Enumeration

課堂補充 by TreapKing
modified by baluteshih

Q&A

- 影片都看過了嗎？
- 對於影片的內容有沒有什麼問題？
  - 如果是三分搜不懂的話，等等我會再解釋。

二分搜的使用關鍵

- 影片中提到了一些二分法的例子
- 有單調性，就能二分
  - 啊單調性到底是什麼？

二分搜的使用關鍵

- 看看下面的這串序列

  000000000011111111111111

- 請你告訴我0跟1的分界點在哪？
  - 當然不是跑迴圈搜，想像序列的長度可能超級長
  - 你每次只能問我某個位置是0還是1
- 這就是可以使用二分搜的時機
  - 戳到1，右邊肯定都不是答案
  - 戳到0，左邊肯定都不是答案

二分搜的使用關鍵

- 丟雞蛋問題
  - 假設我們有一個函數叫 is_borken(x)
  - 如果x這層樓丟下去雞蛋會破，那函式回傳1
  - 反之回傳0
- 我們要找到蛋最高在哪層樓不會破
  - 0跟1的分界點！
  - 二分搜！

二分搜的使用關鍵

- 田忌賽馬
  - 假設我們有一個函式叫 can_win(x)
  - 如果訓練x天可以贏，函式回傳1
  - 反之回傳0
- 我們要找到最少訓練幾天可以贏
  - 0跟1的分界點！
  - 二分搜！

二分搜的使用關鍵

- 所以二分搜時，總是會有一個只回傳0跟1的函式f(x)
  - 存在一個分界，在分界處以上皆回傳1，以下皆回傳0
- 只要f(x)計算的夠快，就可以多花一個log找到分界處！
- 可是在程式上要怎麼寫呢？
  - 接下來我會給三種做法
  - 可以的話回答我，他們分別最後回傳的是分界處的0，還是分界處的1

二分搜的寫法

- 寫法一：左閉右閉

```
1 int binary_search(int l, int r) {
2     while (l < r) {
3         int mid = (l + r) / 2;
4         if(f(mid) == 1) r = mid;
5         else l = mid + 1;
6     }
7     return l;
8 }
```

- 維護「答案可能的區間」

二分搜的寫法

- 寫法二：左閉右開

```
1 int binary_search(int l, int r) {
2     while (r - l > 1) {
3         int mid = (l + r) / 2;
4         if(f(mid) == 1) r = mid;
5         else l = mid;
6     }
7     return l;
8 }
```

- 維護「答案可能的區間 + 一個不可能是答案的邊界」

二分搜的寫法

- 寫法三：位元枚舉

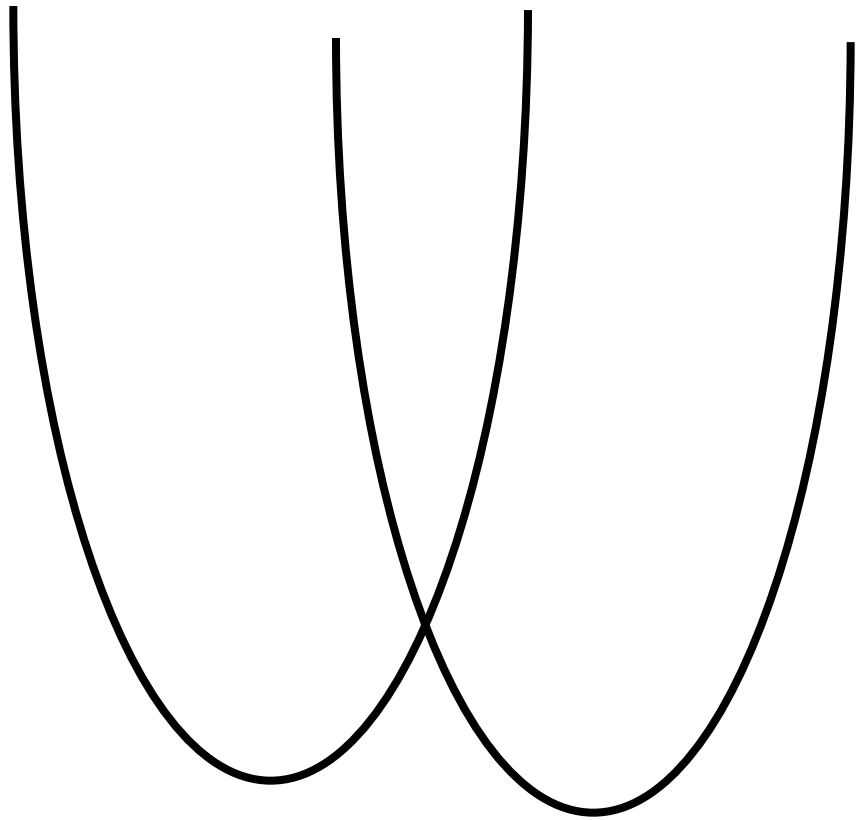
```
1 int binary_search(int r) {
2     int rt = 0;
3     for (int i = __lg(r); i >= 0; --i)
4         if(f(rt + (1 << i)) == 0)
5             rt += 1 << i;
6     return rt;
7 }
```

