1. statement 程式語言
   1. 正確 : 1.2.3.5.6.7.8.10
   2. 修正
      1. [3] : nested loops
      2. [4] :不會被執行
      3. [9] : Exit改成skip
2. Deep Learning
   1. 正確 : 1.3.4.5.6.7.10
   2. 修正
      1. [2] : 不需要
      2. [8] : 能符合預期
      3. [9] : 可以
3. 分散式系統如何應用在AI

🡪 利用分散式系統可以有效處理大量資料的特性，讓AI模型在進算或是訓練時，可以藉由將資料分散給多個節點電腦一起進行計算後再整理合併的方式，加快訓練與計算的速度，並且其增加容錯率降低單一節點的系統附載

1. 是非與解釋
   1. 是；可以用線性排序演算法，例如Radix Sort進行排序可以在O(n)之時間複雜度完成
   2. 否；此IP為class b之ip，subnet mask為255.255.0.0與題目相同因此只有一個網路，從128.96.開始
   3. 否；(50 > 100 && 50 < 100 || 50!= 50) = 0，(50!= 50 && 50 > 100 || 50 < 100) = 1
   4. 否；此為以遞迴求最大公因數之程式碼，因此gcd(3, 9)應為3不為9
   5. 是；因為以相鄰矩陣儲存相連之邊，因此計算最常路徑需掃描所有邊，有就是o(n^2)
2. Graph
   1. 邊皆為正，因此可用dijkstra algo計算1 to all最短路徑問題，其時間複雜度為O(N+M)即為S(n, m, c)所需時間複雜度，在用kruskal algo，挑出最小邊的同時判斷是否能形成迴圈，其時間複雜度為O(m)，因此可在S(n, m, c)所需時間複雜度內完成
   2. 運用proroity queue根據邊建立最大堆積樹，並從刪除節點有需時間複雜度為n，因此其時間複雜度為O(mlogn)
3. 修改程式碼
   1. int commonArray[] = {} 🡪 陣列初始化
   2. int n = 0;
   3. fibArray[i] == primeArray[j]
   4. commonArray[n] = primeArray[j]
   5. %d, n+1
4. 時間複雜度計算

