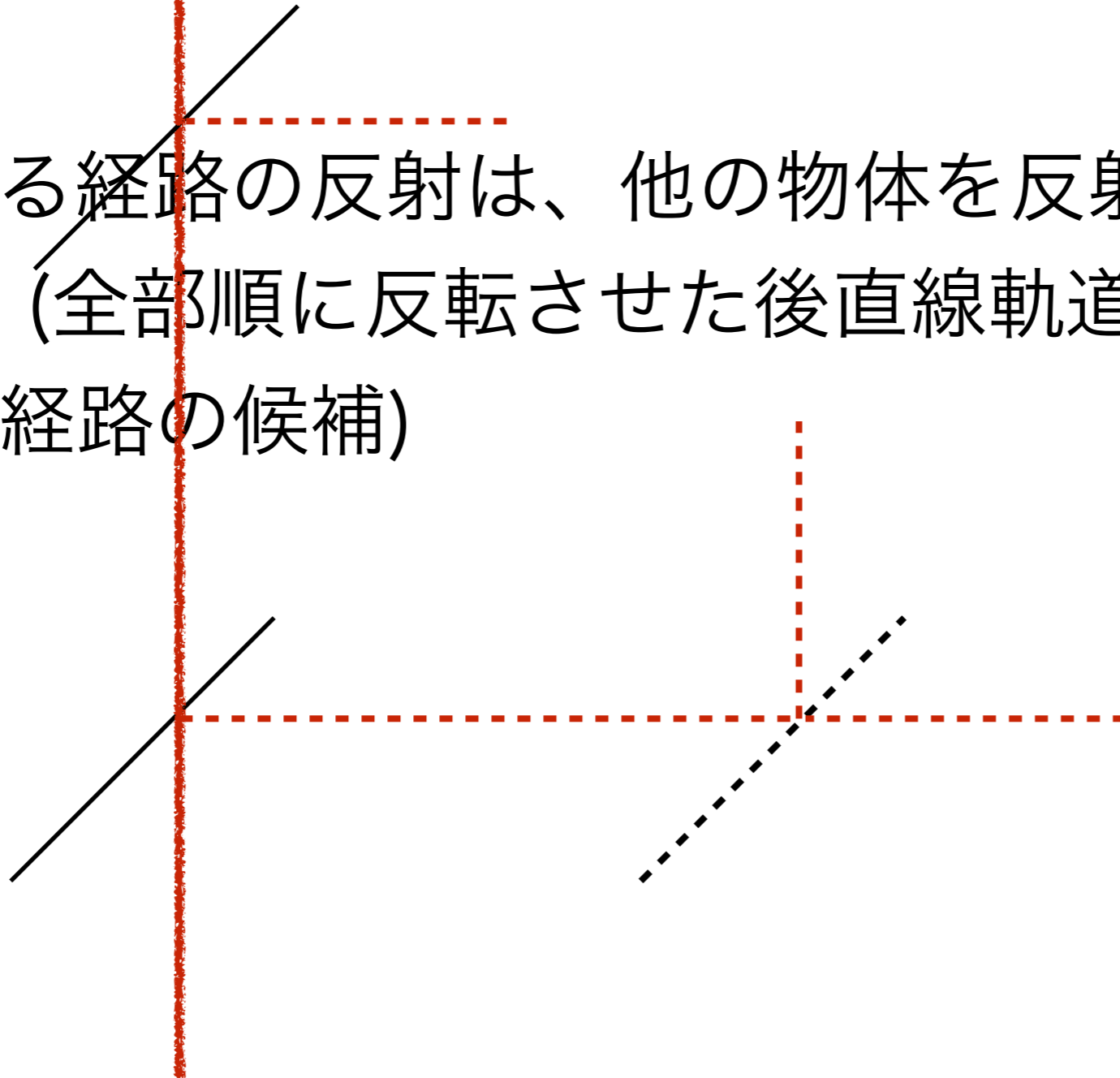
# 例1. レーザー光の反射

- 鏡による経路の反射は、他の物体を反射させて考えると楽



# 例1. レーザー光の反射

- 鏡による経路の反射は、他の物体を反射させて考えると楽 (全部順に反転させた後直線軌道となるものが最短経路の候補)



