

# **Programming Assignment 3**

Job scheduling

## 一、解法與實作過程

使用貪婪演算法來解決此問題。首先，將所有工作依照截止期限排好，接著將排序過的工作依序加入排程中。每加入一項工作，即檢查排程是否仍為可行序列(所有工作都能在截止期限前完成的)。若是，則繼續加入下一個元素；若非，將排程中工作時間最長的工作刪除，維持其為可行序列。加入所有元素並完成檢查後，排程即為能有最大利潤(完成最多項工作)的最佳序列。

因為不斷遇到尋找最大、最小值的過程，所以使用 Priority Queue 資料結構來完成演算法。

另外，特別為排程設計了一個 schedule 類別，其資料成員包含整數 total\_time (總工作時間)，和一個 time 越大，優先次序則越高的優先佇列 J(可行集合)。

借助 Priority Queue 來完成排序。一開始先將輸入資料推入一個 deadline 越大，優先次序則越低的優先佇列 S 中。接著將 S 中的元素 pop 取出。deadline 最小的工作會越先拿出來。

宣告一排程 schedule MaxProfit。呼叫 schedule 類別的成員函式 add\_job，將拿出來的元素加至排程中。add\_job 函式內容包含將新元素推入 J 中；total\_time+= 新元素的工作時間；以及檢查是否仍為可行序列。

### ***Lemma 16.12***

For any set of tasks  $A$ , the following statements are equivalent.

1. The set  $A$  is independent.
2. For  $t = 0, 1, 2, \dots, n$ , we have  $N_t(A) \leq t$ .
3. If the tasks in  $A$  are scheduled in order of monotonically increasing deadlines, then no task is late.

根據定理，J 中的元素集合為可行集合，若且為若 J 中的工作依截止期限大小的非遞減順序排列之序列必為可行序列。所以現在只需檢查 total\_time 會不會超過最新推入的元素之 deadline 即可。如果超過了，則將 J 中工作時間最長的元素移除(pop 掉最優先元素)，並更新 total\_time。

最後，S 中的元素全部被取出。J 中的元素個數就是排程可以獲得的最大利潤。

## 二、虛擬碼

將所有工作按照截止期限排成序列 S

J=空集合;

total\_time=0;

while( S 不為空){

    將所選的元素加入 J 中;

    total\_time +=所選元素的工作時間

    if(total\_time>所選元素的截止期限){

        total\_time -= J 中最優先元素的工作時間;

        J.pop();

    }

    S.pop();

}

## 三、複雜度分析

將輸入資料推入 S 和拿出來-> 花費  $O(n \lg n)$

將所選的元素加入 J 中-> 花費  $O(\lg n)$

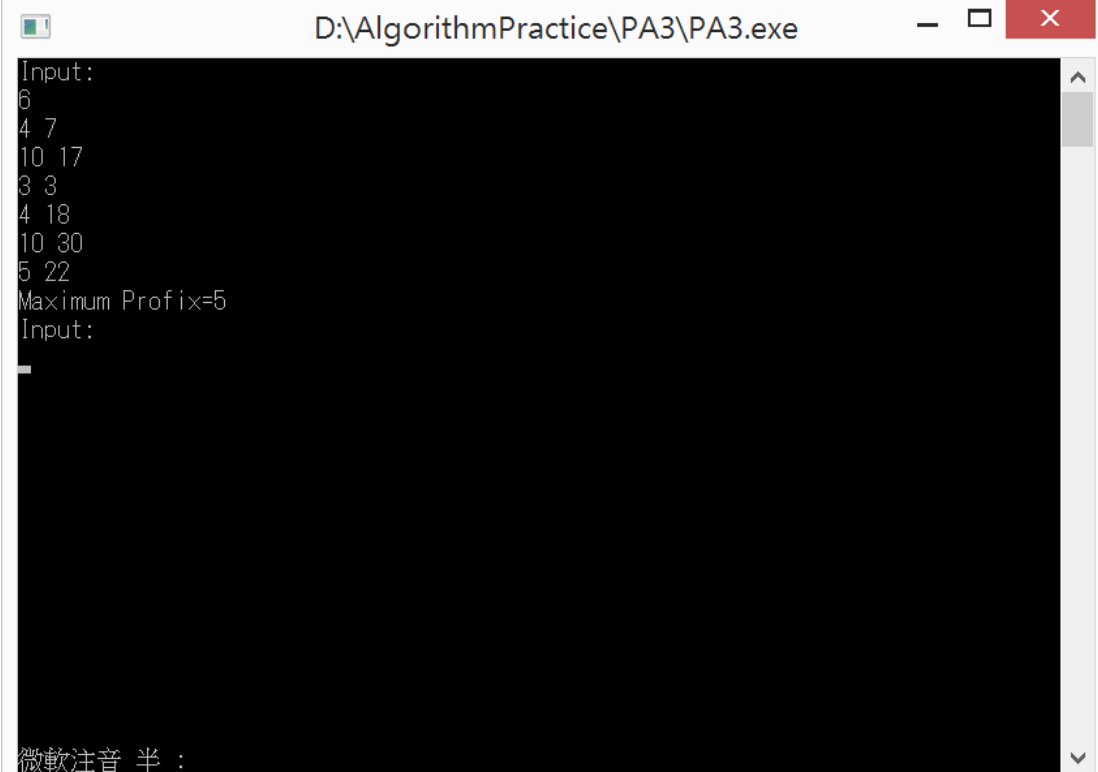
若不可行，J.pop() ->  $O(\lg n)$

以上兩行做了 n 次共花費  $O(n \lg n)$

故整體時間複雜度為  $O(n \lg n)$

#### 四、測試資料輸入

執行 PA3.exe



```
Input:
6
4 7
10 17
3 3
4 18
10 30
5 22
Maximum Prefix=5
Input:
_
```

微軟注音 半 :