分治 Divide and Conquer

Lecture by WiwiHo
Credit: baluteshih, yp155136, TreapKing

Sprout

開始之前

- 影片看了嗎？
- 作業寫了嗎？
- 如果還沒，趕快回去叫兩個星期前的自己不要懶惰 (X)
- 我們會先簡單複習過影片的東西

大綱

- 前言
- 遞迴演算法複雜度分析
- 你應該要會的分治
  - 最大連續和
  - 平面最近點對
- 分治演算法賞析 (?)
  - 多項式乘法 - Karatsuba algorithm
  - 尋找第 K 大 - Median of medians
  - 更多神奇演算法
- 總結

前言

影片講了很多題

- 分治構造
  - 棋盤挖空一格放 L 形
  - 沒長度 ≥ 3 等差數列的 permutation
- 藏著分治的問題
  - 二分搜尋法
  - 快速冪
- 分治與排序
  - Merge sort
  - Quick sort
- 進化版問題
  - 逆序數對

分治是什麼

- 把東西切一半然後遞迴下去做

分治是什麼

- 把東西切一半然後遞迴下去做
  - 當然不可能只有這樣
- 把問題**分**成小問題，再把小問題的答案**合併**
  - 這個合併可以是單純的把一些東西合起來
  - 也可以非常複雜，像是逆序數對的合併要算跨兩邊的答案
  - 同樣地，分也可以像逆序數對那樣很單純地分兩邊，或是像 Quick sort 用特別的方式分兩堆

何時使用分治

- 經驗！
  - 刷題！
- 面對一個問題時，怎麼枚舉都找不到優化的切入點
- 突發奇想從中間切一半，就可能找到很棒的 conquer 性質
- 當這個性質足夠讓我們在良好的複雜度解決 conquer 時，就可能可以在犧牲頂多一個 log 的情況下完成整個問題
- 「在 conquer 時能夠省去維護儲存所有資訊的力氣，只需要專心處理跨分隔線的資訊」，就是分治的精髓

分治小技巧

- 養成良好習慣
  - 左閉右閉（我個人常用的方式）
  - 左閉右開
  - ......(?)
- divide 完之後遞迴下去，直接假設遞迴後得到的結果是理想的，這樣 conquer 的時候會比較好思考
- 當 n 很小的時候，可以暴力做，可能會減少遞迴的 cost
- EX: merge sort 到 n 很小時，就隨便用一個 $O(n^2)$ 的 sort
- 多寫題目 (?)

遞迴演算法複雜度分析

複雜度分析

- 影片上用到的複雜度分析方法都是算「每一層要多久」和「有幾層」

複雜度分析

- 影片上用到的複雜度分析方法都是算「每一層要多久」和「有幾層」
- 具體一點

$\text{mergesort}(1, n)$

$\text{mergesort}(1, n/2)$ $\text{mergesort}(n/2, n)$

$\text{mergesort}(1, n/4)$ $\text{mergesort}(n/4, n/2)$ $\text{mergesort}(n/2, 3n/4)$ $\text{mergesort}(3n/4, n)$

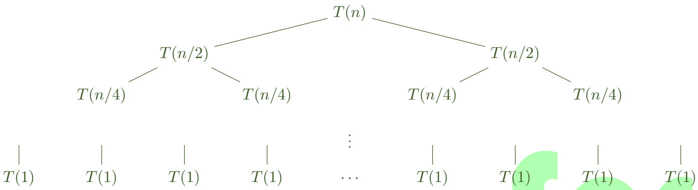
$\vdots$

$\text{mergesort}(1, 1)$ $\text{mergesort}(2, 2)$ $\cdots$ $\text{mergesort}(n-1, n-1)$ $\text{mergesort}(n, n)$