- 直接用二進位找到答案
- 適用的狀況比較少，但基於他搜尋的方法有可能在某些題目會出現優勢

題目在說什麼

- 給你一堆二次函數，要請你找出最小值。
- 影片中提到的「超車」到底是什麼意思？
- 函數的圖形長什麼樣子？

題目在說什麼

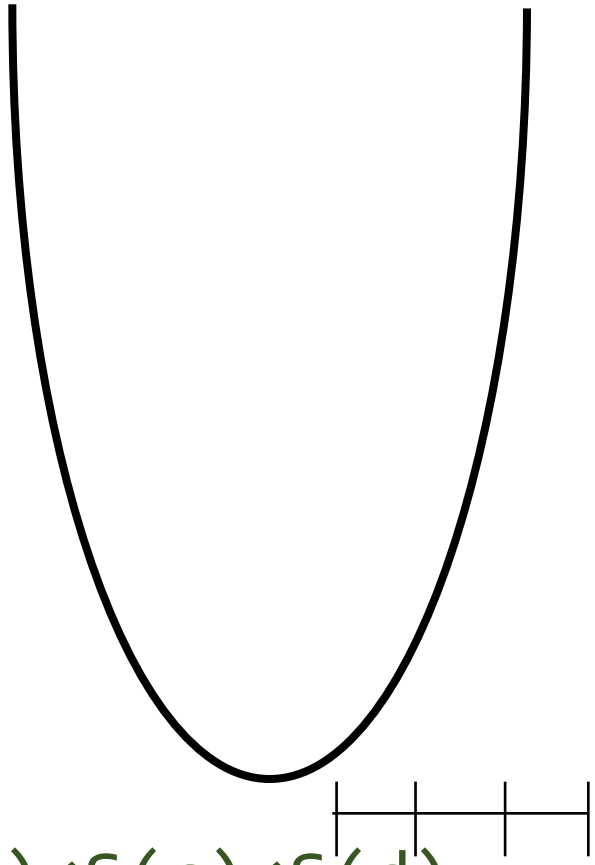


影片的片段

三分怎麼判？

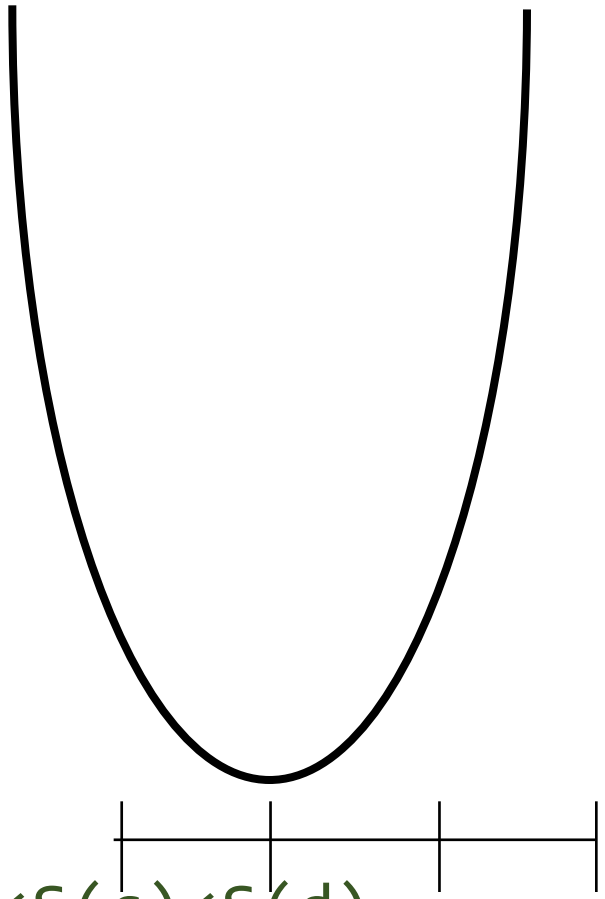
- 考慮三分後從左到右四個採樣點的關係……
- Case 1: $S(a) < S(b) < S(c) < S(d)$
  - 此時最小值一定不在最右邊
- Case 2: $S(a) > S(b) < S(c) < S(d)$
  - 此時最小值一定不在最右邊
- Case 3: $S(a) > S(b) > S(c) < S(d)$
  - 此時最小值一定不在最左邊
- Case 4: $S(a) > S(b) > S(c) > S(d)$
  - 此時最小值一定不在最左邊

影片的片段



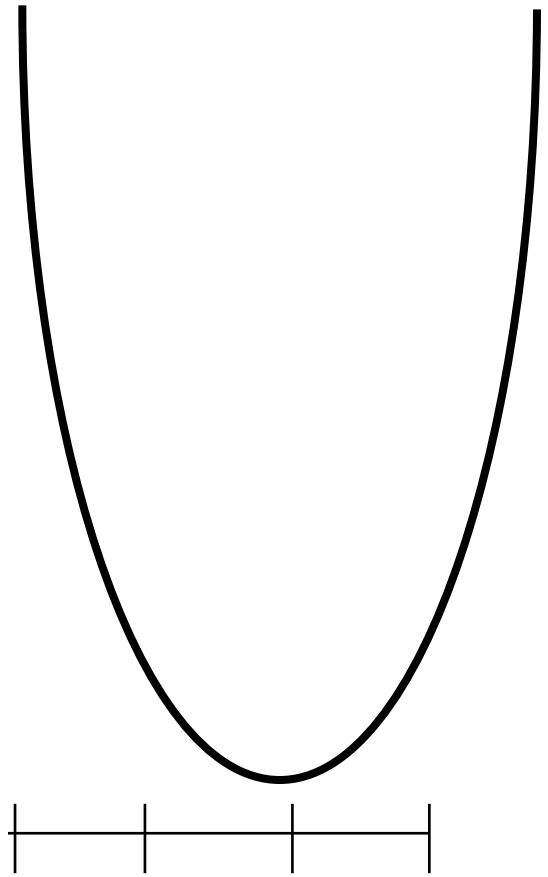
- Case1: S(a)<S(b)<S(c)<S(d)

影片的片段



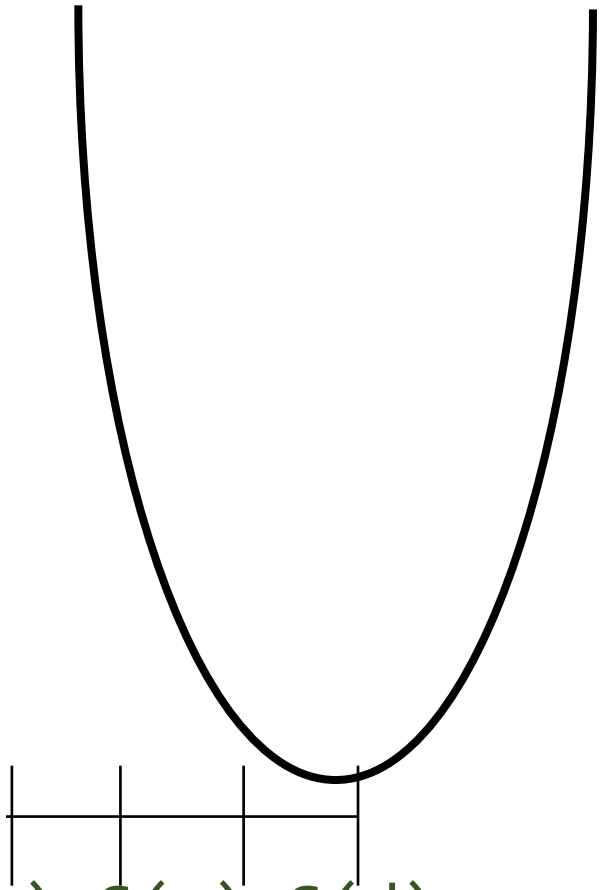
- Case2: S(a)>S(b)<S(c)<S(d)

影片的片段



- Case3: S(a)>S(b)>S(c)<S(d)

