CSE 417 HW2 YANG ZHANG

1.

— tree edge
--- back edge

edge orders are listed leftside of edges.

Articulation points : G . H

Vertex	DFS#	LOW
A	4	2
B	3	2
C	1	1
D	7	6
E	8	6
F	5	2
G	2	1
H	6	1
I	9	2
J	10	10

2.

Component 1 : { {A,B}, {A,F}, {B,G}, {F,G} }

Component 2 : { {C,G}, {C,H}, {G,I}, {G,H}, {H,I} }

Component 3 : { {D,E}, {D,H}, {I,E} }

Component 4 : { {H,J} }

3. Since articulation points are the edge of biconnected components, in other words, a graph are divided into biconnected components by articulation points.

Modification to the art. pt. algorithm : At initidization stage, initilice a stack At DFS stage, whenever encounters a new edge (or back edge), push it on the stack until DFS found a articulation point. then pop the stack.

sudo code :

initialization : dfscount = 0 ; v.dfs = -1 for all v ; initialize stack S

```
DFS_bi_comp(v):
    v.dfs = dfscount++
    v.low = v.dfs
    for each edge (v, x)
      if (x.dfs = -1)     // new edge
          S.push((v,x))
          DFS_bi_comp(x)
          v.low = min(v.low, x.low)
          if (x.low >= v.dfs)
                output_bi_comp((v,x))
      else if (x is not v's parent) //back edge
          S.push((v,x))
          v.low = min(v.low, x.dfs)

output_bi_comp((v,x)):
    do
      e = S.pop()
      print e
    while (e != (v,x))
```

Simulation for finding the first bi_comp :

start from C

```
dfs(C):
    C.dfs = 1, C.low = 1
    edge (C, G) new edge, stack {(C,G)}
    dfs(G):
        G.dfs = 2, G.low = 2
        edge(G,B) new edge, stack {(C,G), (G,B)}
            dfs(B):
                B.dfs = 3, B.low = 3
                edge(B,A) new, stack {(C,G), (G,B), (B,A)}
                    dfs(A):
                        A.dfs = 4, A.low = 4
                        edge(A,F) new, stack {(C,G), (G,B), (B,A), (A,F)}
                            dfs(F):
                                F.dfs = 5, F.low = 5
                                edge(F G) back edge
                                stack {(C,G), (G,B), (B,A), (A,F), (F,G)
                                F.low = G.dfs = 2 < A.dfs
                        A.low = F.low = 2 < B.dfs
                B.low = A.low = 2 = G.dfs
                G is articulation point
                output edge up to (G,B) from stack
```

output: (F,G), (A,F), (B,A), (G,B)

which is exactly the 2st bi-comp