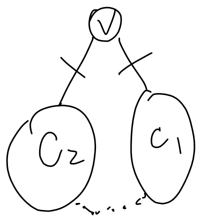
5.

(1) **(3 points)** Let T be a DFS tree of G with root $r \in V$. That is, $T = (V, E_T)$, where E_T is a subset of E, connecting all vertices in G. Given T, please briefly describe how to find $s(r,T)$ in $O(1)$ time complexity and **prove the correctness**.

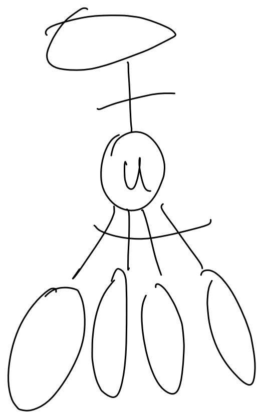
(1) $deg(v)$ 即為答案



若 remove v , C_1 C_2 為原與 v 相連的 component

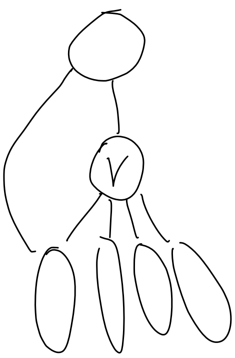
if $C_1.C_2$ 有連接，根據 Tree def. C_1, C_2 會是同一 Component

(2) $deg(u)$ 即為答案



我們可以把 u 的 edge 分為連接父親以及連接 child，其中連接父親只會有一條，child 的部則是和(1)小題一樣，所以是 $deg(u)$

(3) # of $up_T(w_i) = depth(v)$, $1 \le i \le k$



如果 $up_T(w_i) < depth(v)$, exist back edge to ancestor of v , are the same component if remove v.

if $up_T(w_i) = depth(v)$, same rule to (1) and (2)

(since $(w_i, v) \in E$, $up_T(w_i) \not> depth(v)$)

(4) observing from (3) we define $Low_T(v)$

$Low_T(v) = \min\{ depth_T(v), Low_T(w) \mid (u,w)$ is a back edge for $u \in$ descendant of $v \}$

which is the minimum Low that the subtree of v and it's neibor's depth

first DFS for the DFS Tree and $depth_T(v)$

then use DFS again but when the precedure return is the $Low_T(w) = Low_T(v)$ the count++ for v, note that w is child of v

and if (v,w) is a back edge just update the $Low_T(v)$ if $depth_T(w)$ is smaller.

Finally, we have the count $= s(v,G)$ for $v \in V$ exept the root we use for the first DFS, and it is easy to get $s(r,G)$ by (1). $O(|V|+|E|)$

6.

(1) kruskal algorithm but adding edges in decreasing order. $O(E \log V)$

(2) If there's another path that width is larger then there exist $e \in E$ in max spanning tree s.t. $w(e) < w(e')$, for e' in another path then $e \notin$ max spanning tree

(3) ($\Rightarrow$) Suppose (u,v) is a downwards critical

$\exists w$, that shortest path from s to w get smaller when (u,v) get smaller

implies that the shortest path include (u,v) or (v,u)

s — u — v —— w

by the property of shortest path, shortest path from s to v is include in s to w. so s to v go though u. implies it ends at (u,v)

($\Leftarrow$) suppose there's shortest path s to v end at (u,v). then when decreasing (u,v). the shortest path decreased implies (u,v) is downwards critical

(4) road $(u, v) \in E$ is upwards critical if and only if there is a unique shortest path from s to u or v that ends at (u,v).
(i.e. for all other path to s to v(u) end with other edge does not form a shortest path)

($\Rightarrow$)
suppose (u,v) is upwards critical
$\exists$ w, that shortest path from s to w get larger when (u,v) get larger
implies that the shortest path include (u,v) or (v,u)

is longer

s — u — v —— w

by the property of shortest path, shortest path from s to v is include in s to w. so s to v go though u. implies it ends at (u,v)

($\Leftarrow$)
suppose there's unique shortest path s to V end at (u,V). then when increasing (u,V). the shortest path increased implies (U,V) is upwards critical

(5) Dijstra algorithm twice
diffirent is that when relaxing a vertex V to U. comparing
$(d(s,u), d(s,v)+(v,u))$

if they have the same value. add (v,u) to a temporary downward for U
else if d(s,u) is larger, then clear the temporary downward of U and add (v,u) in it.
When U got pop from the priority queue

The edges in temporary downward of u are downward critical.

for upward.

comparing

$(d(s,u), d(s,v)+(v,u))$

if $d(s,u)$ is larger then

set (v,u) as a temporary upward

else if the values are same

we clear the temporary upward for u if there is one.

When u pop form the priority queue

The temporary upward for u is a upward critical if there is one.