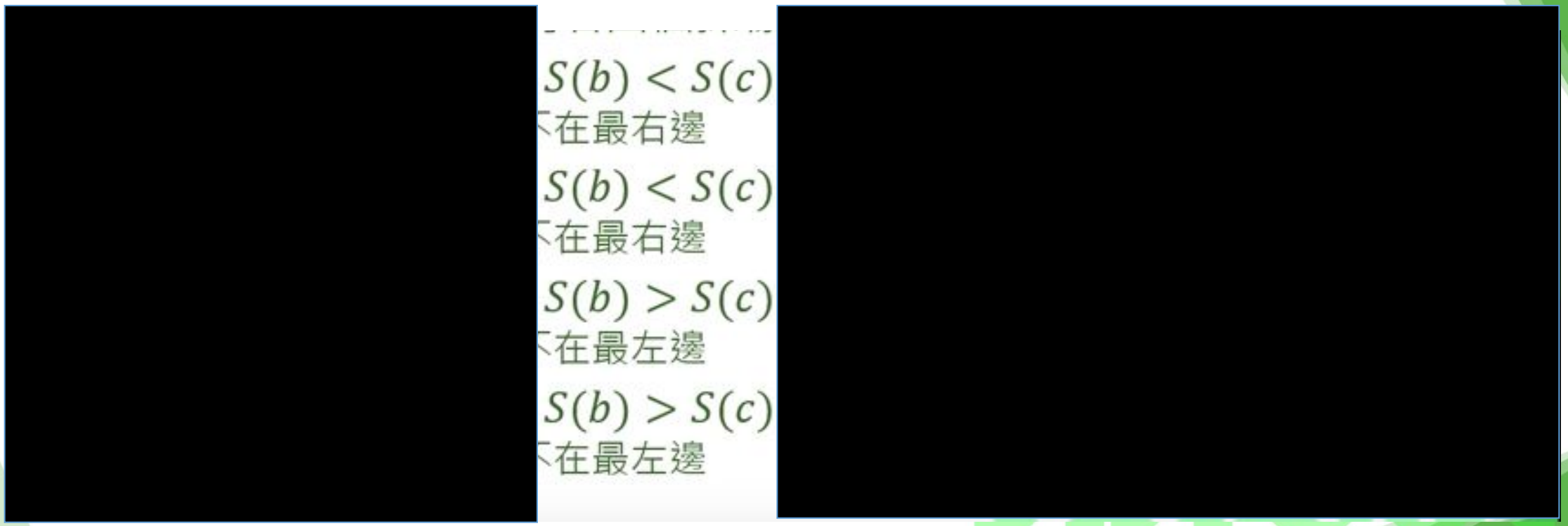
影片的片段



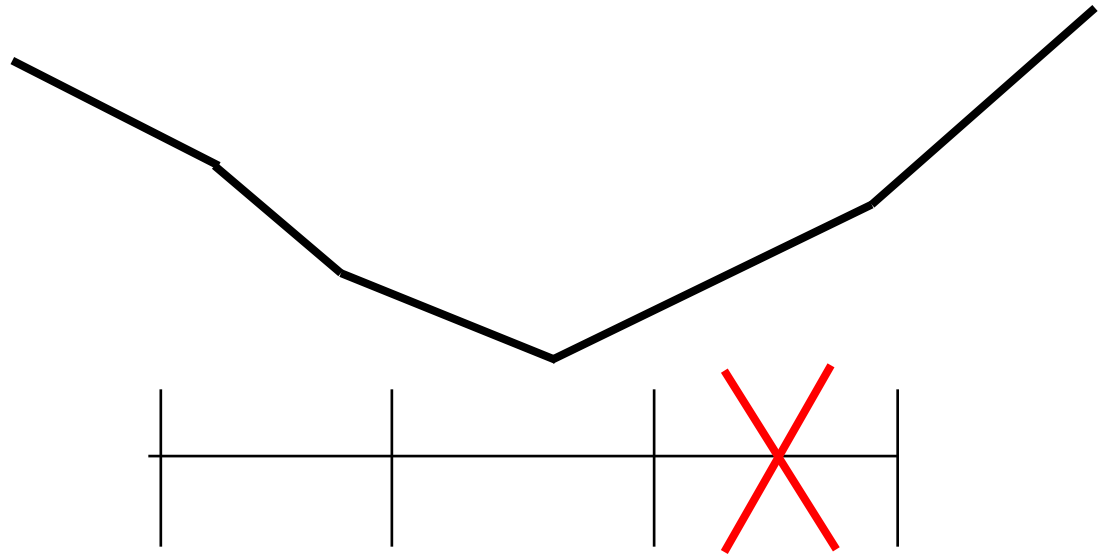
- Case2: S(a)>S(b)>S(c)>S(d)

