

113 學年度技藝競賽培訓

Ching-Chang Lin
cclin@ba.tpcu.edu.tw

Taipei City University of Science and Technology — November 25, 2024

113 學年度全國高級中等學校技藝競賽官網

為提升全國高級中等學校學生的技藝學習動機，促進學校間技藝的交流與觀摩，並提升學生技藝水準，檢視學生學習成效及技職教育成果，每年 11 月舉辦的全國高級中等學校學生技藝競賽，統計 110 學年度五大類競賽職種，分別有工業類 28 項、商業類 11 項、農業類 9 項、家事類 8 項、海車水產類 7 項。旨在強化學生的學習動機，促進技藝交流與觀摩，並提升技藝水準。

統計 113 學年度商業類競賽職種 (A) 平面設計、(B) 網頁設計、(C) 程式設計、(D) 文書處理、(E) 電腦繪圖、(F) 會計資訊、(G) 商業簡報、(H) 職場英文，八項競賽項目，計有 137 所學校、511 位學生參與競賽。

112 術科重點

詳細說明文件請見：商科技藝競賽線上評分說明文件成績公告

Question 328

Problem A

二數有權重的相加-(單列)



Info:

送分題

```
1 a,b = map(int,input().split())
2 print(4*a+6*b)
```

採用 `sys.stdin.readline()` 或 `sys.stdin.readlines()` 可以提高效率, 但需要引入 `sys` 模組

```
1 import sys
2
3 a,b = map(int,sys.stdin.readline().split())
4 print(4*a+6*b)
```

Question 329

Problem B

剪刀石頭布單行版

i

Info:

送分題

```
1 a,b = input().split()
2
3 if a=='Y' and b=='X' or a=='X' and b=='O' or a=='O' and b=='Y':
4     print(1)
5 elif a==b:
6     print(0)
7 else:
8     print(2)
```

Question 330

Problem C

閏年

i

Info:

送分題

```
1 y=int(input())+1911
2
3 if y%400 == 0 or y%4 == 0 and y%100:
4     print('True')
5 else:
6     print('False')
```

Question 331

Problem D ASCII 碼

i

Info:

簡單題

```
1 n = int(input())
2 if 65 <= n <= 90:
3     print(1)
4 elif 97 <= n <= 122:
5     print(2)
6 else:
7     print(0)
```

Question 332

Problem E 2 的幂

i

Info:

簡單題

```
1 n = int(input())
2 for i in range(1000000000):
3     la = pow(2,i)
4     if la == n:
5         print(2**i)
6         break
7     elif la > n:
8         print(2**(i-1))
9         break
```

i

Info: 運用 $\log_2(n)$ 解法，效率較高，但是需要引入 math 模組

```
1 import math
2 n = int(input())
3 print(2**int(math.log2(n)))
```

Question 333

Problem F

質因數分解

i

Info:

基礎數學題，解法 1: 逐一判斷

```
1 n = int(input())
2 ans = []
3 while n > 1:
4     for i in range(2,n+1):
5         if n % i == 0:
6             ans.append(i)
7             n //= i
8             break
9
10 print(*ans)
```

i

Info: 基礎數學題，解法 2:

```
1 n=int(input())
2 ans=[]
3 for i in range(2,n+1):
4     while n % i == 0:
5         ans.append(i)
6         n //= i
7
8 print(*ans)
```

Question 334

Problem G

因數加總

i

Info:

簡單題

```
1 n = int(input())
2 ans = 0
3
4 for i in range(1,n+1):
5     if n%i == 0:
6         ans += i
7
8 print(ans)
```

Question 335

Problem H

數字運算 (減 1 或除以 10)

i

Info:

基礎數學題: 採用 map 函數將輸入的字串轉換成整數

```
1 n,k = map(int,input().split())
2
3 for i in range(k):
4     if n%10 == 0:
5         n//=10
6     else:
7         n -= 1
8
9 print(n)
```

i

Info:

基礎數學題: 採用 [] 串列解析式將輸入的字串轉換成整數

```
1 n,k=[int(n1) for n1 in input().split()]
2 for i in range(k):
3     if n%10==0:n=n//10
4     else:n-=1
5 print(n)
```


