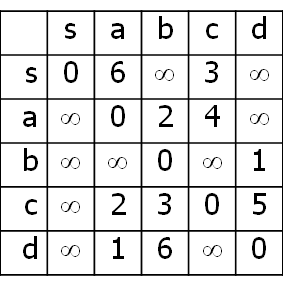
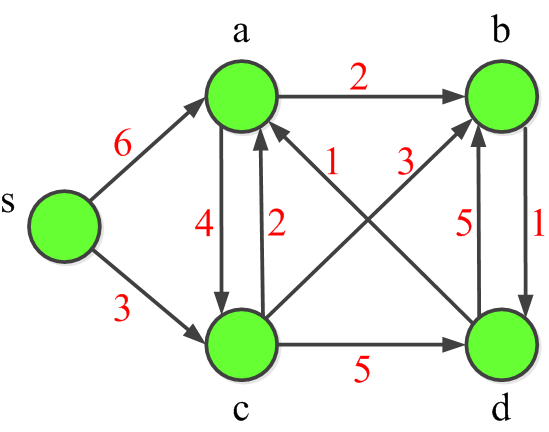
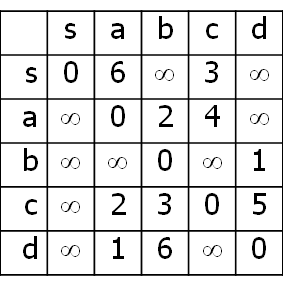
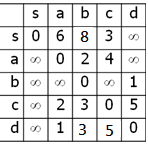
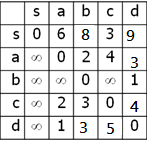
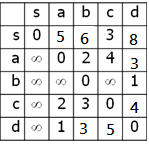
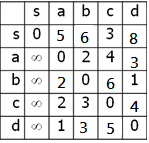
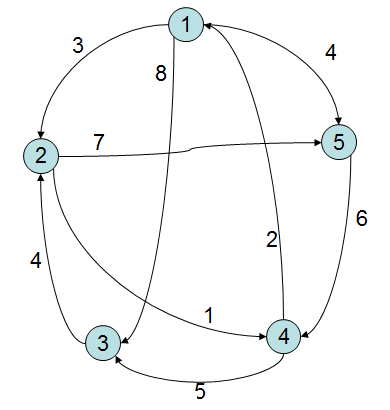
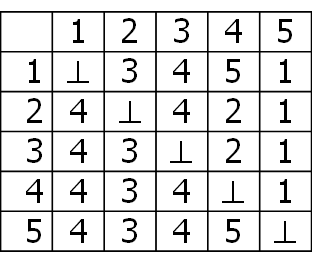
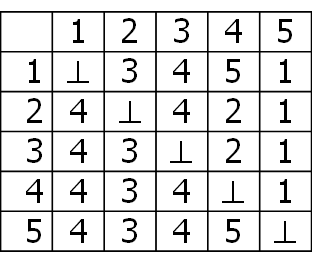
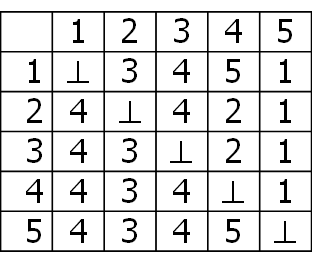
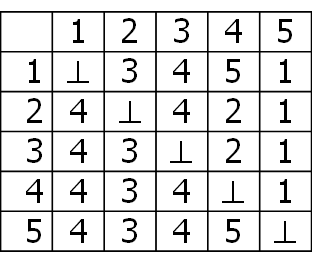
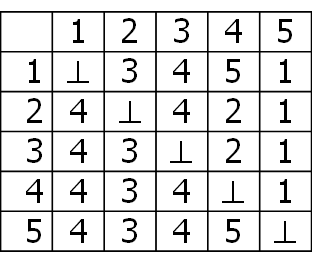
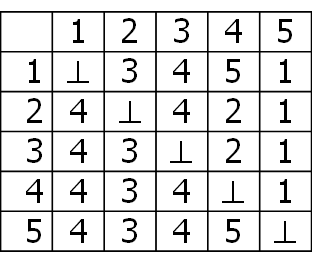
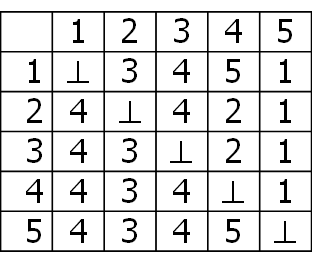
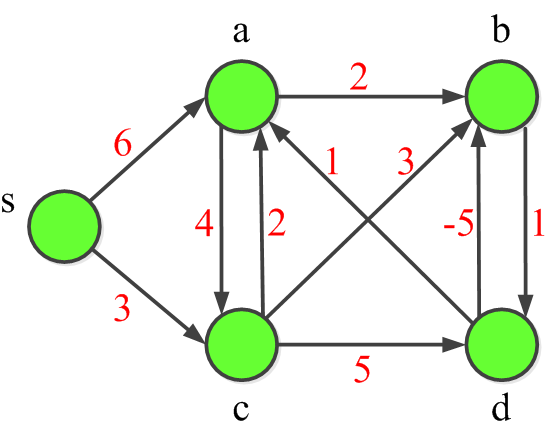
A. 利用Floyd-Warshall演算法求以下圖(graph)全對最短路徑(all-pair shortest path)距離(成本)(此圖的啟始距離矩陣如下，以經過的中間節點為s, a, b, c, d的順序寫出距離矩陣的改變過程。)  
  