影片的片段

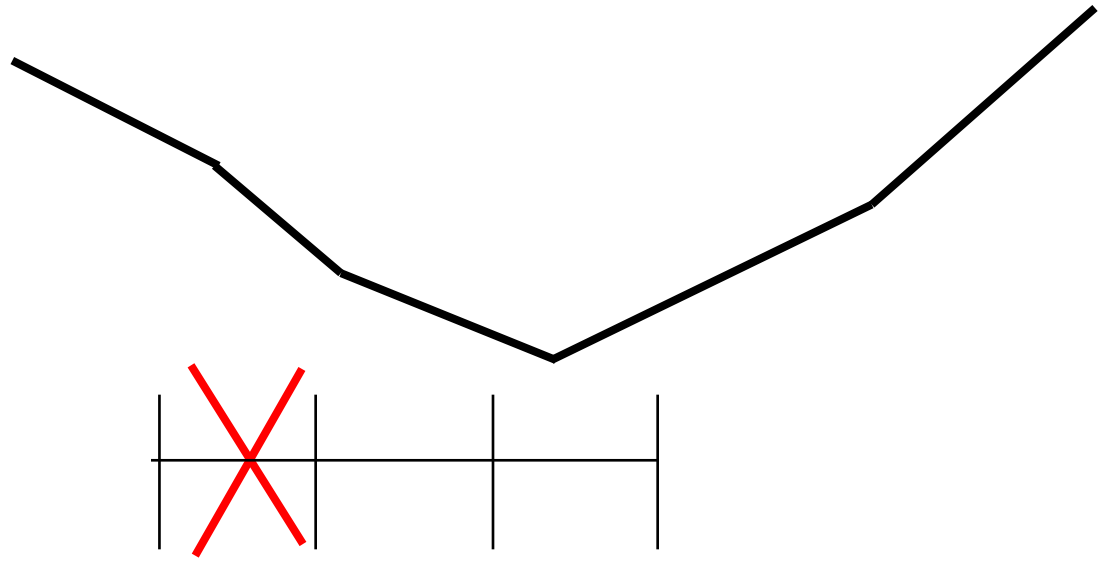
- 試著遮掉一些東西看看。
- 只要比較中間兩項的大小就好！



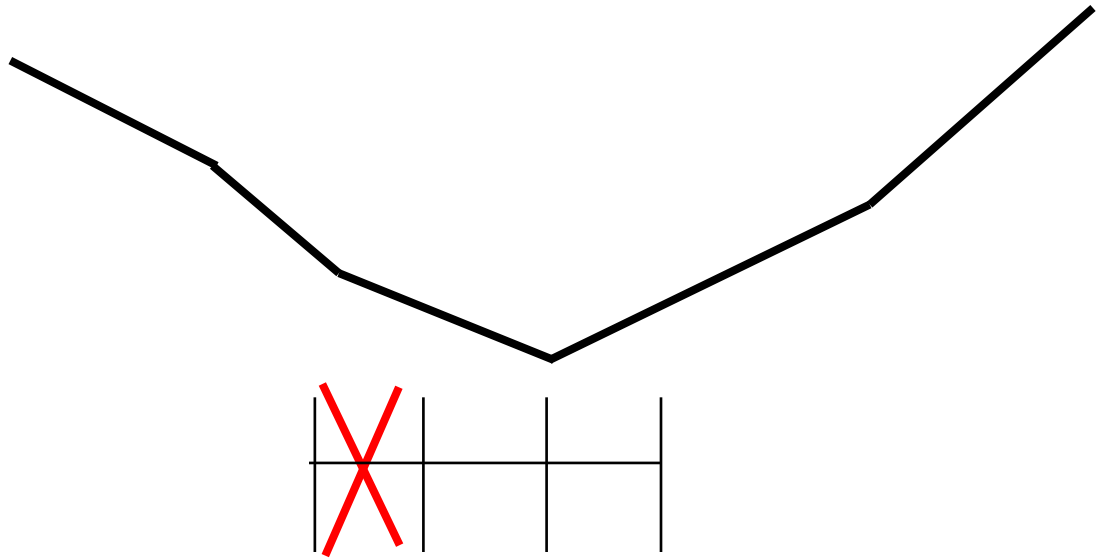
圖解三分搜



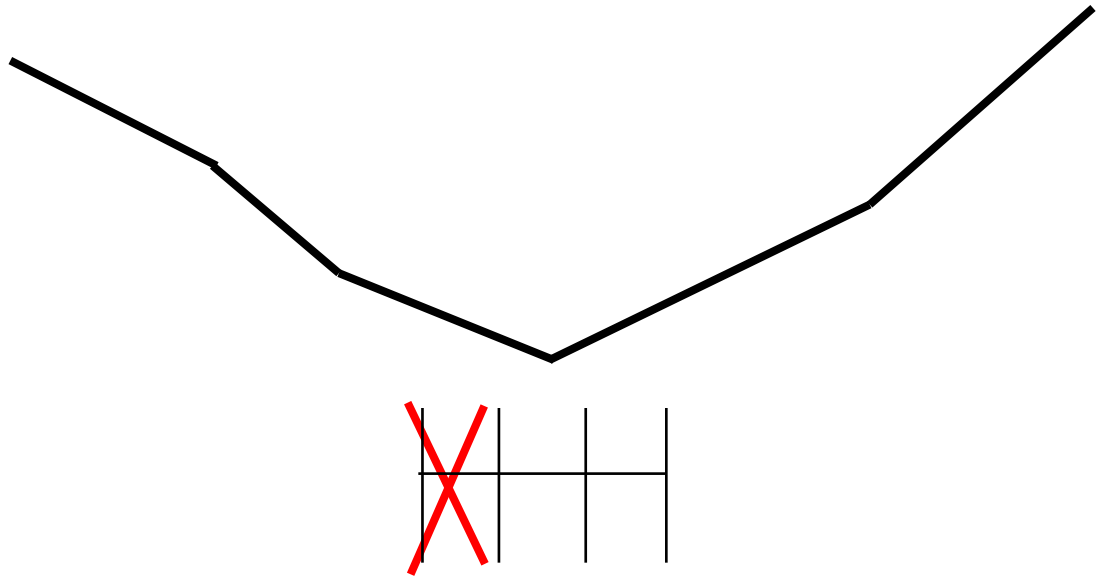
圖解三分搜



圖解三分搜



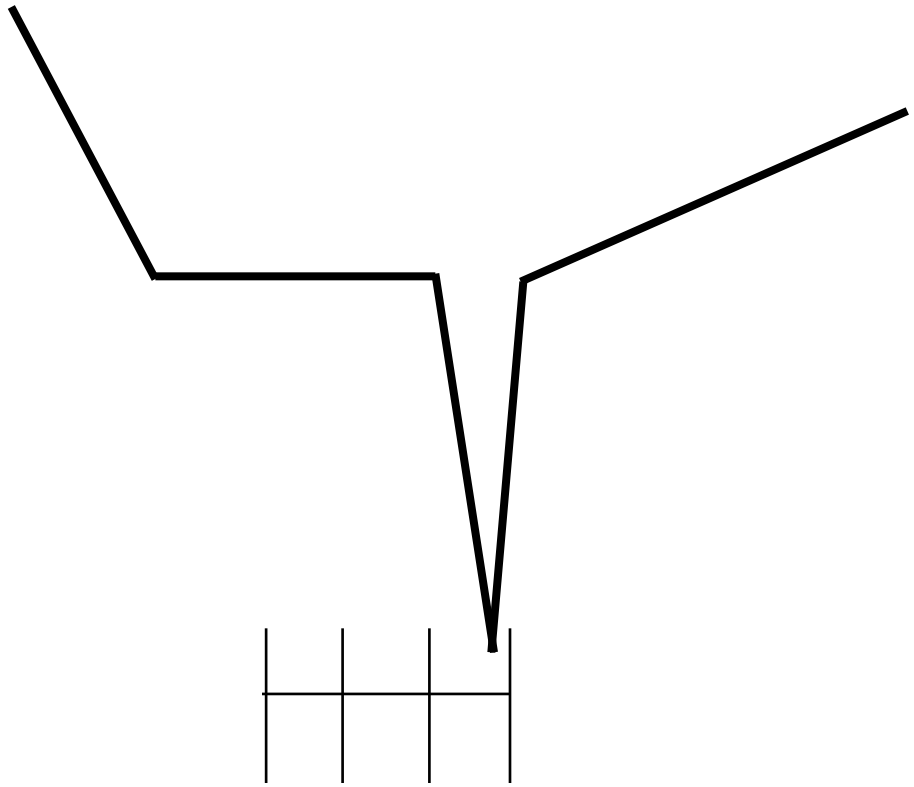
