計算幾何 Geometry

Lecture & modified by Colten
Credit by yp155136, baluteshih, TreapKing

目錄

- Q & A + Review
- 弧度
- 三角函數
- IEEE 754
- 線段交點
- 極角排序
- 凸包
- More topics

Before the Lecture

- 後面的投影片沒特別說明，都只考慮二維的情況
  - 當然還有很多高維度的幾何，但是不在今天的討論範圍，有興趣的學員可以自行上網參考相關資料XD

Q & A

- 影片有什麼問題嗎？

Review

- 內積、外積

```
double operator*(const Pt &p1, const Pt &p2) {
    return p1.X * p2.X + p1.Y * p2.Y;
}
double operator^(const Pt &p1, const Pt &p2) {
    return p1.X * p2.Y - p1.Y * p2.X;
}
```

Review

- 多邊形有向面積

多邊形的有向面積

一般而言，會挑選原點作爲參考點

若一個多邊形的頂點依序爲：

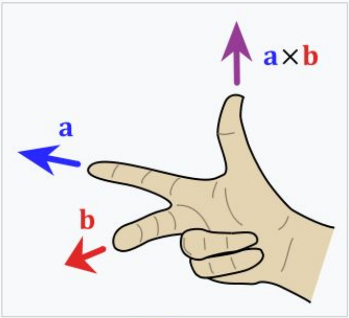
$$P_0, P_1, \ldots, P_{N-1}, P_N = P_0$$

則多邊形的有向面積公式爲：

$$\text{Area} = \frac{1}{2} \sum_{i=0}^{N-1} \overrightarrow{P_i} \times \overrightarrow{P_{i+1}}$$

Review

- 方向函式：$\overrightarrow{OA}$轉到 $\overrightarrow{OB}$的方向是？



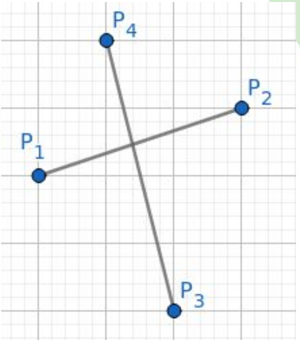
用右手定則求叉積的方向

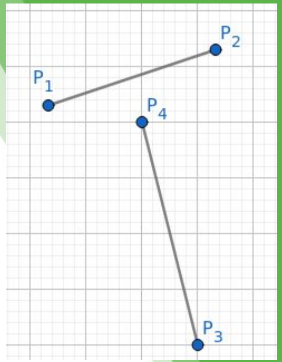
```
int sign(double a) {
    if (fabs(a) < eps) return 0;
    return a > 0 ? 1 : -1;
}
int ori(const Pt &o, const Pt &a, const Pt &b) {
    double cross = (a - o) ^ (b - o);
    return sign(cross);
}
```

- 正的是逆時針，負的是順時針，零是共線

Review

- 線段~~香蕉~~相交

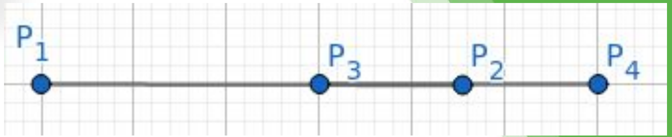





```
bool banana(const Pt &p1, const Pt &p2, const Pt &p3, const Pt &p4) {
    int a123 = ori(p1, p2, p3);
    int a124 = ori(p1, p2, p4);
    int a341 = ori(p3, p4, p1);
    int a342 = ori(p3, p4, p2);
    return a123 * a124 <= 0 && a341 * a342 <= 0;
}
```

- 兩側都要判！

Review



- 線段~~香蕉~~相交 – 勿忘平行線！

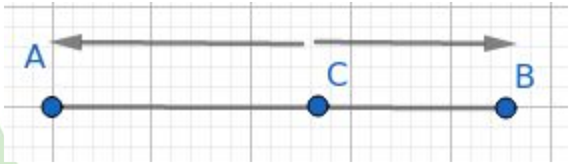
```
bool banana(const Pt &p1, const Pt &p2, const Pt &p3, const Pt &p4) {
    int a123 = ori(p1, p2, p3);
    int a124 = ori(p1, p2, p4);
    int a341 = ori(p3, p4, p1);
    int a342 = ori(p3, p4, p2);
    if (a123 == 0 && a124 == 0)
        return one of the points is located on the opposite segment
    return a123 * a124 <= 0 && a341 * a342 <= 0;
}
```

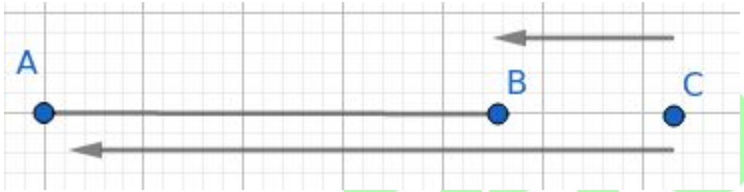
- 怎麼判點在線段內？

Review

- 點在線段內: C在線段 $\overline{AB}$上嗎？(使用內積判斷)
- 內積意義:投影長 ＊ 投影長 （同向時內積為正）

```
bool btw(const Pt &a, const Pt &b, const Pt &c) {
    if (ori(a, b, c) != 0) return 0;
    return sign((c - a) * (c - b)) <= 0;
}
```





