CS 170 Homework 3

Due **2025/3/1, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

Solution: I worked on this homework with the following collaborators:

- none,which is only me,Sillycheese

2 Depth First Search

(a) (4 points) In each of the following cases, PreVisit and PostVisit have been defined for you. After execution, the array A[v] will hold a value for each vertex v. Describe in words what A[v] represents.

i. Describe in words what A[v] represents.

Solution: this is the lognest path form root of subtrees to leaf.

ii. Describe in words what A[v] represents.

Solution: this is maximum degree.

(b) (6 points) In each of the following cases, write down pseudocode for PreVisit and PostVisit routines to perform the computation needed.

i. For each vertex v, compute the maximum weight of an edge along the path from root r to vertex v and store it in array A[v].

Solution:

```
procedure PreVisit(u, v)
A[v] ← max(A[u], w(u, v))


procedure PostVisit(u, v)
return
```

ii. For each vertex v, compute the maximum weight of any edge in the subtree rooted at vertex v and store it in array A[v].

Solution:

```
procedure PreVisit(u, v)
return
```

```
procedure PostVisit(u, v)
A[u] ← max(A[u],A[v],w(u, v))
```

iii. For each vertex v, compute the maximum pre-order number of any of its children and store it in array A[v]. If v has no children, then A[v] should be 0.

Solution:

```
procedure PreVisit(u, v)
t← t+1
A[u] ← t


procedure PostVisit(u, v)
t← t+1
```

3 Biconnected Components

(a) Suppose that |V | 2. Can you always find a vertex v V that is not critical? What about an edge that is not critical?

Solution: DFS,and find that leaf

if all Vs are critical

(b) Give a linear time algorithm to find all the critical edges of G.

Solution: PreVisit , and maintain a low value. if pre(v) < low(n), then u-v is critical.

(c) Modify your algorithm above to find all the critical vertices of G.

Solution: DFS

4 Topological Sort Proofs

Solution: SKIP!all is because of my Ph.D Exam coming:(

5 Distant Descendants

(a) Write an O(|V |) algorithm that computes the total size of the subtree (number of descendants plus 1 for the vertex itself) of each vertex v in an array s[v]. Give a brief justification that your algorithm is correct and runs in O(|V |) time. Do not just cite an algorithm from class; reproduce anything you use in your solution.

Solution: To find the size of subtree. we need to DFS(v),and maintain s[u]=s[u]+s[v] recursively.

(b) Write an O(|V |) algorithm that computes the K-th level ancestor of each vertex v (null if the depth of v is less than K) in an array a[v]. Give a brief justification that your algorithm is correct and runs in O(|V |) time. Make sure your algorithm runs in O(|V |) time and not O(K|V |) time.

Solution: say a array called a to note the members of ancestors.

if len(a) > K,then return a[len(a)-1-k]

else return null

then recursively loop.

for (u,v) in G
$a.push_back(v)$
DFS(v)
$a.pop_back(v)$
clean the path for the next search.
remember to set a visited array to optimize it!

(c) Write an O(|V |) algorithm to compute d[v] for each vertex v using s[v] and a[v]. Give a brief justification that your algorithm is correct and runs in O(|V |) time.

Solution: if a[v] exists
d[a[v]]=s[v]+d[a[v]]
loop and loop