Question 336

Problem I

數字相加

i

Info:

```
1 for _ in range(int(input())):
2     n = int(input())
3     A = list(map(int, input().split()))
4     q = []
5     tar = A.pop()
6     for i in range(n-1):
7         if tar - A[i] in q:
8             print('YES')
9             break
10        else:
11            q.append(A[i])
12    else:
13        print('NO')
```

i

Info: 另一種解法: 使用 `itertools.permutations()` 函數生成所有可能的排列組合

```
1 from itertools import permutations
2 for _ in range(int(input())):
3     m=int(input())
4     x=list(map(int, input().split()))
5     tar=x[-1]
6     x=x[:-1]
7     perms=list(permutations(x,2))
8     for i in perms:
9         if sum(i)==tar:
10            print('YES')
11            break
12    else:
13        print('NO')
```

Question 337

Problem J

七段顯示器

i

Info:

```
1 for _ in range(int(input())):  
2     n = int(input())  
3     ans = '1' * (n//2)  
4     if n%2:  
5         ans = '7' + ans[1:]  
6     print(ans)
```

Question 338

Problem K 字串

i

Info:

```
1 for _ in range(int(input())):
2     n = int(input())
3     ans = set()
4     S = list(input())
5     for i in range(n-1):
6         new = S[:]
7         new[i] = ''
8         new[i+1] = ''
9         s = ''.join(new)
10        ans.add(s)
11
12    print(len(ans))
```

Question 339

Problem K2

字串

i

Info: 此題為 Problem K 的進階考題，有時間限制，屬於另一種解法，但是效率較高

```
1 def main():
2     t=int(input())
3     for _ in range(t):
4         n=int(input())
5         s=input()
6         ans=0
7         for i in range(1,n):
8             x=i-2
9             if x<0:
10                 ans+=1
11                 elif s[x]!=s[i]:
12                     ans+=1
13         print(ans)
14
15 main()
```

i

Info: 此題為 Problem K 的進階考題，有時間限制，屬於另一種解法，但是效率較高

```
1 for _ in range(int(input())):
2     n = int(input())
3     ans = set()
4     S = list(input())
5     ans = n - 1
6     # 1
7     i = 0
8     while i < len(S)-1:
9         curr = 1
10        j = i + 1
11        while j < len(S):
12            if S[j] == S[i]:
13                curr += 1
14            else:
15                ans -= max(0, curr - 2)
16                i = j-1
17                break
18            j += 1
19        else:
20            ans -= max(0, curr - 2)
21            i = j
22            i += 1
23
24    # 2
25    for i in range(2, len(S)):
26        if S[i] == S[i-2] and S[i] != S[i-1]:
27            ans -= 1
28
29    print(ans)
```

Question 340

Problem L

中位數

i

Info:

```
1 A = []
2 for i in range(int(input())):
3     A.append(int(input()))
4     n = len(A)//2
5     A.sort()
6     if len(A)%2 ==1:
7         print(A[len(A)//2])
8     else:
9         print((A[n]+A[n-1])/2)
```

Question 341

Problem M

分組

i

Info:

```
1 for _ in range(int(input())):
2     n,x = map(int,input().split())
3     A = list(map(int,input().split()))
4     A.sort(reverse=True)
5     ans = 0
6     p = 1
7     i = 0
8     while i < len(A):
9         if A[i] * p >= x:
10             ans += 1
11             i += p
12         else:
13             p += 1
14             i += 1
15     print(ans)
```

Question 342

Problem N

函數

i

Info:

```
1 n = int(input())
2 if n%2 == 0:
3     print(n//2)
4 else:
5     print(n//2-n)
```


Question 343

Problem O

洗牌

i

Info:

```
1 n,m = map(int,input().split())
2
3 A = list(map(int,input().split()))
4 for i in range(m):
5     new = []
6     a = A[:n//2]
7     b = A[n//2:]
8     for i in range(len(a)):
9         new.append(a[i])
10        new.append(b[i])
11    A = new[:]
12
13 print(*A)
```

Question 344

Problem P

魔術方陣

i

Info:

```
1 A = []
2 total = 0
3 for i in range(3):
4     arr = list(map(int, input().split()))
5     total += sum(arr)
6     A.append(arr)
7
8 total //= 2
9 A[0][0] = total-A[1][0]-A[2][0]
10 A[1][1] = total-A[0][1]-A[2][1]
11 A[2][2] = total-A[0][2]-A[1][2]
12 for i in A:
13     print(*i)
14
15 '''
16 0 1 6
17 3 0 7
18 4 9 0
19 '''
```

程式碼第 13 行:

`print(*i)` 中的星號 `*` 是 Python 的「開箱運算子」(unpacking operator)，Unpacking 顧名思義就是將打包好的資料取出來。