Review

- 線段~~香蕉~~相交 – 完整程式碼

```
bool banana(const Pt &p1, const Pt &p2, const Pt &p3, const Pt &p4) {
    int a123 = ori(p1, p2, p3);
    int a124 = ori(p1, p2, p4);
    int a341 = ori(p3, p4, p1);
    int a342 = ori(p3, p4, p2);
    if (a123 == 0 && a124 == 0)
        return btw(p1, p2, p3) || btw(p1, p2, p4) ||
               btw(p3, p4, p1) || btw(p3, p4, p2);
    return a123 * a124 <= 0 && a341 * a342 <= 0;
}
```

Review: 誤差分析

- 能不用浮點數盡量不用浮點數
  - 有些題目是請你輸出面積到小數一位，但是（面積 * 2）根據題目一定會是整數
  - 這時候就用 long long int 存答案，最後輸出的時候再處理小數部份就好了

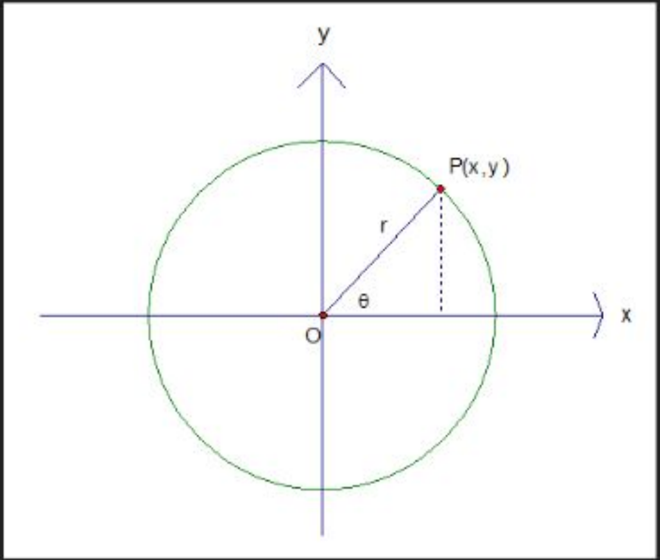
- 如果要用小數呢？
  - 可以參考影片分析誤差
  - 或者是直接設 10^{-6} ~ 10^{-9} 附近， WA 了再改XD

弧度

- $2\pi=360°$
- 同理, $\pi=180°$
- $x°=(x/360)*2\pi$
- $x = (x/2\pi)*360°$
- 弧度沒有單位
- C/C++ 裡的三角函數都是用弧度

三角函數

廣義三角函數



$$cos(\theta) = \frac{x}{r}$$

$$sin(\theta) = \frac{y}{r}$$

$$tan(\theta) = \frac{y}{x}$$

$$atan2(y, x) = \theta$$

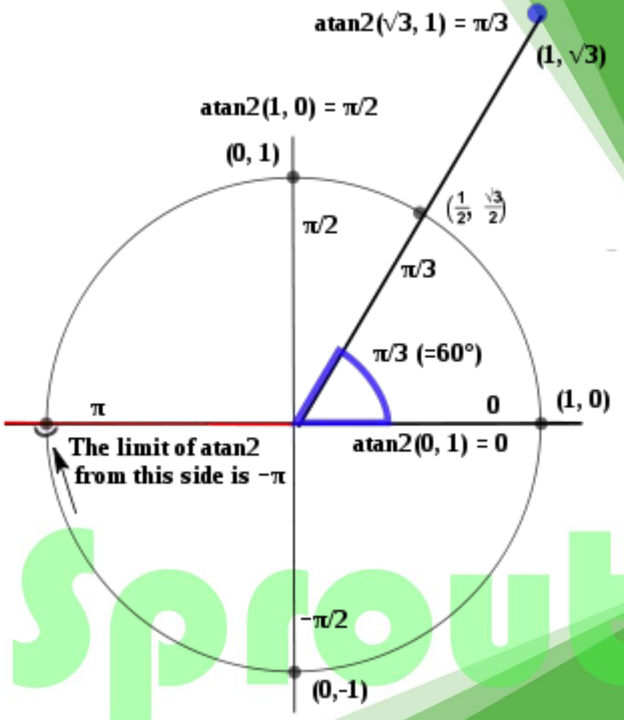
三角函數

- 注意到 cmath, math.h 裡面的三角函數
- 實做上是用泰勒展開式......等等高級數學技巧來逼近的
- 不要把他們當成 O(1)

atan2 (正切)