圖解三分搜



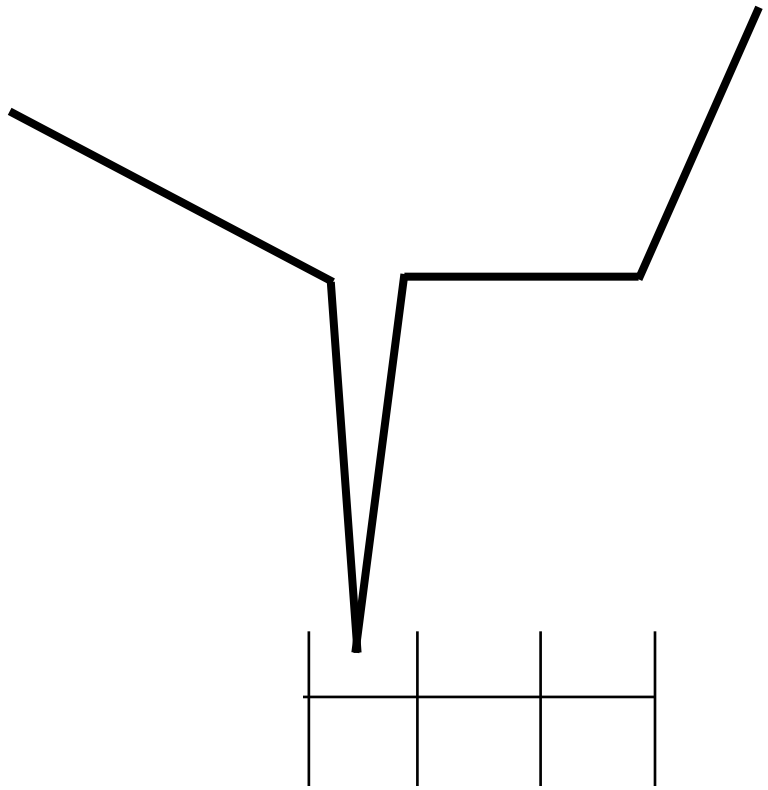
水平線

- 有些題目的單調性會是非嚴格遞增，二分搜的時候這並不成問題，因為二分搜時會是越右(左)邊越好之類的，所以你會知道要往哪一邊砍。
- 那麼三分搜呢？如果函數圖形有水平線也可以三分搜？
- 答案是不太行。
  - 廢話要是可以的話我幹嘛特別提出來。
  - 那為什麼不行呢？不太行又是怎麼一回事？

