Dan Kondratyuk
Data Structures I
Charles University
19 November 2017

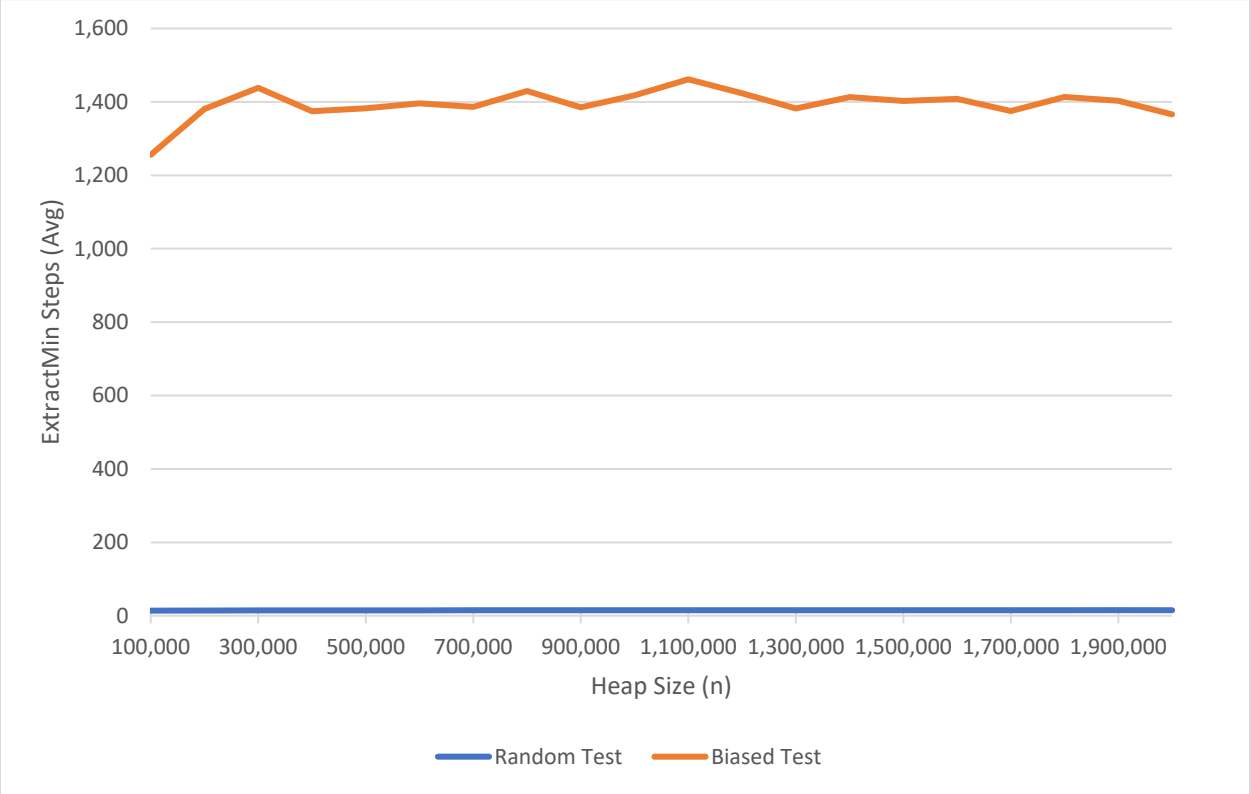
Assignment 2

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 and Biased Test

The random test intersperses $Insert$, $DecreaseKey$, and $ExtractMin$ operations uniformly, while the biased test uses much fewer $ExtractMin$ operations.

Figure 1: Random Test vs. Biased Test



The most prominent feature in the graph is that the biased test uses a few orders of magnitude more steps on average than the random test: the random test has an average of 15 steps, while the biased test has an average of 1,400 steps. This is because the biased test issues many more $Insert$ and $DecreaseKey$ commands before calling $ExtractMin$. An $Insert$ operation causes the root list to grow by one, and a $DecreaseKey$ operation can possibly increase the size of the root list even more when performing cascading cuts. Therefore, the root list will grow very wide between $ExtractMin$ operations, increasing the number of nodes joined in the consolidation phase of $ExtractMin$ substantially. As the number of steps of $ExtractMin$ is defined as $children\ appended + nodes\ joined\ in\ consolidation$, the number of steps in the biased test will be much higher. As opposed to the biased test, the random test will consolidate the heap more frequently, making the number of joins small, and hence making the average number of steps also relatively small.