- atan2(y, x)
- 回傳值的值域為 (-pi, pi]
  - 但 C++ 可能回傳 -pi！

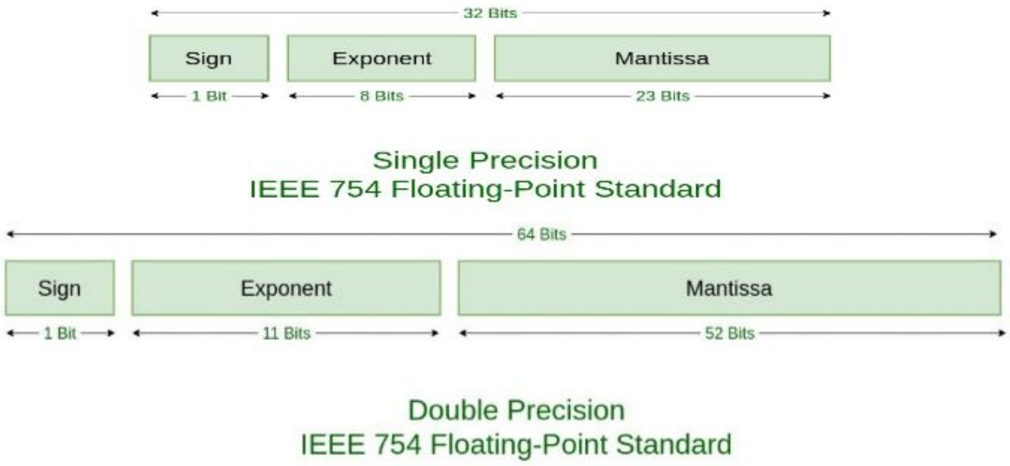
$$\operatorname{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$



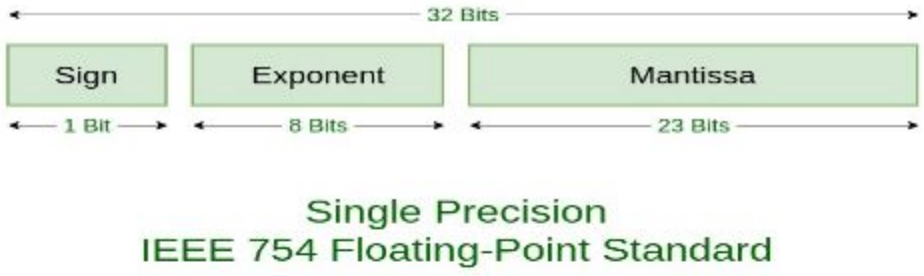
- source: wiki

IEEE 754

- Source: https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/
- 用科學記號(EX: 1.001 * 2^11010)存小數

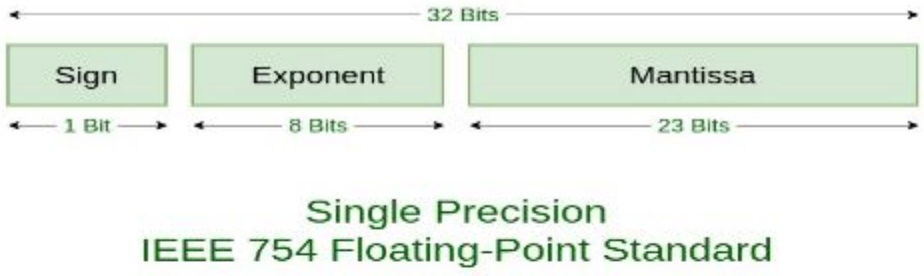


IEEE 754

- 

- Sign ： 正負號
- Exponent ： 指數部份(2 的冪次)
- Mantissa ： 小數部份

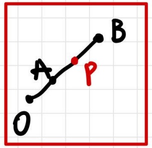
IEEE 754

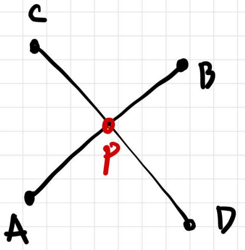
- 

- Sign ： 正負號
- Exponent ： 指數部份(2 的冪次)
- Mantissa ： 小數部份
- Q:只要是存小數，就會有誤差嗎？

線段交點

$$\vec{A} + i\vec{AB} = \vec{P}$$





$$\vec{CP} \times \vec{CD} = 0$$

$$\Rightarrow \underbrace{\boxed{(\underbrace{\vec{A} + i\vec{AB}}_{\vec{P}} - \vec{C})}}_{\vec{CP}} \times \vec{CD} = 0$$

$$\Rightarrow (\underbrace{\boxed{\vec{A} - \vec{C}}}_{\vec{CA}} + i\vec{AB}) \times \vec{CD} = 0$$

$$\Rightarrow (\vec{CA} + i\vec{AB}) \times \vec{CD} = 0$$

$$\Rightarrow \vec{CA} \times \vec{CD} + i\vec{AB} \times \vec{CD} = 0$$

$$\Rightarrow \vec{CA} \times \vec{CD} = -(i\vec{AB} \times \vec{CD})$$

$$\Rightarrow \vec{CA} \times \vec{CD} = \vec{CD} \times i\vec{AB}$$

$$\Rightarrow \frac{\vec{CA} \times \vec{CD}}{\vec{CD} \times \vec{AB}} = i$$

線段交點

- 實作 (from 108 師大附中校隊培訓)
- 記得共線的情況要另外處理!

```
pair<T, T> intersection(pair<T, T> a, pair<T, T> b, pair<T, T> c, pair<T, T> d){
    assert(intersect(a, b, c, d));
    return a + cross(a - c, d - c) * (b - a) / cross(d - c, b - a);
}
```

