中原大學資訊工程系 演算法分析第二次機測

Deadline: 6 / 16 / 2021 (星期三)

注意：本學期取消期末實體考試，改以機測成績為主(占總成績 35%)

【程式設計說明】

1. 為了因應新冠疫情與避免群聚感染，本次機測改由每位同學獨立解題。
2. 程式設計必須使用 Python 程式語言，版本採用 3.8。
3. 可參考課本、相關書籍或 Algorithms.py 等解題，解題方法及演算法不限，但絕對嚴禁抄襲其他同學的程式。若發現抄襲屬實，兩位同學均以零分計。
4. 所有輸入及輸出均為標準格式，即程式在命令提示字元環境下執行時可以鍵盤輸入資料，本機測不採讀檔方式進行。
5. 請同學保持程式設計的良好習慣，充分使用程式註解。
6. 每一支程式均須附上姓名及學號，例如:

```
#    演算法分析機測
#    學號: 10427001
#    姓名: 江○○
#    中原大學資訊工程系
```

程式命名依[學號+題號]為原則。例如:
10427001_1.py
10427001_2.py

【機測須知】

1. 評分以解題成功之題數多寡與執行時間決定。
2. 程式必須能處理不同之輸入資料 (但輸入格式與範例相同)，並輸出正確結果 (輸出格式必須與範例相同)。程式之輸出結果錯誤、輸出格式與範例不符、或在執行後超過 60 秒仍未結束，均視為失敗。若程式測試失敗給予基本分數，未繳交程式則以零分計。
3. 本機測於規定之期限前，上傳至 i-learning 作業區，逾期不得補繳。
4. 助教將使用不同之輸入資料作為測試與評分依據，請同學應在繳交前充分測試程式。
5. 機測成績納入學期總成績計算，請同學把握!

指導教授: 張元翔

I. 雙子星塔 (Twin Towers)

很久很久以前，在古帝國有兩座高塔位於兩座城市中，他們的形狀不太相同。但是他們都是用圓柱形的石塊一個堆在另一個上面建起來的。每個圓柱形石塊的高度都相同 (假設為 1)，但是半徑卻不一。所以，雖然兩座高塔的形狀不一樣，但事實上他們可能有許多石塊是相同的。

在高塔建成的一千年後，國王要求建築師拿掉高塔的某些石塊，使得兩座高塔的形狀大小和高度一樣。但同時要盡可能讓高塔的高度越高越好。新高塔的石塊的順序也必須和原來的高塔一樣。國王認為這樣可以代表兩座城市之間的和諧與平等。他為這兩座高塔命名為「雙子星塔」

現在，你的任務是就是算出這雙子星塔的高度。

輸入說明:

輸入含有多組測試資料，每組測試資料 3 列，代表原來兩座高塔的資料。每組測試資料的第一列有 2 個整數 N1 和 N2 ($1 \le N1$、$N2 \le 100$)，代表這兩座高塔原來的高度，0 0 代表結束。接下來的一列有 N1 個正整數，代表第一座高塔石塊的半徑 (由高到低)。再接下來的一列有 N2 個正整數，代表第二座高塔石塊的半徑 (由高到低)。

如果讀到 0 0，代表輸入結束。

輸出說明:

對每一組測試資料，輸出這是第幾組測試資料以及這兩座塔後來的高度。每組測試資料後請空一列。

輸入範例:

```
7 6
20 15 10 15 25 20 15
15 25 10 20 15 20
8 9
10 20 20 10 20 10 20 10
20 10 20 10 10 20 10 10 20
0 0
```