# 例1. レーザー光の反射

- ・ 要するに、反射可能な鏡をどの順番で使うかを全部試し、反転をほぐした後の目的地の座標とスタートとの距離を計算すればよい
- ・ 反転の仕方？  
直線abを軸に点pを反転するとどこに移るか？  
-  $hLP(a,b,p)*2-p$  が移動先 (hLP: 垂線の足)

# 例2. 三角形の分類

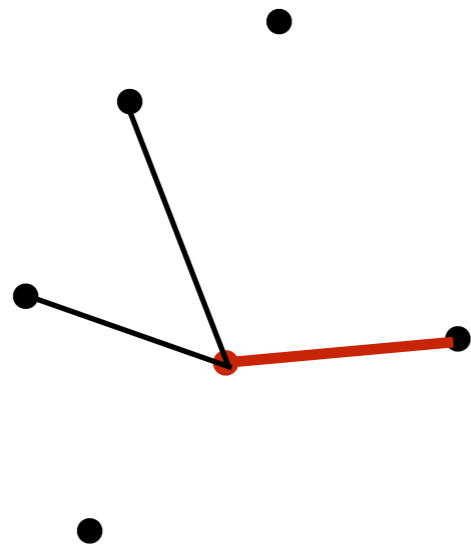
- ・ 座標平面上に  $N$  点ある (3点が同一直線上にない)
- ・  ${}_N C_3$  通りの三角形が作れるが、そのうち鋭角、直角、鈍角三角形はいくつあるか？
- ・  $N \leq 2,000$

# 例2. 三角形の分類

- ・ 一つ一つ三角形を取り出して角度を計算すると  $O(N^3)$  で遅い
- ・ まとめて数える必要がある
- ・ 鈍角三角形: 鈍角が1個ある  
直角三角形: 直角が1個ある  
鋭角三角形: その他
- ・ 角度に関している……？

# 例2. 三角形の分類

- ・ 角度に関係している……？  
→ 平面走査②を使ってみよう



赤い辺を固定したとき、正の方向でいくつ直角・鈍角が作れるか数える

(これは角度でソートして二分探索すれば求められる)

# 例2. 三角形の分類

- ・ 鋭角三角形の個数は、三角形の総数から鈍角三角形と直角三角形の個数を引けば良い
- ・ 各頂点を中心に平面走査を行い、それぞれソートと二分探索が必要なので、計算量は  $O(N^2 \log N)$

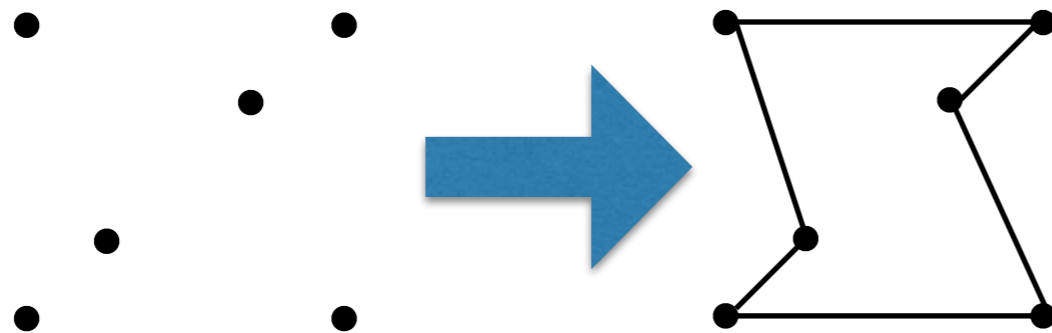
# 例3. Rural Planning

- ・ N点与えられる。N点をうまくつないで次のような多角形を作れ。
  - 与えられたN点は全てこの多角形の周上にある
  - この多角形の面積は, N点の凸包の面積の半分以上



