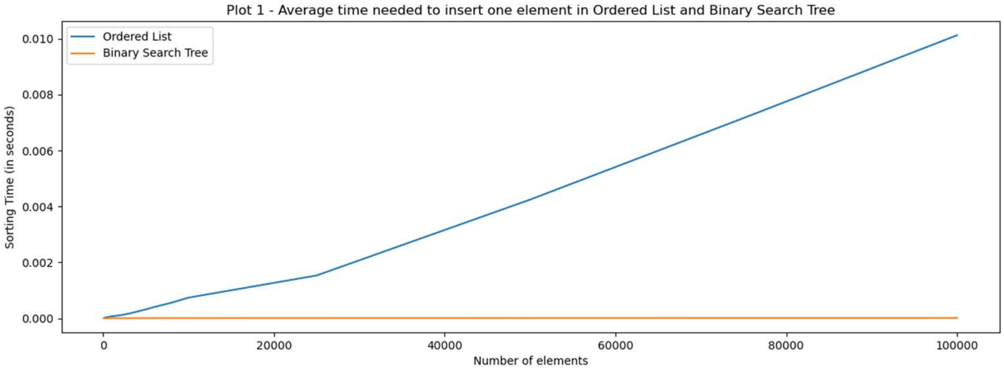
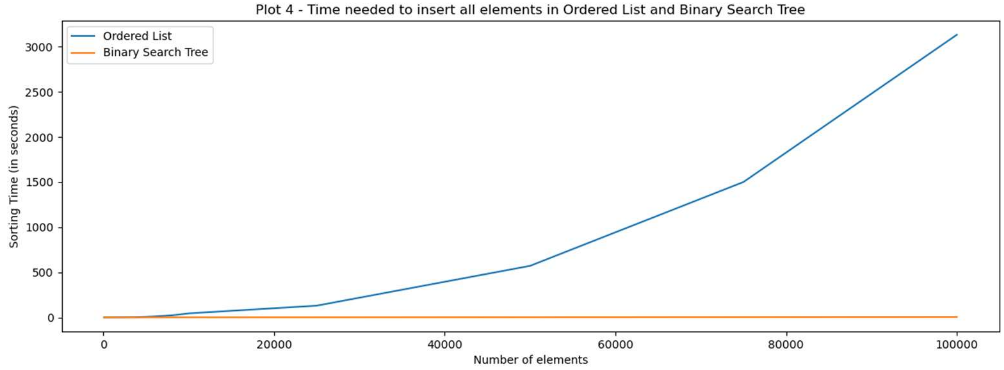
Conclusions Rafał Stachowiak

Adding Elements to Ordered List has complexity O(n) The list is sorted from the beginning.

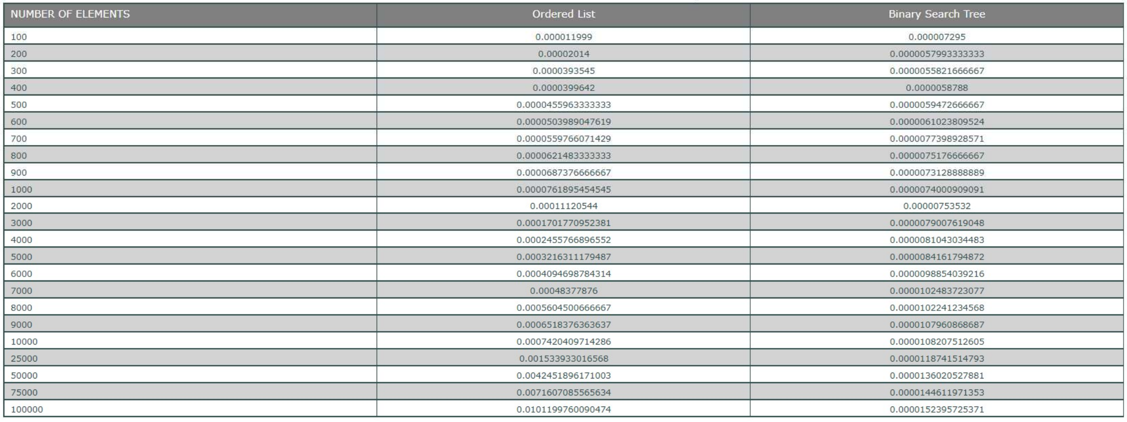
Inserting an element to the binary tree has complexity O(log(n)), so it is much better than ordered list.


