Branch-and-bound Programming Assignment #4

1. Write a branch-and-bound algorithm for the problem of scheduling with

deadlines discussed in Section 4.3.2.

1. Develop a Java Program to implement your branch-and-bound algorithm.
2. Given an example with at least 10 jobs to show your program working properly.

請繳交Java source code 和一份說明文件。The document must contain your algorithm and screenshots for running the example described in step 3。

無說明文件，以零分計算此作業。

## 介紹

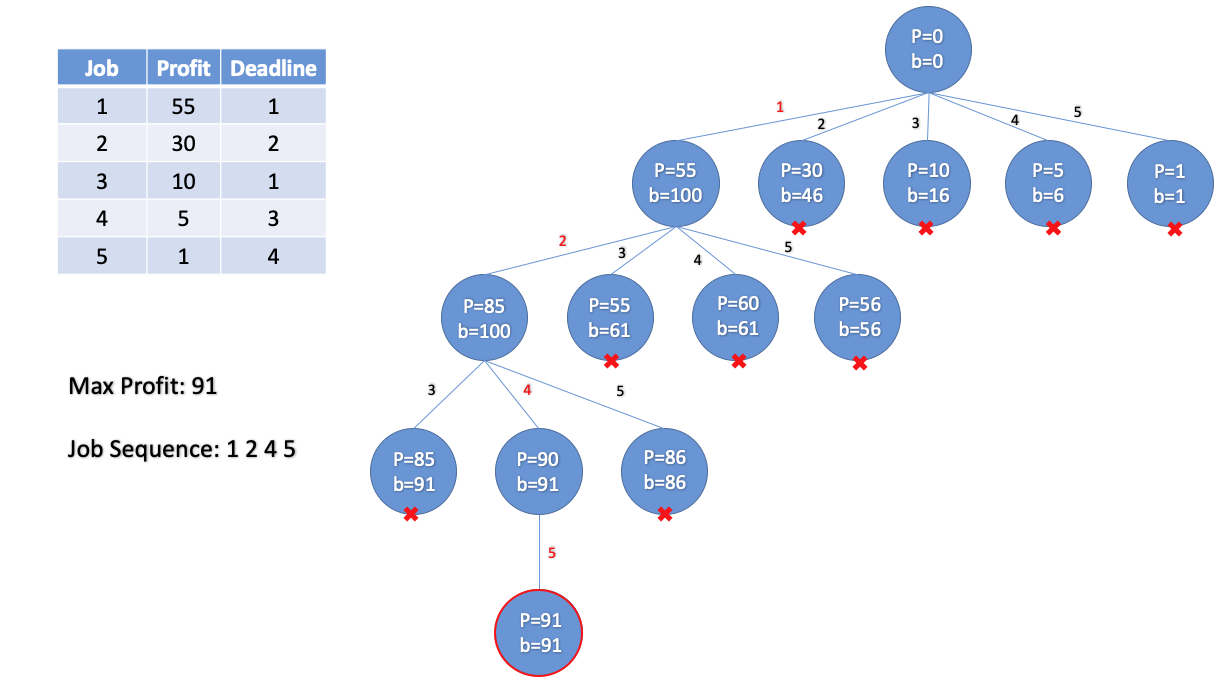
branch and bound(分支定界)目標是找出滿足條件的一個解，所謂分支就是採用廣度優先的策略，以Queue佇列方式下去實作，在每個節點中拋棄不滿足約束條件的節點，故不繼續擴展該節點以下的子樹。以下會以 branch and bound 來實作 scheduling with deadlines 排程的問題。

