Problem 5

> Refs:
>
> None

(1)

Algorithm

$$s(r, T) = \deg(r) \text{ in } E_T$$

Time Complexity

Finding $\deg(v)$ with adjacency list for any vertex v uses $O(1)$ time.

Correctness

For any vertex v such that edge $(r, v) \in E_T$, the sub-tree with root at v is a connected component after the removal of r. And since T is a tree, each of these sub-trees isn't connected to each other without r. Therefore the number of connected components after removing r is the degree of r in E_T.

(2)

Algorithm

$$s(v, G) = 2$$

Time Complexity

Since it's always 2, it takes $O(1)$ time.

Correctness

Because there are no edges from descendants of v to ancestor of v, and G is undirected, all edges between descendants of v are tree edges, and they form a connected component after removing v. All the other vertices are connected to r, and they form the second connected component after removing v.

(3)

Algorithm

$$s(v, G) = |\{\mathrm{up}_T(w_t) \mid \mathrm{up}_T(w_t) = \mathrm{depth}(v) \text{ and } 1 \leq t \leq k\}| + 1$$

Correctness

Because G is undirected, every edge is either a tree edge or a back edge. This means the vertex that $\mathrm{up}_T(w_k)$ implies is either v or an ancestor of v.

If this vertex is v, $\mathrm{up}_T(w_k) = \mathrm{depth(v)}$, and after removing v, w_k and its descendants forms a connected component themselves.

If this vertex is an ancestor of v, $\mathrm{up}_T(w_k) < \mathrm{depth(v)}$, and after removing v, w_k and its descendants joins the connected component of ancestors of v.

Therefore $s(v, G)$ is the number of w_k such that $\mathrm{up}_T(w_k) = \mathrm{depth(v)}$ plus the connected component of ancestors of v.

(4)

Algorithm

1. Choose any vertex as root r. Run DFS from r.
    1. During DFS, maintain the depth of each vertices in the DFS tree with: $v.\mathrm{depth} = v.\pi.\mathrm{depth} + 1$
    2. After visiting every neighboring vertices of v, calculate $\mathrm{up}_T(v)$ with: $\mathrm{up}_T(v) = \min_{w \in W,\, u \in U}(\mathrm{up}_T(w), u.\mathrm{depth})$, where W is the set of all children of v and U is the set of all neighbors of v.
    3. Also calculate $s(v, G)$ with the method in (3) if $v \neq r$.
2. For r, $s(r, G) = \deg(r)$ in DFS tree.

Time Complexity

- Initialization for DFS takes $O(|V|)$ time.
- For each vertex v:

    - Visiting every neighbor vertices v takes $O(|E_v|)$ time. E_v is the set of edges with v at one end.
    - Calculating $s(v, G)$ and $\mathrm{up}_T(v)$ takes $O(|E_v|)$ time.
- And calculating $s(r, G)$ takes $O(|E_r|)$ time (checking all neighboring vertices' predecessor).
- Total time complexity is $O(|V|) + \Sigma_{v \in V}\ O(|E_v|) = O(|V|) + O(2 \cdot |E|) = O(|V| + |E|)$

Correctness

$\mathrm{up}_T(v)$ can be calculated by:

$$\mathrm{up}_T(v) = \min_{w \in W,\, u \in U} (\mathrm{up}_T(w), u.\mathrm{depth})$$

where W is the set of all children of v and U is the set of all neighbors of v. Because $\min_{w \in W} \mathrm{up}_T(w)$ includes all neighbors of "descendants of v excluding v", only neighbors of v haven't been considered.

$s(w, G)$ and $\mathrm{up}_T(w)$ are calculated after visiting w. $u.\mathrm{depth}$ is also calculated after visiting all children because u must be either v's ancestor or descendant (no cross edge in undirected graph). Therefore, all needed values for $s(v, G)$ and $\mathrm{up}_T(v, G)$ are available during calculation.

For root r, because there are no cross edges, $s(r, G)$ is number of sub-trees, which is also $\deg(r)$.