Plot 1 - Average time needed to insert one element in Ordered List and Binary Search Tree


Plot 4 - Time needed to insert all elements in Ordered List and Binary Search Tree

Here we have a table with all the times in seconds shown on the plots:

Table 1 - Average time of inserting
one element in n-element Ordered List and Binary Search Tree



NUMBER OF ELEMENTS	Ordered List	Binary Search Tree
100	0.000011999	0.000007295
200	0.00002014	0.0000057993333333
300	0.0000393545	0.0000055821666667
400	0.0000399642	0.0000058788
500	0.0000455963333333	0.0000059472666667
600	0.0000503989047619	0.0000061023809524
700	0.0000559766071429	0.0000077398928571
800	0.0000621483333333	0.0000075176666667
900	0.0000687376666667	0.0000073128888889
1000	0.0000761895454545	0.0000074000909091
2000	0.00011120544	0.00000753532
3000	0.0001701770952381	0.0000079007619048
4000	0.0002455766896552	0.0000081043034483
5000	0.0003216311179487	0.0000084161794872
6000	0.0004094698784314	0.0000098854039216
7000	0.00048377876	0.0000102483723077
8000	0.0005604500666667	0.0000102241234568
9000	0.0006518376363637	0.0000107960868687
10000	0.0007420409714286	0.0000108207512605
25000	0.001533933016568	0.0000118741514793
50000	0.0042451896171003	0.0000136020527881
75000	0.0071607085565634	0.0000144611971353
100000	0.0101199760090474	0.0000152395725371

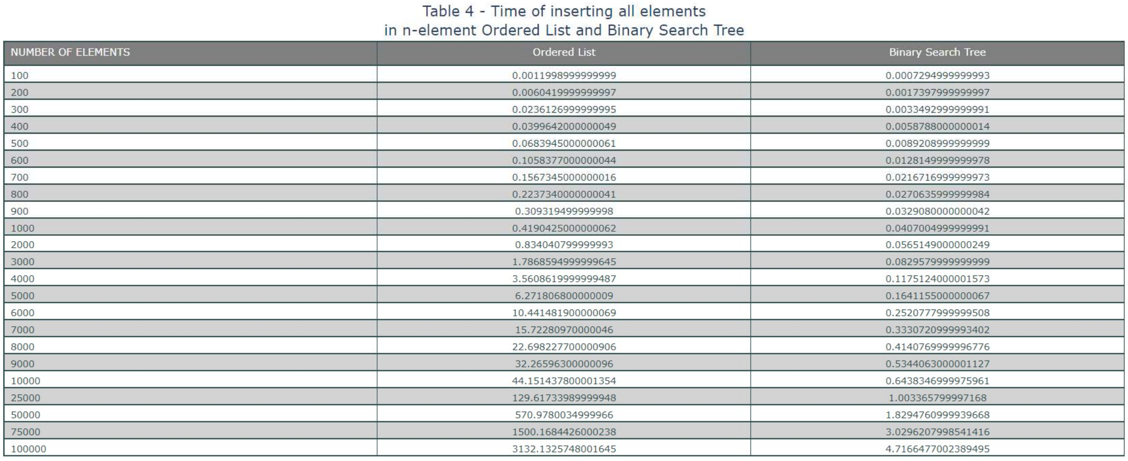
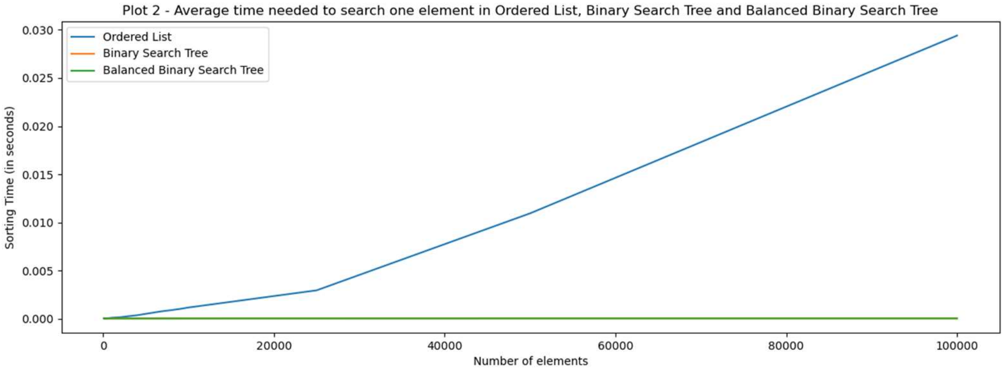


