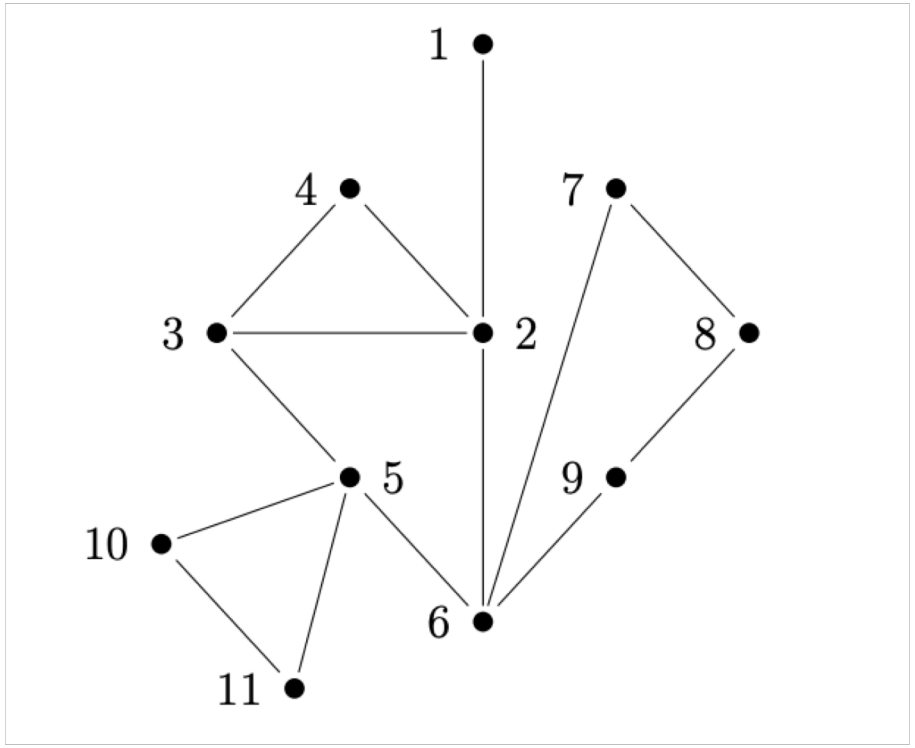
演算法 #9

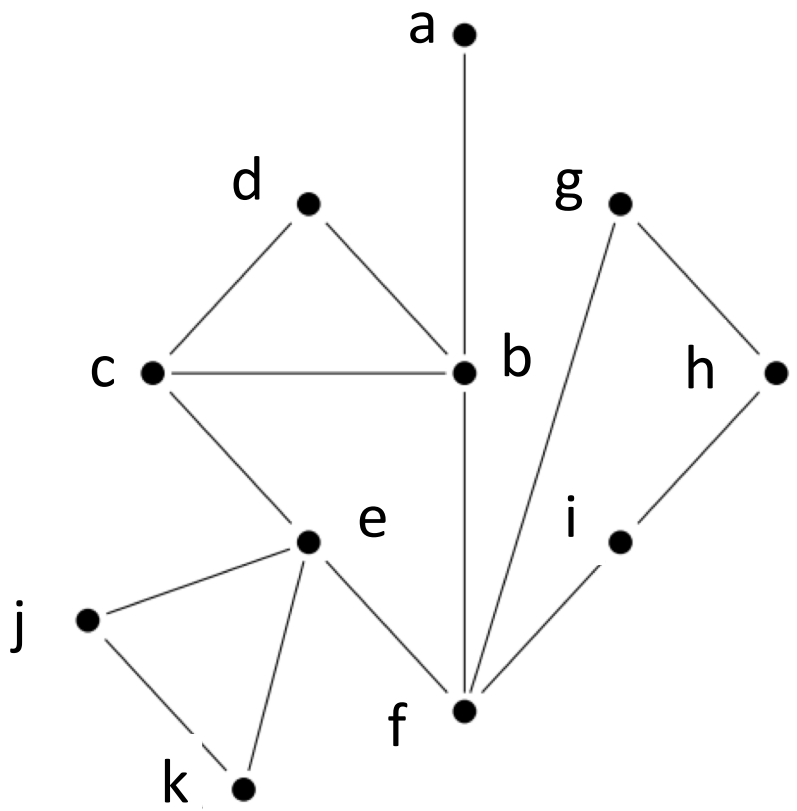
賴冠廷 陳昀靖

Q1.

a) Run the biconnected component algorithm on the graph in Fig. 1. The algorithm should follow the DFS numbers that are given in the figure. Show the High values as computed by the algorithm in each step.