## 演算法

首先將工作的Profit由大至小排序

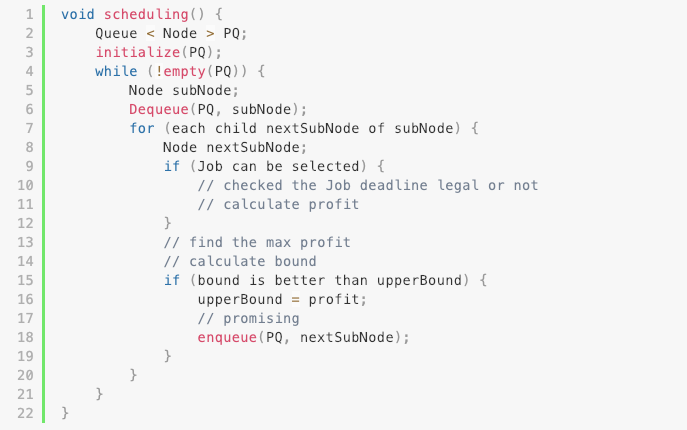
* Profit 算法:  
  計算目前節點內可執行的工作 Profit 加總(需檢查 Deadline)
* Bound 算法:  
  包含目前節點內可執行的工作尋找前k大的工作 Profit

以下範例是五個工作的空間狀態樹，Profit 已經先行排序(ps.壓縮檔內的 PPT 有完整計算過程)



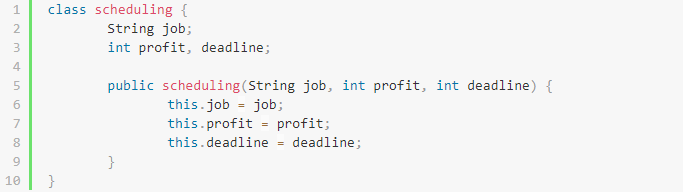
### Pseudocode

排程問題演算法架構如下，首先建立一個 Queue 佇列來儲存每個子節點 Node，並清空初始化佇列。接著依照廣度優先搜尋的順序逐一清空走訪佇列內的子節點。此外在迴圈中去比對每個工作的deadline是否已到若無代表該工作能進行並標記起來。計算 profit、bound 最後再比對 bound 是否小於等於 upperbound，若成立則 nonpromising 該節點不繼續擴展。此外 upperbound 為每次節點中的最大 profit，故每次計算完都要進行檢查並且更新 upperbound。全部結束後若尚有子節點(promising)再將放入 Queue 中繼續走訪。



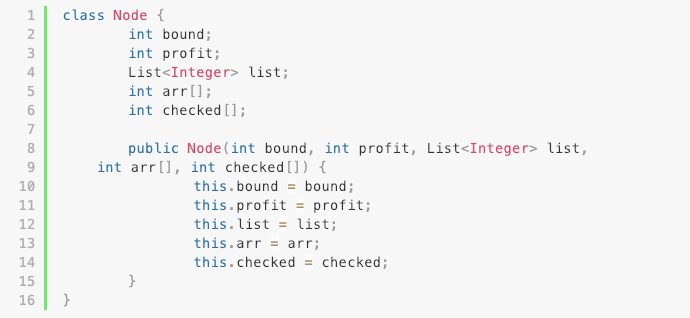
## 建立scheduling類別

為了資料方便處理這邊建立一個 scheduling 的類別專門放置每一個工作內容的利益和截止時間。該列別建立三個變數分別為字串型態的工作名稱(Job)，以及兩個整數型態的變數分別為利益(Proft)與截止時間(Deadline)。最後在建立建構子來初始化每個變數值。



## 建立Node節點

我們必須將每個節點的資訊儲存下來，所以每個節點會有 bound、profit、list(目前工作串列)、arr(陣列儲存工作順序)、checked(陣列標記某個工作是否已經排定工作0與1表示)，最後自建立建構子初始化節點中的每一個變數。



## main() 主程式

在主函式中主要目地是讀取使用者所輸入的測試資料首先要輸入整數N代表以下會有N個工作。接下來會要求使用者輸入N筆工作資料分別為(job name、profit、deadline)。

下圖程式第二行建立一個自訂義 scheduling 型態的 LinkedList 取名為 list 專門來儲存所有的工作項目與內容。第三行建立一個整數型態的陣列 solution[] 專門來儲存最佳解的工作順序。第四行有多個整數變數第一個 N 為工作數量。第二個 maxProfit 為所有工作序列中最大的利益值，並將它初始值為最小值，最後一個變數 maxDeadline 儲存工作序列中最大的截止時間(deadline)。