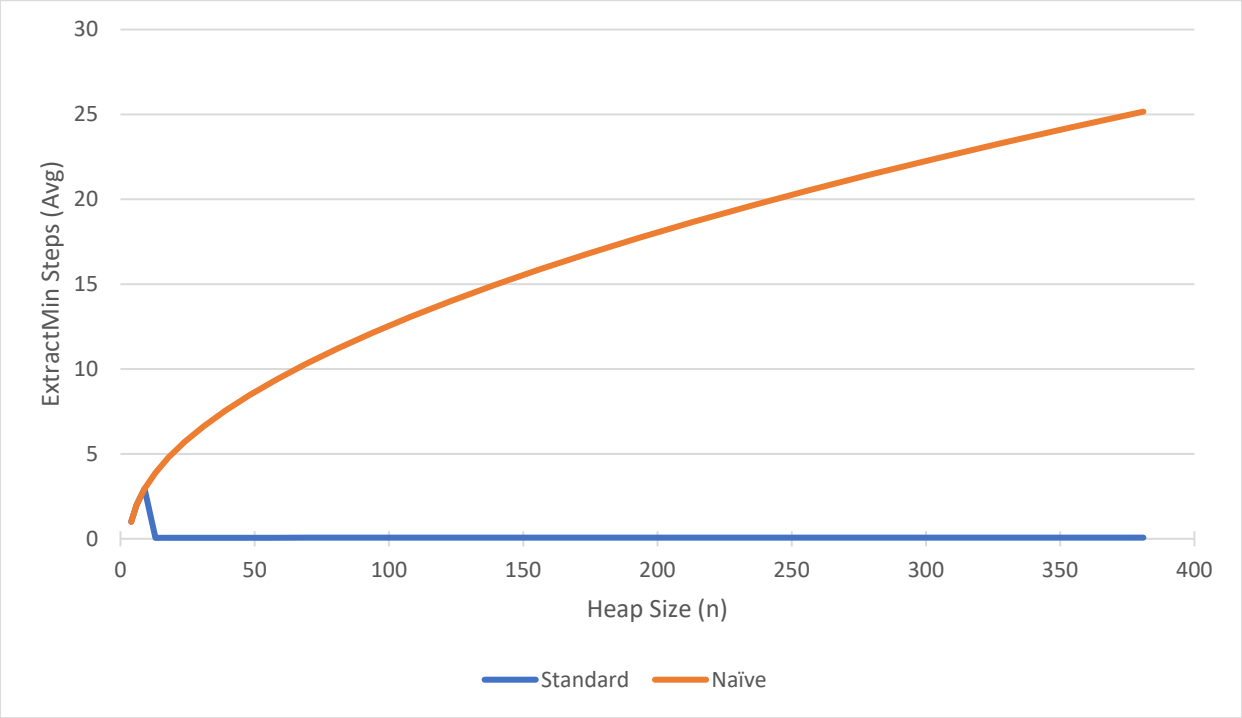
Another feature is that the biased test has a lot more variance in its data points than in the random test. This is simply due to the biased test having much fewer $ExtractMin$ operations, resulting in a sparser number of data samples. If the number of $ExtractMin$ operations in the biased test matched that of the random test, the biased test curve should be much smoother.

The last feature in the graph to note is the shape of the curve for the standard Fibonacci heap in the random test, which is approximately logarithmic. 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.

II. 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. Stars are constructed by recursively joining smaller stars.

Figure 2: Special Test



Asymptotically, the naïve heap tends to $O(\log n)$. Because it lacks cascading cuts, what ends up happening is that there will be a tree that has a depth of 1 but have $O(\log n)$ children (a star). Upon calling $ExtractMin$, there will be $O(\log n)$ nodes appended and all these nodes will be joined together. As the star construction is recursive, the number of nodes appended and joined is the sum of smaller stars in the process. This all adds up to the logarithmic curve above.

Asymptotically, the average number of steps for the standard heap tends to a constant of ~0.0065, or $O(1)$. The construction of stars fails in the standard heap because the node marking and cascading cuts ensure that the structure of the heap does not stray too far from that of a binomial heap. After a tree with height 2 is constructed (3 levels), when a cut is initiated on the bottom node, its marked parent is cut too, always reducing the number of children of a root list node in the process. This ensures that a large star is never created, bounded by a maximum of 3 children. As a result, the number of appends and joins is bounded by this relationship. Furthermore, this causes most of the $ExtractMin$ operations to delete nodes immediately after inserting them as to not append or join any nodes, decreasing the average to below 1 step per operation.

This also explains why the standard heap follows the same trend as the naïve heap in the first few data points: a star can be constructed up to 3 children, which happens in the first three tree sizes. A cascading cut wouldn't happen until a tree size of 13 in the star construction, where the number of steps drops to a constant amount.