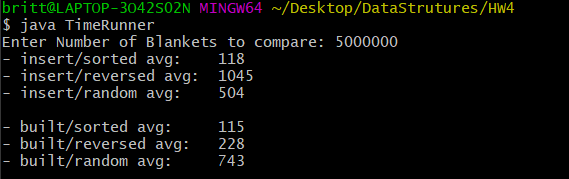
Heap Building Analysis

Purpose: This program builds a heap of blankets, because that just sounds nice. Blankets are ranked on their fluffiness value. The program builds 3 arrays: sorted, reversed, and random. Each array is inserted into the heap blanket by blanket, and then built as a whole.



Sample Test Above

Testing: Each number of blankets was tested 10 times and averaged for consistency and understanding.

\*\*Lower values were consistently 0, not worth analyzing for this project

Analysis:

* **Inserts**
  + **Sorted** – Sorted insert performed by far the best. Because the data was in order via merge sort, insert() only had to place the next data at the end of the binary heap, no percolation necessary.
  + **Reversed** – Reversed insert did awful. Each time an element was inserted, the heap had to shift everything existing within the structure to accommodate for the new element, essentially the root.
  + **Random** – Obviously… kind of varies based on the configuration of the array before building the heap. Generally better than reversed, worse than sorted. This is to be expected since reversed will be the hardest to insert and sorted the easiest.
* **Builds**
  + **Sorted** – Build sorted fares well. Again, we see how the data is placed in the correct position and does not need to be percolated. The cost remains relatively low.
  + **Reversed** – Build reversed is okayish. Sorted is obviously a better option, but in comparison to insert random and insert reverse, this performs much better.
  + **Random** – Build random was very similar to insert random, which is not surprising. Each element going in had some chance to be either a great or horrible option for the heap. The two random types fluctuated a bit with different values of N, not following any solid trend. It depends on the shuffling of the elements.

In general, smaller values of N are negligible for differences in time. As values of N get larger, the graphs start looking more similar. We can clearly see the sorted elements can create a heap much quicker than reversed elements. Although, accounting for the cost of sorting, it may be better to use random insert on unsorted data. Reversed insert is a bit of an awful idea, just sort it at that point.



Small pile of blankets, very efficient