Thomas Koker and Grace Seiche

Professor Whitehill

CS201x/B02

6 December 2016

Project 3 Write Up

Identification of Mystery Abstract Data Types Using Time Cost Analysis

Team Number: 1015

Results:

Structure Index	**Structure Type**
0	Binary Search Tree
1	Heap
2	Doubly Linked List
3	Hash Set
4	Binary Search Tree

Raw Data:

Structure	N	T add	T remove	T contains	T removeLargest	T containsLargest
0	1	55	111	111	111	
0	2	55	139	138	166	
0	5	56	187	180	222	
0	10	56	229	218	277	
0	20	64	284	271	278	
0	30	67	305	291	333	
0	40	74	330	315	389	
0	50	86	342	335	333	
0	60	101	355	343	333	
0	70	104	374	358	333	
0	80	122	376	370	389	
0	90	123	381	375	389	
0	100	130	390	386	444	
0	200	247	435	439	444	
0	300	354	460	475	555	

0	400	409	486	492	556	
0	500	440	495	504	611	
0	600	454	511	525	555	
0	700	464	527	538	555	
0	800	480	533	546	556	
0	900	495	546	557	556	
0	1000	517	550	557	611	
0	2000	581	625	616	612	
0	3000	609	667	646	667	
0	4000	634	691	686	778	
0	5000	647	702	689	777	
0	6000	662	717	704	723	
0	7000	670	729	715	723	
0	8000	683	746	724	778	
0	9000	691	750	734	834	
0	10000	709	774	760	834	
1	1	85	170	85	170	85
1	2	101	213	125	170	85
1	5	107	391	253	340	85
1	10	110	615	466	341	85
1	20	104	1067	889	426	84
1	30	107	1470	1302	426	85
1	40	102	1922	1728	511	85
1	50	96	2401	2226	512	85
1	60	106	2706	2712	512	85
1	70	102	3153	2941	597	85
1	80	101	3598	3541	596	85
1	90	96	4127	3748	596	85
1	100	95	4465	4291	597	85
1	200	92	8679	8550	681	85
1	300	95	12695	13085	767	85
1	400	90	17429	17197	768	85
1	500	93	21515	21450	767	85
1	600	88	25924	25409	853	85
1	700	98	30959	30212	852	85
1	800	90	34204	34931	852	85
1	900	90	38796	39327	852	85
1	1000	99	43251	41976	853	85
1	2000	89	85177	86275	938	85
1	3000	110	131130	130522	1024	85
1	4000	87	164799	174926	1023	85
1	5000	96	218319	219048	1108	85
1	6000	89	253897	257491	1108	85

1	7000	124	307217	299078	1108	85
1	8000	87	340916	327947	1108	85
1	9000	89	377410	377575	1193	85
1	10000	91	436011	417331	1194	85
2	1	4	4	8	4	
2	2	4	5	9	8	
2	5	4	12	16	19	
2	10	4	22	26	39	
2	20	4	42	46	79	
2	30	4	60	67	119	
2	40	4	82	85	159	
2	50	4	103	107	199	
2	60	4	122	127	239	
2	70	4	145	144	279	
2	80	4	159	164	319	
2	90	4	188	182	359	
2	100	4	198	205	399	
2	200	4	417	406	799	
2	300	4	612	599	1199	
2	400	4	796	781	1599	
2	500	4	994	983	1998	
2	600	4	1244	1242	2398	
2	700	4	1415	1436	2799	
2	800	4	1622	1615	3198	
2	900	4	1787	1820	3599	
2	1000	4	1939	2005	3998	
2	2000	4	4010	3960	7994	
2	3000	4	6023	6031	11997	
2	4000	4	7890	8041	15993	
2	5000	4	9800	10052	19996	
2	6000	4	11568	12218	23993	
2	7000	4	14061	14397	27997	
2	8000	4	15926	16171	31999	
2	9000	4	17913	17991	35992	
2	10000	4	19613	19260	39992	
3	1	29	58	58	58	
3	2	29	58	58	58	
3	5	29	58	58	58	
3	10	29	58	58	58	
3	20	29	57	58	58	
3	30	30	57	58	58	
3	40	30	57	58	58	
3	50	30	57	58	58	