複雜度分析：Merge sort

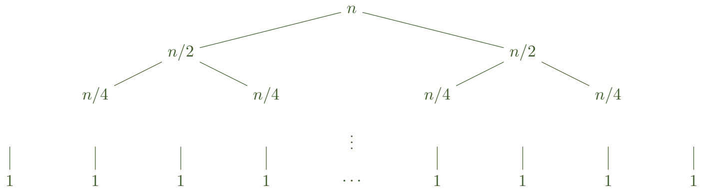
$T(n) =$ 把長度為 n 的陣列排序好要花的時間

$$T(n) = 2T\left(\frac{n}{2}\right) + n, \quad T(1) = 1$$



複雜度分析：Merge sort

那個多加上去的 n 的總和就是答案！



很明顯地可以看出來一層總和是 n，共有 $O(\log n)$ 層，總時間是 $O(n \log n)$

Recursion Tree Method

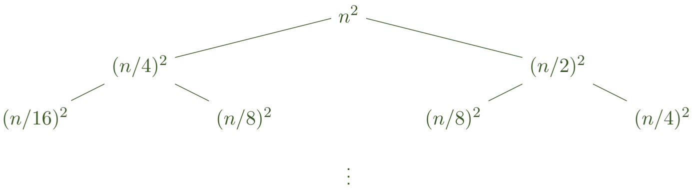
- 這叫作 Recursion Tree Method

Recursion Tree Method

- 這叫作 Recursion Tree Method
- 看一個複雜一點的例子：

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$$

Recusion Tree Method



- 第一層是 n^2，第二層是 $\frac{5}{16}n^2$，第三層是 $\frac{25}{256}n^2$

Recusion Tree Method

$$n^2$$

$$(n/4)^2 \qquad (n/2)^2$$

$$(n/16)^2 \qquad (n/8)^2 \qquad (n/8)^2 \qquad (n/4)^2$$

$$\vdots$$

- 第一層是 n^2，第二層是 $\frac{5}{16}n^2$，第三層是 $\frac{25}{256}n^2$
- 所以深度 i 那層是 $\left(\frac{5}{16}\right)^i n^2$ (??)
- 深度有 $O(\log n)$ 層 (??)

Recusion Tree Method

$$n^2$$

$$(n/4)^2 \qquad\qquad (n/2)^2$$

$$(n/16)^2 \qquad (n/8)^2 \qquad (n/8)^2 \qquad (n/4)^2$$

$$\vdots$$

- 第一層是 n^2，第二層是 $\frac{5}{16}n^2$，第三層是 $\frac{25}{256}n^2$
- 所以深度 i 那層是 $\left(\frac{5}{16}\right)^i n^2$ (??)
- 深度有 $O(\log n)$ 層 (??)

$$1 + \left(\frac{5}{16}\right) + \left(\frac{5}{16}\right)^2 + \cdots \leq \frac{1}{1 - \frac{5}{16}}$$

- 總時間複雜度是 $O(n^2)$ (??)

Substitution Method

- 剛剛那個跟打表找規律有什麼不一樣 = =
- 不如我們用數學歸納法證明看看

Substitution Method

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$$

- 我們要證明存在整數 c, n_0，使得對於所有整數 $n \geq n_0$，都有 $T(n) \leq cn^2$
  - 我們直接讓 $n_0 = 1$，並且假裝我們知道 c，實際上我們等一下會列出 c 要滿足的條件

Substitution Method

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$$

- 我們要證明存在整數 c, n_0，使得對於所有整數 $n \geq n_0$，都有 $T(n) \leq cn^2$
  - 我們直接讓 $n_0 = 1$，並且假裝我們知道 c，實際上我們等一下會列出 c 要滿足的條件
- 當 $n = 1$ 時，$T(n) = 1 \leq cn^2 = c$，只要 $c \geq 1$ 就滿足條件

Substitution Method

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$$

- 我們要證明存在整數 c, n_0，使得對於所有整數 $n \geq n_0$，都有 $T(n) \leq cn^2$
  - 我們直接讓 $n_0 = 1$，並且假裝我們知道 c，實際上我們等一下會列出 c 要滿足的條件
- 當 $n = 1$ 時，$T(n) = 1 \leq cn^2 = c$，只要 $c \geq 1$ 就滿足條件
- 當 $n = k > 1$ 時，假設對於 $n' < k$ 都有 $T(n') \leq c(n')^2$，那麼：
  - 我們知道：$T(n) \leq c\left(\frac{n}{4}\right)^2 + c\left(\frac{n}{2}\right)^2 + n^2 = \left(1 + \frac{5c}{16}\right)n^2$
  - 我們想要：$T(n) \leq cn^2$，也就是 $\left(1 + \frac{5c}{16}\right) \leq c$
  - 只要 $c \geq \frac{16}{11}$ 就滿足條件，$T(n)$ 也 $\leq cn^2$

Substitution Method

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$$

- 我們要證明存在整數 c, n_0，使得對於所有整數 $n \geq n_0$，都有 $T(n) \leq cn^2$
  - 我們直接讓 $n_0 = 1$，並且假裝我們知道 c，實際上我們等一下會列出 c 要滿足的條件
- 當 $n = 1$ 時，$T(n) = 1 \leq cn^2 = c$，只要 $c \geq 1$ 就滿足條件
- 當 $n = k > 1$ 時，假設對於 $n' < k$ 都有 $T(n') \leq c(n')^2$，那麼：
  - 我們知道：$T(n) \leq c\left(\frac{n}{4}\right)^2 + c\left(\frac{n}{2}\right)^2 + n^2 = \left(1 + \frac{5c}{16}\right) n^2$
  - 我們想要：$T(n) \leq cn^2$，也就是 $\left(1 + \frac{5c}{16}\right) \leq c$
  - 只要 $c \geq \frac{16}{11}$ 就滿足條件，$T(n)$ 也 $\leq cn^2$
- 所以只要我們選 $c = \frac{16}{11}$ 根據數學歸納法，就有 $T(n) \leq cn^2$