Unpacking Operator(開箱運算子) 分為:

- `*` 符號：可用於任何可疊代的 (Iterable) 物件。
- `**` 符號：只能用於字典 (Dictionary) 物件。

主要用途為分解可疊代的 (Iterable) 物件元素，在進行建立或合併時非常的實用。

舉例: `i` 是一個串列 (例如 `[1, 2, 3]`)，使用 `*i` 可以將串列中的每個元素取出，並分別傳遞給 `print` 函數。這樣可以實現將串列中的元素逐一列印出，並以空格分隔，而不會有串列的中括號 `[]`，若搭配每次迴圈執行時，`print(*i)` 會將當前串列的所有元素列印在同一行，並以空格分隔，這樣就實現了將二維串列中的每一行元素依序列印出的效果。PS:

進階學習: [Python 教學]Python Unpacking 實用技巧分享

Question 345

Problem Q

數字全部相加成本

i

Info:

```
1 while 1:
2     n = int(input())
3     if not n:
4         break
5     A = list(map(int,input().split()))
6     ans = 0
7     while len(A)>1:
8         A.sort()
9         A[0] += A[1]
10        ans += A[0]
11        A.pop(1)
12    print(ans)
```

Question 346

Problem R

樹的直徑

i

Info: 問題分析:

- 樹的直徑: 樹中任意兩個節點之間最長的路徑長度。
- 輸入格式: 第一行為節點數, 接下來的 $n-1$ 行表示邊的連接。
- 輸出格式: 輸出樹的直徑。

解題思路:

- 建立樹的結構: 使用鄰接列表或鄰接矩陣來表示樹的結構, 並根據輸入的節點和邊, 構建每個節點的相鄰節點。
- BFS 函數: 使用 BFS 計算從起始節點出發到其他節點的距離, 並找到最遠的節點及其距離。
- 第一次 BFS: 從節點 1 出發, 找到距離最遠的節點。
- 第二次 BFS: 從第一次找到的最遠節點出發, 再次進行 BFS, 計算出樹的直徑。
- 輸出結果: 最終輸出樹的直徑。

討論:

為何從最遠節點出發可以找到直徑?

當我們第一次 BFS 從隨便選的一個節點 (例如節點 1) 開始時, 可以確保找到的最遠節點 (如 A) 已經在樹的某個「邊界」上。這樣, 再從 A 開始, 因為樹是無環結構, 所以 A 到樹的另一端 (即 B) 的路徑必然是樹的最長路徑, 也就是直徑。

結論:

這兩次 BFS 過程等效於從樹的一端找另一端, 利用無環圖的特性, 保證了找到的路徑是最長的, 也就是樹的直徑。



Info:

方法 1:

```
1 # 定義一個函數來進行 BFS 並找出最遠距離及節點
2 def bfs(start):
3     distance = [-1] * (n + 1)
4     distance[start] = 0
5     queue = [start]
6     head = 0
7     while head < len(queue):
8         node = queue[head]
9         head += 1
10        for neighbor in edges[node]:
11            if distance[neighbor] == -1:
12                distance[neighbor] = distance[node] + 1
13                queue.append(neighbor)
14    max_dist = max(distance)
15    farthest_node = distance.index(max_dist)
16    return max_dist, farthest_node
17
18 # 建立樹的鄰接表表示法
19 edges = {}
20 n = int(input())
21 for _ in range(n - 1):
22     a, b = map(int, input().split())
23     if a not in edges:
24         edges[a] = [b]
25     else:
26         edges[a].append(b)
27     if b not in edges:
28         edges[b] = [a]
29     else:
30         edges[b].append(a)
31
32 # edges = [(1, 2), (1, 3), (3, 4), (3, 5)]
33 # 第一次 BFS 從節點 1 開始
34 _, farthest_from_1 = bfs(1)
35
36 # 第二次 BFS 從第一次找到的最遠節點開始
37 diameter, _ = bfs(farthest_from_1)
38 # 輸出樹的直徑
39 print(diameter)
```

Info:

方法 2:

```

1 def tree_diameter(edges):
2     n = len(edges) + 1
3     graph = {}
4     for u, v in edges:
5         if u not in graph:
6             graph[u] = []
7         graph[u].append(v)
8         if v not in graph:
9             graph[v] = []
10        graph[v].append(u)
11
12    dis_1to = [-1] * (n+1)
13    dis_1to[1] = 0
14    que = [1]
15    head = 0
16    while head < len(que):
17        node = que[head]
18        head += 1
19        for v in graph[node]:
20            if dis_1to[v] == -1:
21                dis_1to[v] = dis_1to[node] + 1
22                que.append(v)
23
24    i = dis_1to.index(max(dis_1to))
25    dis_ito = [-1] * (n+1)
26    dis_ito[i] = 0
27    que = [i]
28    head = 0
29    while head < len(que):
30        node = que[head]
31        head += 1
32        for v in graph[node]:
33            if dis_ito[v] == -1:
34                dis_ito[v] = dis_ito[node] + 1
35                que.append(v)
36    return max(dis_ito)
37
38 # 輸入範例 edges = [(1, 2), (1, 3), (3, 4), (3, 5)]
39 edges = []
40 n = int(input())
41 for _ in range(n-1):
42     a, b = map(int, input().split())
43     edges.append((a, b))
44
45 # 計算並輸出樹的直徑
46 diameter = tree_diameter(edges)
47 print(diameter)

```



Info:

方法 3:

```
1 graph = {}
2 n = int(input())
3 for _ in range(n-1):
4     a,b = map(int,input().split())
5     if a not in graph:
6         graph[a] = [b]
7     else:
8         graph[a].append(b)
9     if b not in graph:
10        graph[b] = [a]
11    else:
12        graph[b].append(a)
13
14 dis_1to = [-1] * (n+1)
15 dis_1to[1] = 0
16 que = [1]
17 head = 0
18 while head<len(que):
19     node = que[head]
20     head += 1
21     for v in graph[node]:
22         if dis_1to[v] == -1:
23             dis_1to[v] = dis_1to[node]+1
24             que.append(v)
25
26 i = dis_1to.index(max(dis_1to))
27
28 dis_ito = [-1] * (n+1)
29 dis_ito[i] = 0
30 que = [i]
31 head = 0
32 while head<len(que):
33     node = que[head]
34     head += 1
35     for v in graph[node]:
36         if dis_ito[v] == -1:
37             dis_ito[v] = dis_ito[node]+1
38             que.append(v)
39
40 print(max(dis_ito))
```

Question 347

Problem S

迴圈

i

Info:

方法 1 解題思路:

- 我們首先構建圖的鄰接表並計算每個節點的度數。
- 然後，我們遍歷所有節點，對於每個未訪問且度數為 2 的節點，使用 DFS 找到其所在的連通子圖。
- 若該子圖中的所有節點度數都為 2，則該子圖為循環圖。
- 最後，返回所有符合條件的循環圖數量。

在此程式碼中，`component` 用於儲存 DFS（深度優先搜尋）探索到的同一連通子圖中的所有節點。其主要用途如下：

1. 確認連通子圖是否為循環圖：

- 當我們找到一個尚未被訪問且度數為 2 的節點時，從該節點執行 DFS，收集與其相連的所有節點，形成一個連通子圖，並將這些節點存入 `component` 列表中。
- DFS 結束後，`component` 中的節點即為這個連通子圖的所有節點。

2. 檢查條件：

- 程式會檢查 `component` 中每個節點的度數是否均為 2，以確認該連通子圖是否滿足「循環圖」的條件。
- 若 `component` 中所有節點的度數都為 2，則此連通子圖為一個循環圖，並將計數器 `cycle_count` 增加 1。

因此，`component` 的作用是幫助我們檢查一個連通子圖是否為循環圖。如果不使用 `component`，我們無法確認 DFS 遍歷的子圖中所有節點的度數是否都為 2。

Info:

方法 1

```

1 from collections import defaultdict
2
3 def count_cycle_graphs(n, edges):
4     # Step 1: Initialize graph and degree count
5     graph = defaultdict(list)
6     degree = [0] * (n + 1)
7     # Step 2: Build the graph and degree count
8     for u, v in edges:
9         graph[u].append(v)
10        graph[v].append(u)
11        degree[u] += 1
12        degree[v] += 1
13    # Step 3: Function to perform DFS and return nodes in a component
14    def dfs(node, visited):
15        stack = [node]
16        component = []
17        while stack:
18            u = stack.pop()
19            if not visited[u]:
20                visited[u] = True
21                component.append(u)
22                for v in graph[u]:
23                    if not visited[v]:
24                        stack.append(v)
25        return component
26    # Step 4: Count cycle graphs
27    visited = [False] * (n + 1)
28    cycle_count = 0
29
30    for i in range(1, n + 1):
31        if not visited[i] and degree[i] == 2:
32            component = dfs(i, visited)
33            # Check if all nodes in the component have degree 2
34            if all(degree[node] == 2 for node in component):
35                cycle_count += 1
36
37    return cycle_count
38
39 # 讀取輸入
40 n, m = map(int, input().split())
41 edges = []
42 for _ in range(m):
43     u, v = map(int, input().split())
44     edges.append((u, v))
45
46 # Output result
47 print(count_cycle_graphs(n, edges))

