演算法入門

Fishhh

2023-09-09

目錄

前言

前言

- 相信大家在生活中都會或多或少的聽過” 演算法 (Algorithm)” 這個名詞

前言

- 相信大家在生活中都會或多或少的聽過” 演算法 (Algorithm)” 這個名詞
- 雖然這聽起來很難接近，但他白話文的解釋就是：” 解決問題的方法”

前言

- 相信大家在生活中都會或多或少的聽過” 演算法 (Algorithm)” 這個名詞
- 雖然這聽起來很難接近，但他白話文的解釋就是：” 解決問題的方法”
- 例如說，在算一道數學題時會使用固定的方法一樣，這也是一種演算法

前言

- 相信大家在生活中都會或多或少的聽過” 演算法 (Algorithm)” 這個名詞
- 雖然這聽起來很難接近，但他白話文的解釋就是：” 解決問題的方法”
- 例如說，在算一道數學題時會使用固定的方法一樣，這也是一種演算法
- 我們這堂課的內容不像是平常會聽到的類似 FB、IG、YT 的演算法

前言

- 相信大家在生活中都會或多或少的聽過” 演算法 (Algorithm)” 這個名詞
- 雖然這聽起來很難接近，但他白話文的解釋就是：” 解決問題的方法”
- 例如說，在算一道數學題時會使用固定的方法一樣，這也是一種演算法
- 我們這堂課的內容不像是平常會聽到的類似 FB、IG、YT 的演算法
- 就是一些對於競賽相關、較好理解的一些演算法。

前言

- 在計算機科學的方面來說，大家在追求的演算法需要符合以下兩點
  1. 需要的時間少（時間複雜度）
  2. 需要的空間小（空間複雜度）

尋找最大值

尋找最大值

題目敘述

給定一個長度為 n 的陣列，陣列內的每個數為 $a_i (1 \leq a_i \leq 10^9)$
請輸出陣列中最大的數。

先讓大家想一下吧　如果有想法的就可以直接開始實作了！

實作方法

- 先假設一個變數 x 並且把它設為一個極小的值
- 從頭到尾走過一次陣列，如果目前遇到的數字大於 x 那我們就把他撿起來，當作目前為止的最大值。這樣全部走過一次後 x 就會是我們要找的最大值了！

参考程式

```
int main(){
    int n,ary[200];
    FOR1(i,n){
        GET(ary[i]);
    }
    int x = ary[1]; // x = -999999 也行
    FOR1(i,n){
        if(ary[i]>x){
            x = ary[i];
        }
    }
    PUT(x);
    new_line;
}
```

實作時間！

尋找最小值

邪惡的 Eason 偷偷把題目改成尋找最小值了 ><

排序演算法

排序 Sorting

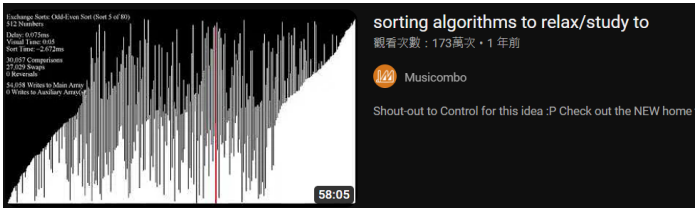
- 何謂排序?
- 簡單來說，就是讓一堆資料按照你想要的順序排好

排序 Sorting

- 何謂排序?
- 簡單來說，就是讓一堆資料按照你想要的順序排好
- 例如：把數字由大往小排序、將書本按照編號排序、把大家的名字按照筆畫排序等等 ...

