HSYLC Homework 1 Solutions

1) $f(n)$ is $O(g(n))$

$\Rightarrow$ exists constants c_1, N_1 such that

$$0 \le f(n) \le c_1 g(n) \quad \text{for all } n \ge N_1$$

$g(n)$ is $O(h(n))$

$\Rightarrow$ exists constants c_2, N_2 such that

$$0 \le g(n) \le c_2 h(n) \quad \text{for all } n \ge N_2$$

$$\Rightarrow \quad 0 \le c_1 g(n) \le c_1 c_2 h(n) \quad \text{for all } n \ge N_2$$

$$\Rightarrow \quad 0 \le \underline{f(n)} \le c_1 g(n) \le \underline{c_1 c_2 h(n)} \quad \text{for all } n \ge \max(N_1, N_2)$$

$\Rightarrow$ $f(n)$ is $O(h(n))$.

2) You don't necessarily have $O(1)$ random access to elements in a stack or queue with minimal functionality

3) In the worst case, our search space decreases by a factor of $2/3$ each time, so the number of times we search before we get to a size of ≈ 1 is roughly k, where $\left(\frac{2}{3}\right)^k n = 1 \Rightarrow k = \log_{3/2} n$. Since $\log_2 n = (\log_2 3/2) \log_{3/2} n$, this is the same asymptotic complexity of normal binary search, $O(\log n)$.

HSYLC Homework 2 Solutions

1) By the Master Theorem, we have the following:

a. $a = 3,\ b = 2,\ k = 2 \Rightarrow a < b^k$

$\Rightarrow T(n) = O(n^2)$

b. $a = 7,\ b = 2,\ k = 1 \Rightarrow a > b^k$

$\Rightarrow T(n) = O(n^{\log_2 7})$

2) DFS: 1, 2, 4, 3, 6, 7, 5

BFS: 1, 2, 3, 4, 5, 6, 7

3) Use the shortest path algorithm (application of BFS) to find the distance from the source to all vertices, then take the maximum over all such distances.

(since it is a tree, there is only one path between any two vertices)