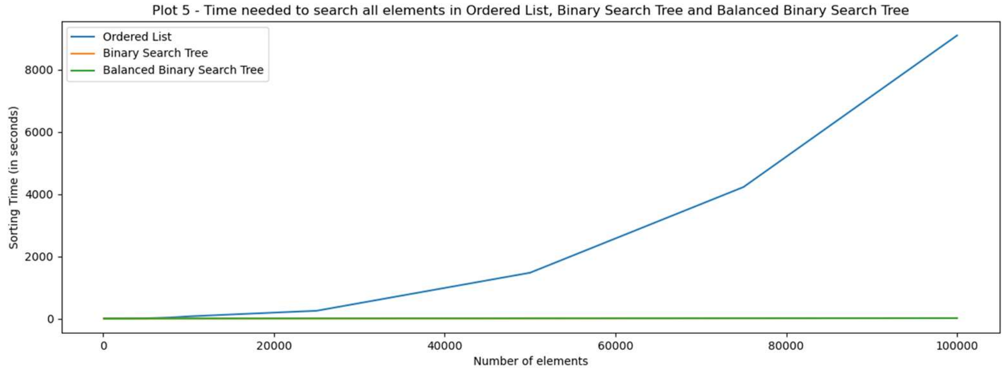
Table 4 - Time of inserting all elements
in n-element Ordered List and Binary Search Tree

NUMBER OF ELEMENTS	Ordered List	Binary Search Tree
100	0.0011998999999999	0.0007294999999993
200	0.0060419999999997	0.0017397999999997
300	0.0236126999999995	0.0033492999999991
400	0.0399642000000049	0.0058788000000014
500	0.0683945000000061	0.0089208999999999
600	0.1058377000000044	0.0128149999999978
700	0.1567345000000016	0.0216716999999973
800	0.2237340000000041	0.0270635999999984
900	0.309319499999998	0.0329080000000042
1000	0.4190425000000062	0.0407004999999991
2000	0.834040799999993	0.0565149000000249
3000	1.7868594999999645	0.0829579999999999
4000	3.5608619999999487	0.1175124000001573
5000	6.271806800000009	0.1641155000000067
6000	10.441481900000069	0.2520777999999508
7000	15.72280970000046	0.3330720999993402
8000	22.698227700000906	0.4140769999996776
9000	32.26596300000096	0.5344063000001127
10000	44.151437800001354	0.6438346999975961
25000	129.61733989999948	1.003365799997168
50000	570.9780034999966	1.8294760999939668
75000	1500.1684426000238	3.0296207998541416
100000	3132.1325748001645	4.7166477002389495

Searching an element in ordered List has complexity O(n) and Binary search tree is O(log(n)) In this approach also binary tree is better. Here we add to consideration also balanced Binary search tree which is similar to BST and even after magnification of the graph these structures does not differ so much



