基礎演算法 1

baluteshih

2019 年 9 月 25 日

1 何謂演算法？

解決問題的方法，便統稱演算法，這裡就不再贅述，可以自行上網搜尋他的意義 (?)。
千萬不要被「演算法」這三個字嚇著了，只要你有心去理解他的運作方式，人人都學的起演算法。

2 <algorithm>

<algorithm> 是一個提供多種常用演算法的函式庫，不過通常為了應付多種不同的資料結構和情況，<algorithm> 通常要求使用者傳入的會是指標或迭代器。以下 Iterator 以 Iter 簡稱：

2.1 排序

1. sort(Iter first,Iter last,Compare comp)：把 [first,last) 依照 comp 定義出的小於由小排到大，comp 的撰寫方式必須形如 bool comp(T a,T b)，若 a<b(這裡的小於可以讓你自己定義) 則回傳 true。如果沒傳入 comp，則會用預設的小於排序。保證複雜度 $O(Cn\log n)$，其中 C 為 comp 的複雜度，以下會略去其複雜度。

2. stable_sort(Iter first,Iter last,Compare comp)：同 sort，不過他會保證同樣的數字會依照原先序列的順序排序。若允許額外空間，複雜度 $O(n\log n)$，否則為保證複雜度 $O(n\log^2 n)$。

3. nth_element(Iter first,Iter nth,Iter last,Compare comp)：把 stable_sort 後在 nth 位置的元素 x 排到該位置，並且會讓該在 x 前面的元素全部出現在 x 前面，反之亦同。期望複雜度 $O(n)$，最壞有機率到 $O(n\log n)$。

2.2 針對已排序好的函式用的工具

1. lower_bound(Iter first,Iter last,T t,Compare comp)：等價 set 的 lower_bound。
2. upper_bound(Iter first,Iter last,T t,Compare comp)：等價 set 的 upper_bound。
3. equal_range(Iter first,Iter last,T t,Compare comp)：等價 multiset 的 equal_range。
4. merge(Iter1 first1,Iter1 last1,Iter2 first2,Iter2 last2,Iter3 res,Compare comp)：假設 [first1,last1) 和 [first2,last2) 是兩個經 cmp 排序好的序列，將兩個序列合併並排序，輸出至以 res 為首的序列並回傳指向序列末端的迭代器。要記得讓 res 後面有足夠的空間。複雜度與元素個數呈線性。
5. unique(Iter first,Iter last)：把 [first,last) 的重複元素刪除向前靠齊，並回傳指向序列末端的迭代器。複雜度 $O(n)$。

2.3 字典序

1. next_permutation(iter first,iter last,Cmp cmp)：將 [first,last) 變成原本的某個排序，使得字典序是比原本大的所有排序中最小的。如果成功了，這個函式會回傳 true；否則回傳 false，並將 [first,last) 變成字典序最小的那個排序。
2. prev_permutation(iter first,iter last,Cmp cmp)：將 [first,last) 變成原本的某個排序，使得字典序是比原本小的所有排序中最大的。如果成功了，這個函式會回傳 true；否則回傳 false，並將 [first,last) 變成字典序最大的那個排序。

2.4 其他好用的工具

1. swap(T &a,T &b),max(T a,T b),min(T a,T b)：內建的東東，用法應該差不了多少就不多說了 (?)。複雜度都是 $O(1)$。
2. fill(iter first,iter last,T a)：把 [first,last) 填滿成 a。複雜度 $O(n)$。
3. reverse(iter first,iter last)：把 [first,last) 反轉。複雜度 $O(n)$。
4. max_element(iter first,iter last,Cmp cmp)、min_element(iter first,iter last,Cmp cmp)：根據 comp 回傳 [first,last) 裡最大值或最小值所在位置的迭代器。複雜度 $O(n)$。
5. random_shuffle(iter first,iter last)：隨機打亂 [first,last) 的元素。複雜度 $O(n)$。

2.5 習題

1. (ZJ a233) 裸排序題。

2. (IOI 2000/TIOJ 1617) 本題為互動題，給你一個你看不見的序列，你可以在有限次數內詢問某三個位置的中位數在哪裡，請你回傳整個序列中位數的位置。(nth_element)
3. (TIOJ 1187) 裸最大最小值。
4. (ZJ a524) 裸 prev_permutation。

3 枚舉

只學函式庫當然不夠，我們總是要來些真槍實彈 (?)。
枚舉是演算法中最直觀也最純粹的作法，好比說找出一個序列的最大值，那就是枚舉每一個數字並不斷比較、淘汰後得出答案。
漂亮的枚舉可以適當的降低實作複雜度，甚至是在適當的搭配其他東西後啟發出一套更快的做法。

