1. More buildHeap proof for those who are curious. Suppose we have a binary tree of size $N$, and suppose the tree is completely filled in. Then $N = 2^k - 1$, where the height of the tree is $k - 1$. (If you’re not sure about this, think about some small examples, e.g. $N = 7$). We established in class today that the summation for operations required in buildHeap is

$$\sum_{i=1}^{k-1} (k - i)2^{i-1}$$

This is because at the top level (i.e. $i = 1$) we could percolate down the entire tree, the height of which is $k-1$; but we’d only have to do that for one item. At the second level (i.e. $i = 2$) we could percolate down the entire tree again, but from the second level, the height of which is $k-2$. However, we might have to do the percolation for both items at the second level. And so on, so that at the $i$th level, we could percolate down the height of the tree $(k - i)$ and we may have to do that for the $2^{i-1}$ items at the $i$th level.

We solve the summation as follows:

$$S = \sum_{i=1}^{k-1} (k - i)2^{i-1} = (k - 1)(1) + (k - 2)(2) + (k - 3)(4) + \ldots + (2)(2^{k-3}) + (1)(2^{k-2})$$

Now let’s look at multiplying $S$ by 2 and subtracting $S$ from 2$S$.

$$2S = (k - 1)(2) + (k - 2)(4) + \ldots + (2)(2^{k-2}) + (1)(2^{k-1})$$

$$-S = -(k - 1)(1) - (k - 2)(2) - (k - 3)(4) - \ldots - (1)(2^{k-2})$$

$$2S - S = -(k - 1)(1) + 2 + 4 + \ldots + 2^{k-2} + (1)(2^{k-1})$$

$$S = -k + 1 + 2 + 4 + \ldots + 2^{k-2} + 2^{k-1}$$

So $S = -k + \sum_{i=0}^{k-1} 2^i$. From page 4 in Weiss, that sum is $2^{k+1}-1$, thus $S = 2^k - 1$. Since $N = 2^k - 1$, $k = \log_2(N-1)$, and $S = 2^{\log_2(N-1)} - \log_2(N-1) - 1 = N - 1 - \log_2(N-1) - 1 = O(N)$.

2. More on $d$-heap deleteMin for those who are curious. The question is, how does the running time of deleteMin on a $d$-heap ($d \log_d n$) compare to the running time of deleteMin on a binary heap ($2 \log_2 n$)? First we change the base, so the comparison is clearer.

$$d \log_d n = \frac{d \log_2 n}{\log_2 d} = \frac{d}{\log_2 d} \log_2 n$$
So we want to know how $d/\log_2 d$ compares with 2. If we set $d = 3$, we get $3/\log_2 3 = 3/1.54 < 2$. If we set $d = 4$, we get $4/\log_2 4 = 4/2 = 2$. As $d$ increases, we see that $d/\log_2 d$ increases (since $d$ grows faster than $\log_2 d$). Thus the answer is “it depends on $d$”; but a more detailed answer is “for values of $d > 4$, $d \log_d n > 2 \log_2 n$”. So for $d > 4$, deleteMin runs slower on a $d$-heap than on a binary heap.