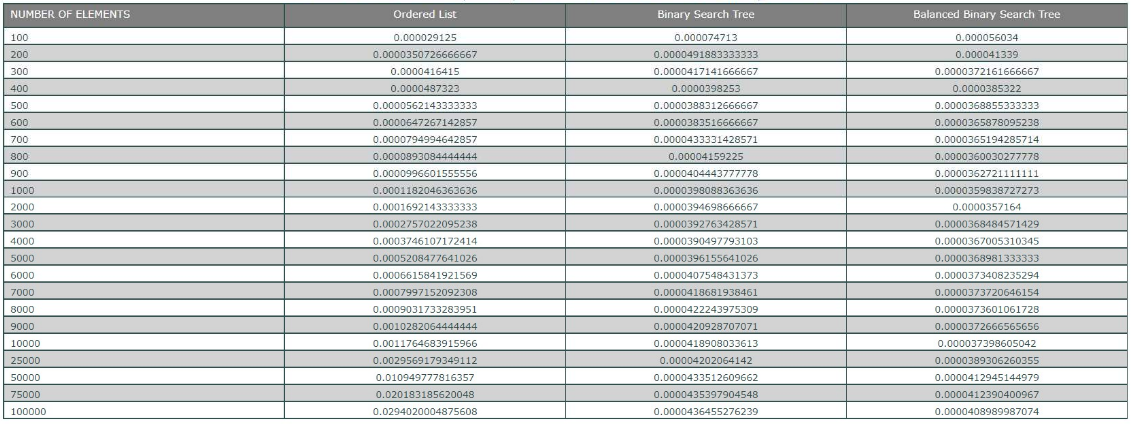
Plot 2 - Average time needed to search one element in Ordered List, Binary Search Tree and Balanced Binary Search Tree



Plot 5 - Time needed to search all elements in Ordered List, Binary Search Tree and Balanced Binary Search Tree

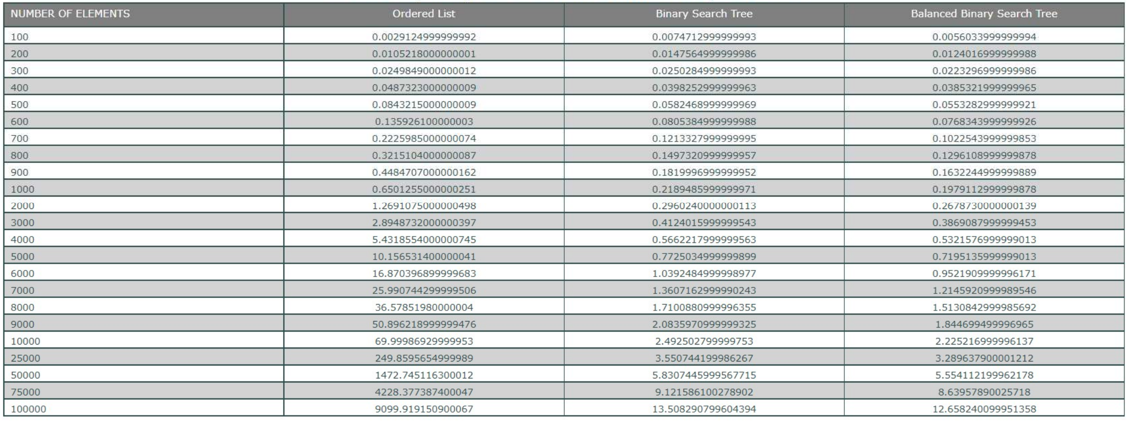
Here we have a table with all the times in seconds shown on the plots:

Table 2 - Average time of searching one element
in n-element Ordered List, Binary Search Tree and Balanced Binary Search Tree



