Tarjan

進階教學組

August 14, 2022

目錄

1 序言

DP 是藉由紀錄小問題和把大問題拆成需多的小問題來節省時間的一個演算法，作用十分廣泛，基本上 Greedy 可以解的題目 DP 都可以解，但 DP 可以解的題目 Greedy 不一定可以解，且 DP 可以把一些窮舉需要 $O(2^N)$ 的問題縮短到 $O(N^K)$ 其中 $K \in$ 常數

2 費氏數列

費氏數列想必大家都曾經聽聞了

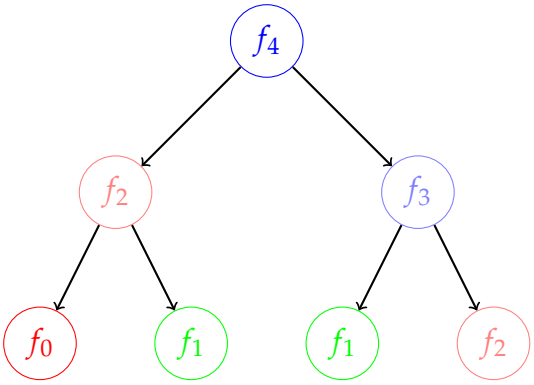
$$\begin{cases} f_0 = 1 \\ f_1 = 1 \\ f_n = f_{n-1} + fn - 2 \qquad n \geq 2 \end{cases}$$

接下來會介紹幾種解費氏數列的方式。

2.1 遞迴

```
func f( int n ) {
    if( n == 0 or n == 1)
        return 1
    else
        return f(n-1)+f(n-2);

}
```



可以發現，我們重複計算了很多次一樣的值，複雜度直接爆炸。
時間複雜度 $O(\phi^N)$

2.2 bottom-down

其實只要多開個陣列來記錄就可以達到很快的效能了

```
int dp[N]
dp[0] <- 1
dp[1] <- 1

func f( int N ) {
    if( dp[N] != 0 )
        return dp[ N ];
    else
        dp[ N ] = f(N-1) + f(N-2);
        return dp[ N ];
}
```

可以發現，這樣子每一個 f_i 只會被計算到一次，然後最多只會有 N 個格子，所以複雜度會是 $O(N)$

2.3 bottom-up

雖然遞迴的複雜度是 $O(N)$，但他的常數實在有點大，所以我們為什麼不嘗試看看直接用 *for* 寫呢? 根據遞迴公式，我們便可以直接使用 *for* 迴圈來寫這一題。

$$\begin{cases} f_0 = 1 \\ f_1 = 1 \\ f_n = f_{n-1} + fn - 2 \qquad n \geq 2 \end{cases}$$

```
int dp[ N ]
dp[ 0 ] <- 1
dp[ 1 ] <- 1

for i in [2, N]
    dp[ i ] = dp[ i-1 ] + dp[ i-2 ]
```

這樣子的時間複雜度也會是 $O(N)$ 不過常數小了許多。