Q1.




```
procedure BC(v) ;

begin
    v.DFS_Number := DFS_N ;
    DFS_N := DFS_N - 1 ;
    insert v into Stack ; { Stack is initially empty }
    v.High := v.DFS_Number ; { initial value }
    for all edges (v, w) do
        insert (v, w) into Stack ;
        { each edge will be inserted twice (for both directions) }
        if w is not the parent of v then
            if w.DFS_Number = 0 then
                BC( w ) ;
                if w.High ≤ v.DFS_Number then
                { v disconnects w from the rest of the graph }
                    remove all edges and vertices from Stack until v is
                        reached, and mark the subgraph they form
                        as a biconnected component ;
                    insert v back into Stack ;
                    { v is part of w's component and possibly others }
                v.High := max ( v.High , w.High )
            else { (v, w) is a back edge or a forward edge }
                v.High := max ( v.High , w.DFS_Number )
end
```

Q1.

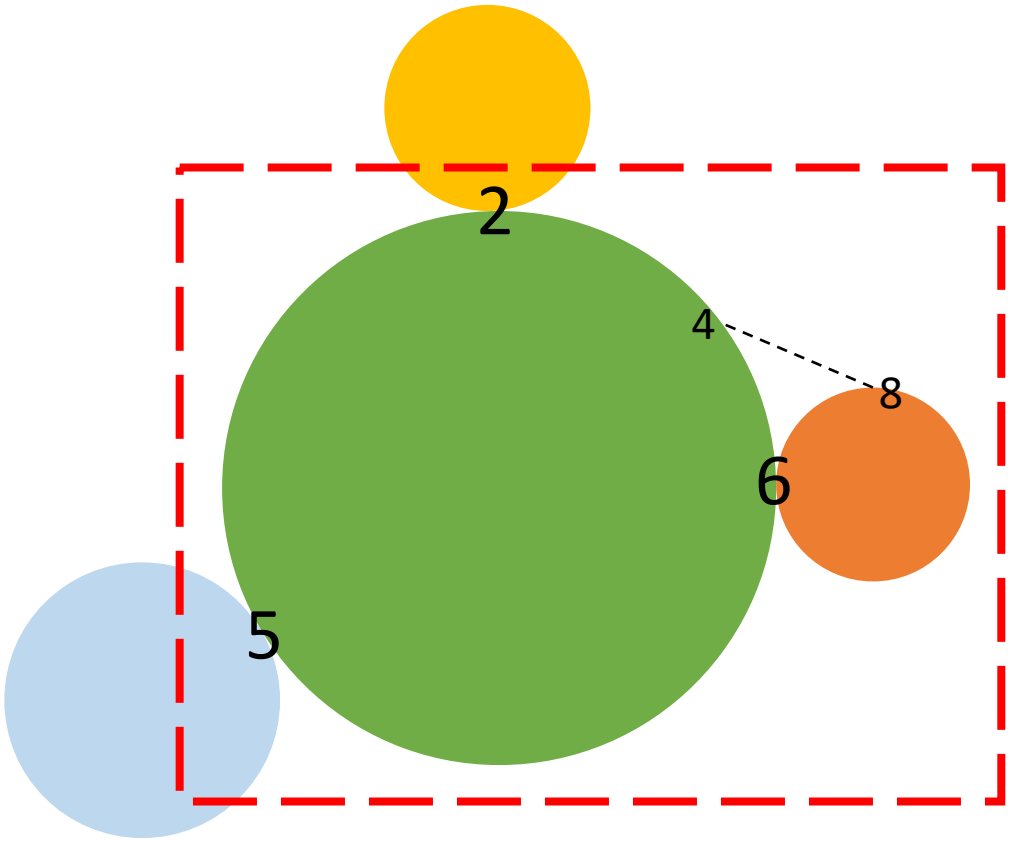
	1	2	3	4	5	6	7	8	9	10	11
1	11										
2	11	10									
3	11	10	9								
4	11	10	9	10							
3	11	10	10	10							
5	11	10	10	10	7						
6	11	10	10	10	7	10					
7	11	10	10	10	7	10	5				
8	11	10	10	10	7	10	5	4			
9	11	10	10	10	7	10	5	4	6		
8	11	10	10	10	7	10	5	6	6		
7	11	10	10	10	7	10	6	6	6		
6	11	10	10	10	7	10	6	6	6		
5	11	10	10	10	10	10	6	6	6		
10	11	10	10	10	10	10	6	6	6	2	
11	11	10	10	10	10	10	6	6	6	2	7
10	11	10	10	10	10	10	6	6	6	7	7
5	11	10	10	10	10	10	6	6	6	7	7
3	11	10	10	10	10	10	6	6	6	7	7
2	11	10	10	10	10	10	6	6	6	7	7
1	11	10	10	10	10	10	6	6	6	7	7

