Sarah Teichman

Project I Report

	Type:	N=10	N=100	N=1000	N=10000	N=100000
N/10	hybrid	1.12E-05	6.70E-06	1.34E-04	9.20E-05	5.12E-03
N/10	redblack	4.96E-06	1.73E-06	1.16E-05	1.87E-05	2.23E-04
100N	hybrid	2.40E-04	4.14E-04	6.08E-03	7.27E-01	7.93E01
100N	redblack	6.28E-05	1.89E-04	1.52E-03	2.10E-02	4.41E-01

For the Hybrid set implementation:
the m = n/10 set has a $\theta(f(n) = \theta((n/10)*\sqrt{(n/10)})$
the m = 100n set has a $\theta(g(n) = \theta((100*n)*\sqrt{(100*n)})$

I calculated $\theta(f(n))$ and $\theta(g(n))$ based on the my code. There are several parts to the add method. First, the method calls contains. Contains has two parts, a binary search and a sequential search. The binary search happens every time and is $\theta(\log(n))$ and the sequential search happens less than $\sqrt{(n)}$ times and is $\theta(n)$. This means that the contains method contributions on average: $\log(n) + \sqrt{(n)}$ to the total run time. However, $\log(n)$ is $O(\sqrt{(n)})$, so the $\log(n)$ term can be discarded, as it does not affect the asymptotic running time. After contains is called, the item is added to the end of the ArrayList unsorted, which happens in constant time. The merge method is not called every time, only n^(1/x) terms for some x, so the merge method also does not affect the asymptotic running time. Therefore, the total time asymptotic running time for n adds is $n*\sqrt{(n)}$.

For the TreeSet implementation:
the m = n/10 set has a $\theta(f(n) = \theta((n/10)*\log(n/10))$
the m = 100n set has a $\theta(g(n) = \theta((100*n)*\log(100*n))$

I calculated $\theta(f(n))$ and $\theta(g(n))$ based on the documentation for Java's TreeSet. The add method for a balanced tree is at worst $\log(n)$, and this is multiplied by the number of times that the add is done.

Note: $f(n) = \lambda g(n)$ for both of these sets, where λ is a constant scaling factor based on the number of items added.