I. 最長共同子序列 (Longest Common Subsequence)

電腦科學領域中，**最長共同子序列** (Longest Common Subsequence) 問題，簡稱 LCS 問題，是相當具有代表性的問題。LCS 問題描述如下：給定兩個序列 (或字串)，目的是找到兩個序列連續的共同字元，稱為共同序列 (Common Subsequence)，且長度必須最長。舉例說明：

給定兩個序列 (或字串)：

X = < A, B, C, B, D, A, B >
Y = < B, D, C, A, B, A >

則這兩個序列的 LCS 為：

Length of LCS = 4
LCS = < B, C, B, A >

註：若 LCS 的解非唯一解，你只要列出其中一解即可。

試採用**動態規劃法** (Dynamic Programming) 的設計策略進行程式設計，解決 LCS 問題。

輸入說明:
輸入含有多組測試資料，每組測試資料 3 列，代表兩個序列。每組測試資料的第一列有 2 個整數 m 和 n (1 ≤ m、n ≤ 100)，分別代表這兩個序列的長度，0 0 代表結束。接下來的第二、三列分別有 m、n 個字元，字元為英文的大寫或小寫字母。

輸出說明:
對每一組測試資料，輸出 LCS 的最佳解，包含 LCS 的長度與 LCS。

輸入範例:
7 6
A B C B D A B
B D C A B A
7 6
B C D A A C D
A C D B A C
0 0

輸出範例:

Case #1
Length of LCS = 4
LCS = BCBA

Case #2
Length of LCS = 4
LCS = CDAC

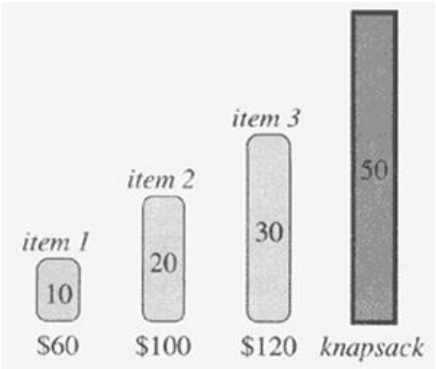
II. 0-1 背包問題 (0-1 Knapsack)

電腦科學領域中，**0-1 Knapsack 問題** (又稱為 Bin-Packing 問題) 是具有代表性的問題，問題描述如下: 有一個小偷到一家店內偷東西，他發現店內有 *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。

試設計程式解決 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。

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，採用可變長度編碼 (Variable-Length Codeword) 的方式。

以下述字元編碼為例，試參考課本描述的演算法，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)。最後，給定一特定二元碼，試使用霍夫曼碼對其進行解碼。

輸出說明:
就每組輸入列出結果，包含：(1) 每一個字元的霍夫曼碼；及(2) 解碼之結果。

輸入範例:
6
a 45
b 13
c 12
d 16
e 9
f 5
01001101
6
A 2
B 6
C 15
D 12

E 8
F 3
010101001100
0

輸出範例:

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

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

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

圖形演算法中，**單一源點最短路徑問題** (Single-Source Shortest Paths Problem) 是最常見的問題之一。Dijkstra 演算法是最具代表性的演算法，主要是針對圖形中權重值均為正數的情況。下圖為 Dijkstra 演算法操作過程，資料結構牽涉圖形表示法及最小優先佇列的實現：