NUMBER OF ELEMENTS	Ordered List	Binary Search Tree	Balanced Binary Search Tree
100	0.000029125	0.000074713	0.000056034
200	0.0000350726666667	0.0000491883333333	0.000041339
300	0.0000416415	0.0000417141666667	0.0000372161666667
400	0.0000487323	0.0000398253	0.0000385322
500	0.0000562143333333	0.0000388312666667	0.0000368855333333
600	0.0000647267142857	0.0000383516666667	0.0000365878095238
700	0.0000794994642857	0.0000433331428571	0.0000365194285714
800	0.0000893084444444	0.00004159225	0.0000360030277778
900	0.0000996601555556	0.0000404443777778	0.0000362721111111
1000	0.0001182046363636	0.0000398088363636	0.0000359838727273
2000	0.0001692143333333	0.0000394698666667	0.0000357164
3000	0.0002757022095238	0.0000392763428571	0.0000368484571429
4000	0.0003746107172414	0.0000390497793103	0.0000367005310345
5000	0.0005208477641026	0.0000396155641026	0.0000368981333333
6000	0.0006615841921569	0.0000407548431373	0.0000373408235294
7000	0.0007997152092308	0.0000418681938461	0.0000373720646154
8000	0.0009031733283951	0.0000422243975309	0.0000373601061728
9000	0.0010282064444444	0.0000420928707071	0.0000372666565656
10000	0.0011764683915966	0.0000418908033613	0.000037398605042
25000	0.0029569179349112	0.00004202064142	0.0000389306260355
50000	0.010949777816357	0.0000433512609662	0.0000412945144979
75000	0.020183185620048	0.0000435397904548	0.0000412390400967
100000	0.0294020004875608	0.0000436455276239	0.0000408989987074

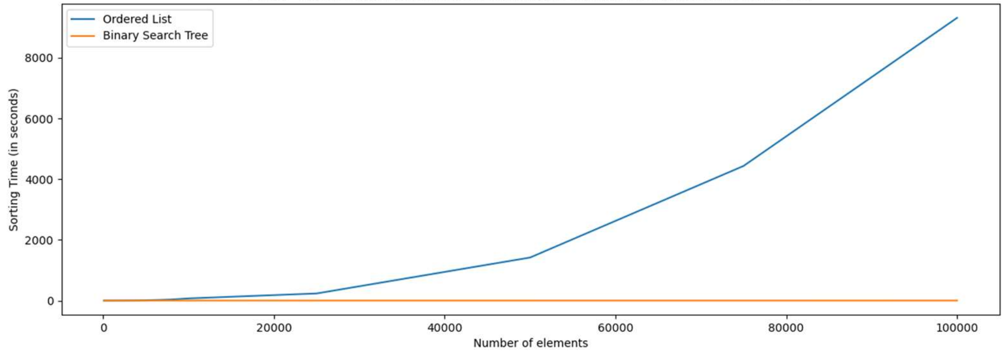
Table 5 - Time of searching all elements in n-element
Ordered List, Binary Search Tree and Balanced Binary Search Tree