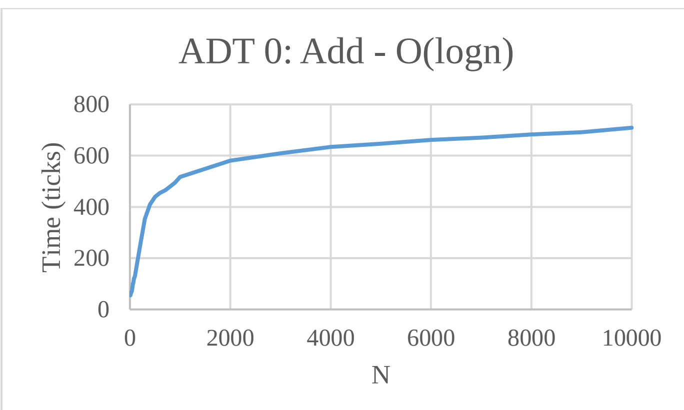
3	60	30	56	58	58	
3	70	31	56	58	58	
3	80	31	56	58	58	
3	90	31	55	58	58	
3	100	32	55	58	58	
3	200	35	52	57	58	
3	300	37	50	57	58	
3	400	39	48	56	58	
3	500	41	47	54	58	
3	600	41	46	54	58	
3	700	42	46	54	58	
3	800	43	45	52	58	
3	900	44	45	51	58	
3	1000	44	46	50	58	
3	2000	49	48	50	58	
3	3000	50	48	50	58	
3	4000	51	49	50	58	
3	5000	50	49	50	58	
3	6000	51	49	51	58	
3	7000	51	49	50	58	
3	8000	51	50	51	58	
3	9000	50	50	50	58	
3	10000	51	50	51	58	
4	1	57	115	115	115	
4	2	57	144	144	173	
4	5	58	195	188	231	
4	10	59	240	222	289	
4	20	70	286	268	346	
4	30	73	316	297	347	
4	40	77	344	330	347	
4	50	95	364	348	404	
4	60	103	375	364	462	
4	70	122	390	391	346	
4	80	118	392	378	520	
4	90	113	395	392	461	
4	100	136	407	403	462	
4	200	261	451	457	520	
4	300	365	478	492	520	
4	400	408	502	515	520	
4	500	458	523	528	577	
4	600	467	528	546	520	
4	700	501	553	569	635	
4	800	502	556	571	578	

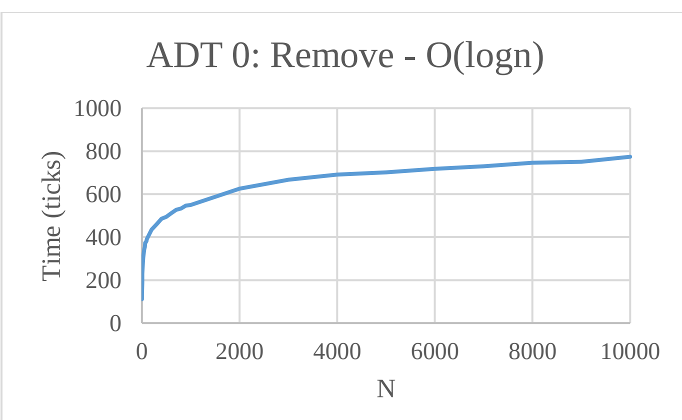
4	900	519	564	570	578	
4	1000	537	581	591	578	
4	2000	621	659	653	635	
4	3000	659	701	690	693	
4	4000	676	732	713	693	
4	5000	697	743	723	751	
4	6000	687	754	731	866	
4	7000	698	760	745	809	
4	8000	718	781	764	925	
4	9000	751	814	791	809	
4	10000	746	807	783	751	

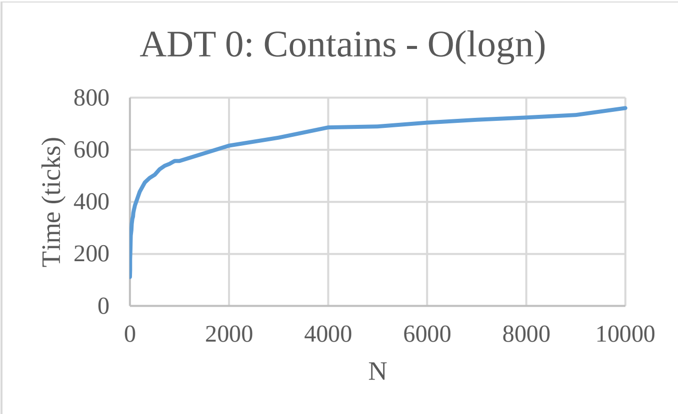
Conclusions (with Graphs):

In order to determine the time costs, we analyze the graphs to determine the slopes of the data. The O(1) time cost has a slope of zero, which is a straight horizontal line. The O(n) time cost has a slope of n, which is a straight diagonal line. The O(logn) time cost has a slope of log(n), which is a line that begins almost completely vertical and continues curving until it is almost horizontal.

