1 (easy) Suppose that a root $x$ in a Fibonacci heap is marked. Explain how $x$ came to be a marked root.

Solution Note that while we the root nodes to always be unmarked, the nodes can be marked. Specifically, a root node is marked if some of its children have been promoted.

2 (easy) Give a sequence of $m$ MAKE-SET, UNION and FIND-SET operations, $n$ of which are MAKE-SET operations, that takes $\Omega(m \log n)$ time when we use union by rank only.

Solution Perform $n$ MAKESET operations. Then, using $n - 1$ UNION operations, create a binomial tree of degree $\lceil \log_2 n \rceil$. Recall that a binomial tree of degree $k + 1$ is formed by taking the UNION of any two elements in different binomial trees of degree $k$. For example, UNION(1,2), UNION(3,4), UNION(1,3) creates a binomial tree of degree 2.

The resulting binomial tree has one node at depth $\lceil \log_2 n \rceil$. Say it has value $k$. Then the last $m - n - \lceil \log_2 n \rceil + 1$ operations of the sequence are FIND-SET($k$). Each of these takes time proportional to $\lceil \log_2 n \rceil$. The total time for the entire sequence is bounded below by the time to perform all the FIND-SET operations, which is in $(m - 2n)(\log n)$. Provided $m$ is sufficiently large, i.e. $m - 2n \in \Omega(n)$, it follows that $(m - 2n) \log n = \Omega(m \log n)$, as desired.

3 (easy) Give an efficient algorithm to determine if an undirected graph is bipartite.

Solution An undirected graph is bipartite if its vertices can be divided into two sets $V_0$ and $V_1$ such that every edge goes between these sets. To determine this we will classify the vertices into $V_0$ and $V_1$ during breadth-first search. After choosing the class of the root, all other choices will be forced in the connected component of the root. If therefore we observe an edge between vertices in the same class then the graph will not be bipartite.

4 (hard) The diameter of a tree $T = (V, E)$ is given by

$$\max_{u,v \in V} \delta(u, v)$$

that is, the diameter is the largest of all shortest-path distances in the tree. Give an efficient algorithm to compute the diameter of a tree, and analyze the running time of your algorithm. $|V|$ is the number of nodes in $T$.

Solution Run BFS on any node $s$ in the graph, remembering the node $u$ discovered last. Run BFS from $u$ remembering the node $v$ discovered last. $d(u, v)$ is the diameter of the tree.

Correctness: Let $a$ and $b$ be any two nodes such that $d(a, b)$ is the diameter of the tree. There is a unique path from $a$ to $b$. Let $t$ be the first node on that path discovered by BFS.
If the paths $p_1$ from $s$ to $u$ and $p_2$ from $a$ to $b$ do not share edges, then the path from $t$ to $u$ includes $s$ so
\[
\begin{align*}
d(t, u) &\geq d(s, u) \\
d(t, u) &\geq d(s, a) \\
d(t, u) &\geq d(t, a) \\
d(b, u) &\geq d(b, a)
\end{align*}
\]

Since $d(a, b) \geq d(u, b)$, $d(a, b) = d(u, b)$.

If the paths $p_1$ and $p_2$ do share edges, then $t$ is on $p_1$. Since $u$ was the last node found by BFS, $d(t, u) \geq d(t, a)$. Since $p_2$ is the longest path, $d(t, a) \geq d(t, u)$. Thus $d(t, a) = d(t, u)$ and $d(u, b) = d(a, b)$.

$d(a, b) \geq d(u, v)$ and $d(u, v) \geq d(u, b)$ so all three are equal. Thus $d(u, v)$ is the diameter of the tree.

5 (easy) Give a counterexample to the conjecture that if there is a path from $u$ to $v$ in a directed graph $G$, and if $d[u] < d[v]$ in a depth-first search of $G$, then $v$ is a descendant of $u$ in the depth-first forest produced.

Solution

We perform a DFS starting at vertex $s$. We then discover vertex $u$. Since the only edge out of $u$ is $(u, s)$, and $s$ has been found, we finish $u$. Next, we discover and finish $v$. Finally, we finish $s$. 