# 例3. Rural Planning

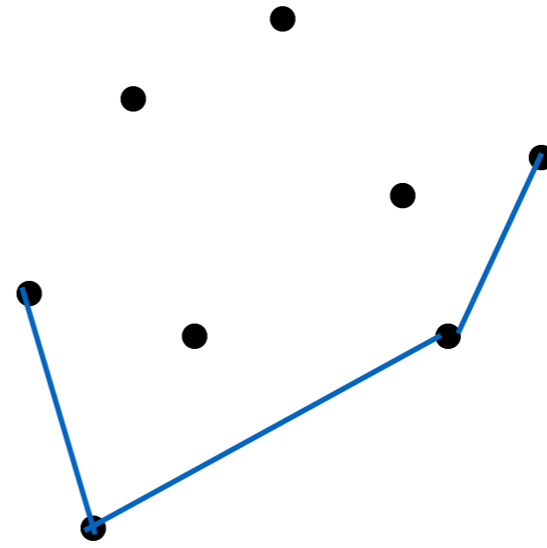
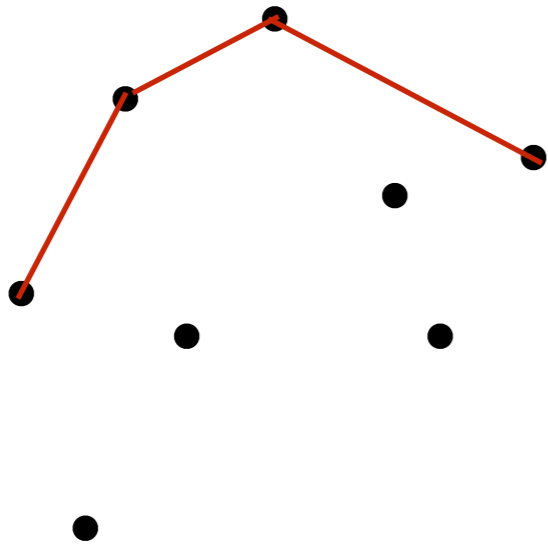
- ・ ハア？
- ・ 例すら意味不明



# 例3. Rural Planning

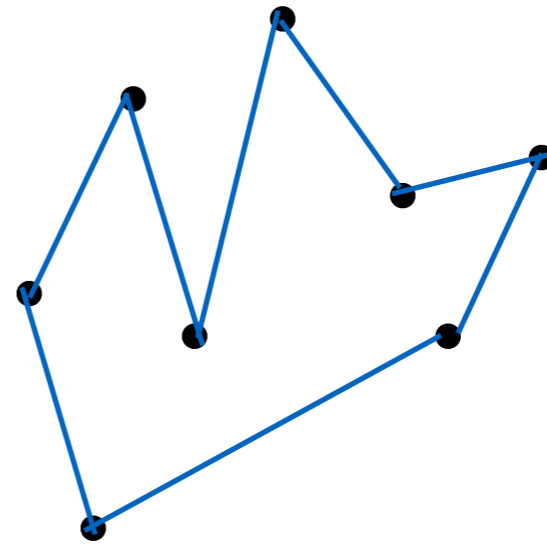
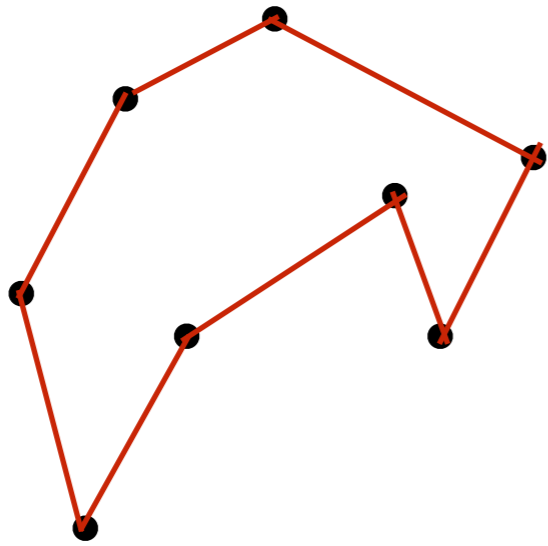
- ・ 凸包に思いを馳せよう
- ・ 凸包を作る時は上凸包と下凸包の2つをつなげて作って……
- ・ ……2つ？