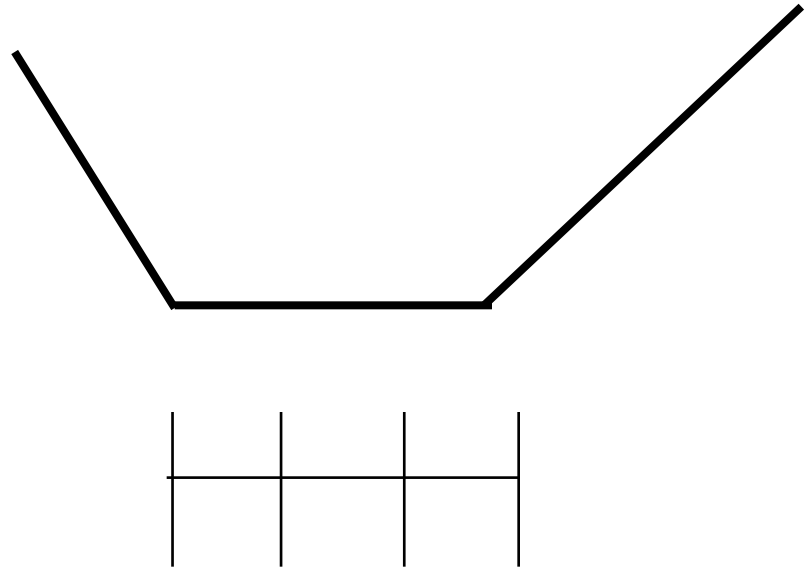
水平線-圖解



水平線-圖解



水平線-圖解



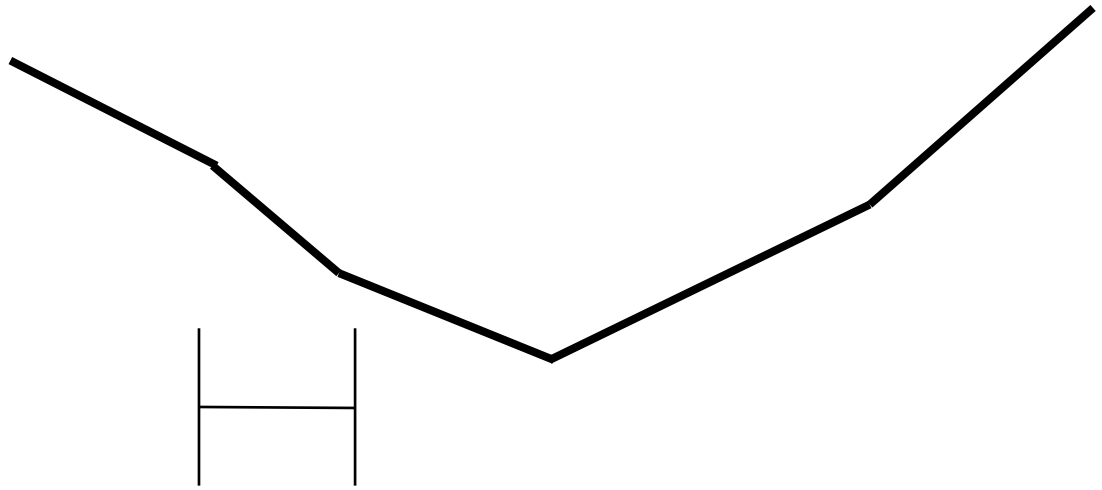
水平線-結論

- 三分搜在有水平線的情況下，必須確保水平線的地方一定是答案才可以用三分搜找極值。
- 否則的話，三分搜會壞掉。

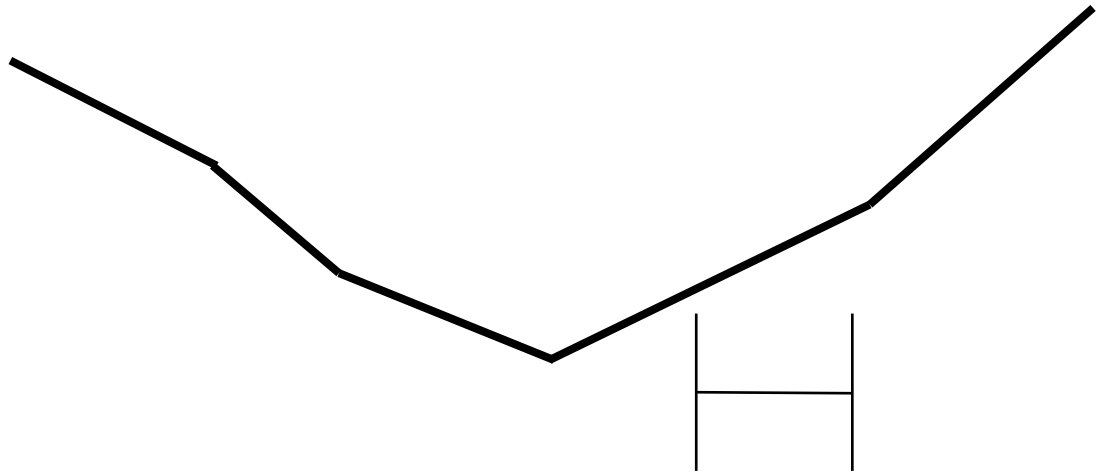
整數三分搜

- 如果今天我們要三分搜的點全是整數會發生什麼事？
- 注意到「決策點有限」

整數三分搜



整數三分搜

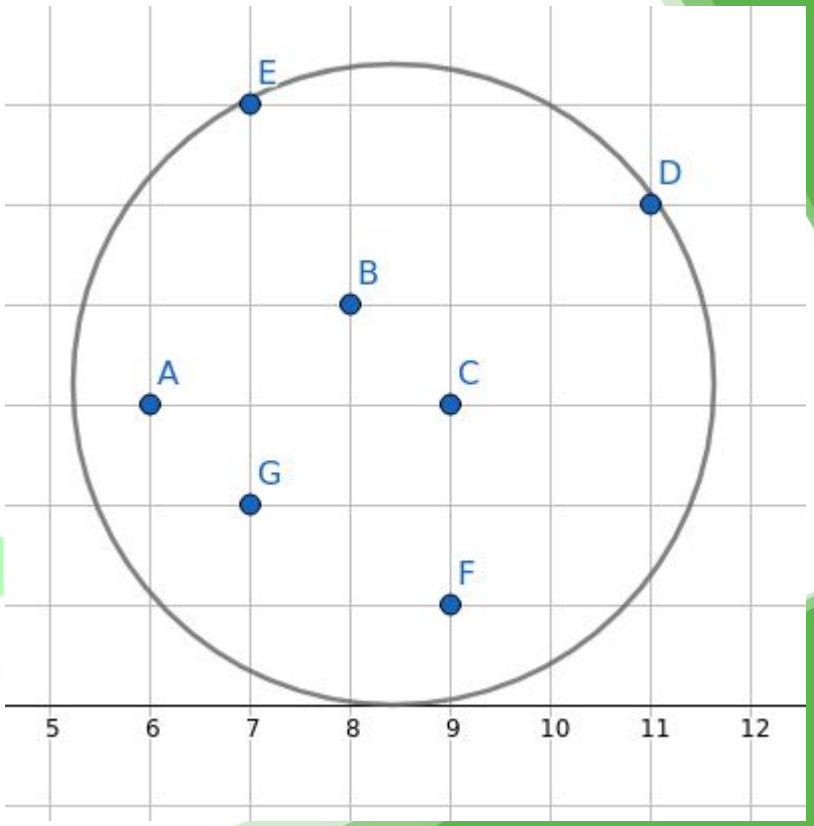


整數三分搜

- 整數三分搜可以被轉換成二分搜！
- 類似於「微分」的概念，詢問mid跟mid+1來比大小，再選擇要拋棄哪邊就好了

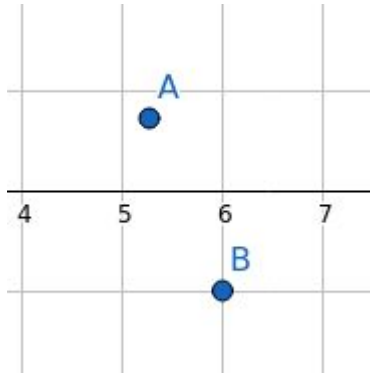
例題:最小包覆圓的限制情況

- 給你二維平面上的N個點，問你包覆所有點的最小圓半徑是多少？
- 特殊限制:這個包覆圓必須恰與x軸交一點
- N<=10^5
- |座標範圍|<=10^7
- 保證沒有點的y座標是0