排序 Sorting

- 排序演算法有千百種，現在 YT 上的那種排序影片就可以做到一個小時（x



排序 Sorting

今天主要會介紹幾個排序的演算法

1. Bubble Sort 泡泡排序
2. Merge Sort 合併排序

Bubble Sort

- 先來看看範例影片，稍微觀察一下 Bubble Sort 在幹嘛吧
- Bubble Sort

Bubble Sort

- 如果我們要由小排到大的話，Bubble Sort 就是一直把最大的拿到最右邊
- 再來拿第二大、第三大 ... 直到全部都走過，也就是成功把陣列由小排到大的時候了！

参考程式

```
int main(){
    int n,ary[200];
    FOR1(i,n){
        GET(ary[i]);
    }
    FOR1(i,n){
        int x = 0,id = 0;
        FOR1(j,n){
            if(ary[j]>x){
                id = i;
                x = ary[j];
            }
        }
        ans[i] = x;
        ary[id] = -1;
    }
    FOR1(i,n){
        PUT(ans[i]);
    }
    new_line;
}
```

實作時間！

杰哥的排序

邪惡的 Eason 偷偷又把題目改掉了 ><

Merge Sort

- 先來看看影片吧，觀察一下 Merge Sort 執行起來會長怎樣

Merge Sort

- 先來看看影片吧，觀察一下 Merge Sort 執行起來會長怎樣
- 最後那一幕，就是在把兩段已經排序好的數字合併成一個完整的已經排序好的數列

Merge Sort

- 先來看看影片吧，觀察一下 Merge Sort 執行起來會長怎樣
- 最後那一幕，就是在把兩段已經排序好的數字合併成一個完整的已經排序好的數列
- 接下來我們可以討論一件事情，就是我們要怎麼把兩段已經排序好的數列合併起來

Merge Sort

- 讓我用畫圖的解釋一下吧
- diagrams

Merge Sort

- 我們已經學完 Merge Sort 裡面的 Merge 了
- 說到 Merge(合併) 那就一定會有 Split(拆開) 呀
- 那我們來介紹一下 Merge Sort 裡面的 Split 吧
- diagrams

Merge 實作

```cpp
int main(){
    int n,ary[200]={},ary2[200]={},ans[400]={};
    GET(n);
    FOR1(i,n){
        GET(ary[i]);
    }
    FOR1(i,n){
        GET(ary2[i]);
    }
    int t_ary = 1,t_ary2 = 1;
    FOR1(i,2*n){
        if(t_ary==n+1){
            ans[i] = ary2[t_ary2];
            t_ary2+=1;
        }
        else if(t_ary2==n+1){
            ans[i] = ary[t_ary];
            t_ary+=1;
        }
        else if(ary[t_ary]>ary2[t_ary2]){
            ans[i] = ary2[t_ary2];
            t_ary2+=1;
        }
        else{
            ans[i] = ary[t_ary];
            t_ary+=1;
        }
    }
    FOR1(i,2*n){
        PUT(ans[i]);
    }
    new_line;
}
```

實作時間！

合併大師

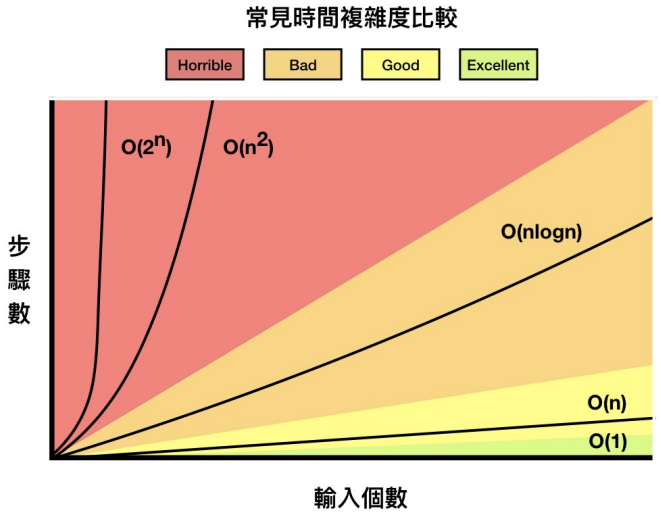
題目被改成兩個長度不同的數列了！
想辦法解決吧 ww

時間複雑度

時間複雜度


不要跟我談什麼
演算法複雜度分析！
Quick Sort？
Heap Sort？
Merge Sort？
老夫就問哪個排序法
能讓我早點下班！？

時間複雑度


常見時間複雑度比較
Horrible
Bad
Good
Excellent
O(2^n)
O(n^2)
O(nlogn)
O(n)
O(1)
步驟數
輸入個數

時間複雜度

- 講了那麼多，總而言之時間複雜度就是在判斷你的程式執行速度的好壞
- 也就等價於是演算法的好壞，或是說處裡方式的好壞
- 為什麼評估這個很重要呢?
- 這樣才可以單純看程式碼，就知道程式是好的還是壞的