Substitution Method

- Recursion Tree Method 只能拿來猜一個複雜度，實際上不太嚴謹
- Substitution Method 可以在你已經有一個**猜測**的前提下，**證明**這個複雜度
- 要熟悉複雜度的定義！

Master Theorem

- 主定理
- 前面那些很麻煩？沒關係，大多數時候都可以直接用主定理！

Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- Case 1: $\exists \epsilon > 0,\ f(n) \in O(n^{\log_b a - \epsilon})$，
  那麼 $T(n) \in \Theta(n^{\log_b a})$
- Case 2: $\exists k \geq 0,\ f(n) \in \Theta(n^{\log_b a} \log^k n)$，
  那麼 $T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$
- Case 3: $\exists \epsilon > 0,\ f(n) \in \Omega(n^{\log_b a + \epsilon})$，且存在 $k < 1$，對於足夠大的 n，都有
  $af\left(\frac{n}{b}\right) \leq kf(n)$
  那麼 $T(n) \in \Theta(f(n))$

Master Theorem

- 白話一點 QQ
- 不嚴謹的說，就是比較 $f(n)$ 跟 $O(n^{\log_b a})$ 的關係
- 如果差不多，那就加個 $\log$，否則就是比較大的那個
- 例子：
  - Case 1: $T(n) = 9T(\frac{n}{3}) + n$
  - Case 2: $T(n) = 2T(\frac{n}{2}) + n$
  - Case 3: $T(n) = T(\frac{n}{2}) + n^2$

複雜度分析：小結

- 數學理論的部份就講到這邊啦
- 什麼東西都記不起來，也可以暫時只記得 Master Theorem 就好 XD
- 接下來要講實際上會遇到的題目，~~前面睡著的可以起床了~~

你應該要會的分治

最大連續和

Problem 最大連續和[1]

給你一個長度為 N 的序列 $a_1, a_2, \ldots, a_N$，請你找到 (L, R)，滿足 $a_L + \cdots + a_R$ 最大。

- $N \leq 2 \times 10^5$

[1]可以傳的地方：CSES Maximum Subarray Sum

最大連續和

- $O(N^3)$
- $O(N^2)$
- $O(N \log N)$
- $O(N)$
- 那換一個限制，只能用「分治法」來做
  - 因為我們現在在教分治嘛

最大連續和

- 答案有 $O(N^2)$ 種

最大連續和

- 答案有 $O(N^2)$ 種
- 我們有沒有辦法好好的 divide 成一半，然後想盡辦法 conquer 起來呢？

最大連續和

- 答案有 $O(N^2)$ 種
- 我們有沒有辦法好好的 divide 成一半，然後想盡辦法 conquer 起來呢？
- 既然都那麼說了那當然可以
- 分治的常用想法：把序列切成兩半，兩邊各自遞回求解，再算跨越兩邊的解

最大連續和

- Solve(L, R)：
  - 輸入是原序列上的一段區間 $[L, R]$
  - 回傳值是 $[L, R]$ 裡的最大連續和大小

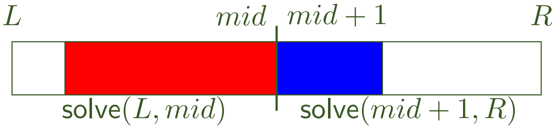
最大連續和

- Solve(L, R)：
  - 輸入是原序列上的一段區間 $[L, R]$
  - 回傳值是 $[L, R]$ 裡的最大連續和大小
- 我們只要專注於找出跨越中線的最大連續和，其餘的部分就由子問題幫我們解決！

最大連續和

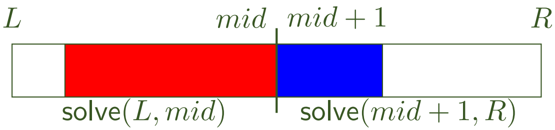
- Solve(L, R)：
  - 輸入是原序列上的一段區間 $[L, R]$
  - 回傳值是 $[L, R]$ 裡的最大連續和大小
- 我們只要專注於找出跨越中線的最大連續和，其餘的部分就由子問題幫我們解決！
- 跨越兩邊的區間會長什麼樣子呢？

最大連續和



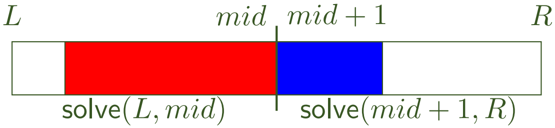
- 任何跨越兩邊的區間都可以分為左半和右半兩個部分

最大連續和



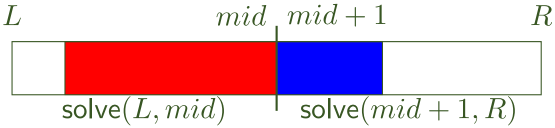
- 任何跨越兩邊的區間都可以分為左半和右半兩個部分
- 左半總是貼著中間，右半也總是貼著中間