Q1.

b) Add the edge (4,8) to the graph and discuss the changes this makes to the algorithm.

Q1.

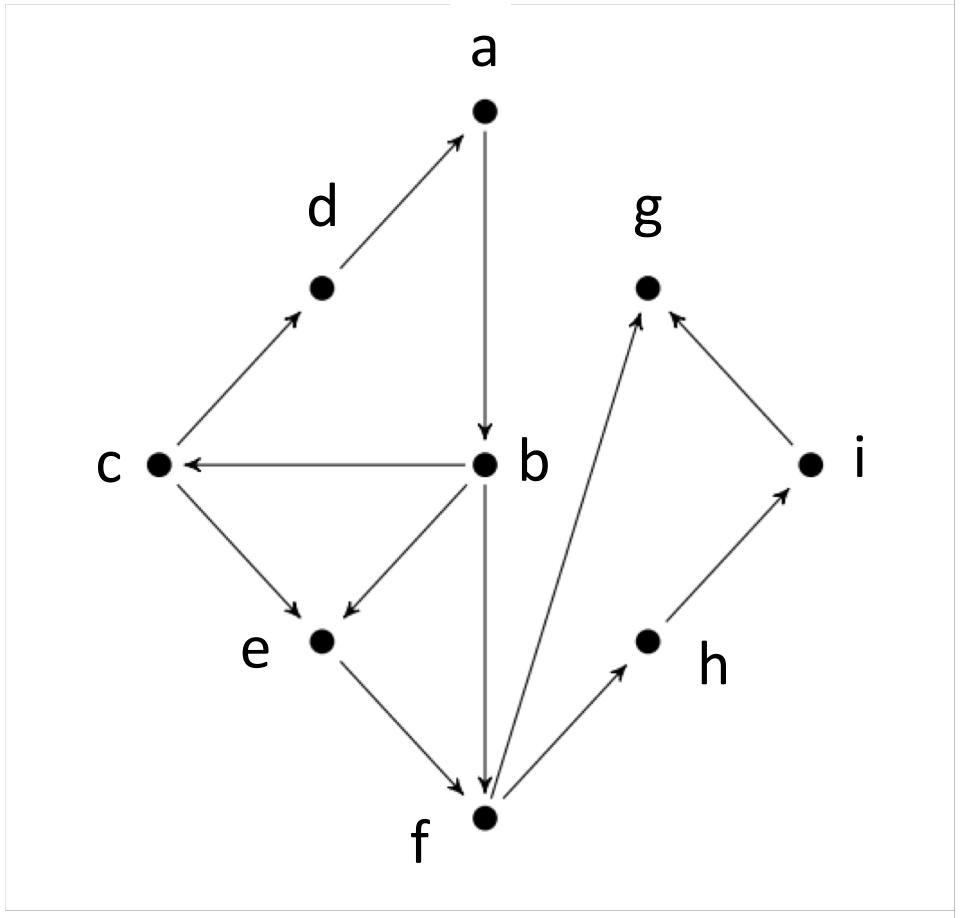
- 6 is no more an articulation point. The (2,5,6,…) BCC and (6,8,…) BCC will merge into one BCC.



Q2.

a) Run the strongly connected component algorithm on the graph in Fig. 2. The algorithm should follow the DFS numbers that are given in the figure. Show the High values as computed by the algorithm in each step.