Structure 0:

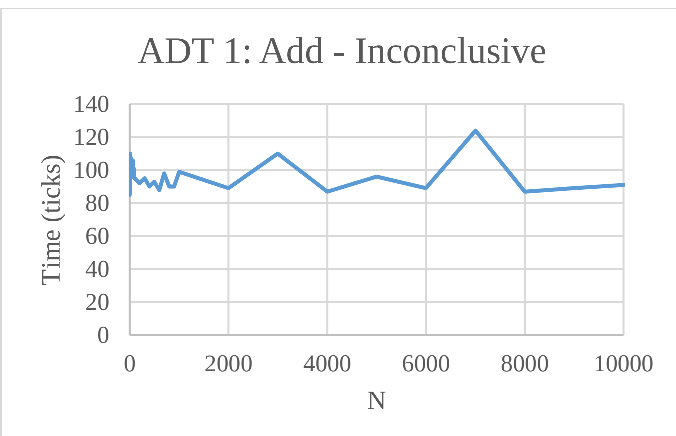


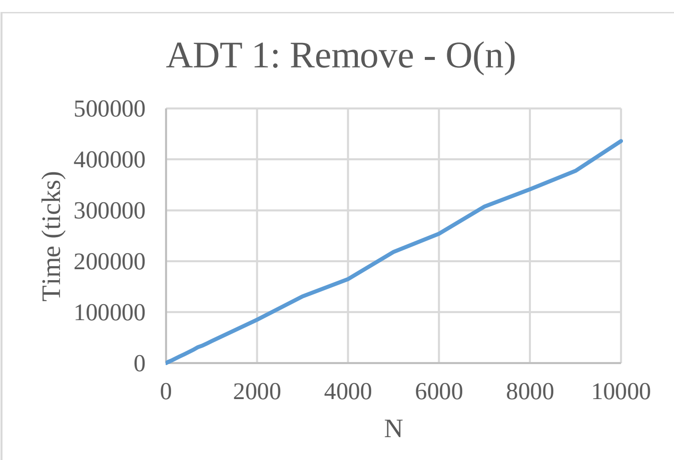


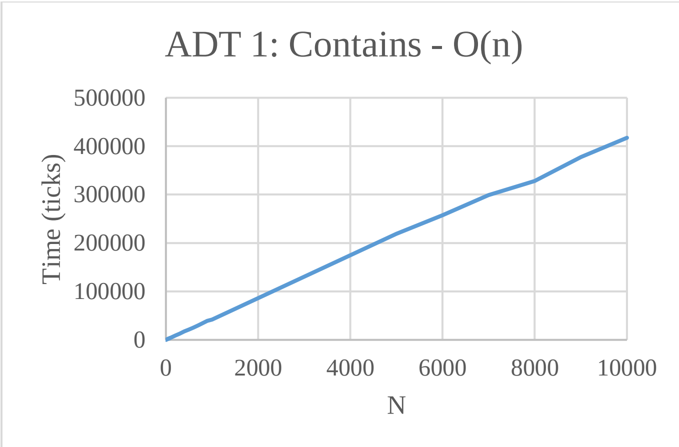


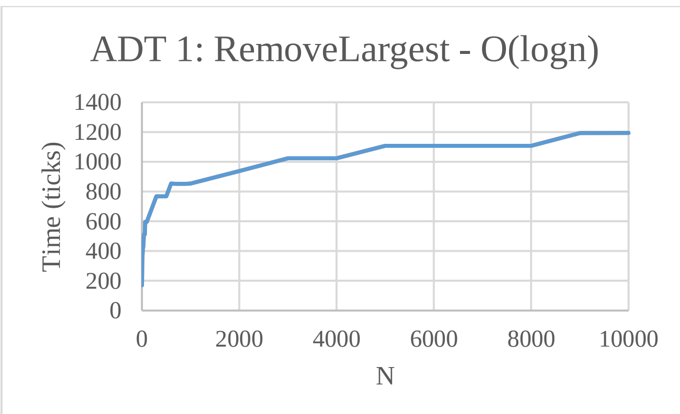
We classify the 0th data structure as a Binary Search Tree (BST). In the worst case, a BST should perform the add, remove, and contains methods all at an O(logn) time. Graphing the data for the 0th data structure shows that the time cost of add is O(logn), the time cost of remove is O(logn), and the time cost of contains is O(logn). (See graphs.) Since this data corresponds with the expected time costs for a binary search tree, the 0th data structure is a binary search tree.

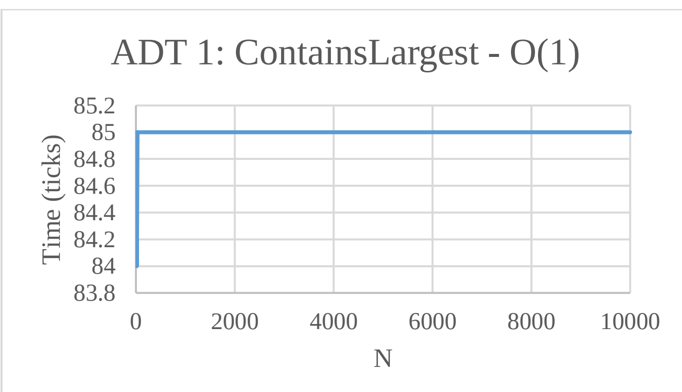
Structure 1:





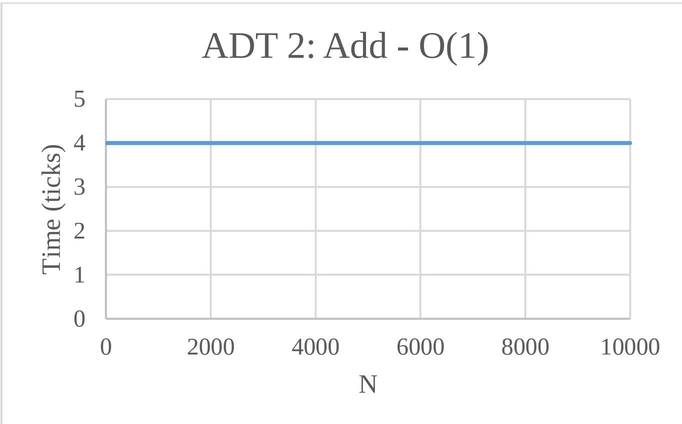


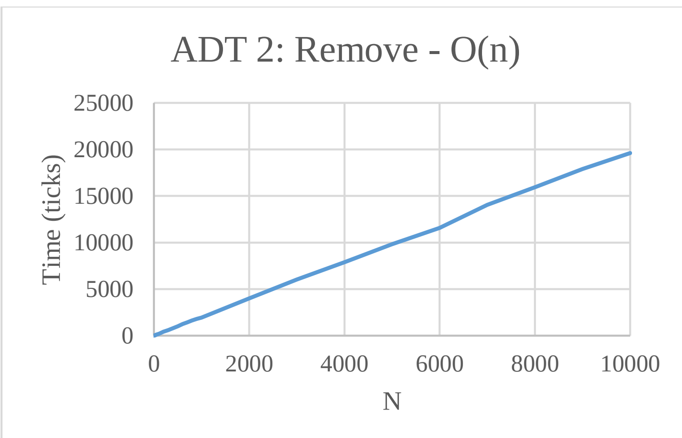


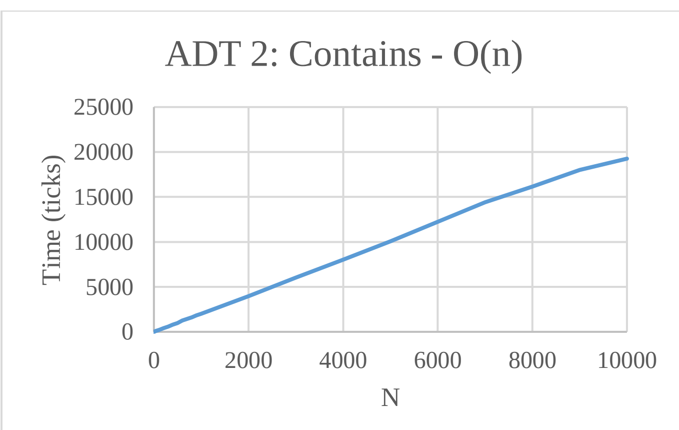


We classify the 1st data structure as a Heap. In the worst case, a heap should perform add in O(logn) time, remove in O(n) time, and contains in O(n). According to the time cost graphs for these functions, the first data structure performs remove in O(n) time and contains in O(n) time. However, the time for the add method is inconclusive. (See graphs.) Thus, more testing is required. In addition to the aforementioned functions, a heap should remove the largest element in O(logn) time and should find the largest element in O(1) time. The graphs for remove largest element and contains largest element display O(logn) and O(1) time, respectively, which also coincides with the heap