# 例3. Rural Planning



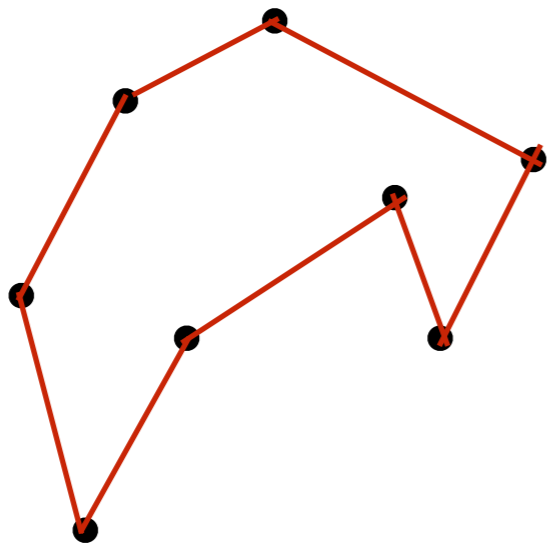
- ・ 2つに分ける

# 例3. Rural Planning

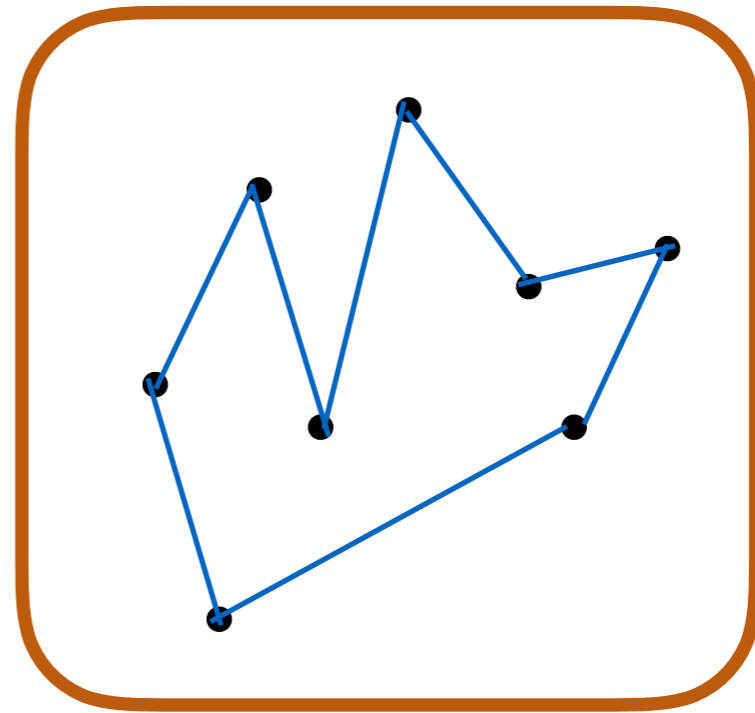


- ・ それぞれを使って多角形を1つ作ってみる(使われなかった点をソートして、小さい順に点つなぎ)