例題:最小包覆圓的限制情況

- 同時有y>0跟y<0的點出現的話，肯定無解
- 如果全部的y<0，那我們可以把全部的y取相反數之後做y>0的情況
  - 所以接下來我們假設所有的y>0



例題:最小包覆圓的限制情況

- 有什麼東西是我們可以枚舉的？

例題:最小包覆圓的限制情況

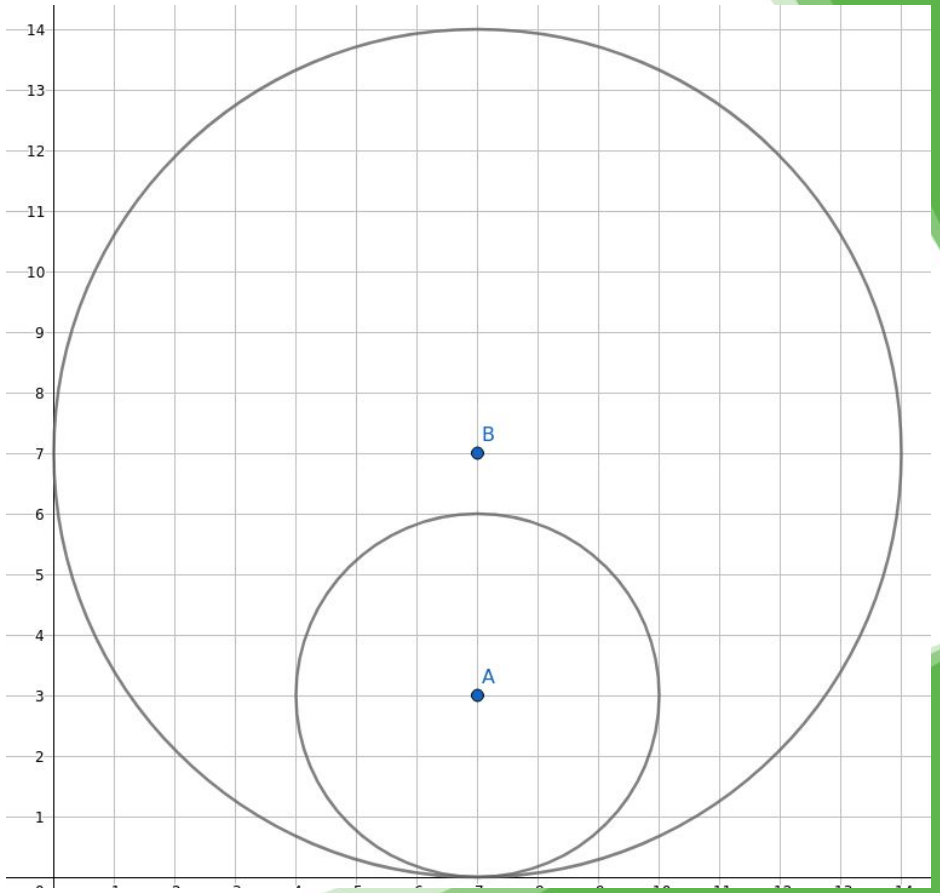
- 有什麼東西是我們可以枚舉的？

例題:最小包覆圓的限制情況

- 有什麼東西是我們可以枚舉的？
  - 圓心的x座標！

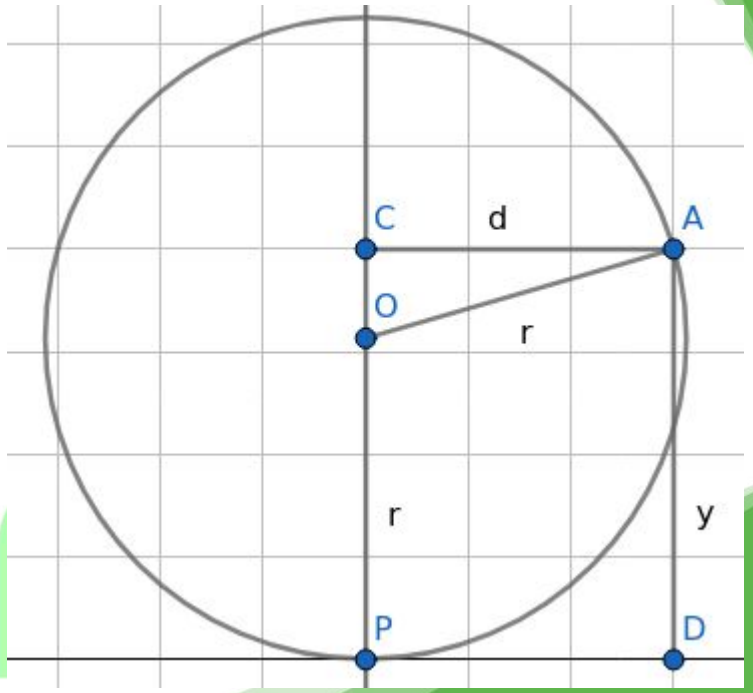
例題:最小包覆圓的限制情況

- 倘若我們枚舉到圓心的x座標必須要在p……
- 圓心越高，可以包到的點就越多！
  - 好像可以再二分搜圓心的高度...?
  - 當然不用這麼麻煩



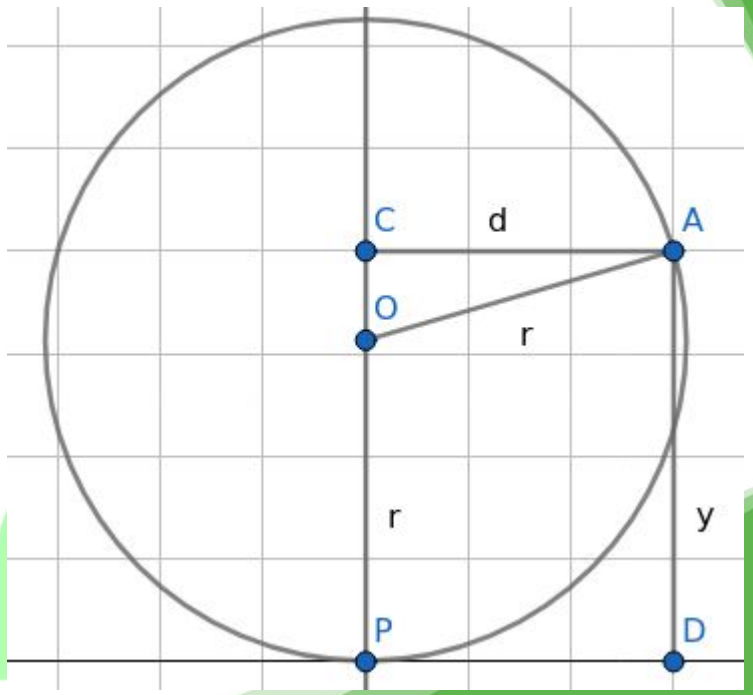
例題:最小包覆圓的限制情況

- 倘若我們枚舉到圓心的x座標必須要在p……
- 對於一個點(x,y)，如果圓心的x座標在p，那圓心至少要多高才可以蓋到他？



例題:最小包覆圓的限制情況

- 倘若我們枚舉到圓心的x座標必須要在p……
- 對於一個點(x,y)，如果圓心的x座標在p，那圓心至少要多高才可以蓋到他？
- $(y-r)^2+d^2=r^2$
  $\Rightarrow y^2+d^2=2yr$
  $\Rightarrow r=\frac{y^2+d^2}{2y}$
- r只要超過某個值
- 就可以蓋到他！



例題:最小包覆圓的限制情況

- 所以我們只需對於每個點，求出最小可以蓋到他的r
- 最後取max，就是圓心在p的答案了
  - 可是可能的圓心x座標有很多種啊？
- 看看我們取max的那些r:

例題:最小包覆圓的限制情況

- 所以我們只需對於每個點，求出最小可以蓋到他的r
- 最後取max，就是圓心在p的答案了
  - 可是可能的圓心x座標有很多種啊？
- 看看我們取max的那些r:
- $r = \frac{y^2+d^2}{2y}$

  $\Rightarrow r = \frac{y^2+(x-p)^2}{2y}$

例題:最小包覆圓的限制情況

- 所以我們只需對於每個點，求出最小可以蓋到他的r
- 最後取max，就是圓心在p的答案了
  - 可是可能的圓心x座標有很多種啊？
- 看看我們取max的那些r:
- $r = \frac{y^2+d^2}{2y}$

  $\Rightarrow r = \frac{y^2+(x-p)^2}{2y}$

- 看起來像是二次函數

例題:最小包覆圓的限制情況

- 所以我們只需對於每個點，求出最小可以蓋到他的r
- 最後取max，就是圓心在p的答案了
  - 可是可能的圓心x座標有很多種啊？
- 看看我們取max的那些r:
- $r = \frac{y^2+d^2}{2y}$