時間複雜度

- 這堂課我們只會教最最最簡單的判斷方式
- 實際上打競賽還需要多分析均攤以及 `log` 的狀況，但對於剛入門的而言是不太需要判斷這個

時間複雜度

- 那 . . . 我們要怎麼判斷呢

時間複雜度

- 那 . . . 我們要怎麼判斷呢
- 很簡單，估計迴圈的執行次數然後取最高位就好了

時間複雜度

- 那 ... 我們要怎麼判斷呢
- 很簡單，估計迴圈的執行次數然後取最高位就好了
- 聽起來是不是很抽象 XD
- 那我們來帶一些範例吧

時間複雜度

請判斷下列程式片段的複雜度

```
1  FOR(i,n){
2          FOR(j,n){
3              cout<<i;
4          }
5      }
```

```
1  FOR(i,n*n){
2          FOR(j,n){
3              cout<<i;
4          }
5      }
```

```
1  FOR(i,n){
2      cout<<i<<" ";
3  }
```

時間複雑度

```
1  FOR(i,n){
2      cout<<i<<" ";
3  }
4  FOR(i,n){
5      FOR(j,m){
6          cout<<"!";
7      }
8  }
```

時間複雜度

- 恭喜你學會時間複雜度最基礎的判斷方式了！
- 接下來介紹一下 Sorting 的時間複雜度
- Sorting：目前最快是 $O(n \log n)$，如果你想到更快的請趕快來聯絡我 ww
- 一般的電腦大概一秒鐘可以跑 $10^8 \sim 10^9$ 的運算，如果 n 帶進去後發現會超過，那就代表這個程式是會得到 TLE 的！

前綴和

前綴和

先來看題目:D

- 題目

前綴和

```cpp
int ary[200010];
int main(){
    int n,q;
    GET(n);
    GET(q);
    FOR1(i,n){
        GET(ary[i]);
    }
    int l,r;
    FOR(i,q){
        GET(l);
        GET(r);
        int sum = 0;
        FORS(i,l,r){
            sum+=ary[i];
        }
        PUT(sum);
        new_line;
    }
}
```

219821257	Aug/22/2023 00:50 UTC+8	Fishhh0710	D - 一維區間和問題 (1D-Prefix)	GNU C++20 (64)	Time limit exceeded on test 3	1000 ms	800 KB

前綴和

- TLE 了 QQ
- 我們來分析一下複雜度吧！

前綴和

- TLE 了 QQ
- 我們來分析一下複雜度吧!
- 結果竟然是 $O(n^2)$，$10^{5\times 2} = 10^{10}$ 一定會 TLE 的

前綴和

- TLE 了 QQ
- 我們來分析一下複雜度吧!
- 結果竟然是 $O(n^2)$，$10^{5\times 2} = 10^{10}$ 一定會 TLE 的
- 所以我們就來優化它吧!

前綴和

- 國中可能有算過這樣的數學題：你知道 $a+b+c+d+e$ 以及 $a+b+c+d+e+f+g+h$ 的數值，想要求 $f+g+h$ 是多少

前綴和

- 國中可能有算過這樣的數學題：你知道 $a+b+c+d+e$ 以及 $a+b+c+d+e+f+g+h$ 的數值，想要求 $f+g+h$ 是多少
- 無獎徵答，有人要來回答一下這題怎麼做嗎?