點到線段的距離

- 請問 C 到線段 $\overline{AB}$ 的最短距離是？
- 要求：可以證明，當給定的點都是格子點時，答案的平方是有理數，請輸出這個有理數

點到線段的距離

- 請問 C 到線段 $\overline{AB}$ 的最短距離是？
- 要求：可以證明，當給定的點都是格子點時，答案的平方是有理數，請輸出這個有理數

- Case 1: $A = B$

點到線段的距離

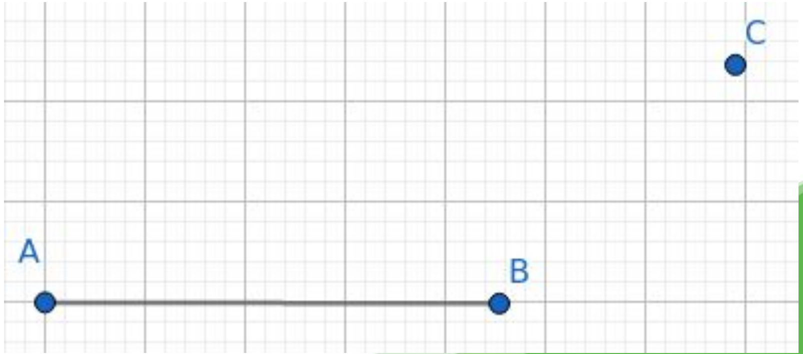
- 請問 C 到線段 $\overline{AB}$ 的最短距離是？
- 要求：可以證明，當給定的點都是格子點時，答案的平方是有理數，請輸出這個有理數

- Case 1: $A = B$
- $|A - C|$

點到線段的距離

- 請問 C 到線段 $\overline{AB}$ 的最短距離是？
- 要求：可以證明，當給定的點都是格子點時，答案的平方是有理數，請輸出這個有理數

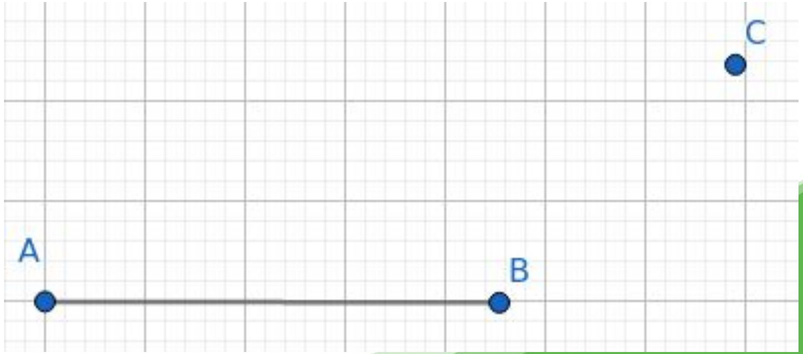
- Case 2: C 到 $\overline{AB}$ 的垂足在 $\overline{AB}$ 外



點到線段的距離

- 請問 C 到線段 $\overline{AB}$ 的最短距離是？
- 要求：可以證明，當給定的點都是格子點時，答案的平方是有理數，請輸出這個有理數

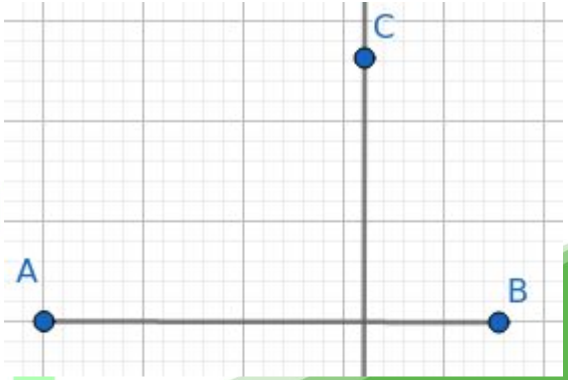
- Case 2: C 到 $\overline{AB}$ 的垂足在 $\overline{AB}$ 外
- $\min(|C-A|, |C-B|)$



點到線段的距離

- 請問 C 到線段 $\overline{AB}$ 的最短距離是？
- 要求：可以證明，當給定的點都是格子點時，答案的平方是有理數，請輸出這個有理數

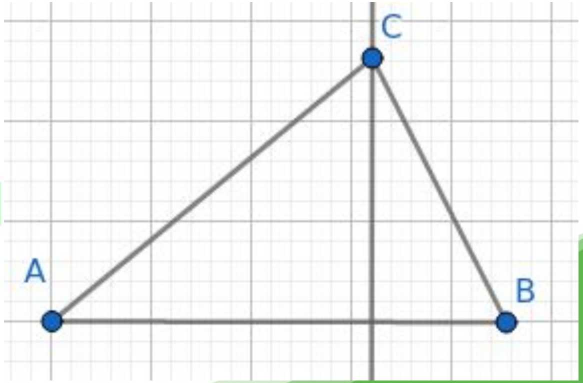
- Case 3: C 到 $\overline{AB}$ 的垂足在 $\overline{AB}$ 上



點到線段的距離

- 請問 C 到線段 $\overline{AB}$ 的最短距離是？
- 要求：可以證明，當給定的點都是格子點時，答案的平方是有理數，請輸出這個有理數

- Case 3: C 到 $\overline{AB}$ 的垂足在 $\overline{AB}$ 上
- $\triangle ABC$ 以 $\overline{AB}$ 為底的高



