Dan Kondratyuk
Data Structures I
Charles University
19 November 2017

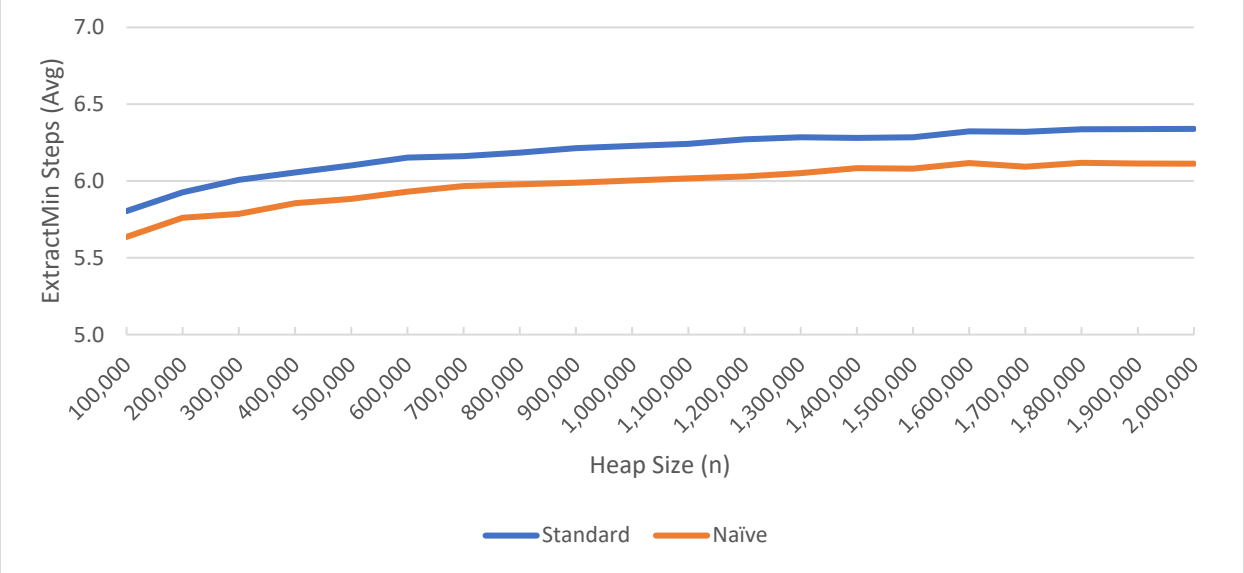
Assignment 2

This assignment implements a Fibonacci heap for analyzing its algorithmic complexity. There are two types of heaps implemented: (1) a standard Fibonacci heap which utilizes marks and cascading cuts in the decrease key operation, and (2) a naïve version which utilizes only single cuts. The following graphs plot the average number of steps of an $ExtractMin$ as a function of the heap size.

I. Random Test

The random test intersperses the $Insert$, $DecreaseKey$, and $ExtractMin$ operations uniformly.

Figure 1: Random Test



The first feature to notice is the shape of the curve for the standard Fibonacci heap is approximately logarithmic. This is because the number of steps of $ExtractMin$ is defined as $children\ appended + nodes\ joined\ in\ consolidation$. The more nodes there are, the more children there are per node and the more children will need to be appended and joined during $ExtractMin$. Fibonacci heaps have been proven to have an amortized complexity of $O(\log n)$

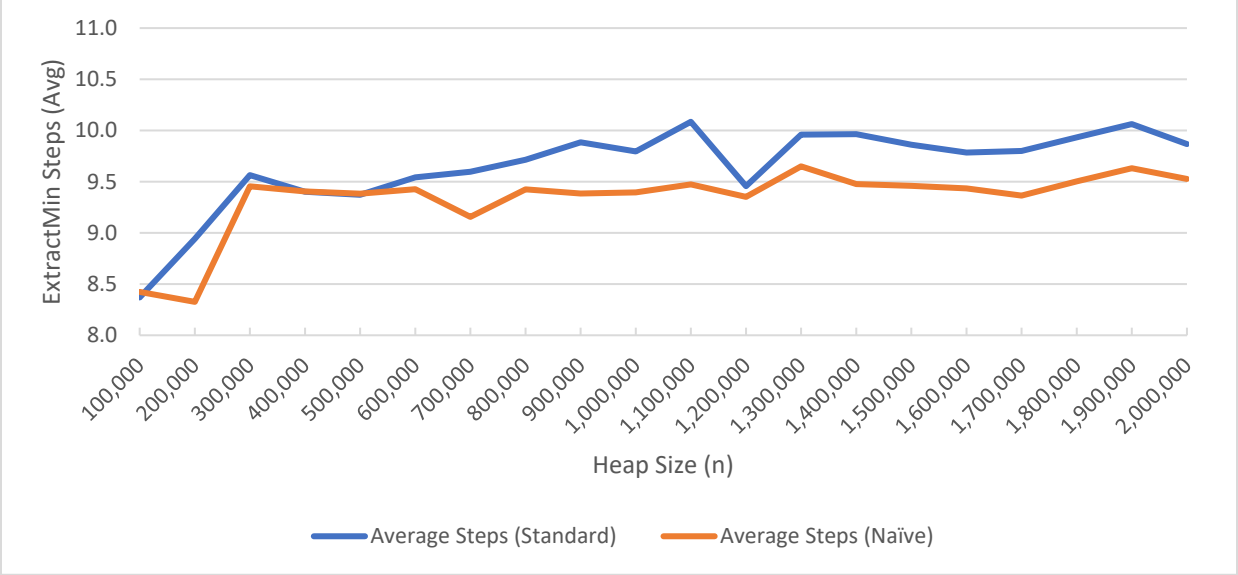
for $ExtractMin$ where n is the number of nodes in the heap. The number of children of any given node and the number of joins necessary for consolidation is on average $O(\log n)$, so the logarithmic curve is as expected.