Q2.




```
procedure SCC(v) ;

begin
    v.DFS_Number := DFS_N ;
    DFS_N := DFS_N − 1 ;
    insert v into STACK ;
    v.High := v.DFS_Number ; { the initial value }
    for all edges (v, w) do
        if w.DFS_Number = 0 then
            SCC(w) ;
            v.High := max ( v.High , w.High )
        else
            if w.DFS_Number > v.DFS_Number and w.Component = 0 then
            { (v, w) is a cross edge or a back edge that we need to consider }
                v.High := max ( v.High , w.DFS_Number ) ;
    if v.High = v.DFS_Number then { v is a root of a component }
        Current_Component := Current_Component + 1 ;
        repeat  { mark the vertices of the new component }
            remove x from the top of STACK ;
            x.Component := Current_Component ;
        until x = v
end
```

Q2.

	1	2	3	4	5	6	7	8	9
1	9								
2	9	8							
3	9	8	7						
4	9	8	7	9					
3	9	8	9	9					
5	9	8	9	9	5				
6	9	8	9	9	5	4			
⑦	9	8	9	9	5	4	3		
6	9	8	9	9	5	4	3		
8	9	8	9	9	5	4	3	2	
⑨	9	8	9	9	5	4	3	2	1
⑧	9	8	9	9	5	4	3	2	1
⑥	9	8	9	9	5	4	3	2	1
⑤	9	8	9	9	5	4	3	2	1
3	9	8	9	9	5	4	3	2	1
2	9	9	9	9	5	4	3	2	1
①	9	9	9	9	5	4	3	2	1

Q2.

b) Add the edge (7,1) to the graph and discuss the changes this makes to the algorithm.

Q2.

- The whole graph will become one SCC.

Q3

- Let $G = (V, E)$ be a connected weighted undirected graph and T be a minimum-cost spanning tree (MCST) of G. Suppose that the cost of one edge $\{u, v\}$ in G is changed (increased or decreased); $\{u, v\}$ may or may not belong to T. Design an efficient algorithm either to find a new MCST or to determine that T is still an MCST. Explain why your algorithm is correct and analyze its time complexity.

Q3.

	$\{u, v\}$ in T	$\{u, v\}$ NOT in T
$\{u, v\}$ increase	• Delete {u, v} edge from T • T becomes two subtrees T1, T2. • Find the minimum cost edge f that connecting T1 and T2. • If f = {u, v} same MCST else add the edge k to T.	Same MCST
$\{u, v\}$ decrease	Same MCST	• Add {u, v} edge to T. • There is a cycle in T. • Find the maximum cost edge k in the cycle. • Delete the edge k from T.

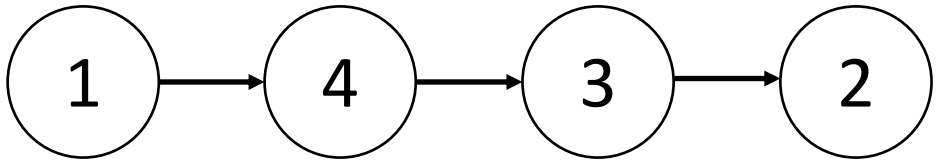
Q3.

	$\{u, v\}$ in T	$\{u, v\}$ NOT in T
$\{u, v\}$ increase	• Find the minimum cost edge f that connecting T1 and T2. • $O(\|V_1\| + \|E_1\|) + O(\|V_2\| + \|E_2\|) + O(E)$	Same MCST
$\{u, v\}$ decrease	Same MCST	• There is a cycle in T. $O(\|V\| + \|E\|)$ • Find the maximum cost edge k in the cycle. $O(\|V\|)$ • Delete the edge k from T.

Q4.

- Algorithm *Improved_Transitive_Closure* given in Algorithm 1 (Fig. 7.24) has three nested loops. The first one (the outer one) chooses a column, the second one chooses a row, and the third one operates on the chosen row. Suppose that we exchange the first two loops such that the first one chooses a row and the second one chooses a column. In other words, we simply exchange the first two lines in the program, as is shown in algorithm *WRONG_Transitive_Closure* in Algorithm 2 (Fig. 7.46). Show that this modification does not work, by giving an example for which the transitive closure is not computed.

Q4.



Initial Matrix

	1	2	3	4
1	F	F	F	T
2	F	F	F	F
3	F	T	F	F
4	F	F	T	F

Correct Matrix

	1	2	3	4
1	F	T	T	T
2	F	F	F	F
3	F	T	F	F
4	F	T	T	F

Wrong Matrix

	1	2	3	4
1	F	F	T	T
2	F	F	F	F
3	F	T	F	F
4	F	T	T	F