點到線段的距離

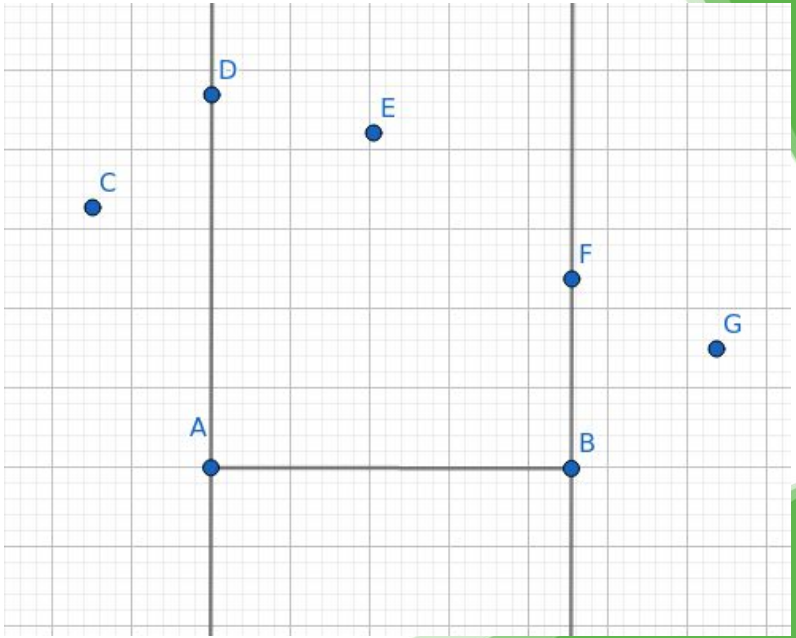
- Case 1: $A = B$
- Case 2: C到 $\overline{AB}$的垂足在 $\overline{AB}$外
- Case 3: C到 $\overline{AB}$的垂足在 $\overline{AB}$上
- 所以到底怎麼判垂足在哪？

點到線段的距離

- Case 1: $A = B$
- Case 2: C到 $\overline{AB}$的垂足在 $\overline{AB}$外
- Case 3: C到 $\overline{AB}$的垂足在 $\overline{AB}$上
- 所以到底怎麼判垂足在哪？