中間節點為s:  
  
中間節點為a:  
  
中間節點為b:  
  
中間節點為c:  
  
中間節點為d:  
  
B. 求出以下給定圖(graph)的Floyd-Warshall演算法的啟始前節點矩陣(陣列)，並求出最後的前節點矩陣(陣列)。  
  
C. 以下是Floyd-Warshall演算法針對具有五個節點(記為1、2、3、4、5)的圖產生的前節點矩陣(陣列)，說明如何藉以找出節點1到節點3的 最短路徑，及節點5到節點2的最短路徑。  
  
  
  
  
  
  
  
  
(1)節點1到節點3: 1->4->3  
  
節點1到節點4: 1->5->4  
  
節點1到節點5: 1->5  
  
因此從節點1到節點3: 1->5->4->3  
  
  
  
  
  
  
  
  
  
  
  
(2)節點5到節點2: 5->3->2  
  
節點5到節點3: 5->4->3  
  
節點5到節點4: 5->4  
  
因此從節點5到節點2: 5->4->3->2  
D. 針對以下的給定圖，列出Bellman-Ford最短路徑演算法執行過程， 說明Bellman -Ford最短路徑演算法如何檢查出一給定圖具有負循環(negative-weight cycle)。  
  
iteration 1: d[a] = 6 ,d[c] = 3   
iteration 2 :d[a]=5 (5<6) ,d[b]=6 (6<8) ,d[c]=3 (3<10) d[d]= 8  
iteration 3: d[a] = 5(6<9) , d[b]=3(3<6), d[d] =7 (7<8)  
iteration 4: d[b] = 2 d[d] = 4  
檢查negative cycle: d[b] >d[d]+w[d][b]  
 2 > 4+(-5) //找到最小邊後仍有更小的邊，故判斷產生負循環  
  
E. 證明Bellman-Ford最短路徑演算法可以檢查出一給定圖具有負循環(negative-weight cycle)，也就是累積邊加權為負的循環。  
F. 寫一個演算法來解決最長單調遞增子序列問題(the longest monotonically increasing subsequence problem)，並分析演算法的時間複雜度。  
Algorithm LIS  
Input:n個elements的序列S

Output:LIS長度

length[]初始值為1  
for i=0 to n-1  
 for j=i+1 to n  
 if(s[i] < s[j]) // s[i]後面是否可以接 s[j]  
 length[j] = max( length[j] , length[i]+1)//比較從0~j 和 0~I +1 的長度  
 prev[j] =i //紀錄j接在i後  
 end if  
 end for  
end for  
x = 0  
for i=0 to n  
 x= max(x,length[i])  
end for  
return x  
時間複雜度: (n+1)(n-1)/2 +cn = (n2 -1)/2+cn = O(n2)  
空間複雜度:用length[n]紀錄長度-> O(n)