前綴和

- 國中可能有算過這樣的數學題：你知道 $a+b+c+d+e$ 以及 $a+b+c+d+e+f+g+h$ 的數值，想要求 $f+g+h$ 是多少
- 無獎徵答，有人要來回答一下這題怎麼做嗎?
- 只要把後面的那個數值減去前面的，就可以得到我們要的 $f+g+h$ 了
- 那如果題目改成：你知道 $a_1+a_2+a_3+a_4$ 以及 $a_1+a_2+a_3+a_4+a_5+a_6$ 求 a_5+a_6

前綴和

- 國中可能有算過這樣的數學題：你知道 $a+b+c+d+e$ 以及 $a+b+c+d+e+f+g+h$ 的數值，想要求 $f+g+h$ 是多少
- 無獎徵答，有人要來回答一下這題怎麼做嗎?
- 只要把後面的那個數值減去前面的，就可以得到我們要的 $f+g+h$ 了
- 那如果題目改成：你知道 $a_1+a_2+a_3+a_4$ 以及 $a_1+a_2+a_3+a_4+a_5+a_6$ 求 a_5+a_6
- 這不是一模一樣嗎 ww

前綴和

- 國中可能有算過這樣的數學題：你知道 $a+b+c+d+e$ 以及 $a+b+c+d+e+f+g+h$ 的數值，想要求 $f+g+h$ 是多少
- 無獎徵答，有人要來回答一下這題怎麼做嗎?
- 只要把後面的那個數值減去前面的，就可以得到我們要的 $f+g+h$ 了
- 那如果題目改成：你知道 $a_1+a_2+a_3+a_4$ 以及 $a_1+a_2+a_3+a_4+a_5+a_6$ 求 a_5+a_6
- 這不是一模一樣嗎 ww
- 看完上面那一條等式後，可以發現如果我們有 $a_1+\ldots+a_{l-1}$ 以及 $a_1+\ldots+a_r$ 的和，那我們就可以用一次的減法得到 $l\sim r$ 的總和了!

前綴和

- 按照前面的思想，我們就得知一件事情，我們只要手上同時握有 $a_1, a_1 + a_2, ..., a_1 + ... + a_n$ 這些數字
- 我們就可以透過一次的減法得到我們要的區間和（也就是 $O(1)$）
- 所以我們可以來思考一下這些數字該怎麼得到

前綴和實作

```
1  FOR1(i,n){
2      GET(ary[i]);
3  }
4  FOR1(i,n){
5      FOR1(j,i){
6          pre[i]+=ary[j];
7      }
8  }
```

前綴和實作

- 看來我們需要一個更快的作法

前綴和實作

```cpp
int ary[200010],pre[200010]={};
int main(){
    int n,q;
    GET(n);
    GET(q);
    FOR1(i,n){
        GET(ary[i]);
    }
    pre[0] = 0;
    FOR1(i,n){
        pre[i] = pre[i-1]+ary[i];
    }
}
```

前綴和

- 經過上面的程式，我們已經可以得到所謂的” 前綴和” 陣列
- 接下來就是要處裡詢問的部分了
- 結論已經在前面講過了，如果詢問的是 $[l, r]$ 那麼就輸出 $pre[r] - pre[l - 1]$ 即可

練習時間

- 接下來就先把第一題完成吧！
- 這次沒有改題目了

練習時間?!

序列相似度（Simility）

這題被大改了，(其實是我自己在 SCIST S3 出的期中考題目 ww 然後又被我稍微改了一下
想想看這題與前綴和的關聯吧

二維前綴和

二維前綴和

- 二維？ 聽 .. 聽起來好難

二維前綴和

- 二維？ 聽 .. 聽起來好難
- 為了方便理解，接下來我都用圖解的吧！

二維前綴和

- 二維？ 聽 .. 聽起來好難
- 為了方便理解，接下來我都用圖解的吧！
- 這個部分不會有實作，就單純講講觀念 w

二維前綴和

- diagrams

動態規劃 DP

動態規劃

- 這個主題在競程的道路上算是最難上手的一個
- 新手常常會在所謂的 DP 上碰壁
- 這堂課就帶大家好好的認識所謂的”DP” 吧!