最大連續和



- 任何跨越兩邊的區間都可以分為左半和右半兩個部分
- 左半總是貼著中間，右半也總是貼著中間
- 也就是說，「跨越中間的區間」就是「從中間往左邊選一些、往右邊選一些得到的區間」

最大連續和



- 任何跨越兩邊的區間都可以分為左半和右半兩個部分
- 左半總是貼著中間，右半也總是貼著中間
- 也就是說，「跨越中間的區間」就是「從中間往左邊選一些、往右邊選一些得到的區間」
- 兩半肯定各自都是越大越好！

$$\max_{L\le l\le mid<r\le R}\{a_l+\cdots+a_r\} = \max_{L\le l\le mid}\{a_l+\cdots+a_{mid}\}+\max_{mid<r\le R}\{a_{mid+1}+\cdots+a_r\}$$

最大連續和

```
int solve(int L, int R) {
    if (L == R) {
        return a[L];
    }
    int mid = (L + R) >> 1;
    int ans = max(solve(L, mid), solve(mid + 1, R));
    int lmax = a[mid], lpre = a[mid];
    for (int i = mid - 1; i >= L; --i) {
        lpre += a[i];
        lmax = max(lmax, lpre);
    }
    int rmax = a[mid + 1], rpre = a[mid + 1];
    for (int i = mid + 2; i <= R; ++i) {
        rpre += a[i];
        rmax = max(rmax, rpre);
    }
    return max(ans, lmax + rmax);
}
```

最大連續和：複雜度分析

- $T(n) = 2T(n/2) + O(n) \implies T(n) = O(n \log n)$

最大連續和：複雜度分析

- $T(n) = 2T(n/2) + O(n) \implies T(n) = O(n \log n)$
- Challenge：用分治法可以做到 $O(n)$ 嗎？

平面最近點對

Problem 平面最近點對[2]

在平面上給你 N 個點，要你找出歐氏距離最短的兩個點。

- $N \leq 2 \times 10^5$

[2]可以傳的題目：CSES Minimum Euclidean Distance

平面最近點對

- 開心 $\binom{N}{2} = O(N^2)$ 當然不是我們要的
- 如果在平面上想要做 divide and conquer，該怎麼分割問題？

平面最近點對

- 開心 $\binom{N}{2} = O(N^2)$ 當然不是我們要的
- 如果在平面上想要做 divide and conquer，該怎麼分割問題？
- 我們可以先把所有輸入的點照 x 座標排序後，在中間畫一條分隔線

平面最近點對

- 開心 $\binom{N}{2} = O(N^2)$ 當然不是我們要的
- 如果在平面上想要做 divide and conquer，該怎麼分割問題？
- 我們可以先把所有輸入的點照 x 座標排序後，在中間畫一條分隔線
- 這樣可能的答案就分成三種情況
  - 兩個點都在左邊
  - 兩個點都在右邊
  - 一個點在左邊、一個點在右邊

平面最近點對

- 開心 $\binom{N}{2} = O(N^2)$ 當然不是我們要的
- 如果在平面上想要做 divide and conquer，該怎麼分割問題？
- 我們可以先把所有輸入的點照 x 座標排序後，在中間畫一條分隔線
- 這樣可能的答案就分成三種情況
  - 兩個點都在左邊
  - 兩個點都在右邊
  - 一個點在左邊、一個點在右邊
- 一些平面上的 D&C 題都會做類似的事

平面最近點對

- 開心 $\binom{N}{2} = O(N^2)$ 當然不是我們要的
- 如果在平面上想要做 divide and conquer，該怎麼分割問題？
- 我們可以先把所有輸入的點照 x 座標排序後，在中間畫一條分隔線
- 這樣可能的答案就分成三種情況
  - 兩個點都在左邊
  - 兩個點都在右邊
  - 一個點在左邊、一個點在右邊
- 一些平面上的 D&C 題都會做類似的事
- 顯然只有點對「橫跨分隔線」的情況需要討論，如果這個情況可以解決的話，其他兩個情況只要遞迴下去解就好了

平面最近點對

- 如果只是要算分隔線兩邊的最近點對，有什麼好方法嗎？

平面最近點對

- 如果只是要算分隔線兩邊的最近點對，有什麼好方法嗎？
- 因為 x 座標的大小關係已經確立了，所以可以把兩邊直接照 y 座標排序
- 然後就發現還是好困難……

平面最近點對

- 定神一想，會發現如果遞迴下去後找到的最近點對的距離是 d ，那麼我們根本就不需要考慮那些距離超過 d 的點對

平面最近點對

- 定神一想，會發現如果遞迴下去後找到的最近點對的距離是 d ，那麼我們根本就不需要考慮那些距離超過 d 的點對
- 所以離分隔線超過 d 的點都不需要去考慮

平面最近點對

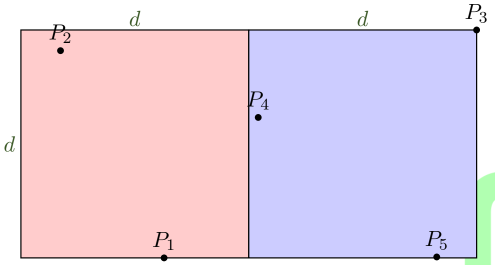
- 定神一想，會發現如果遞迴下去後找到的最近點對的距離是 d ，那麼我們根本就不需要考慮那些距離超過 d 的點對
- 所以離分隔線超過 d 的點都不需要去考慮
- 對於每個點，也只有 y 座標差距不超過 d 的點可能可以讓你找到更近的點對