3.1 適當的枚舉

有時候可以不用真的把元素跑完，可以適當的在某個確定有答案的時機跳出。
有時候也可以透過 sort 把枚舉的次數減少，根據題目的性質也可以刪減枚舉的次數。
枚舉是活的，請不要把他寫死。

3.1.1 習題

1. (ZJ a010) 因數分解。
2. (ZJ b893) 給定你一個多項式，請求出他的所有根 (保證根為整數)。
3. (ZJ c268) 給你一堆數字，請你判斷存不存在三個數字能形成三角形的三邊長。

3.2 遞迴與剪枝

撰寫遞迴函式 solution() 用來解決當前問題，並在合理的情況下擺上可能的解後遞迴下去，最後設置檢查答案正確的基底。
這種算法往往會造成及龐大的時間複雜度，於是若發現已經沒有合理的擺放方法了卻還不是答案，就立刻回傳，這種做法便叫做「剪枝」。

3.2.1 習題

1. (TIOJ 1235) 求解數獨。
2. (ZJ a160) 八皇后問題。
3. (ZJ b510) 皇后 + 城堡的問題。

3.3 一些特定的窮舉

3.3.1 0/1 枚舉

利用數字的二進位來輔助決定哪些元素要用，進而枚舉到所有元素出現與否的目的。

Algorithm 1: 0/1 Enumeration

```
1 void Enumerate(int arr[],int n){
2     for(int x=(1<<n)-1;x>=0;--x)
3         for(int i=0,t=x;i<n;++i,t/=2)
4             if(t&1) //use arr[i]
5 }
```

3.3.2 字典序枚舉

利用 next_permutation 或 prev_permutation 達到枚舉排列種類的目的。

3.3.3 習題

由於這類做法通常是用來拿部分分數的，所以題目不多 (?)

1. (TIOJ 2060) 題敘略。
2. (ZJ c272) 對於任一種排列都有他的代價，請你在這些代價中回答 Q 次小於等於給定的 qs 的最大代價。

3.4 爬行法/Sliding Windows

常常我們會需要枚舉所有元素，再枚舉這個元素所能造成最符合答案的解。
此時如果能觀察到題目的單調性，就可以透過 sort 等方式，省去多餘的枚舉步驟，透過有順序地枚舉元素來省去後面元素枚舉的時間。

3.4.1 習題

這類做法通常會搭配其他算法一起出現，這裡就只附上裸題了，往後還請各位多加運用。

1. (ZJ b542) 給定一個長度為 N 的序列，請回答 Q 次是否存在任兩個數的差距為給定的 k。($O(NQ)$ 可過)。
2. (2018 TOI 初選 pD/TIOJ 2054) 給你一些點，請你找到一個 $l \times w$ 平行兩座標軸的矩形，使得被矩形包覆住的點最多。

3.5 折半枚舉

遇上複雜度大的算法，像是 $O(2^N)$ 這種算法，由於只要 N 差了一個倍數就可以使時間變化很多，所以我們可以把元素對半切開再去嘗試用其他方式組合起來，達到降低複雜度的目的。

3.5.1 習題

1. (POJ 3977) 有一個 N 個數的集合，找一個非空子集，使該子集中所有元素的和的絕對值最小，若有多個，則輸出個數最小的那個。
2. (背包問題，No judge) 給你 $N(N \leqq 40)$ 個物品，物品有各自的重量和價值，現在你有一個負重上限 W 的包包，請問在每種物品都只能選取一次的情況下，包包內的物品價值總和最大可以到多少？

3.6 更多習題

1. (ZJ c435) 找到兩個數字 $i, j(i < j)$ 使 $a[i] - a[j]$ 盡量大。
2. (atcoder IntegerotS) 給你一些帶權重和價值物品，一堆物品的權重是這些物品的權重做 OR 運算之後的值，試問在權重 $\leqq K$ 時，可以拿到的最大價值總和為多少？
3. (104 全國賽 pC/ZJ c424) 題敘略。
4. (ZJ c412) 試問在一個字串中，有多少個子序列為「OwO」?