動態規劃

- 先從最好上手的” 爬樓梯問題” 開始吧

動態規劃

- 先從最好上手的” 爬樓梯問題” 開始吧
- 題目是這樣的，給你一個長度為 n 的樓梯，每次可以往上爬 **1** 階，請問有幾種方式可以爬到第 n 階

動態規劃

- 先從最好上手的” 爬樓梯問題” 開始吧
- 題目是這樣的，給你一個長度為 n 的樓梯，每次可以往上爬 1 階，請問有幾種方式可以爬到第 n 階
- 答案十分明顯，就是一階
- 如果改成一次可以向上爬 1 階或是 2 階呢?
- 先給大家一個數學題，一共有 5 階，每次可以向上爬 1 階或是 2 階，請問有幾種方式?

動態規劃

- 先從最好上手的” 爬樓梯問題” 開始吧
- 題目是這樣的，給你一個長度為 n 的樓梯，每次可以往上爬 1 階，請問有幾種方式可以爬到第 n 階
- 答案十分明顯，就是一階
- 如果改成一次可以向上爬 1 階或是 2 階呢?
- 先給大家一個數學題，一共有 5 階，每次可以向上爬 1 階或是 2 階，請問有幾種方式?
- 開始列舉吧!
- diagrams

動態規劃

- 這跟動態規劃有什麼關係?
- 而且” 動態”,” 規劃” 這兩個詞彙要怎麼連結在一起的呀?
- 其實從中文的角度來說” 動態” 這個詞彙比較貼近 DP 在做的事情
- 等我講完爬樓梯問題怎麼用程式處裡，再回來說說吧!

爬樓梯問題

- 先知道一件事情：如果今天要爬到第 1 階，那麼唯一的爬法就是從 0 爬一階到 1，所以方法數是 1
- 如果是要到第 2 階呢? 這時候就有兩種方法了，就是從第 0 階一次爬兩階到 2 或是先到 1 再到 2
- 可以觀察到一件事情，如果要爬第 i 階，就只能從 $i-1$ 階或是 $i-2$ 階爬上來。
- 那麼爬到第 i 階的方法數就會是，爬到 $i-1$ 階的方法數加上 $i-2$ 的方法數!

爬樓梯問題

- 從前面的過程，很明顯爬到第 1 階的方法數是 1，第二階則是 2
- 如果要求第三階呢?

爬樓梯問題

- 從前面的過程，很明顯爬到第 1 階的方法數是 1，第二階則是 2
- 如果要求第三階呢?
- 顯而易見，就是 $1+2$
- 繼續推下去，第四、第五階的方法數就呼之欲出了

爬樓梯問題

```
1  int main(){
2      int n;
3      GET(n);
4      int dp[20]={};
5      dp[1] = 1;
6      dp[2] = 2;
7      FORS(i,3,n){
8          dp[i] = dp[i-1]+dp[i-2];
9      }
10     PUT(dp[n]);
11 }
```

回到 DP

- 我們來討論 **DP** 的一些名詞吧
- **Base Case**：就是在電腦開始計算之前需要的資料，像是爬樓梯問題需要知道 $\mathrm{dp}[1], \mathrm{dp}[2]$
- **狀態**：因為我們設了一個叫做 **dp** 的陣列，要賦予每個位置一個意義，例如說：dp_i 就是爬到第 i 階的方法數
- **轉移**：透過轉移這個操作，我們可以透過一些運算推知其他東西，例如：$\mathrm{dp}_i = \mathrm{dp}_{i-1} + \mathrm{dp}_{i-2}$ 這就是爬樓梯問題的轉移式

回到 DP

- 所以，DP 就是一連串的狀態進行轉移，最後得到答案！
- 在遇到更難的問題時，往往要想的是去觀察出狀態，思考如何轉移
- 而”觀察”這件事常常被說成是在通靈啦 w

練習時間

DP 大師的考驗

題目又被改了 QQ