平面最近點對

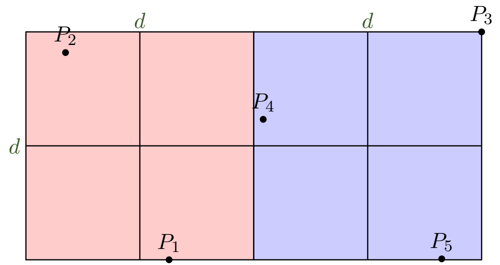
- 定神一想，會發現如果遞迴下去後找到的最近點對的距離是 d ，那麼我們根本就不需要考慮那些距離超過 d 的點對
- 所以離分隔線超過 d 的點都不需要去考慮
- 對於每個點，也只有 y 座標差距不超過 d 的點可能可以讓你找到更近的點對
- 這樣子複雜度會是好的嗎？
- 聽起來只是個壓常數的剪枝，但在什麼情況下，這個做法一樣會退化成 $O(N^2)$ 呢？

平面最近點對

- 因為已經知道各自的最近點對的距離是 d，所以每個點附近的點不會太多！
- 每個點究竟得去看幾個點呢？
- 每個點只需要看它上面 $2d \times d$ 的那個區域就好了（其實只要看對面那側）
- 假設我們現在關心的是 P_1，這個框框最多有多少點呢？



平面最近點對



- 每個小格子裡最多只有一個點
- P_1 只要往上看 7 個點！

平面最近點對

- 複雜度？

平面最近點對

- 複雜度？
- $T(n) = 2T(n/2) + O(n \log n)$，因為我們要對 y 座標排序，所以會有個 $\log$
- 這是 $O(n \log^2 n)$，好像很慢...

平面最近點對

- 複雜度？
- $T(n) = 2T(n/2) + O(n\log n)$，因為我們要對 y 座標排序，所以會有個 $\log$
- 這是 $O(n\log^2 n)$，好像很慢...
- 靈光一閃，想起 merge sort，後面那個 $O(n\log n)$，只要我們一邊分治一邊排序，就可以壓到 $O(n)$

平面最近點對

- 複雜度？
- $T(n) = 2T(n/2) + O(n \log n)$，因為我們要對 y 座標排序，所以會有個 log
- 這是 $O(n \log^2 n)$，好像很慢...
- 靈光一閃，想起 merge sort，後面那個 $O(n \log n)$，只要我們一邊分治一邊排序，就可以壓到 $O(n)$
- $T(n) = 2T(n/2) + O(n) \implies T(n) = O(n \log n)$

平面最近點對

- 實際上可以掃比 7 個更少的點
  - 5 個點
  - 3 個點
- 也有非分治的作法，有興趣可以上網查查

分治演算法賞析 (?)

分治演算法賞析

- 剛才和影片中都講了一些用到的想法比較基礎的分治演算法
- 接下來是一些經典的利用到分治的演算法
- 不過會更奇形怪狀通靈，可能主要是欣賞用 XD

多項式乘法

Problem 多項式乘法

給你兩個 n 次多項式，請你求出兩個多項式的乘積。

多項式乘法

- 暴力：用個雙重迴圈跑一跑乘一乘加一加，$O(n^2)$

多項式乘法

- 暴力：用個雙重迴圈跑一跑乘一乘加一加，$O(n^2)$
- 可能會有人想到 FFT（Fast Fourier Transform，快速傅立葉轉換），但我現在沒有要講這個

多項式乘法

- 把多項式表達為

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = \sum_{i=0}^{n} a_ix^i$$

$$g(x) = b_0 + b_1x + b_2x^2 + \cdots + b_nx^n = \sum_{i=0}^{n} b_ix^i$$

- 順便不失一般性的假設 $n+1$ 是 2 的冪次（如果不是的話，就加入一些係數為 0 的高次項）

多項式乘法

- 既然是在教分治，那就先把多項式切兩半看看

$$t = \frac{n-1}{2}$$
$$A = a_0 + a_1 x + \cdots + a_t x^t$$
$$B = a_{t+1} + a_{t+2} x + \cdots + a_n x^t$$
$$C = b_0 + b_1 x + \cdots + b_t x^t$$
$$D = b_{t+1} + b_{t+2} x + \cdots + b_n x^t$$