4 搜尋

當枚舉問題存在著單調性，有時我們甚至可以直接用特殊的方法來大量淘汰掉不需要枚舉的元素，這就是搜尋的精神。

4.1 二分搜尋法

對於一個函數 $f(n)$，如果存在一個 x，滿足對於所有 $\geqq x$ 的值 a 都會使 $f(a)$ 回傳 $true$，反之則回傳 $false$ 的話，對於這樣的單調性，我們就能使用二分搜來找到這個 x。

Algorithm 2: Binary Search

```
int binary_search(int l,int r){
    while(l<r){
        int m=(l+r)/2;
        if(f(m)==true) r=m;
        else l=m+1;
    }
    return l;
}
```

其想法來源是，如果我們已經確保 x 所在的區間為 $[l,r]$，那麼只要戳中間的點 mid，驗證其是否為 true，如果是的話就可以把區間換成 $[l,mid]$，否則就換成 $(mid,r]$ 繼續做下去。
由於每次都至少能把區間砍成一半，所以複雜度會是 $O(k\log n)$，其中 k 是計算 $f(n)$ 的複雜度。

4.1.1 對答案二分搜

對於某些「至少要多少才會成立」或是「最多不超過多少」的題目，如果你發現你有辦法在良好的時間檢查出「如果代價是 x，那可不可以達成目標」，且發現這個 x 存在單調性的話，就可以把這個做法模擬成二分搜時用來檢查的函數來「對答案二分搜」。
這類題目存在著不少變形，大概要靠多寫題目來慢慢掌握。

4.1.2 習題

當題目不存在單調性的時候，你可以自己嘗試去製造出一個，像是排序之類的。

1. (ZJ d732) 給你一個嚴格遞增的數列，之後會問你 k 個問題，對於每個問題會告訴你一個 x，請回答該數列中是否存在一個值與 x 相等。
2. (ZJ b844) 給你一堆按鈕，每個按鈕旁邊的數字都顯示為 0，按下第 K 個按鈕可以把第 K 個以後的數字 1 變 0，0 變 1，讓你按下按鈕 N 次之後，有 Q 個詢問，問第 P 個數字為 0 還 1？

3. (ZJ b870) 數線上有有 N 棵樹，你能夠興建 K 座滅火器，試求樹和滅火器間距離最大值的最小可能。

4. (ZJ c250) 把序列中兩兩數字的差拿出來排序，試求第 K 大的差。

5. (TIOJ 1598) 假設有無限多個房客，前 n 個房客手上恰好各有一些鬆餅，而這些房客決定和其它房客共享鬆餅。在每一分鐘，每個房客可以吃掉一個鬆餅，或者分一些些鬆餅給另一個房客（不一定要是前 n 個房客）。規定房客們分鬆餅的總次數不能超過一個數 B，試問鬆餅最快可以在幾分鐘內吃完？

6. (IOI 2013/TIOJ 1839) 有 $n \leqq 5000$ 個開關，分別（不照順序地）連著 n 個門。對於每個開關，要嘛開的時候門會開，要嘛關的時候門會開，反之則反。你最多可以詢問 70000 次，對於每次詢問，你可以給一個 n 個開關的配置，程式會告訴你第一個關著的門是幾號門。請找出開關和門之間的對應關係，以及會讓所有門都開的開關配置。

7. (TIOJ 1926) 有一張 $N \times M$ 的不可見表格，你每次可以詢問任兩個點的大小關係，請你找到一個點使得他的上下左右都比他大。

4.2 三分搜尋法

當要搜尋的函數為 U 型函數 (ex: $y = x^2$)，也就是存在著某個點 x 滿足他的左右邊都各自嚴格遞增或遞減，那我們就可以用三分搜尋法來找到 x。
以下假設 $f(n)$ 的開口向上。

Algorithm 3: Trinary Search

```
1 double trinary_search(double l,double r){
2     static const double EPS=1e-7;
3     while(r-l>EPS){
4         double ml=(l+l+r)/3,mr=(l+r+r)/3;
5         if(f(mr)>f(ml)) r=mr;
6         else l=ml;
7     }
8     return l;
9 }
```

同樣地，如果確保 x 所在的區間為 $[l, r]$，那只要把區間三等分，得到兩個等分點 $m1 = \frac{l+l+r}{3}$ 跟 $m2 = \frac{l+r+r}{3}$。如果 $f(m1) \leqq f(m2)$，就把 r 更新成 $m2$，反之把 l 更新成 $m1$。

4.2.1 整數函數的三分搜

前面的三分搜是假設函數值域為實數的情況下實作的，那如果遇到整數的值域怎麼辦？
這時因為我們能很清楚地得知決策點可能性有限的關係，所以我們能戳中間連續的兩點找出函數的「斜率」，就可以轉化成二分搜問題。

Algorithm 4: Integer Trinary Search

```
int trinary_search(int l,int r){
    while(r-1>l){
        int m=(l+r)/2;
        if(f(m+1)>f(m)) r=m;
        else l=m;
    }
    if(f(l)<f(l+1)) return l;
    return r;
}
```