A bit less intuitive is that the naïve implementation has a slightly better number of steps, but only by a constant amount. A naïve heap does not implement the cascading cut, so only one node is appended, and fewer nodes are joined as a result. There is the issue that a node may accumulate a lot of children (further increasing the number of steps), but this is offset by the fact that operations are random and so worst-case trees are not common.

II. Biased Test

The biased test intersperses operations like the random test, but uses much fewer $ExtractMin$ operations.

Figure 2: Biased Test



The biased test shows that the average number of steps in both curves has increased from ~6.0 to ~ 9.5. This test issues a lot of $Insert$ and $DecreaseKey$ commands, so naturally the list of nodes at the root will grow much larger before $ExtractMin$ consolidates them. This further increases the average number of children appended and nodes joined in consolidation.

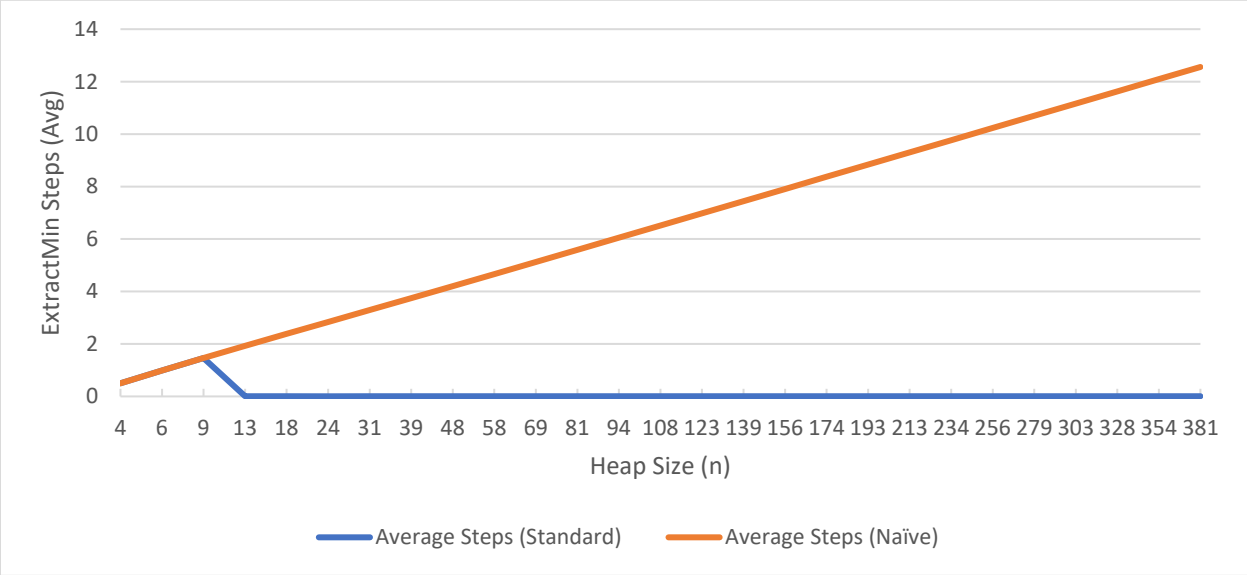
The number of children appended is much smaller than the number of nodes joined, as there will be many 0-degree children from the $Insert$ operations before an $ExtractMin$ operation. This explains why the cascading cut doesn't affect the number of steps between naïve

and standard implementations except by a small constant amount. As in the previous graph the curves are approximately logarithmic, but here contain a lot more noise. The number of these 0-degree children will affect how many joins are necessary, but this has a 50% chance of occurring for every operation between $ExtractMin$ operations, thus causing the plot to fluctuate.

III. Special Test

The special test creates a worst-case scenario for the naïve heap by constructing a star, where a node has a depth of 1 but contains many children.

Figure 3: Special Test



Asymptotically, the naïve heap tends to $O(n)$. Because it lacks cascading cuts, what ends up happening is that there will be a tree that has a depth of 1 but have $O(n)$ children. Upon calling $ExtractMin$, there will be $O(n)$ nodes appended and $O(n)$ joins.

Asymptotically, the average number of steps for the standard heap tends to a constant of ~0.007, or $O(1)$. This is because after a tree with height 3 is constructed, when a cut is initiated on the bottom node, its parent is cut too, always reducing the tree height to what it was previously. This ensures that a large star is never created, bounded by a maximum of 3 children. This also explains why the standard heap starts increasing as in the naïve case, because a parent node wouldn't be cut until after a tree size of 13.