時間複雜度

sam571128

September 9, 2021

你是否常常遇到這種情況?

1980	6060	Time Limit Exceeded	0
1980	8760	Time Limit Exceeded	0
1984	9048	Time Limit Exceeded	0
1976	3952	Time Limit Exceeded	0
1984	12512	Time Limit Exceeded	0
1976	4420	Time Limit Exceeded	0
1980	12952	Time Limit Exceeded	0
1984	10576	Time Limit Exceeded	0
1980	6940	Time Limit Exceeded	0
1984	15504	Time Limit Exceeded	0
1972	10144	Time Limit Exceeded	0
1968	10248	Time Limit Exceeded	0

你是否常常遇到這種情況?

Lang	Verdict	Time	Memory
GNU C++17	Time limit exceeded on test 2	2000 ms	0 KB
GNU C++17	Time limit exceeded on test 2	2000 ms	0 KB
GNU C++14	Time limit exceeded on test 2	2000 ms	0 KB

演算法的效率?

電腦看似萬能，但一秒能運算的次數還是有限制的

因此分析一個演算法的會運行的時間就是很重要的一件事

而我們今天要來講的就是演算法的時間分析

時間複雜度 (Time Complexity)

在分析演算法的運行時間時，我們一般會引入三個函數

大 O 符號 (Big O Notation)

大 Ω 符號 (Big Ω Notation)

大 Θ 符號 (Big Θ Notation)

而這些函數一般都會被我們拿來分析一個函數的「時間複雜度」

但我們接下來的課程都會以 Big O 為主

Big O

大 O 符號

- 最常拿來分析時間複雜度的函數
- 表達演算法複雜度量級的**上界**
- 寫出函數最大項、並省略常數

Big O

範例:

$$O(n^2 + 9000n + 300) = O(n^2)$$

$$O(2000n \log n + n + 1) = O(n \log n)$$

$$O(500n + 1000) = O(n)$$

$$O(100) = O(1)$$

Big O

而基本上大 O 符號有個嚴格定義，寫成以下

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = k \Rightarrow f(n) \in O(g(n))$$

不過在競賽中，只需要會判斷一個程式的複雜度即可

演算法的執行時間?

現在正常的電腦一秒鐘可以跑 10^8 到 10^9 次運算

而根據我們上面所提的大 O 符號，將 n 代入函數

我們可以得到程式大約的執行時間 (不過常數影響也滿大)

如: $n = 10^4 \Rightarrow O(n^2)$ 大約跑一秒

常用的時間複雜度

常數時間: $O(1)$

- 加減乘除餘 (+,-,*,/,%)
- 等於、大於、小於 ($==$, $>$, $<$)
- 賦值 ($=$)

常用的時間複雜度

對數時間: $O(\log n)$

- 二分搜尋 (Binary Search)
- 一些 STL 的 find(), count()
- Heap 的插入與查詢

(因為這裡有九年級的，提一下 log 這個符號)

(對數 $\log$ 的定義是 $a^b = c \Rightarrow \log_a c = b$)

常用的時間複雜度

線性時間: $O(n)$

- 線性搜尋
- memset(), fill(), iota()
- min_element(), max_element()

常用的時間複雜度

$O(n \log n)$

- 多數排序演算法 (Merge Sort, Quick Sort, std::sort())
- 調和級數的上界 $(\frac{n}{1} + \frac{n}{2} + \cdots)$
- 多數分治演算法 (之後會提一個可以分析遞迴的定理)

常用的時間複雜度

$O(n^2)$

- 插入排序
- 氣泡排序

實際判斷看看

```cpp
int sum = 0;
for(int i = 1;i <= n;i++){
    sum += i;
}
cout << sum << "\n";
```

實際判斷看看

```cpp
int sum = 0;
for(int i = 1;i <= n;i++){
    sum += i;
}
cout << sum << "\n";
```

複雜度: $O(n)$

實際判斷看看

```
for(int i = 1;i <= n;i++){
    for(int j = 1;j < n;j++){
        if(arr[j] > arr[j+1]){
            swap(arr[j],arr[j+1]);
        }
    }
}
```

實際判斷看看

```
for(int i = 1;i <= n;i++){
    for(int j = 1;j < n;j++){
        if(arr[j] > arr[j+1]){
            swap(arr[j],arr[j+1]);
        }
    }
}
```

複雜度: $O(n^2)$

實際判斷看看

```
int f(int x){
    if(x==0) return 1;
    return f(x-1)+f(x-1);
}
```

實際判斷看看

```
int f(int x){
    if(x==0) return 1;
    return f(x-1)+f(x-1);
}
```

複雜度: $O(1+2+4+\cdots+2^{n+1}) \Rightarrow O(2^n)$

實際判斷看看

```
for(int i = 2;i <= n;i++){
    for(int j = 1; i * j <= n;j++){
        prime[j] = false;
    }
}
```

實際判斷看看

```
for(int i = 2;i <= n;i++){
    for(int j = 1; i * j <= n;j++){
        prime[j] = false;
    }
}
```

複雜度: $O(\frac{n}{2} + \cdots + \frac{n}{n}) \leq O(n \int_1^n \frac{1}{x} dx) \approx O(n \log n)$

