Prob. 1	Prob. 2	Prob. 3

Problem 1.

1. In the worst case, we insert the smallest number into tree, so we need to swap from the leaf to the root in *bubble-up* way. That results in $\lfloor \log(n) \rfloor$. In the worst case after the delete, we replace the root with the largest value then we need to swap from the root to the leaf in *bubble-down* way. That results in $\lfloor \log(n) \rfloor$.

2. For amortized analysis, we can sum depths of all nodes as a potential function. Therefore, since tree is balanced, we sum *logarithms* of all nodes and we have the following for n items of a heap:

$$\Phi = \sum_{i=2}^{n} \log i$$

insert operation has a logarithmic amortized cost:

$$\textit{amortized cost} = \textit{actual cost} + \Delta\Phi = \log(n) + \log(n+1) \equiv O(\log n)$$

However, *delete* operation has better amortized cost:

$$\textit{amortized cost} = \textit{actual cost} + \Delta\Phi = \log(n) - \log(n) \equiv O(1)$$

Problem 2.

Problem 3.