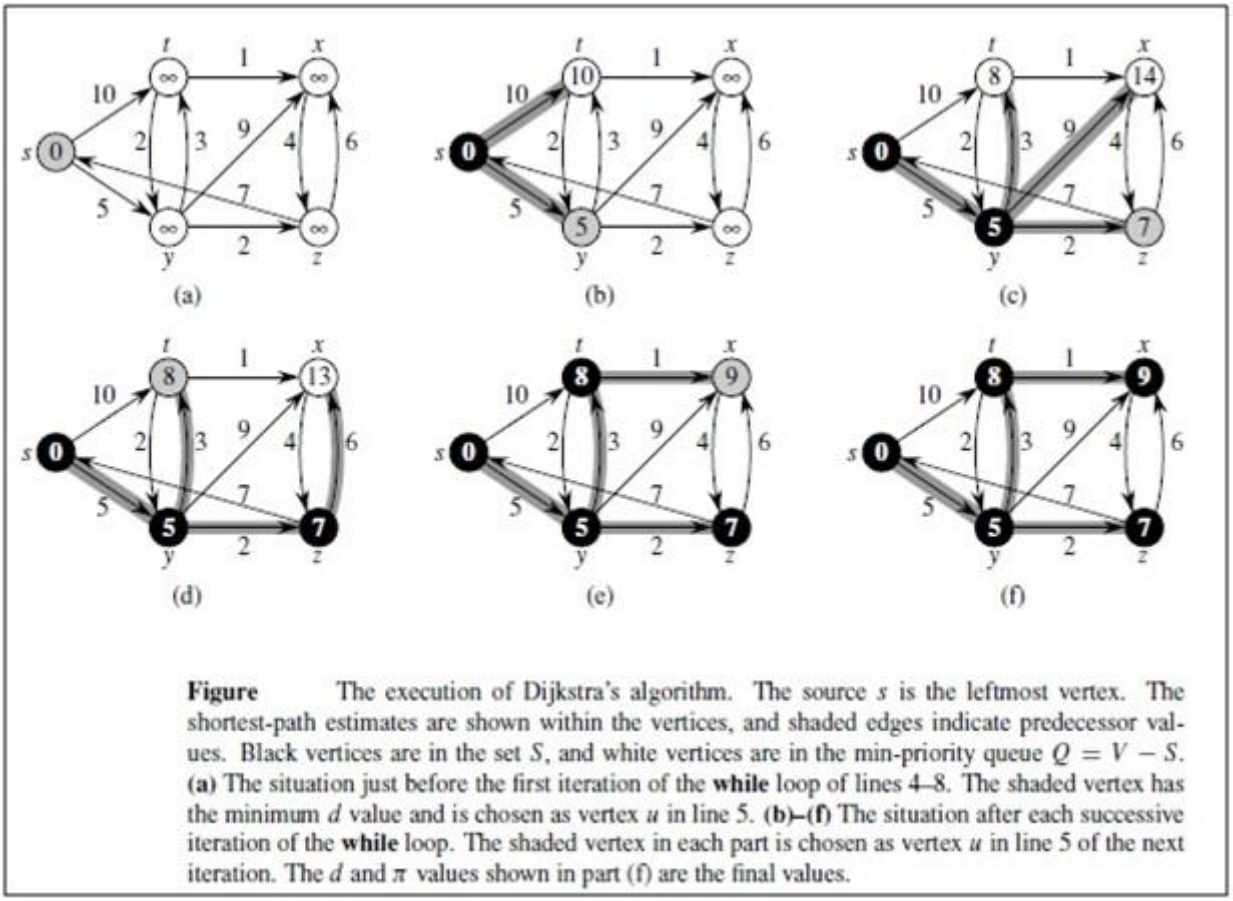


Figure The execution of Dijkstra's algorithm. The source s is the leftmost vertex. The shortest-path estimates are shown within the vertices, and shaded edges indicate predecessor values. Black vertices are in the set S, and white vertices are in the min-priority queue $Q = V - S$. **(a)** The situation just before the first iteration of the **while** loop of lines 4–8. The shaded vertex has the minimum d value and is chosen as vertex u in line 5. **(b)–(f)** The situation after each successive iteration of the **while** loop. The shaded vertex in each part is chosen as vertex u in line 5 of the next iteration. The d and π values shown in part (f) are the final values.