```

Info:**方法 2**

程式碼解析：尋找無向圖中的循環

程式碼目的

這段 Python 程式碼旨在找出給定無向圖中的循環數量。

核心概念

- **無向圖**：一種圖結構，其中邊沒有方向性。
- **循環 (Cycle)**：一條從一個節點出發，經過若干個節點後又回到起點的路徑。
- **深度優先搜索 (DFS)**：一種圖的搜索算法，從起始節點開始，深度優先地探索圖中的節點。

程式碼解說**1. find_cycles 函數**

- **dfs 遞迴函數**
 - visited 字典：用來標記已訪問過的節點，避免重複訪問。
 - path 列表：用來記錄當前搜索路徑上的節點。
 - cycle_count 變數：用來計數找到的循環數量。
 - 遞迴邏輯：
 - * 將當前節點標記為已訪問。
 - * 將當前節點加入路徑。
 - * 對每個未訪問的鄰居節點遞迴調用 dfs 函數。
 - 循環檢測：
 - * 如果遇到已經在路徑中的節點，且不是起點節點，則找到一個循環。
 - * 檢查循環中的所有節點是否度數都為 2，以確保是簡單循環。
 - * 如果是有效循環，則增加 cycle_count。
 - 回溯：從路徑中移除當前節點，以便繼續探索其他分支。

2. 主程式

- 讀取圖的節點數 n 和邊數 m。
- 建立一個默認字典 graph 來表示圖，其中鍵是節點，值是該節點的鄰接節點列表。
- 對於每條邊，將兩個端點互相添加到鄰接表中。
- 呼叫 find_cycles 函數來計算循環數量。
- 列印結果。

關鍵點

- **DFS 算法**：核心算法是深度優先搜索，逐層深入探索圖的節點。
- **循環檢測**：透過檢查當前節點是否已經在路徑中來判斷是否形成循環。
- **簡單循環驗證**：確保循環中的每個節點的度都為 2，以避免重複計數。
- **回溯**：遞迴函數在探索完一個分支後，需要回溯到上一個節點，繼續探索其他分支。

通過以上步驟，該程式可以有效地找到給定無向圖中的循環數量。



Info:

方法 2

```
1 from collections import defaultdict
2
3 def find_cycles(graph):
4     def dfs(node, visited, path):
5         nonlocal cycle_count
6         visited[node] = True
7         path.append(node)
8
9         for neighbor in graph[node]:
10             if neighbor not in visited:
11                 if dfs(neighbor, visited, path):
12                     return True
13             elif neighbor == path[0] and len(path) > 2:
14                 # 找到一個循環，且所有節點的度都為2
15                 if all(len(graph[n]) == 2 for n in path):
16                     cycle_count += 1
17                 return True
18
19         path.pop()
20     return False
21
22     cycle_count = 0
23     visited = {}
24     for node in graph:
25         if node not in visited:
26             dfs(node, visited, [])
27     return cycle_count
28
29 n, m = map(int, input().split())
30 graph = defaultdict(list)
31 for _ in range(m):
32     u, v = map(int, input().split())
33     graph[u].append(v)
34     graph[v].append(u)
35
36 result = find_cycles(graph)
37 print(result)
```



Info:

方法 3

```
1 n,m = map(int,input().split())
2
3 graph = {i:[] for i in range(1,n+1)}
4 visited = []
5 for i in range(m):
6     a,b = map(int,input().split())
7     graph[a].append(b)
8     graph[b].append(a)
9
10 ans = 0
11 for i in range(1,n+1):
12     if i not in visited:
13         arr = []
14         que = [i]
15         head = 0
16         while head<len(que):
17             node = que[head]
18             head += 1
19             for v in graph[node]:
20                 if v not in visited:
21                     visited.append(v)
22                     que.append(v)
23                     arr.append(v)
24             if arr and all(len(graph[num]) == 2 for num in arr):
25                 ans += 1
26                 # print(arr)
27
28 print(ans)
29
30 '''
31 5 4
32 1 2
33 3 4
34 5 4
35 3 5
36 '''
```