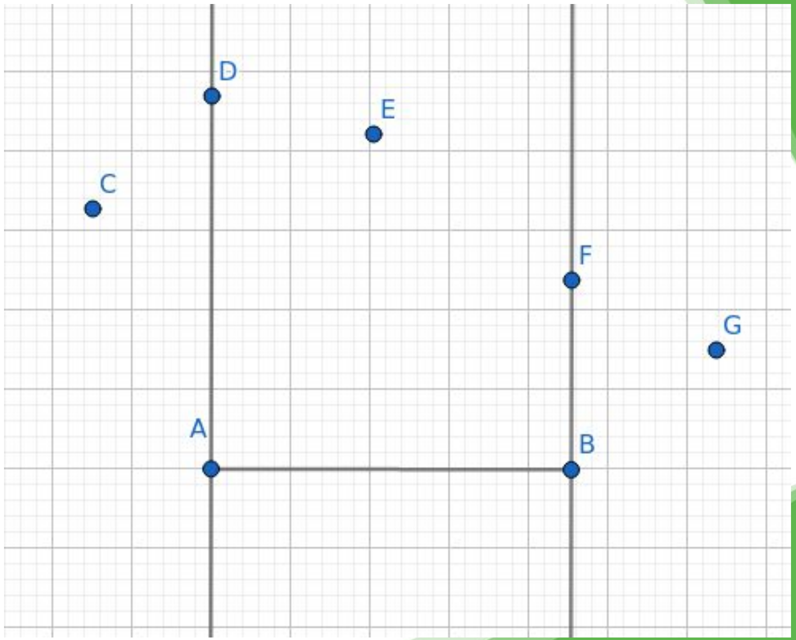
點到線段的距離

- 所以到底怎麼判垂足在哪？



點到線段的距離

- 所以到底怎麼判垂足在哪？
- $\angle CAB$ 跟 $\angle CBA$ 都不可以是鈍角！

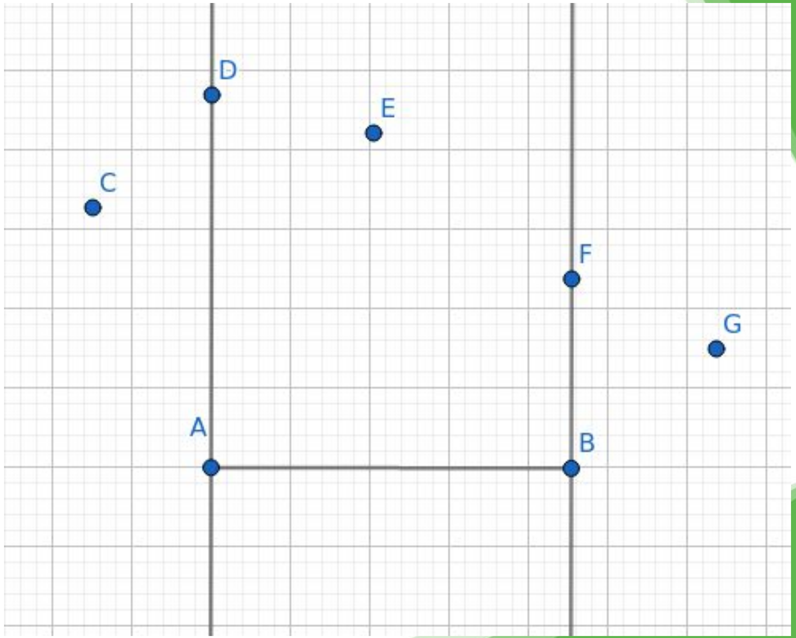


點到線段的距離

- 所以到底怎麼判垂足在哪？
- $\angle CAB$ 跟 $\angle CBA$ 都不可以是鈍角！

$$\overrightarrow{AB} \cdot \overrightarrow{AC} \geq 0$$

$$\overrightarrow{BA} \cdot \overrightarrow{BC} \geq 0$$



點到線段的距離

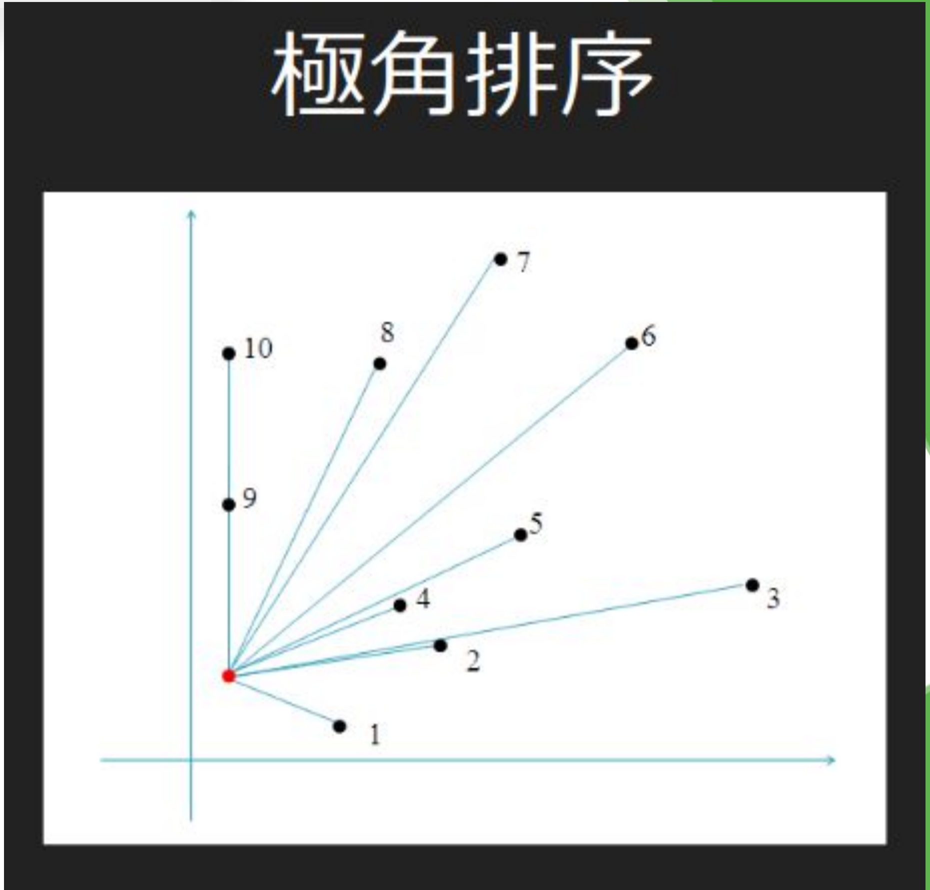
- 實作

```
double PointToSegDist(const Pt &a, const Pt &b, const Pt &c) {
    if (sign(abs(a - b)) == 0) return abs(a - c);
    if (sign((b - a) * (c - a)) >= 0 && sign((a - b) * (c - b)) >= 0)
        return fabs(((b - a) ^ (c - a)) / abs(a - b));
    return min(abs(c - a), abs(c - b));
}
```

極角排序

- 定義：
  給你一堆點，請你把這些點排序。排序的依據是到某一個特定點的角度

- 特定點常常是原點



極角排序

- 方法一: sort by angle

```
bool cmp1(Pt a, Pt b) {
    return atan2(a.Y, a.X) < atan2(b.Y, b.X);
}
```

- 優點:直觀
- 缺點:
  - 慢
  - 浮點數誤差--> 可以使用 atan2l(回傳的型態是 long double)
  - 排序順序是從第三象限到第二象限(因為角度最小值在第三象限)
- 不建議使用

極角排序

- 方法二：分上下半平面、sort by cross

```
bool cmp2(Pt a, Pt b) {
#define is_neg(k) (sign(k.Y) < 0 || (sign(k.Y) == 0 && sign(k.X) < 0))
    int A = is_neg(a), B = is_neg(b);
    if (A != B)
        return A < B;
    return sign(a ^ b) > 0;
}
```