4.2.2 習題

1. (NEOJ 72) 給你一堆二次函數，函數 $f(x)$ 的值為這些二次函數代入 x 的最大值，試問 $f(x)$ 的最小值。
2. (Sky OJ 18) 同上題，但為一次函數，且值域為整數。
3. (TIOJ 1635) 給你一個先遞增後遞減的函數，你可以動態詢問任兩點的高度大小關係，問你最高點的位置。
4. (CF 939E) 請你支援以下兩種操作：1. 加入一個大於等於集合內所有數字的數字。2. 詢問對於所有的子集合 s，$max(s) - mean(s)$ 的最大值為多少，其中 $mean(s)$ 為子集的平均數。

5 貪心法 (Greedy)

「面對一個問題，不斷地使用同一個策略做事。」
貪心法可以很單純，也可以很懸，這裡提到所謂的「策略」也有可能會是很不直觀的做法。
常常某些題目會讓人很直覺的想使用貪心的策略，等到中途才發現自己完全走錯了路，所以通常我們在寫貪心的時候，會特別去注意這個做法是否是對的，可惜的是這是需要花時間練習的。

5.1 證明貪心的思路

如果你提出了一個貪心做法，卻又很不確定他是不是對的，你可以朝以下幾個方向去思考：

1. 試圖構造出反例，發現他不存在。
2. 如果存在更佳解的答案比你做出來的還好，那這組解一定可以再做得更好，進而達到反證出更佳解不存在。
3. 使用遞迴證法：1. 證明基底是對的。2. 假設小問題是好的。3. 你一定可以用最好的方法來將問題簡化成剛才假設是好的小問題。
4. ~~好麻煩喔，感覺寫起來也不慢就寫完丟上去看會不會錯吧。~~

剛開始看可能都會滿頭霧水不知道怎麼證明，所以可能都會用第四種做法 (?)，多練習題目，漸漸就會了解這些東西了。

5.2 一些嘗試貪心的例子

1. (0/1 背包問題) 每次拿將 CP 值（價值除以重量）最高的物品塞進包包。
2. (給你 n 個線段。請問最多能取出幾個線段，內部互不重疊。) 將線段依右端點排序，從最左邊的線段開始取。如果有重疊就不取，沒有重疊就取。
3. (給你一個面額，請你用最少的零錢湊出之) 如果錢幣面額滿足任意兩個面額都恰有一方整除另一方，則每次都選盡量大的面額扣掉再重複執行下去。
4. (給你一個面額，請你用最少的零錢湊出之) 如果錢幣面額沒有限制，則每次都選盡量大的面額扣掉再重複執行下去。

這四個例子最後的結果是 XOOX，讀者可以自己嘗試驗證看看。

5.3 題外話－－ Weighted Matroid

這東西似乎不是很實用，但他是一個貪心法相關的模塊。大致上就是在說，如果你有辦法把一個問題歸約到他身上，就一定存在一個固定且有效的貪心法可以得出最佳解。
有興趣的話可以自己上網搜尋，不過中文好像沒什麼資源就是。

5.4 習題

1. (TIOJ 1072) 有 N 個人各點了一道菜，每道菜各有需要的烹煮時間跟吃完時間，每個時間只能同時煮一道菜。訂定一個煮菜順序使得從第一道菜開始煮到最後一個人把菜吃完所過的時間的最小值。

2. (2009 TOI 入營考 pC/ZJ b231) 有 N 本書，每本書都有所需要的印刷時間和裝訂時間。你可以同時裝訂任意多本書，但同一時間只能印刷至多一本書。每本書需要先印完再裝訂。請問你至少要花多久才能將所有書都印刷裝訂完畢。

3. (TIOJ 1441) 請你適當地刪除一些數字，使得整個序列呈現大小大小... 小大小大的排列方式。

4. (ZJ d418) 給你一個大於等於 0 的整數 N，請你你找到最小的自然數 Q，使得在 Q 中所有數字位數的乘積等於 N。

5. (TIOJ 2009) 現在總共有 n 個介於 1 到 9 的正整數，每次調整密碼鎖上的數字時，必須一次讓其中連續的 k 個數字加上 1(9 會循環回 1)。試問最少要調整幾次，才能調整成另一個給定的序列，又或者是無解？

6. (APCS P4/ZJ c471) 有一堆物品堆成一疊，取出一個物品使用一次的代價是他上方的所有物品的重量和。給定物品的重量和預計取用次數，請找出一種排列使得代價和最小。