輸出範例:

```
Twin Towers #1
Number of Tiles : 4
```

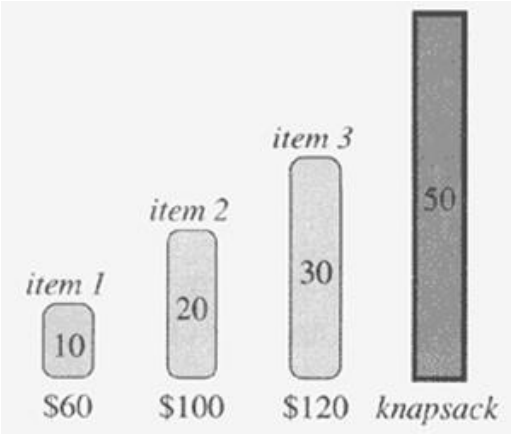
Twin Towers #2
Number of Tiles : 6

II. 0-1 背包問題 (0-1 Knapsack)

0-1 Knapsack 問題 (又稱為 Bin-Packing 問題) 是電腦科學中非常具有代表性的問題，問題描述如下: 有一小偷到一家店內偷東西，他發現 *n* 項物品，每項物品各有不同價值及不同重量，小偷的目的是帶走總價值最高的物品，但他能帶走的背包 (Knapsack) 有重量的限制。試寫程式解決 0-1 背包問題 (即每項物品僅能**取走**或**不取**，無法取走部分)，並須求得**最佳解** (Optimal Solution)。

輸入說明:

輸入物品 Knapsack 重量 *W* 及物品總數 *n*，接著分別是各項物品的重量及價值 (均為正整數)。以下為輸入範例:



輸出說明:

求出可能之最高總價值，並列出取走物件的編號 (須按編號由小到大順序排列，並以逗號隔開)。

輸入範例:

50
3
10 60
20 100
30 120

輸出範例:

Total Value = 220
Items 2, 3

III. 霍夫曼碼 (Huffman Codes)

霍夫曼碼在資料壓縮中是常見的技術之一，被廣泛使用在音訊、影像、視訊等多媒體壓縮應用中。霍夫曼碼的主要原理是由於表示資料的方式可以分成兩種，若使用**固定長度字碼** (Fixed-Length Codeword)，則每一個字元是以固定長度的編碼方式；霍夫曼碼是比固定長度編碼更為有效的編碼方式，採用**可變長度編字碼** (Variable-Length Codeword) 的方式。

以下述字元編碼為例，試設計程式完成霍夫曼碼的編碼 (Encoding) 及解碼 (Decoding)。

	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

輸入說明:
每組輸入包含的字元數 n（均為正整數），0 表示結束，緊接為每一個字元及其發生頻率，所有字元均可能是英文字母大或小寫，且頻率均為正整數 (但不會事先排序)。

輸出說明:
就每組輸入列出結果，包含：每一個字元的霍夫曼碼。

輸入範例:
6
a 45
b 13
c 12
d 16
e 9
f 5
6
A 2
B 6
C 15
D 12
E 8
F 3
0

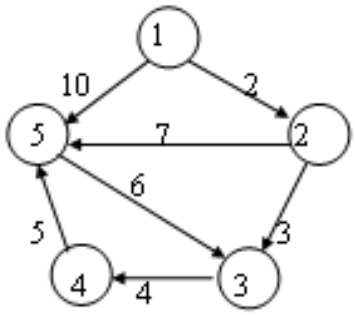
輸出範例:

Huffman Codes #1
a 0
b 101
c 100
d 111
e 1101
f 1100

Huffman Codes #2
A 0100
B 011
C 11
D 10
E 00
F 0101

IV. 單一源最短路徑 (Single-Source Shortest Paths)

給定一有向圖$G=(V,E)$ 與**源頂點** (Source Vertex)，**單一源最短路徑** (Single-Source Shortest Paths) 問題的目的是找到 Source Vertex與其他頂點的最短距離。舉例說明，下圖為典型的有向圖，有向圖的**權重** (weights) 代表兩頂點的距離，在此均為正整數，並具有方向性。假設頂點的編號分別為 1…n，試設計程式輸出Source Vertex與其他頂點的最短距離。



輸入說明：

每組輸入含頂點個數 n (原則上頂點數不超過 20，0 代表結束) 與**邊** (Edges) 的個數，接著為 Source Vertex 的編號，最後列出連接每個邊的兩個頂點與權重，其間以空格隔開。