- $f(x)g(x) = (A + x^{t+1}B)(C + x^{t+1}D) = AC + x^{t+1}(AD + BC) + x^{n+1}BD$

多項式乘法

$$f(x)g(x) = (A + x^{\frac{n+1}{2}}B)(C + x^{\frac{n+1}{2}}D) = AC + x^{\frac{n+1}{2}}(AD + BC) + x^{n+1}BD$$

- A, B, C, D 都是 $(n+1)/2 - 1$ 次的多項式
- 所以做四次比較小的乘法和一點加法就可以解決原問題了！

多項式乘法

$$f(x)g(x) = (A + x^{\frac{n+1}{2}}B)(C + x^{\frac{n+1}{2}}D) = AC + x^{\frac{n+1}{2}}(AD + BC) + x^{n+1}BD$$

- A, B, C, D 都是 $(n+1)/2 - 1$ 次的多項式
- 所以做四次比較小的乘法和一點加法就可以解決原問題了！
- 這樣真的比較快嗎？

Sprout

多項式乘法

$$f(x)g(x) = (A + x^{\frac{n+1}{2}}B)(C + x^{\frac{n+1}{2}}D) = AC + x^{\frac{n+1}{2}}(AD + BC) + x^{n+1}BD$$

- A, B, C, D 都是 $(n+1)/2 - 1$ 次的多項式
- 所以做四次比較小的乘法和一點加法就可以解決原問題了！
- 這樣真的比較快嗎？
- $T(n) = 4T(n/2) + O(n)$，$T(n) \in O(n^2)$，沒有變快 QQ

Karatsuba Algorithm

$$f(x)g(x) = (A + x^{\frac{n+1}{2}}B)(C + x^{\frac{n+1}{2}}D) = AC + x^{\frac{n+1}{2}}(AD + BC) + x^{n+1}BD$$

- 我們要算 $AC, AD + BC, BD$
- 靈光一閃，如果我們計算

$$E = (A + B) \times (C + D) = AC + AD + BC + BD$$

- 就可以把原本的式子化為

$$f(x)g(x) = AC + x^{\frac{n+1}{2}}(E - AC - BD) + x^{n+1}BD$$

- 乘法只剩下三次了！

Karatsuba Algorithm

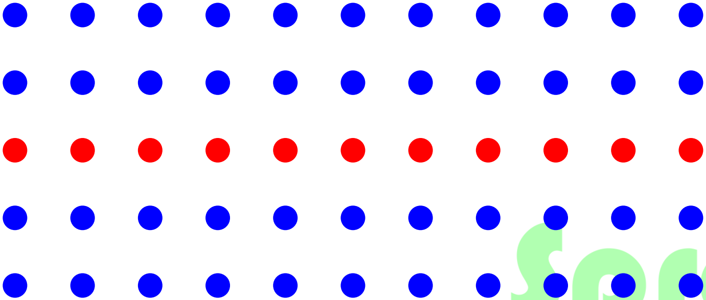
- 分析一下時間
- $T(n) = 3T(n/2) + O(n)$
- 根據 Master Theorem，$T(n) \in O(n^{\log_2 3}) \approx O(n^{1.58})$

尋找第 k 大

- 給你一個序列，找第 k 大
- 幹嘛不要 sort 就好
- 那樣就太遜了，我們要做到 $O(n)$！

尋找第 k 大

- 把序列五個五個分一組，並找到每組的**中位數**
  - 5 是個常數，所以找一組的中位數只要花 $O(1)$ 的時間！

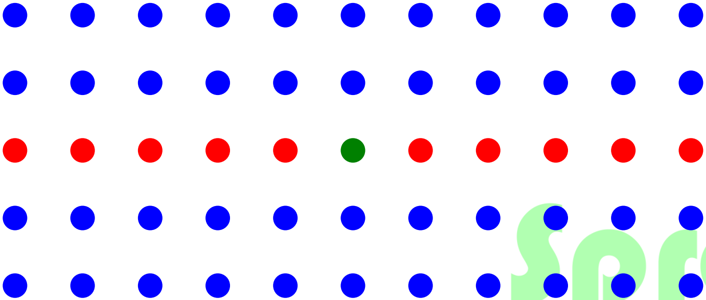


尋找第 k 大

- 把所有中位數蒐集起來，再找到「中位數的中位數」
  - 怎麼再找中位數？

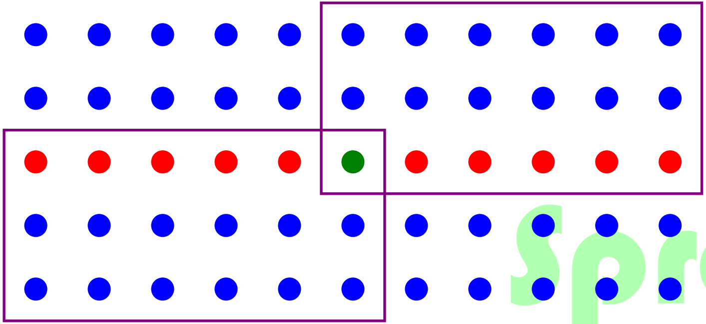
尋找第 k 大

- 把所有中位數蒐集起來，再找到「中位數的中位數」
  - 怎麼再找中位數？**對規模 $n/5$ 的問題呼叫尋找第 $n/10$ 大**
  - $T(n/5)$