structure. (See below.) Since four of the five of the first structure's time costs conclusively match the expected time costs for a heap structure, we can conclude that the 1st data structure is a heap.

Structure 2:

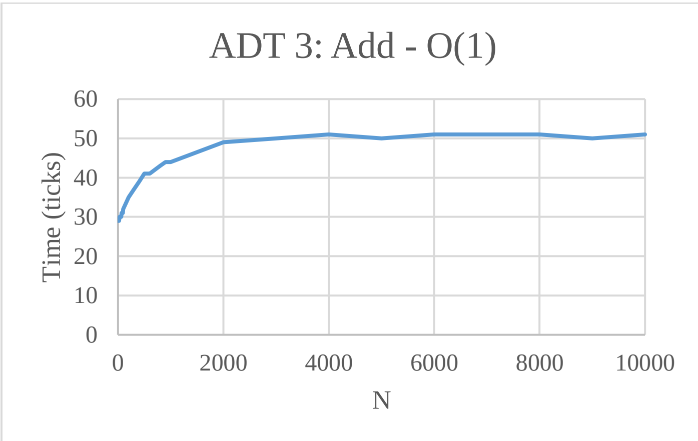


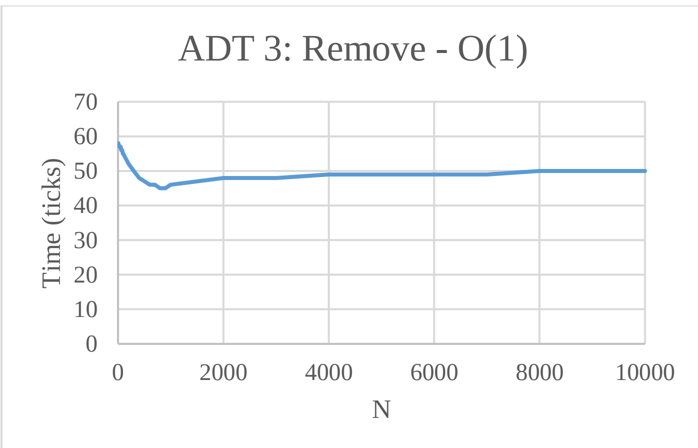


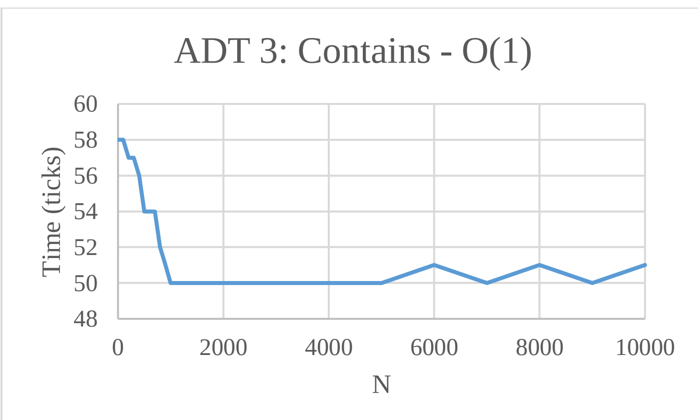


We classify the 2nd data structure as a Doubly Linked List. In the worst case, a doubly linked list should perform add in O(1) time, remove in O(n) time, and contains in O(n) time. The graphs of the add, remove, and contains functions for the 2nd show a time cost of O(1), O(n), and O(n), respectively. Since the time costs of the 2nd data structure match the expected time costs of the doubly linked list, we can conclude that the 2nd data structure is a doubly linked list.

