

TOI推廣計畫

解題-尋找蜂后



題目

科學家發現了一種新型的蜜蜂，它們的蜂巢結構為立方體，蜂巢被蜂巢壁分割為好幾個小空間，而**蜂后住在蜂巢內最大的空間中**。今日為了研究這種蜜蜂，需找到蜂巢中的最大空間，以確認蜂后位置。
蜂巢的**長度為 L** ，**寬度為 W** ，**高度為 H** ，請你寫一個程式計算蜂巢中的最大空間為多少。

輸入格式

每筆測試資料為二列:

1. 有三個正整數 L 、 W 、 H ($1 \leq L, W, H \leq 135$)， L 、 W 、 H 代表蜂巢的長、寬、高。
2. 共有 $L \times W \times H$ 個字元，前 $L \times W$ 個字元為蜂巢的第一層，接下來 $L \times W$ 個字元為蜂巢的第二層，以此類推。每個字元可以是 0 或 1，1 代表蜂巢壁，0 代表蜂巢中的空間。

輸出格式

對每筆資料請輸出一列，請輸出蜂巢中的最大空間。

以輸入範例為例，則蜂巢結構如下圖所示，蜂后所在的空間為灰色區域，共有七格。

010	001	111
101	001	101
010	111	111

輸入範例

333

010101010001001111111101111


輸出範例

7

解題重點:

1. 三維陣列儲存整個蜂巢
2. 尋找蜂巢中最大的空間size
 - **method 1 :**
深度優先搜尋 (DFS)
 - **method 2 :**
廣度優先搜尋 (BFS)





尋找蜂后 尋找蜂巢中最大空間

(深度優先搜尋 DFS)

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 深度優先搜尋 (DFS) :

先任意傳入一個為0的位置，並搜尋上下左右前後是否有尚未拜訪過的空格，並搜尋上下左右的周遭是否有空格，直至整個空間皆被拜訪，拜訪的數量即是此空間的大小。

➤ 以第一組範例測資為例：

```
010 001 111
101 001 101
010 111 111
```

◆尋找蜂后 – 尋找蜂巢中最大空間

➤ 深度優先搜尋 (DFS) :

先任意傳入一個為0的位置，並搜尋上下左右前後是否有尚未拜訪過的空格，並搜尋上下左右的周遭是否有空格，直至整個空間皆被拜訪，拜訪的數量即是此空間的大小。

➤ 以第一組範例測資為例：

010 001 111

101 001 101

010 111 111

往上一層

◆尋找蜂后 – 尋找蜂巢中最大空間

➤ 深度優先搜尋 (DFS) :

先任意傳入一個為0的位置，並搜尋上下左右前後是否有尚未拜訪過的空格，並搜尋上下左右的周遭是否有空格，直至整個空間皆被拜訪，拜訪的數量即是此空間的大小。

➤ 以第一組範例測資為例：

010	001	111	
101	001	101	
010	111	111	同層往下一格

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 深度優先搜尋 (DFS) :

先任意傳入一個為0的位置，並搜尋上下左右前後是否有尚未拜訪過的空格，並搜尋上下左右的周遭是否有空格，直至整個空間皆被拜訪，拜訪的數量即是此空間的大小。

➤ 以第一組範例測資為例：

010	001	111	
101	001	101	
010	111	111	同層往右一格

◆尋找蜂后 – 尋找蜂巢中最大空間

➤ 深度優先搜尋 (DFS) :

先任意傳入一個為0的位置，並搜尋上下左右前後是否有尚未拜訪過的空格，並搜尋上下左右的周遭是否有空格，直至整個空間皆被拜訪，拜訪的數量即是此空間的大小。

➤ 以第一組範例測資為例：

010	001	111	
101	001	101	
010	111	111	同層往上一格

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 深度優先搜尋 (DFS) :

先任意傳入一個為0的位置，並搜尋上下左右前後是否有尚未拜訪過的空格，並搜尋上下左右的周遭是否有空格，直至整個空間皆被拜訪，拜訪的數量即是此空間的大小。

➤ 以第一組範例測資為例：

010	001	111	
101	001	101	已無未拜訪空間
010	111	111	回溯並搜尋
			往上一層

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 深度優先搜尋 (DFS) :

先任意傳入一個為0的位置，並搜尋上下左右前後是否有尚未拜訪過的空格，並搜尋上下左右的周遭是否有空格，直至整個空間皆被拜訪，拜訪的數量即是此空間的大小。

➤ 以第一組範例測資為例：

010 001 111

101 001 101

010 111 111

已無未拜訪空間
回溯並搜尋
往上一層

◆尋找蜂后 – 尋找蜂巢中最大空間

➤ 深度優先搜尋 (DFS) :