尋找第 k 大

- 令剛剛找到的數字是 p，把數字分成 $> p$ 跟 $< p$ 兩堆
  - 用和 Quick sort 把數字分兩邊一樣的方式
- 注意到至少有 $3n/10$ 個數字比 p 小、至少有 $3n/10$ 個數字比 p 大



尋找第 k 大：Median of medians

- 看第 k 大在哪邊，往那邊遞迴就可以了！
- 這什麼神仙操作，複雜度長怎樣呢？

尋找第 k 大：Median of medians

- 看第 k 大在哪邊，往那邊遞迴就可以了！
- 這什麼神仙操作，複雜度長怎樣呢？
- 對規模 $n/5$ 的問題呼叫尋找第 $n/10$ 大：$T(n/5)$

尋找第 k 大：Median of medians

- 看第 k 大在哪邊，往那邊遞迴就可以了！
- 這什麼神仙操作，複雜度長怎樣呢？
- 對規模 $n/5$ 的問題呼叫尋找第 $n/10$ 大：$T(n/5)$
- 遞迴找 k 大：少掉至少 $3n/10$ 個數字，$T(7n/10)$

尋找第 k 大：Median of medians

- 看第 k 大在哪邊，往那邊遞迴就可以了！
- 這什麼神仙操作，複雜度長怎樣呢？
- 對規模 $n/5$ 的問題呼叫尋找第 $n/10$ 大：$T(n/5)$
- 遞迴找 k 大：少掉至少 $3n/10$ 個數字，$T(7n/10)$
- 分組：$O(n)$

尋找第 k 大：Median of medians

- 看第 k 大在哪邊，往那邊遞迴就可以了！
- 這什麼神仙操作，複雜度長怎樣呢？
- 對規模 $n/5$ 的問題呼叫尋找第 $n/10$ 大：$T(n/5)$
- 遞迴找 k 大：少掉至少 $3n/10$ 個數字，$T(7n/10)$
- 分組：$O(n)$
- $T(n) = T(n/5) + T(7n/10) + O(n)$

尋找第 k 大：Median of medians

- 看第 k 大在哪邊，往那邊遞迴就可以了！
- 這什麼神仙操作，複雜度長怎樣呢？
- 對規模 $n/5$ 的問題呼叫尋找第 $n/10$ 大：$T(n/5)$
- 遞迴找 k 大：少掉至少 $3n/10$ 個數字，$T(7n/10)$
- 分組：$O(n)$
- $T(n) = T(n/5) + T(7n/10) + O(n)$
- $T(n) \in O(n)$ by substitution method

更多神奇演算法

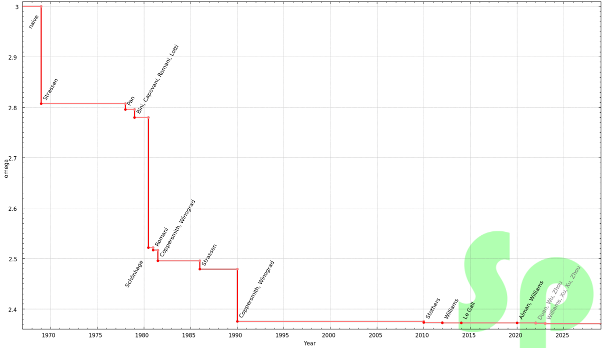
- 世界上充斥著更多不知道怎麼想到的神奇分治演算法

多項式乘法

- 暴力：$O(n^2)$
- Karatsuba：$O(n^{\log_2 3}) \approx O(n^{1.58})$
- FFT：$O(n \log n)$（也是分治！）

矩陣乘法

- 暴力：$O(n^3)$
- Strassen's algorithm：$O(n^{\log_2 7}) \approx O(n^{2.8074})$（也是分治！）



Quick sort 與 Merge sort

- 還記得影片裡有提到出題者可以很壞地讓 Quick sort 複雜度爛掉嗎？

Quick sort 與 Merge sort

- 還記得影片裡有提到出題者可以很壞地讓 Quick sort 複雜度爛掉嗎？
- 搭配一些小修改的話，Quick sort 的**期望**複雜度是 $O(n \log n)$