- 利用外積的正負去判斷
- 正的是逆時針，負的是順時針，零是共線
- a -> b 順時針時，表示 a 比較小
- 下半面 > 上半面

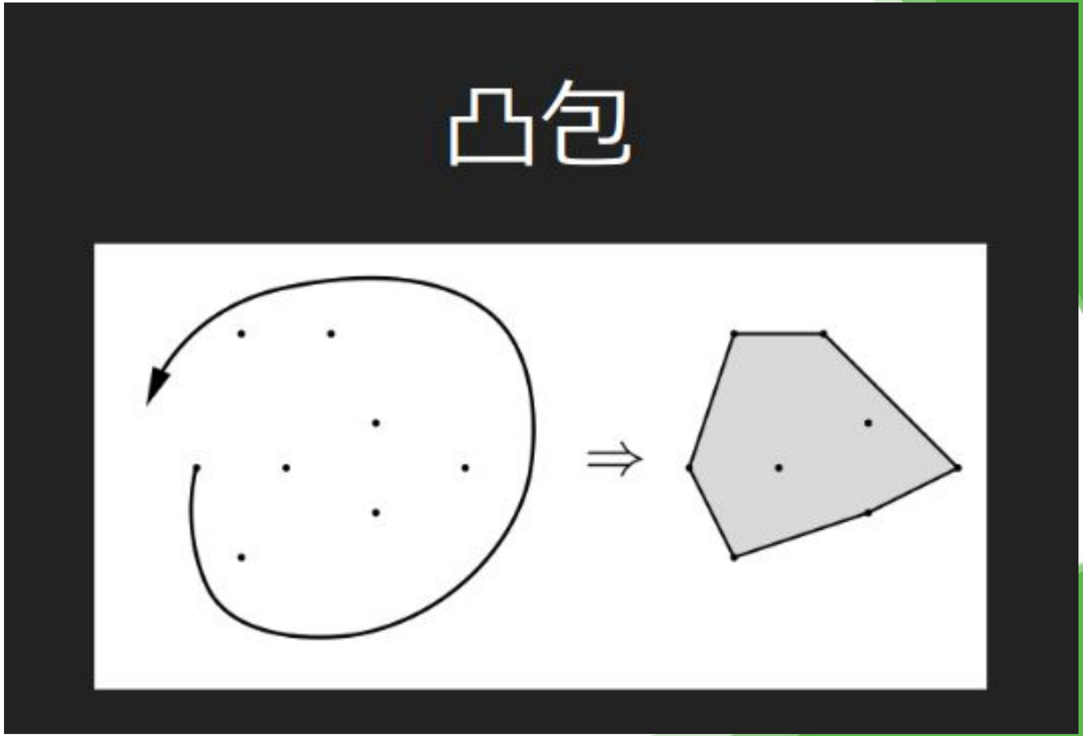
凸包

- 先來講講多邊形的分類(?)
- 簡單多邊形(simple polygon):邊不相交的多邊形
- 凸多邊形(convex polygon):內角都 <= 180 度的多邊形
- 凹多邊形(concave hull):有內角 > 180 度的多邊形

Reference: wiki

凸包

- 凸包定義:
  - 凸多邊形
  - 可以包住所有的點



凸包

- 一些性質:
  - 最邊邊的點一定在凸包裡面
  - 凸包上的點通常不會太多

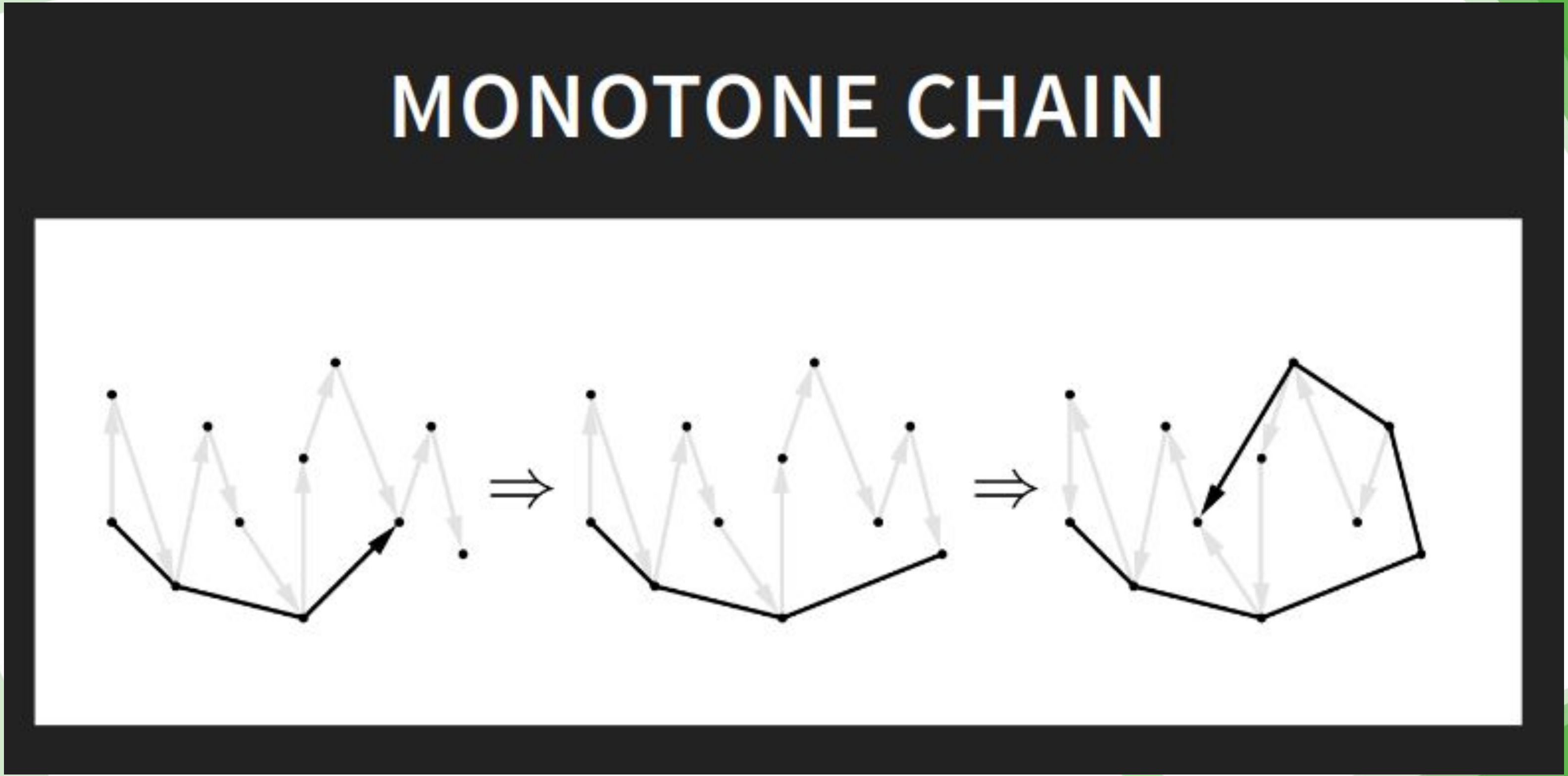
- 凸包求法:
  - Monotone Chain（只適用於二維）
  - Divide and Conquer（三維凸包也可以這樣做）
  - ....

