Lab06-Graph Exploration

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2019.

∗ If there is any problem, please contact TA Mingran Peng.

∗ Name:田雪飞 Student ID:515030910347 Email: 13487426939@qq.com

1. **Solution.** The following is the process of solution.
(a). Answer is 6 and the ccs.cpp program file is in the txf_homework_06 file.
(b). The ccs.gephi file is in the txf_homework_06 file.

□

2. **Proof.** The following is the process of proof.
If we use dfs to traverse a graph, that means we visit the vertex by using a stack.
Proof by contradiction:
Assume: if two intervals are not disjoint or one is contained within the other. and we visit u before v. we know : $PRE[u] < PRE[v]$ and $POST[u] < POST[v]$ but $POST[u] > PRE[v]$ and $POST[v] > POST[u]$ According to the rule of stack. when we visited the u at first time,that means I should push the $PRE[u]$ into the stack and if there no other vertex connect to u,the we call $POST[u]$ when we visit u, then we pop vertex u out of stack. but in the case, we push $PRE[v]$ into stack before we call $POST[u]$;Then the stack exist $PRE[u]$ and $PRE[v]$, meanwhile $PRE[u]$ is on $PRE[v]$. so $PRE[u]$ must waiting for $PRE[v]$ being popped out of stack, so we call the $POST[v]$ before $POST[u]$. so $POST[v] < POST[u]$, this is contradictory to $POST[v] > POST[u]$.
So,$\forall u, v \in V$, intervals $[PRE(u), POST(u)]$, $[PRE(v), POST(v)]$ are either disjoint or one is contained within the other.

□

3. **Solution.** The following is the process of solution.
In the case. we can change this question to the shortest path problem, and two computers communicate with a constant time t. So, we can use the BFS to solve this problem. The following is the pseudo code.

Algorithm 1: The shortest path.

Input: computer s and computer t, and an undirected Graph $G = (V, E)$ represent the relation of computer(connected(1) and unconnected(0)). vertex s ϵ **V**.
Output: The shortest time needed to send message between two computers.

for ***each*** $u \;\epsilon$ ***V*** **do**
  $DIST(u) = \infty$;
  $visited[i] = false$;
$DIST(s) = 0$;
$visited[s] = true$;(Mean the point is visited.)
$Q.PUSH(s)$;(Using a queue Q to do BFS.)
while *Q is not empty* **do**
  $u = Q.POP()$;
  for ***each*** *(u,v)* ϵ ***E*** **do**
    if $DIST(v) = \infty$ ***and*** $visited = false$ **then**
      $Q.PUSH(v)$;
      $visited[s] = true$;
      $DIST[v] = DIST(u) + \mathbf{t}$;
    if $v = \boldsymbol{t}$ **then**
      return *DIST[v]*
return *unconnected*

Time complexity:
(1) Best case: if **s** connected with **v** or connected by a constant k vertex, then time complexity is o(1).
(2)Worst case:if there is no path from **s** to **t**, then we must traverse all vertex and edges.then time complexity is o(V+E).
(3) if two computers connect by k vertex, k=0,1,2,3...n-2; and probability is $\frac{1}{n-1}$. then time complexity is also o(V+E).

□

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.