Quick sort 與 Merge sort

- 還記得影片裡有提到出題者可以很壞地讓 Quick sort 複雜度爛掉嗎？
- 搭配一些小修改的話，Quick sort 的**期望**複雜度是 $O(n \log n)$
- 這裡複雜度不要爛掉的重點是遞迴樹兩邊不要歪掉，就是切兩半的時候不要差太多倍

Quick sort 與 Merge sort

- 還記得影片裡有提到出題者可以很壞地讓 Quick sort 複雜度爛掉嗎？
- 搭配一些小修改的話，Quick sort 的**期望**複雜度是 $O(n \log n)$
- 這裡複雜度不要爛掉的重點是遞迴樹兩邊不要歪掉，就是切兩半的時候不要差太多倍
- 所以，如果用 median of medians 直接拿中位數當 pivot，那可以做到 deterministic 複雜度 $O(n \log n)$
- 但這樣就變 Slow sort 了 XD

Quick sort 與 Merge sort

- 還記得影片裡有提到出題者可以很壞地讓 Quick sort 複雜度爛掉嗎？
- 搭配一些小修改的話，Quick sort 的**期望**複雜度是 $O(n \log n)$
- 這裡複雜度不要爛掉的重點是遞迴樹兩邊不要歪掉，就是切兩半的時候不要差太多倍
- 所以，如果用 median of medians 直接拿中位數當 pivot，那可以做到 deterministic 複雜度 $O(n \log n)$
- 但這樣就變 Slow sort 了 XD
- 反過來說，找第 K 大也有基於隨機，期望複雜度是 $O(n)$ 的演算法

總結

分治

- 雖然我們最後講了一些很通靈的東西，影片裡也有一些看起來很通靈的東西
- 不過分治還是有一些基本策略可循
  - 只需要想怎麼「分」和怎麼「治」，不需要想子問題是怎麼做的
- 不知道是誰說：分治最難的是看出題目要分治

分治

- 我們看了很多在序列、平面的題目了
- 分治其實還能在更奇怪 (?) 的地方分治，例如樹、圖等等
- 往往分治還需要搭配很多噁心的資料結構、演算法
- 分治常常會在偏難的題目中走出一條通路

雜談

- 作為一個競賽選手，我覺得分治是一個在「學習競賽」和「學習演算法」兩種角度上會很不一樣的東西
- 事實上，我上大學後才慢慢比較領悟得到分治的思想
- 在競賽中，大部分的分治題目都可以用一些資料結構做掉
  - 經典的例子是逆序數對可以用 BIT aka Fenwick Tree 做掉
  - 即便很多資料結構往往蘊涵分治的思想，但感覺起來就只是在用工具而已
- 不過其實對分治有點概念對競賽選手是有幫助的
  - 有些難題真的得要會**設計**分治（而不只是直接使用常見的資料結構或演算法）
  - 更能活用基於分治的技術（e.g. 各式線段樹、整體二分、CDQ 分治，等等等）

一些題目

多項式乘法

Problem 2023 資芽一階認證考 pE 多項式乘法（NEOJ 858）

給 N 個多項式，求它們的乘積。輸出每項係數除以 998244353 的餘數。另外，有一個可以在 $O(n \log n)$ 的時間回傳共有 n 項的兩個多項式乘積的 function 已經寫好了，你可以直接使用它。

- $N \le 10^5$
- 多項式的總次數 $\le 10^5$

芽芽國的最短路徑問題

Problem 2022 資芽一階認證考 pE 芽芽國的最短路徑問題（NEOJ 846）

給你一張圖，滿足：

- $N = 1$
- 或滿足以下所有限制
  - 整張圖連通
  - 存在唯一一個「度數」最大的點，並稱其為「關鍵點」
  - 將「關鍵點」移除後，整張圖會剩下恰兩個**點數差不超過 2** 的連通塊，且這兩個連通塊各自滿足這坨條件

有 Q 筆詢問，每筆詢問求兩個給定點之間的最短路徑長。

- $N \leq 10^5$
- $M \leq 3 \times 10^5$
- $Q \leq 3 \times 10^5$

希爾伯特曲線

Problem TIOJ 1994 冰塊線

求 N 階希爾伯特曲線（Hilbert curve）。

- $N \leq 11$

昨天遇到的題目

Problem ECNA 2022 pE Hilbert's Hedge Maze (CF Gym 104614E)

有一個無限大的棋盤，有一道很長的牆壁的形狀是 n 階希爾伯特曲線，求某兩個指定格子之間的最短距離。

- ≤ 100 筆測資
- $n \leq 50$
- 座標範圍 $[-2^{52}, 2^{52}]$

很難的題目

Problem IOI 2006 Day2 pB Joining Points

在平面上的一個正方形範圍內，有 r 個紅色點和 g 個綠色點，保證左下角和右下角各有一個紅色點，左上角和右上角各有一個綠色點。連一些同色點之間的線段，使得綠色點連通、紅色點連通，且任兩條線段不相交。

- $g, r \leq 50000$
- 任三點不共線