  $\Rightarrow r = \frac{y^2+(x-p)^2}{2y}$

- 看起來像是二次函數
- 那我們不就是在求一群二次函數的max嗎？

例題:最小包覆圓的限制情況

- 對於每個p, 都會對應到一群二次函數的max
- 三分搜！
- O(NlogC)

例題:最小包覆圓的限制情況

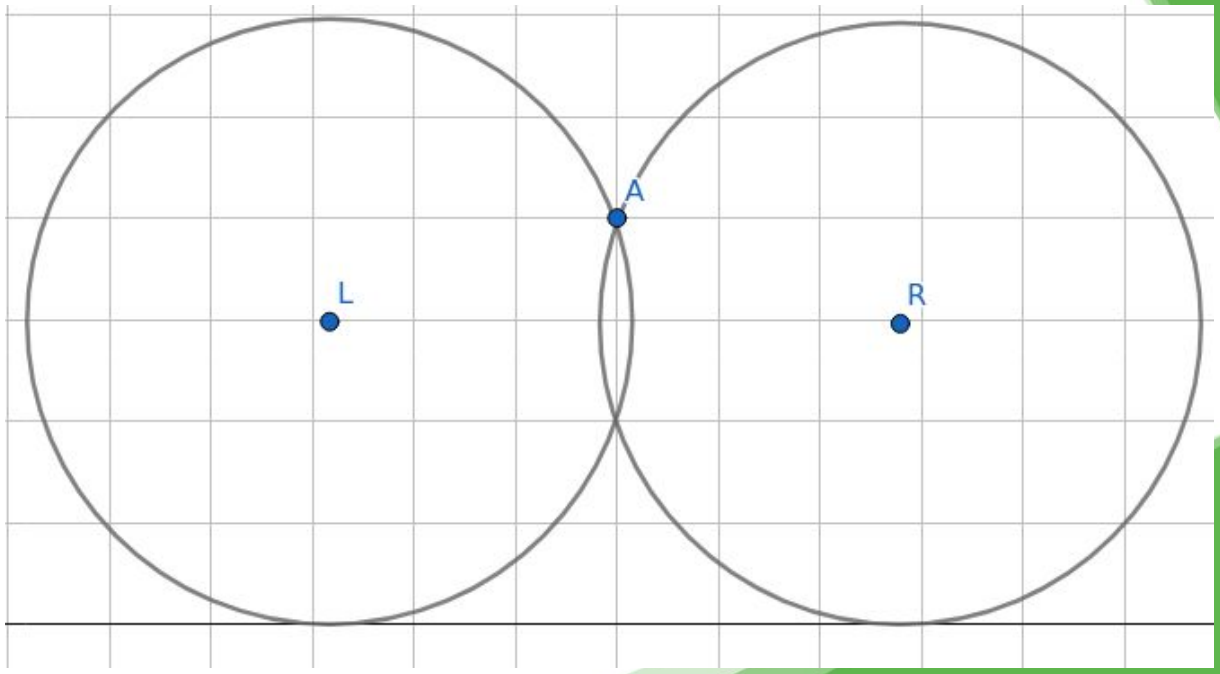
- 有什麼東西是我們可以枚舉的？
  - 半徑！

例題:最小包覆圓的限制情況

- 不妨看看，我們有沒有辦法用半徑r的圓蓋住所有點
- 如果確定了r，表示圓心的y座標已經確定了
  - 所以我們需要找到符合的x座標
- 對於一個點(x,y)，有哪些x座標可以把圓心擺在上面蓋住他呢？

例題:最小包覆圓的限制情況

- 對於一個點(x,y)，有哪些x座標可以把圓心擺在上面蓋住他呢？
- 可以發現是一個區間！(右下圖的L~R)



例題:最小包覆圓的限制情況

- 既然對於每個點我們都能算出一個合法區間
- 那最後只要看這些區間有沒有交集就行了
- 可是要枚舉的半徑有無限多種，該怎麼辦？
- 看起來半徑是否合法存在著單調性
- 二分搜！
- O(NlogC)

影片最後的想一想

- 在 Lotto 中，如果每次不是固定輸出 $M = 6$ 個數字，而是可以自由決定 M，如 $M = k$ 等，那麼 BFS 和 DFS 的空間差距會變得多大？試著以複雜度表示出來，代入幾個數值看看兩者的差異。
- 在數獨問題中，如果完全不做任何加速，並且真的遍歷完所有節點（例如無解狀態），複雜度是多少呢？這結果告訴我們，如果存在暴力搜索以外的解法，我們通常會選擇哪種解法？
- 假如我們用了一個暴力搜索的作法，共搜索了 k 層，那麼通常大部分的時間會花在哪一層的搜索？為什麼？
- 假設你現在有一個「估價函數」，在搜索到某一個節點時，只要呼叫一次估價函數，它就可以回傳這個節點是否有希望拓展出一組可行解，從而提供剪枝參考，但是呼叫一次成本為 $O(n^2)$。這種情況下，要選擇使用該函數（但是讓整體複雜度乘上 $O(n^2)$）好，還是選擇不使用（損失一些剪枝機會）好呢？

影片最後的想一想

- Lotto中，當M=k時，DFS與BFS的空間差距會多大？
- 其實影片裡面有答案
- By二項式定理，$\Sigma^{M}_{i=0}C^{M}_{i}=2^{M}$
- M=10時，$2^{M}=1024$
- M=20時，$2^{M}=1048576$

項目	BFS	DFS
時間複雜度	$O(\sum_{i=0}^{M} C_i^k + MC_M^k)$	$O(\sum_{i=0}^{M} C_i^k + MC_M^k)$
空間複雜度	$O(\sum_{i=0}^{M} C_i^k)$	$O(M)$
實作複雜度	高	低
額外優勢	無	M 較大時仍適用

影片最後的想一想

- 數獨問題如果跑完所有可能的話，複雜度會是多少？
- 每個位子都可以填1~9，有81個格子。
- 這樣有$9^{81} \approx 1.966 * 10^{77}$種可能。

影片最後的想一想

- 假如在暴搜的時候搜索的k層，通常大部分時間會花在哪一層？
- 基本上都會是在第k層。
- 暴搜的狀態樹的節點數通常隨著深度都是指數成長。
- 如果每層都長三倍的話，最後一層的節點數就跟前面所有其他層加起來的兩倍差不多了。

影片最後的想一想

- 如果有個估價函數，花時間計算估價函數可以減少狀態樹的節點數，應不應該用呢？
- 這不好說。
- 看題目而定，通常需要試過了才知道會不會變快。