試根據課本 (講義) 演算法編寫程式實現 Dijkstra 演算法。

輸入說明:

每組輸入包含節點及邊的個數，0 0 表示結束，首先為各節點，均依英文小寫字母表示，s 代表**源節點** (Source Node) 即出發點，在輸入中均先列出，其他節點則以小寫字母 *a*、*b*、...等表示，緊接為連接節點的各邊及其權重值 (不依特定順序)。

輸出說明:

就每組輸入依輸入之節點順序列出源點至各節點之最短路徑，此外，除了源節點外，列出各節點在最短路徑樹中的父節點。

輸入範例:

5 10
s t x y z
s t 10
s y 5
t x 1
t y 2
x z 4
y t 3
y x 9
y z 2
z s 7
z x 6
0 0

輸出範例:

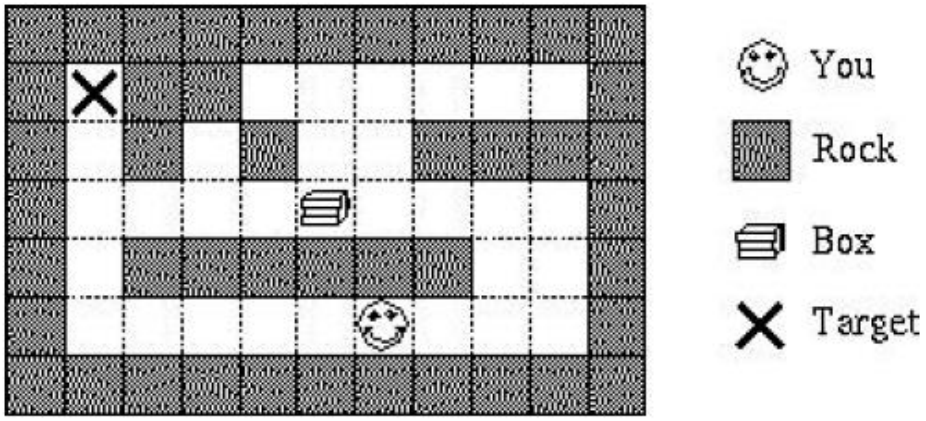
Graph #1
From s to t = 8
From s to x = 9
From s to y = 5
From s to z = 7

s source node
t’s parent node = y
x’s parent node = t
y’s parent node = s
z’s parent node = y

V. 推箱子遊戲 (Pushing Box Game)

假設你身處在二維的迷宮之中 (如圖)，迷宮中可能含有 (或不含) 大石塊 (Rock)，在沒有大石塊阻擋時，你可以向東、西、南、北等四方向一次移動一格。其中一格擺了一個箱子 (Box)，你只可以向箱子移動的方向推，譬如：若你站在箱子的東面，則你只能向西面推箱子; 若你站在箱子的南面，則你只能向北面推箱子; 以此類推。在任何情況下，由於箱子非常沉重，你都無法拉動箱子，萬一你把箱子推到死角，則你將再沒機會移動箱子。

如圖所示，有一目標 (Target)，你的工作是將箱子推到指定的目標，由於箱子非常沉重，因此推動箱子次數必須最少，試寫一程式解決這個問題。



[輸入說明]

輸入含有幾個迷宮，每一個迷宮首先定義迷宮的大小為 r 及 c，分別代表迷宮的列數及行數 (其中，r、$c \le 20$)。緊接為 r 列，每一列含 c 個**字元** (Characters)，其中，大石塊用 # 表示，空格用 . 表示。你的初始位置為 S，箱子的初始位置為 B，目標的位置為 T。當輸入之 r 及 c 為 0 時代表結束。

[輸出說明]

首先，列出迷宮的編號。接著，印出一組推動箱子的方向。若無法將箱子推到指定的目標，則列出 Impossible。若有兩組推動箱子的次數均為最少，則列出總移動數最少的 (即含移動及推動的次數)。使用 E, W, S, N, e, w, s, n 分別代表推動或移動的方向 (東、西、南、北)，大寫表示目前需推動箱子，小寫表示僅需移動。迷宮與迷宮間則以空行隔開。

[輸入範例]

1 7
SB….T

1 7
SB..#.T
7 11
###########
#T##.....#
#.#.#..####
#....B….#
#.######..#
#.....S…#
###########
0 0

本範例與上圖相同

[輸出範例]
Maze #1
EEEEE

Maze #2
Impossible

Maze #3
eennwwWWWWeeeeeesswwwwwwwnNN