CS577: Introduction to Algorithms (Spring 2022)

Discussion 2 Solutions

Problems

1. Order the following functions according to the asymptotically smallest to the asymptotically largest. Assume the base of the logarithm is 2.
   - $\sqrt{5^{\log n}}$
   - $\log(5^n)$
   - 5^n
   - $\sqrt{n}$
   - $n^{\log n}$
   - 3^{n+10}
   - $\log n$
   - $(\log n)^n$
   - $5^{\sqrt{\log n}}$

Solution:

The correct answer (using $\precsim$ to denote the relation "asymptotically smaller than") is:

$$\log n \precsim 5^{\sqrt{\log n}} \precsim \sqrt{n} \precsim \log\left(5^n\right) \precsim \sqrt{5^{\log n}} \precsim n^{\log n} \precsim 3^{n+10} \precsim 5^n \precsim (\log n)^n$$

To find this ranking, we start by giving a ranking of the relatively simpler functions:

$$\log n \precsim \sqrt{n} \precsim \log\left(5^n\right) = n \log 5 \precsim 3^{n+10} \precsim 5^n \precsim (\log n)^n$$

Now we are left to find the positions of $n^{\log n}$, $\sqrt{5^{\log n}}$, and $5^{\sqrt{\log n}}$.

1. $n^{\log n}$

   Let's compare the asymptotic growth of $n^{\log n}$ to c^n for a fixed (positive) constant c. This comparison is not affected by taking the logarithm of both functions (that is, we can compare the logarithms instead of the original functions).

$$\log(c^n) = n \log c$$
$$\log(n^{\log n}) = (\log n)(\log n) = (\log n)^2$$

   and since $(\log n)^2 \precsim n \log c$, we conclude $n^{\log n} \precsim c^n$. Thus $n^{\log n} \precsim 3^{n+10}$. For a lower bound, clearly $n \log 5 \precsim n^{\log n}$.

2. $\sqrt{5^{\log n}}$

$$\sqrt{5^{\log n}} = (5^{\log n})^{\frac{1}{2}} = 5^{\frac{1}{2}\log n} = (2^{\log 5})^{\frac{1}{2}\log n} = 2^{\log(n^{\frac{1}{2}\log 5})} = n^{\frac{1}{2}\log 5}$$

   Since $\log 5$ is between 2 and 3 (5 is between 2^2 and 2^3), $\frac{1}{2}\log 5 > 1$ and thus $n \log 5 \precsim \sqrt{5^{\log n}}$, and on the other hand $\frac{1}{2}\log 5 < 1.5$ and so $\sqrt{5^{\log n}} \precsim n^{1.5} \precsim n^{\log n}$.

3. $5^{\sqrt{\log n}}$

   First we compare to $\sqrt{n}$. We do this by taking logarithms (base 5) of both functions:

$$\log_5 \sqrt{n} = \frac{1}{2}\log_5 n$$
$$\log_5 5^{\sqrt{\log n}} = \sqrt{\log n}$$

   and since $\sqrt{\log n} \precsim \frac{1}{2}\log_5 n$, we conclude $5^{\sqrt{\log n}} \precsim \sqrt{n}$. Now we compare to $\log n$. Again we take the logarithm base 5 of $\log n$, and since $\log_5 \log n \precsim \sqrt{\log n}$ (if this is not clear, try substituting $n = 2^m$) we conclude that $\log n \precsim 5^{\sqrt{\log n}}$.

2. *Kleinberg, Jon. Algorithm Design (p. 110, q. 9).* There's a natural intuition that two nodes that are far apart in a communication network — separated by many hops — have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes.

   Suppose that an n-node (connected) undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $\frac{n}{2}$.

   (a) Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all $s-t$ paths. (In other words, the graph obtained from G by deleting v contains no path from s to t.)

   Solution:

   Let $d(s, v)$ be the distance of a node v from s. Since $d(s, t) > n/2$, any path from s to t must contain a node at each distance within $1, \ldots, \lfloor n/2 \rfloor$ from s. Now, for each distance $k \in \{1, \ldots, \lfloor n/2 \rfloor\}$, we ask *how many* nodes are distance k away from s, that is, how many nodes v are there such that $d(s, v) = k$. Suppose towards contradiction that for each k, there are at least two nodes that are distance k from s. Then the graph would have at least $2 \cdot \lfloor n/2 \rfloor = n$ nodes not including s and t, which is a contradiction because we know the graph has only n nodes total. Therefore there must be some distance $k \in \{1, \ldots, \lfloor n/2 \rfloor\}$ such that there is only one node v which is distance k from s, and thus if we remove v, there cannot be any paths from s to t (by our initial observation that any path from s to t would in particular have to pass through a node which is distance k from s, since $k \leq \lfloor n/2 \rfloor < d(s, t)$).

   (b) Give an algorithm with running time $O(m + n)$ to find such a node v.

   Solution:

   Starting at s, run breadth-first search on the graph.

   Before starting the exploration of a new layer, create a node count and initialize to 0. Every time a node is added to the layer, increment the node count.

   At the end of the layer exploration, if the count is 1, the node in that layer is the node to be removed.

   If not, continue in the same manner.

   Since the work added is less significant than the work the BFS algorithm is doing, this algorithm runs in $O(m + n)$ time.