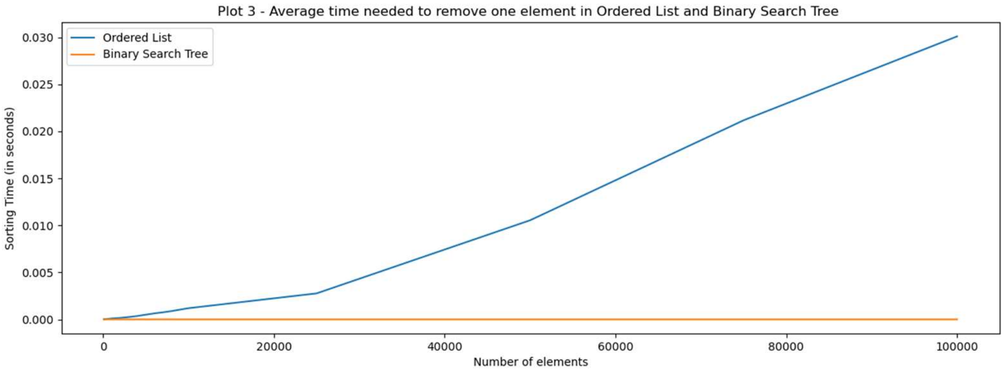


NUMBER OF ELEMENTS	Ordered List	Binary Search Tree	Balanced Binary Search Tree
100	0.0029124999999992	0.0074712999999993	0.0056033999999994
200	0.0105218000000001	0.0147564999999986	0.0124016999999988
300	0.0249849000000012	0.0250284999999993	0.0223296999999986
400	0.0487323000000009	0.0398252999999963	0.0385321999999965
500	0.0843215000000009	0.0582468999999969	0.0553282999999921
600	0.135926100000003	0.0805384999999988	0.0768343999999926
700	0.2225985000000074	0.1213327999999995	0.1022543999999853
800	0.3215104000000087	0.1497320999999957	0.1296108999999878
900	0.4484707000000162	0.1819996999999952	0.1632244999999889
1000	0.6501255000000251	0.2189485999999971	0.1979112999999878
2000	1.2691075000000498	0.2960240000000113	0.2678730000000139
3000	2.8948732000000397	0.4124015999999543	0.3869087999999453
4000	5.4318554000000745	0.5662217999999563	0.5321576999999013
5000	10.156531400000041	0.7725034999999899	0.7195135999999013
6000	16.870396899999683	1.0392484999998977	0.9521909999996171
7000	25.990744299999506	1.3607162999990243	1.2145920999989546
8000	36.57851980000004	1.7100880999996355	1.5130842999985692
9000	50.896218999999476	2.0835970999999325	1.844699499996965
10000	69.99986929999953	2.492502799999753	2.225216999996137
25000	249.8595654999989	3.550744199986267	3.289637900001212
50000	1472.745116300012	5.8307445999567715	5.554112199962178
75000	4228.377387400047	9.121586100278902	8.63957890025718
100000	9099.919150900067	13.508290799604394	12.658240099951358

Deletion of an element has a complexity O(n) in an Ordered linked list and O(log(n)) in Binary Search tree:

Plot 6 - Time needed to remove all elements in Ordered List and Binary Search Tree





Plot 3 - Average time needed to remove one element in Ordered List and Binary Search Tree

Here we have the table with the numbers from the plot:

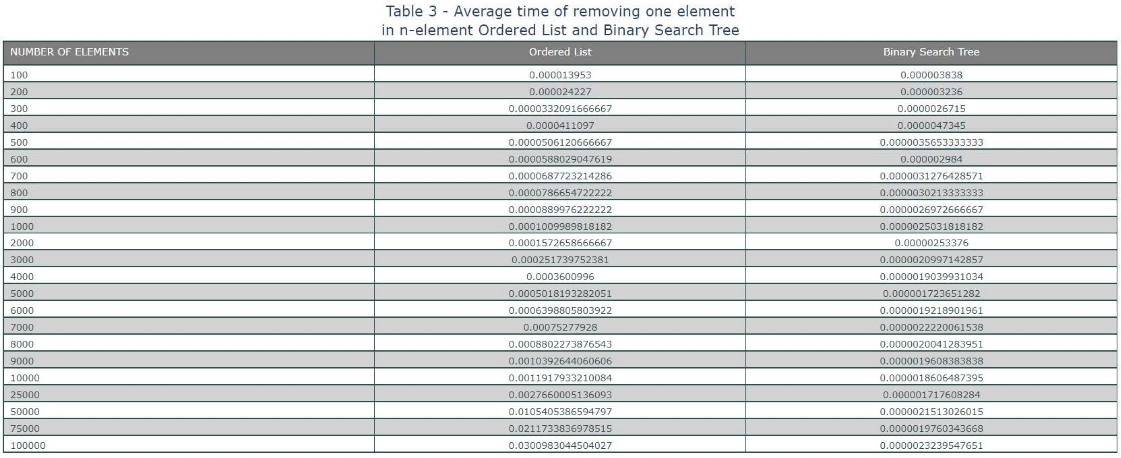


Table 3 - Average time of removing one element
in n-element Ordered List and Binary Search Tree