均攤複雜度

有時候，我們會遇到演算法的複雜度看似很差

但實際將全部的可能性都考慮之後

能夠有一個夠快的時間複雜度

這種我們就稱其為「均攤複雜度」

均攤複雜度

2021 YTP 高中組初賽 p8

有 n 個字串 s_i，以及 q 個詢問，每次詢問兩個字串的最長共同前綴

測資範圍: $1 \leq n, q \leq 5 \times 10^5$ ，總字串長度 $\sum_{i=1}^{n} |s_i| \leq 5 \times 10^5$

備註: 如 abcd 與 abce 的最長共同前綴為 abc

均攤複雜度

2021 YTP 高中組初賽 p8

有 n 個字串 s_i，以及 q 個詢問，每次詢問兩個字串的最長共同前綴

測資範圍: $1 \leq n, q \leq 5 \times 10^5$ ，總字串長度 $\sum_{i=1}^{n} |s_i| \leq 5 \times 10^5$

備註: 如 abcd 與 abce 的最長共同前綴為 abc

這種題目，其實有很多進階的做法可以解決

但是對於還沒學過進階的演算法的大家

這題有辦法做嗎?

均攤複雜度

2021 YTP 高中組初賽 p8

有 n 個字串 s_i，以及 q 個詢問，每次詢問兩個字串的最長共同前綴

測資範圍: $1 \leq n, q \leq 5 \times 10^5$ ，總字串長度 $\sum_{i=1}^{n} |s_i| \leq 5 \times 10^5$

備註: 如 abcd 與 abce 的最長共同前綴為 abc

你一定會想，如果我們暴力做呢?

均攤複雜度

2021 YTP 高中組初賽 p8

有 n 個字串 s_i，以及 q 個詢問，每次詢問兩個字串的最長共同前綴

測資範圍: $1 \leq n, q \leq 5 \times 10^5$ ，總字串長度 $\sum_{i=1}^{n} |s_i| \leq 5 \times 10^5$

備註: 如 abcd 與 abce 的最長共同前綴為 abc

對於 q 次詢問，我們都去暴力找兩個字串最長的前綴

均攤複雜度

2021 YTP 高中組初賽 p8

有 n 個字串 s_i，以及 q 個詢問，每次詢問兩個字串的最長共同前綴

測資範圍: $1 \leq n, q \leq 5 \times 10^5$ ，總字串長度 $\sum_{i=1}^{n} |s_i| \leq 5 \times 10^5$

備註: 如 abcd 與 abce 的最長共同前綴為 abc

對於 q 次詢問，我們都去暴力找兩個字串最長的前綴

時間複雜度: $O(qn)$ **TLE!**

均攤複雜度

2021 YTP 高中組初賽 p8

有 n 個字串 s_i，以及 q 個詢問，每次詢問兩個字串的最長共同前綴

測資範圍: $1 \leq n, q \leq 5 \times 10^5$ ，總字串長度 $\sum_{i=1}^{n} |s_i| \leq 5 \times 10^5$

備註: 如 abcd 與 abce 的最長共同前綴為 abc

要是我們將已經找過答案的兩個字串記錄下來

下次遇到這兩個字串的時候，我們就可以 $O(1)$ 回答答案了

均攤複雜度

2021 YTP 高中組初賽 p8

有 n 個字串 s_i，以及 q 個詢問，每次詢問兩個字串的最長共同前綴

測資範圍: $1 \leq n, q \leq 5 \times 10^5$ ，總字串長度 $\sum_{i=1}^{n} |s_i| \leq 5 \times 10^5$

備註: 如 abcd 與 abce 的最長共同前綴為 abc

仔細想想看，如果我們這樣去做，最差的時間真的會是 $O(qn)$ 嗎?

均攤複雜度

2021 YTP 高中組初賽 p8

有 n 個字串 s_i，以及 q 個詢問，每次詢問兩個字串的最長共同前綴

測資範圍: $1 \leq n, q \leq 5 \times 10^5$ ，總字串長度 $\sum_{i=1}^{n} |s_i| \leq 5 \times 10^5$

備註: 如 abcd 與 abce 的最長共同前綴為 abc

其實不盡然，這樣的時間複雜度最差只可能是 $O(q\sqrt{n})$

(當有 $\sqrt{n}$ 個長度為 $\sqrt{n}$ 的字串時)

而這題就可以用暴力解決了!

P, NP?

在計算機理論中，時間複雜度又被區分為了 P 和 NP

而這兩個東西都有他們各自的定義，不過現階段先有以下想法即可

P 是能夠在**多項式時間**解決的問題

NP 是**無法在多項式時間**解決的問題

(以上內容是錯誤的，有興趣可以上網查，或去資訊之芽等)

之後可能在講一些演算法時會提到這兩個詞

P, NP?

多數問題都是 NP 問題

科學家們還在尋找 NP 問題是否存在多項式時間的解

而「P=NP」是否為真也是目前我們所不知道的

總結

今天大家學會了分析自己演算法的運行時間

今後遇到演算法等等都會需要自行分析

在 CF 的 Group 裡，我放了一些可以去寫寫看的題目

這些都不用用到深入的演算法，所以可以去練習看看