先任意傳入一個為0的位置，並搜尋上下左右前後是否有尚未拜訪過的空格，並搜尋上下左右的周遭是否有空格，直至整個空間皆被拜訪，拜訪的數量即是此空間的大小。

➤ 以第一組範例測資為例：

	010	001	111
完全無未拜訪空間，共計7個空格	101	001	101
→ 此次DFS獲得大小為7的空間	010	111	111


◆尋找蜂后 – 尋找蜂巢中最大空間

➤ 缺點：

- 由於使用遞迴進行回溯，因此在測資較大時，會造成 **runtime error (stack overflow)**。

- 最後一組測資的大小為 135^3 ，無法用此方法解決。



The slide features several cartoon bees. One bee is on the left side, another is at the top left, and a third is at the top center. On the right side, there is a cluster of bees around a yellow beehive hanging from a brown branch. At the bottom left, a bee is positioned above a group of red flowers. At the bottom right, there is a group of yellow flowers.

尋找蜂后 尋找蜂巢中最大空間

(廣度優先搜尋 BFS)

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 廣度優先搜尋 (BFS) :

先任意傳入一個為 0 的位置，並將周圍未拜訪的空格放入佇列中，並使用一個陣列紀錄此佇列中有哪些空格，避免重複拜訪。

➤ 以第一組範例測資為例：

藍色為已拜訪的節點。

紅色為在佇列中，而尚未拜訪的節點。

0	10	001	111
1	01	001	101
0	10	111	111

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 廣度優先搜尋 (BFS) :

先任意傳入一個為 0 的位置，並將周圍未拜訪的空格放入佇列中，並使用一個陣列紀錄此佇列中有哪些空格，避免重複拜訪。

➤ 以第一組範例測資為例：

藍色為已拜訪的節點。

紅色為在佇列中，而尚未拜訪的節點。

010	001	111
101	001	101
010	111	111

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 廣度優先搜尋 (BFS) :

先任意傳入一個為 0 的位置，並將周圍未拜訪的空格放入佇列中，並使用一個陣列紀錄此佇列中有哪些空格，避免重複拜訪。

➤ 以第一組範例測資為例：

藍色為已拜訪的節點。

紅色為在佇列中，而尚未拜訪的節點。

010	001	111
101	001	101
010	111	111

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 廣度優先搜尋 (BFS) :

先任意傳入一個為 0 的位置，並將周圍未拜訪的空格放入佇列中，並使用一個陣列紀錄此佇列中有哪些空格，避免重複拜訪。

➤ 以第一組範例測資為例：

藍色為已拜訪的節點。

紅色為在佇列中，而尚未拜訪的節點。

010	001	111
101	001	101
010	111	111

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 廣度優先搜尋 (BFS) :

先任意傳入一個為 0 的位置，並將周圍未拜訪的空格放入佇列中，並使用一個陣列紀錄此佇列中有哪些空格，避免重複拜訪。

➤ 以第一組範例測資為例：

藍色為已拜訪的節點。

紅色為在佇列中，而尚未拜訪的節點。

010	001	111
101	001	101
010	111	111

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 廣度優先搜尋 (BFS) :

先任意傳入一個為 0 的位置，並將周圍未拜訪的空格放入佇列中，並使用一個陣列紀錄此佇列中有哪些空格，避免重複拜訪。

➤ 以第一組範例測資為例：

藍色為已拜訪的節點。

紅色為在佇列中，而尚未拜訪的節點。

010	001	111
101	001	101
010	111	111

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 廣度優先搜尋 (BFS) :

先任意傳入一個為 0 的位置，並將周圍未拜訪的空格放入佇列中，並使用一個陣列紀錄此佇列中有哪些空格，避免重複拜訪。

➤ 以第一組範例測資為例：

藍色為已拜訪的節點。

紅色為在佇列中，而尚未拜訪的節點。

010	001	111
101	001	101
010	111	111

◆ 尋找蜂后 – 尋找蜂巢中最大空間

➤ 廣度優先搜尋 (BFS) :

先任意傳入一個為 0 的位置，並將周圍未拜訪的空格放入佇列中，並使用一個陣列紀錄此佇列中有哪些空格，避免重複拜訪。

➤ 以第一組範例測資為例：

藍色為已拜訪的節點。

→ 最後求得答案為 7

010	001	111
101	001	101
010	111	111