NUMBER OF ELEMENTS	Ordered List	Binary Search Tree
100	0.000013953	0.000003838
200	0.000024227	0.000003236
300	0.0000332091666667	0.0000026715
400	0.0000411097	0.0000047345
500	0.0000506120666667	0.0000035653333333
600	0.0000588029047619	0.000002984
700	0.0000687723214286	0.0000031276428571
800	0.0000786654722222	0.0000030213333333
900	0.0000889976222222	0.0000026972666667
1000	0.0001009989818182	0.0000025031818182
2000	0.0001572658666667	0.00000253376
3000	0.000251739752381	0.0000020997142857
4000	0.0003600996	0.0000019039931034
5000	0.0005018193282051	0.000001723651282
6000	0.0006398805803922	0.0000019218901961
7000	0.00075277928	0.0000022220061538
8000	0.0008802273876543	0.0000020041283951
9000	0.0010392644060606	0.0000019608383838
10000	0.0011917933210084	0.0000018606487395
25000	0.0027660005136093	0.000001717608284
50000	0.0105405386594797	0.0000021513026015
75000	0.0211733836978515	0.0000019760343668
100000	0.0300983044504027	0.0000023239547651

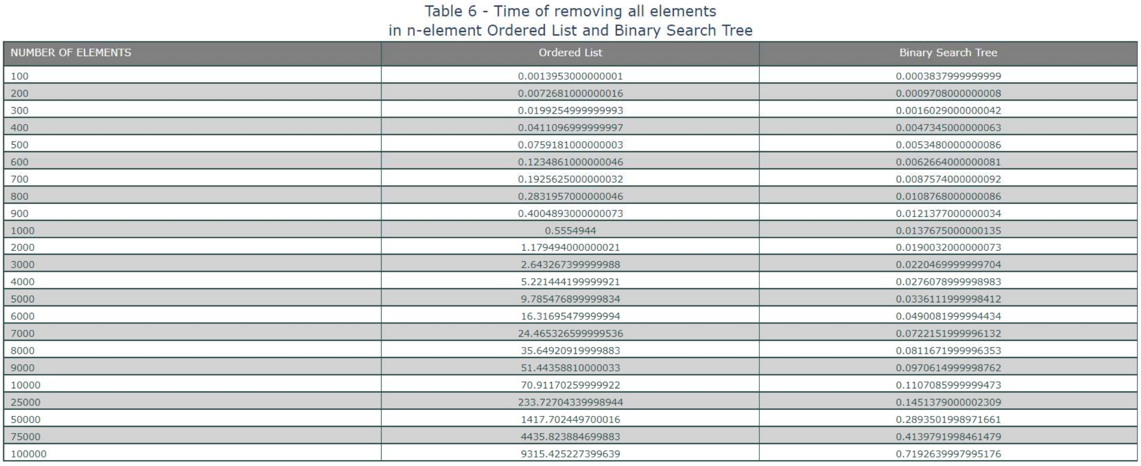


Table 6 - Time of removing all elements
in n-element Ordered List and Binary Search Tree

NUMBER OF ELEMENTS	Ordered List	Binary Search Tree
100	0.0013953000000001	0.0003837999999999
200	0.0072681000000016	0.0009708000000008
300	0.0199254999999993	0.0016029000000042
400	0.0411096999999997	0.0047345000000063
500	0.0759181000000003	0.0053480000000086
600	0.1234861000000046	0.0062664000000081
700	0.1925625000000032	0.0087574000000092
800	0.2831957000000046	0.0108768000000086
900	0.4004893000000073	0.0121377000000034
1000	0.5554944	0.0137675000000135
2000	1.179494000000021	0.0190032000000073
3000	2.643267399999988	0.0220469999999704
4000	5.221444199999921	0.0276078999998983
5000	9.785476899999834	0.0336111999998412
6000	16.31695479999994	0.0490081999994434
7000	24.465326599999536	0.0722151999996132
8000	35.64920919999883	0.0811671999996353
9000	51.44358810000033	0.0970614999998762
10000	70.91170259999922	0.1107085999999473
25000	233.72704339998944	0.1451379000002309
50000	1417.702449700016	0.2893501998971661
75000	4435.823884699883	0.4139791998461479
100000	9315.425227399639	0.7192639997995176

I have to say that I will never use again ordered list because waiting for the result was terrible. Binary search tree is so good that I love this approach.

Memory activity:

Binary tree need more memory to allocate right and left child for every node (even if it is None) Ordered list need only 1 none pointer at the end of the structure, but the time shows that even though BST is better.