# 例3. Rural Planning



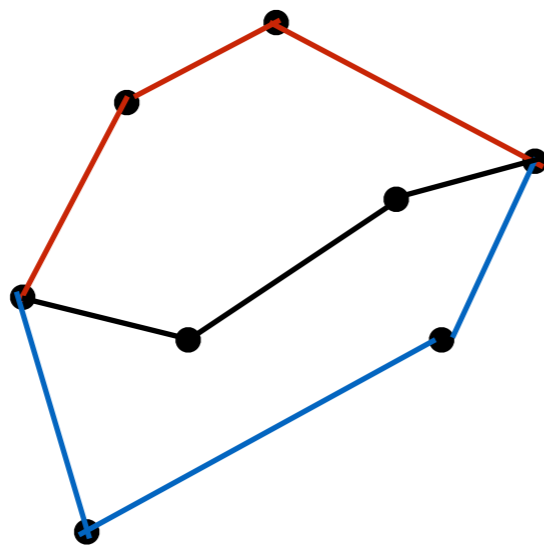
<



- ・ 面積が大きい方が答え！

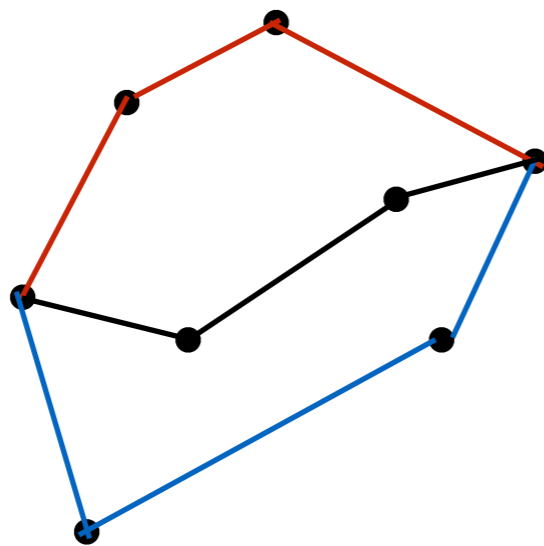
# 例3. Rural Planning

- ・ なぜ正しいのか？
- ・ 凸包の上、下、使われなかった点の点つなぎを図示してみた



# 例3. Rural Planning

- ・ さっき上凸包から作った図形は、赤、黒の辺からなる多角形より面積が大きい (逆もしかり)
- ・ どちらかの色と黒の辺を使った多角形は、凸包の面積の半分以上！



→

**先ほどのどちらかの  
図形は、面積半分以上！**

# 目次

- ・ 1. 平面幾何の構成要素の基本部分
- ・ 2. 基本的な計算幾何のアルゴリズム
- ・ 3. 応用的な使い方・実践例
- ・ 4. 注意点



# 誤差

- ・ 座標が整数で与えられるとき、傾きの差の最小値は
- ・  $1/O(T^2)$  くらい (T: 座標の最大値)
- ・  $T = 10^9$  とかのとき、doubleだと怪しい

# 誤差

- ・ 何がどう怪しいのか？
  - 傾きや座標でソートしたいとき、値が同じになってしまって壊れる  
(doubleの精度 < long longの精度)
  - `sgn(x)`関数で使うEPSが実際の差より大きくなってしまって壊れる
  - $\pi$ の値が実際と乖離していて角度が少しずれてしまう

# 対処法

- ・ 整数・有理数でやる
  - 与えられた点以外に追加しないときは整数でも良いので楽
  - `struct Q{long long a,long long b}`的な有理数クラスを用意しておく (少し大変)
- ・ ハッシュを使う
  - 一致判定をするために  $\text{mod } P$  で座標を持っておく
  - ハッシュを用意しておく (大小判定は厳しい)

# 対処法

- ・ 最初に座標平面を回転させる
  - 各点に  $Pt(\cos(1), \sin(1))$  とかを掛ける
  - 各座標で座標が似かよるのを解消できる
  - 平面走査(数値積分を除く)、最近点对で有効
- ・ long doubleを使う
  - 一番楽? でも効果が怪しい
  - %Lf だったり 1.0L だったり
  - 円があっても使えて強い



# 対処法

- ・  $\pi$  を手入力でなく標準関数を使う
  - `acos(-1.0)` とか
  - やっぱり不安なのでうまくEPSを足し引きしよう

# 変なエラー対策

- ・ 0 で割らないようにする
  - 傾きを計算するときは、y軸に平行な直線の対処法に慎重になるべき
    - (でも基本的に場合分けはしたくないので、代替手法を考えよう)
- ・ sqrtの中身が負にならないようにする
  - 時折誤差で負になってしまう
    - `sqrt(max(0.0,x))` などとする

# おわりに

- ・ 今回の講義では、計算幾何の基本的なプログラム構造について紹介した
- ・ 近年多くの幾何問題では、他のアルゴリズムや考察と複合であるものが多い  
(逆に幾何パートの高速化なんかも面白い)
- ・ しかし、幾何パートが解けないと始まらないので、基礎力を身につける必要がある
- ・ 実装頑張ってください