Keith DeSantis

Algorithms Homework 3

Construction Times in Seconds (10 Trials):

Trial Num.	1	2	3	4	5	6	7	8	9	10	Avg
HT	**0.18**	**0.219**	**0.186**	**0.228**	**0.183**	**0.198**	**0.18**	**0.187**	**0.219**	**0.183**	**0.1963**
BST	**0.074**	**0.06**	**0.072**	**0.053**	**0.05**	**0.065**	**0.1**	**0.082**	**0.057**	**0.063**	**0.0676**

Search 1/10 of Items Times in Seconds (10 Trials):

Trial Num.	1	2	3	4	5	6	7	8	9	10	Avg
HT	**0.026**	**0.045**	**0.043**	**0.026**	**0.024**	**0.03**	**0.042**	**0.046**	**0.04**	**0.034**	**0.0356**
BST	**0.01**	**0.012**	**0.012**	**0.008**	**0.006**	**0.016**	**0.014**	**0.007**	**0.01**	**0.013**	**0.0108**

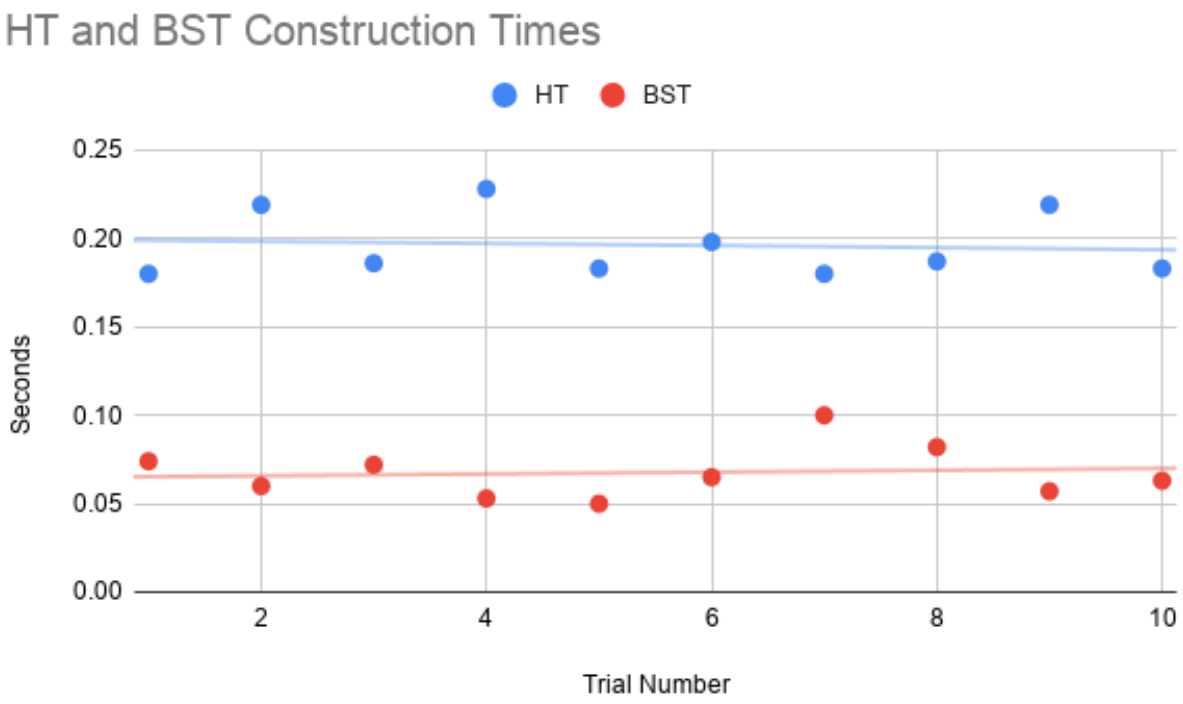

Figure 1. Construction times of Linear Probing Hash Table vs Binary Search Tree over 10 trials, along with average trendlines.

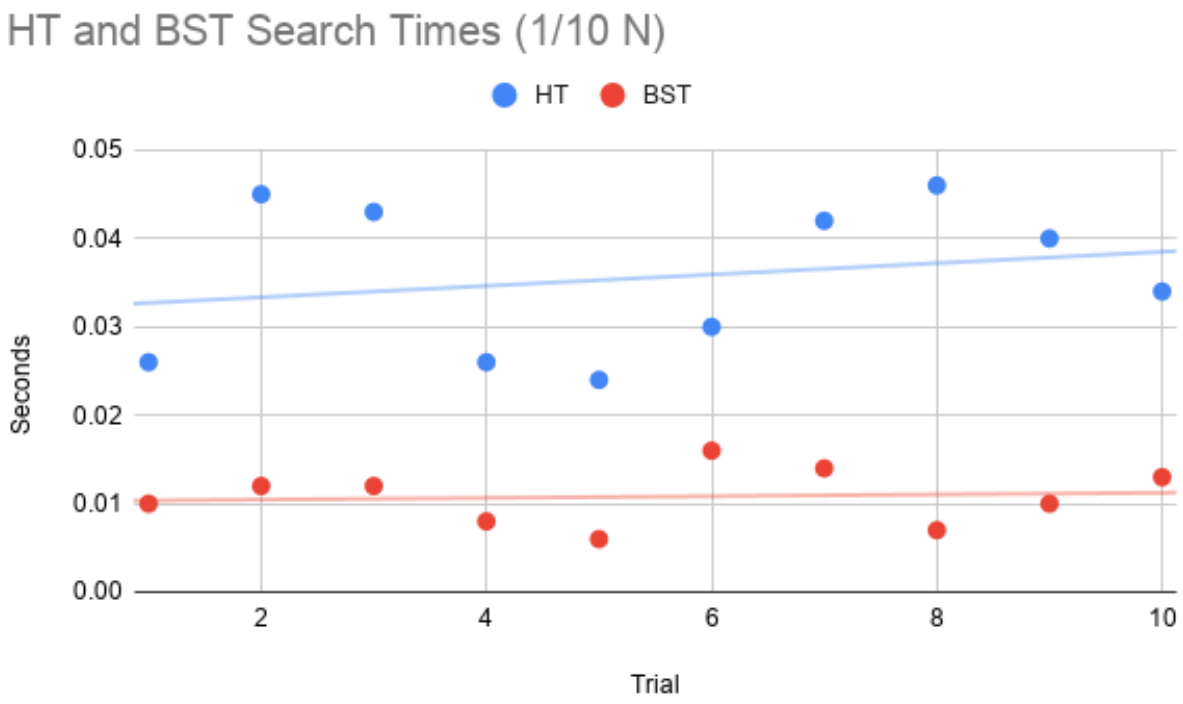

Figure 2. Search times for Linear Probing Hash Table vs Binary Search Tree on 410 searches (1/10 sample size), along with trendlines.

Based on data gathered by both algorithms and search engine implementations, Binary Search Tree is the more efficient and useful for our company. The initial construction and addition of new elements into the database takes less time, nearly by a factor of three, and the process of searching is consistently faster, also to a factor of near three in our test trials. While it may seem a bit more confusing to implement and edit a Binary Search Tree search engine, the pay-offs in performance greatly outweigh any inconvenience and it is with all my confidence I suggest a BST-based algorithm for our company.