輸出說明：

Source Vertex 至其他頂點的最短距離。

輸入範例 (見上圖)：

```
5
7
1
1 2 2
1 5 10
2 3 3
2 5 7
3 4 4
4 5 5
5 3 6
0
```

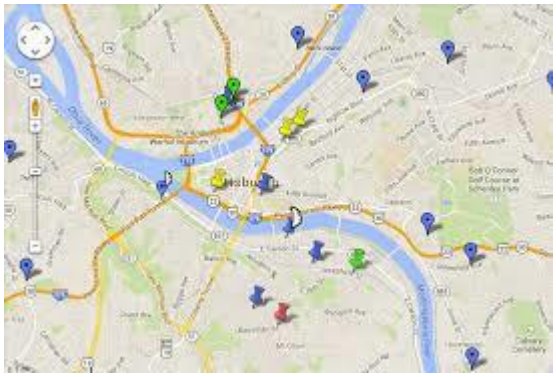
輸出範例：

```
1 to 2 = 2
```

1 to 3 = 5

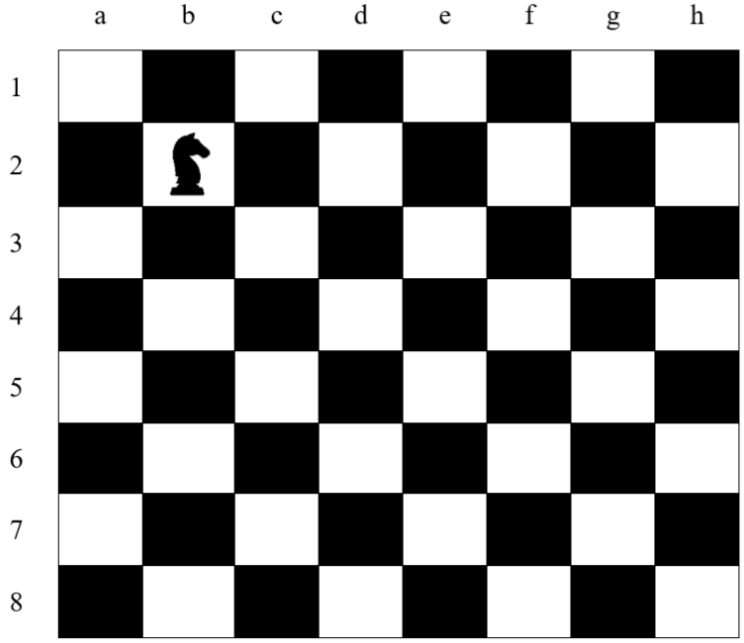
1 to 4 = 9

1 to 5 = 9



V. 西洋棋騎士 (Chess Knight)

西洋棋 (Chess) 是一種二人對弈的戰術棋盤遊戲，也是世界上最流行的遊戲之一。在此，讓我們探討西洋棋**騎士** (Knight) 的移動問題，如下圖。西洋棋盤是一個 8 × 8 的棋盤，每一列使用 1 ~ 8 編號；每一行則使用 a ~ h 編號。



我們想要解決的問題是：「給定兩個位置 X 與 Y，若騎士從 X 到 Y 至少需要走幾步?」

舉例說明，若想將騎士從 b2 移到 c3，至少需要 2 步，即先將騎士從 b2 移到 d1，再從 d1 移到 c3。另一種走法，是先將騎士從 b2 移到 a4，再從 a4 移到 c3，但移動的步數相同。

輸入說明:
兩個西洋棋的座標位置。每個座標位置是由一個小寫英文字母 (a ~ h) 與一個數字 (1 ~ 8) 組成；0 0 代表結束。

輸出說明:
騎士至少需移動的次數。

輸入範例:
b2 c3
a1 b2
a1 h8

0 0

輸出範例:

From b2 to c3, Knight Moves = 2
From a1 to b2, Knight Moves = 4
From a1 to h8, Knight Moves = 6