- 今天只會講 Monotone Chain，對其他算法有興趣的學員可以自行上網搜尋

Monotone Chain

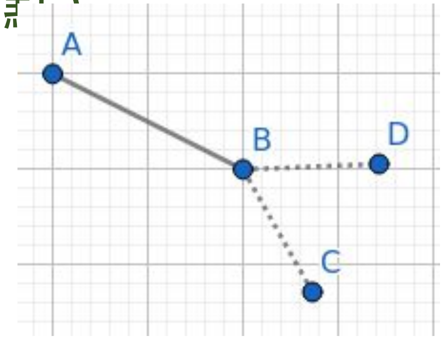
- 先把所有的點按照（x，y）排序
- 先試著把下凸包「圍」出來
- 再把上凸包「圍」出來
- 把上下凸包併在一起，就是凸包了～

Monotone Chain



How to「圍」

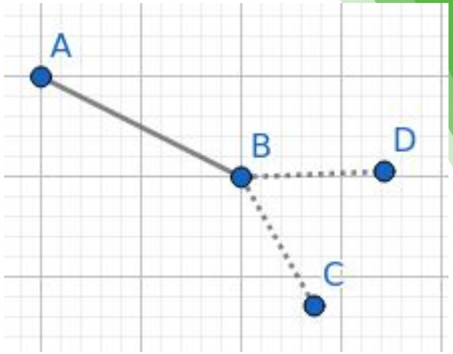
- 先考慮為下半部，上半部的「圍」法是類似的
- 開一個 stack(實作上常用 std::vector)，紀錄當前的下半凸包
- 每看一個新的點是，去看看有哪些點，會因為這個新的到來，導致那些點不再是凸包上的點 (用外積去看順時針還逆時針，順時針表示有更好的點)



Monotone Chain

- 圍下半凸包
- 最後一個點需要滿足:

```
ori(stk[stk.size() - 2], stk.back(), dots[i]) > 0
```




```
vector<Pt> stk(1, dots[0]);
for (int i = 1; i < int(dots.size()); ++i) {
    while (stk.size() > 1 && ori(stk[stk.size() - 2], stk.back(), dots[i]) <= 0)
        stk.pop_back();
    stk.push_back(dots[i]);
}
```

Monotone Chain

- 完整實作（上半部就只是再從右邊做回左邊而已!）

```cpp
vector<Pt> convex_hull(vector<Pt> dots) {
    if (dots.size() == 1) return {dots[0]};
    sort(dots.begin(), dots.end());
    vector<Pt> stk(1, dots[0]);
    for (int i = 1; i < int(dots.size()); ++i) {
        while (stk.size() > 1 && ori(stk[stk.size() - 2], stk.back(), dots[i]) <= 0)
            stk.pop_back();
        stk.push_back(dots[i]);
    }
    int t = stk.size();
    for (int i = dots.size() - 2; i >= 0; --i) {
        while (stk.size() > t && ori(stk[stk.size() - 2], stk.back(), dots[i]) <= 0)
            stk.pop_back();
        stk.push_back(dots[i]);
    }
    stk.pop_back();
    return stk;
}
```

凸包

- 其實凸包有一個更有用的應用，稱作「旋轉卡尺」
- 不過並不在這堂課的範疇內
- 有興趣的同學可以上網自行學習

一些有趣(X)的問題

- 給你平面上 N 個點，問一條直線最多可以穿過幾個點？
- 給你一個簡單多邊形(不一定是凸的)，請判斷一個點是不是在這個多邊形裡面？
- 給你一個線段以及一個圓，請你判斷這個線段跟圓的最短距離？
- 給你平面上 N 個點，請問可以從這 N 個點中畫出多少個三角形？
- 給你一個圓以及一個三角形(三角形有一個頂點是圓心)，求圓跟三角形的交集面積？

More topics

- 兩圓交點
- 圓跟多邊形的交集面積
- 旋轉卡尺
- 掃描線
- 最小包覆圓
- 半平面交
- Bentley-Ottmann algorithm（給你 N 個線段，請判斷這 N 個線段有沒有任意兩個線段相交）
- 模擬退火
- Voronoi Diagram
  - Delaunay Triangulation
- 三維幾何

Sprout