Structure 3:

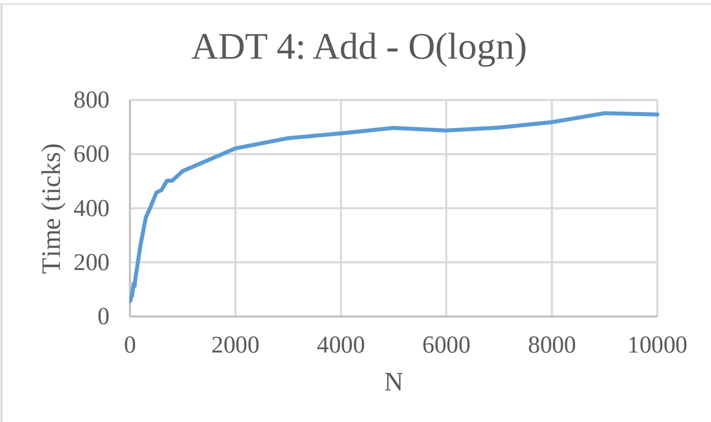


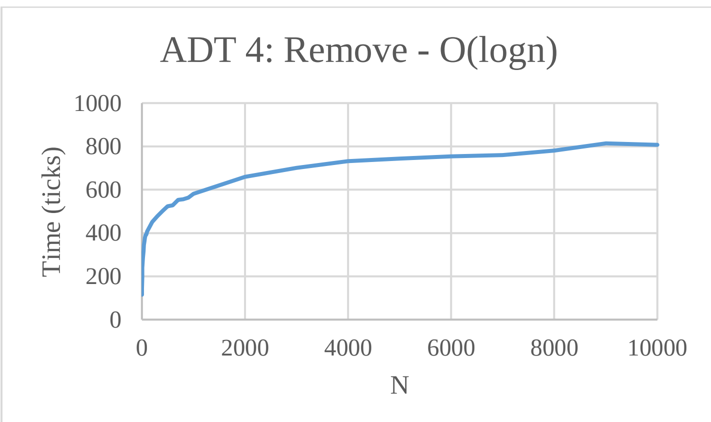


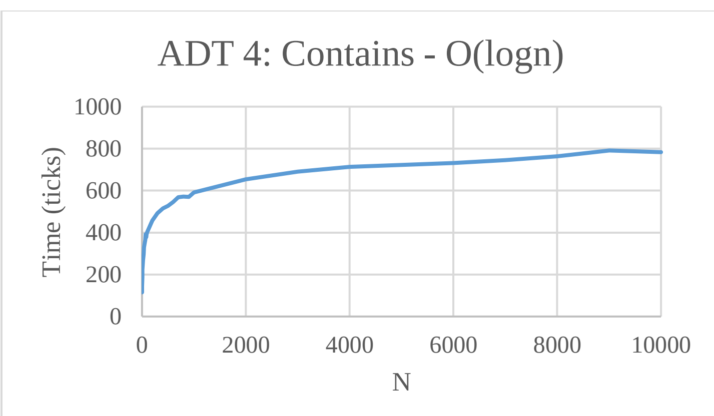


We classify the 3rd data structure as a Hash Set. In the average case, a hash set should perform add, remove, and contains in O(1) time. According to the graphs, the 3rd data structure performs add in O(1) time, remove in O(1) time, and contains in O(1) time. These graphs look slightly out of the ordinary because there tends to be a curve in the beginning of the data. However, when zoomed out to the scale of the other data types, the curve is almost non-existent, and the end behavior is consistent with O(1) time. Therefore, we can conclude that the 3rd data structure performs the three methods in O(1) time, which matches the expected times for hash sets. Thus, the 3rd data structure is a hash set.

Structure 4:







We classify the 4th data structure as a Binary Search Tree. As stated in the conclusion for structure 0, a BST performs add, remove, and contains in O(logn) time in the worst case. The graphs for the 4th data structure show that it performs the add, remove, and contains functions in O(logn) time. Therefore, we can conclude that the 4th data structure is a Binary Search Tree.