測資都依序輸入後我們先將這些資料以 Profit 進行排序，程式第二十三行使用第一個程式作業的合併排序來實作。

程式第二五行進入Scheduling()函式使用branch and bound 並使用最佳優先搜尋(Best-First Search)來尋找最佳工作排程以及計算最大利益。

最後印出結果，首先輸出排序後的工作序列，再來輸出滿足條件的最大利益，緊接著是該最大利益的一組解(工作序列)。



## Scheduling() 函式

此函式是使用最佳優先搜尋法來做排程運算，首先初始化 upperBound 和 bound 變數。profit 變數是儲存每一次排程組合的利益值，arr[] 陣列是儲存每次工作排程的順序，checked[]陣列是紀錄目前某個工作是否已進入排程中1代表有排入工作，反之0尚未排入工作序列中。  
 此演算法是利用 Queue 佇列實作，採先進先出觀念(FIFO)，程式第八行採用 while 迴圈並判斷目前佇列中是否還有數值，直到佇列為空則跳出迴圈。程式第九行使用 poll() 方法用來取出佇列前端物件。第十三至十七行將所有變數初始化並取得上一個節點中計算出來的結果，二十二至二十九行計算profit加總並檢查是否可以執行此工作，若可以執行則將工作排成放入arr[] 陣列中儲存並將此工作的利益加到變數 profit 中，最後 在 checked[] 陣列中標註1代表此工作已被排定。程式碼三十至四十一行是計算 bound，bound 計算方式是包含目前節點內可執行的工作尋找前k大的工作 Profit 並加總。第四十二到四十八行首先判斷是否 promising，若是(promising)則更新 upperBound 以及 maxProfit 接著繼續走訪子節點故將放入佇列中等待走訪(branch)，若 profit 小於 upperBound 則確定了界限(bound)故不做後半部子樹走訪(nonpromising)。



## 排序

在排程 Branch-and-bound 演算法過程中，資料要先行做排序才能執行。原理跟01背包類似，我們這邊是要尋找每個工作最大利益 Profit 故在我們的程式中首先要先以 Profit 由大至小來排序。以下程式碼為第一個程式作業合併排序，在此作業中套用。



## 執行與測試

輸入說明:  
第一行為工作數量N，接下來會等待輸入N筆工作資料分別為 (Job Name、Profit、Deadline)。  
輸出說明:  
首先會輸出排序後的串列，接著輸出計算結果第一行為滿足條件的最大利益，第二行為該最大利益的一組解(工作序列)。

* 測試ㄧ

測資:

5

J1 55 1

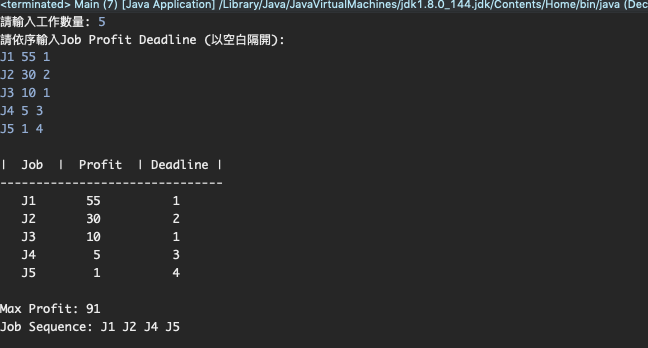
J2 30 2

J3 10 1

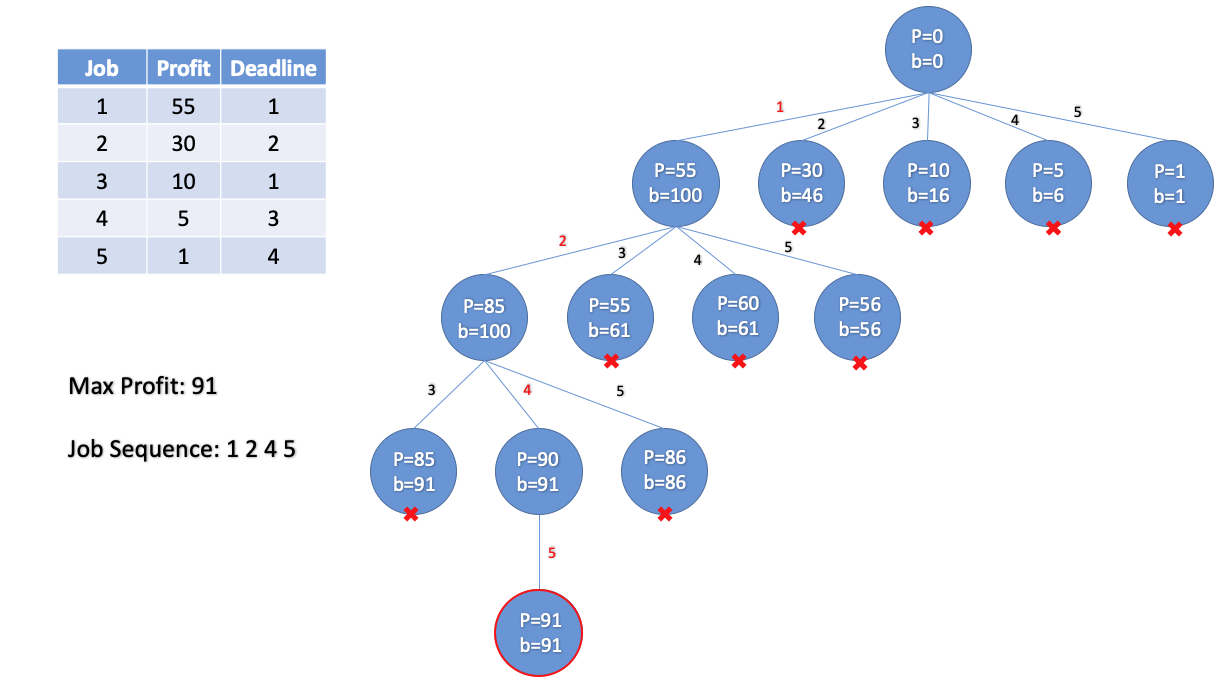
J4 5 3

J5 1 4

55+30+5+1=91



State Space Tree:



* 測試二

測資:

10

J1 80 1

J2 45 2

J3 40 3

J4 30 3

J5 30 2

J6 20 4

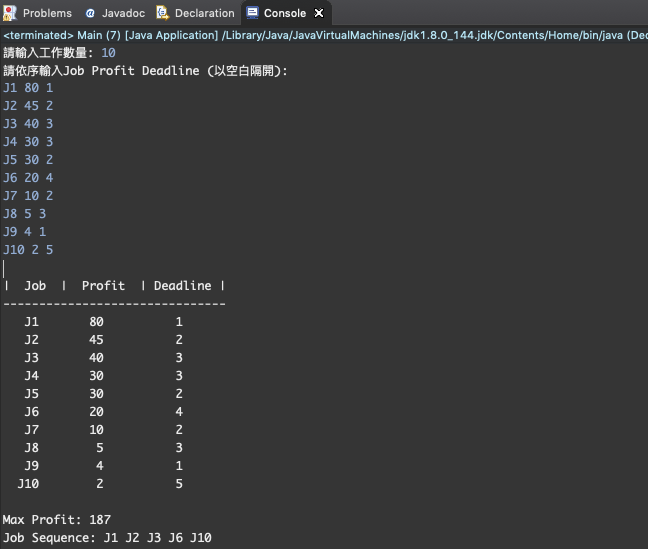
J7 10 2

J8 5 3

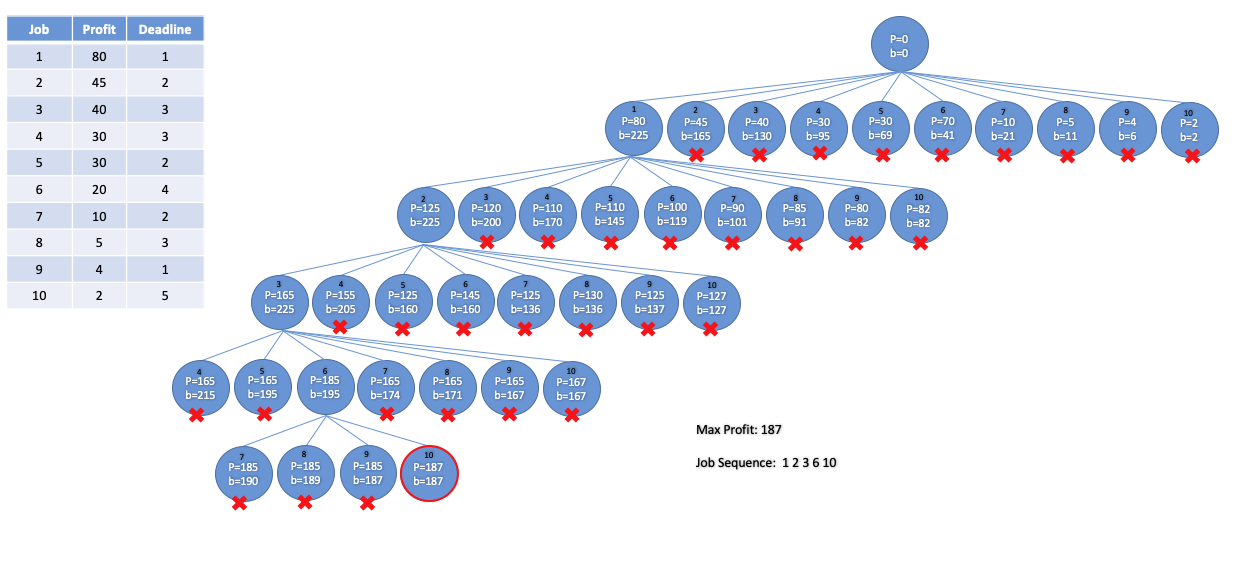
J9 4 1

J10 2 5

80+45+40+20+2=187



State Sspace Tree:



* 測試三

測資:

10

J1 80 1

J2 2 5

J3 40 3

J4 10 2

J5 45 2

J6 20 4

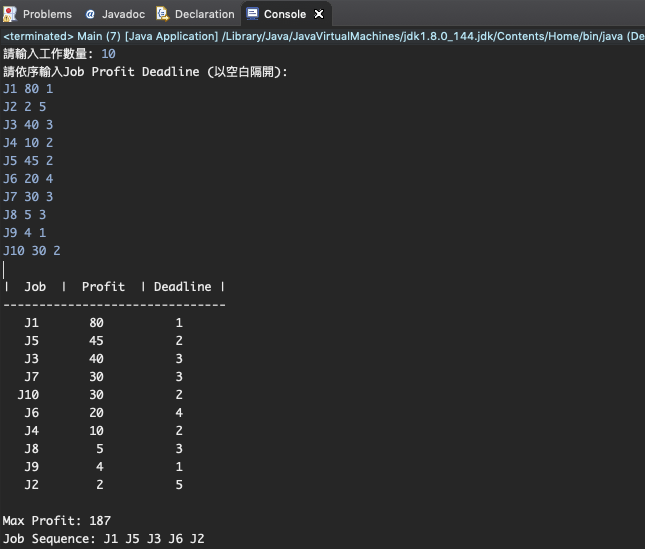
J7 30 3

J8 5 3

J9 4 1

J10 30 2

80+45+40+20+2=187



* 測試四

測資:

10

J8 30 1

J1 60 5

J10 20 3

J3 52 5

J4 50 4

J2 55 4

J9 25 1

J6 40 2

J7 33 4

J5 45 2

45+50+52+55+60=262

