[Avg inserted]

randomized	type	
False	AVL	19424.8
	BST	19424.8
	LIST	19413.6
True	AVL	20000.0
	BST	20000.0
	LIST	20000.0

[Avg insert time]

randomized	type	
False	AVL	3381.882
	BST	7.684
	LIST	2948.296
True	AVL	7760.172
	BST	8.254
	LIST	5784.108

[Avg found]

randomized	type	
False	AVL	4355.0
	BST	4355.0
	LIST	4312.8
True	AVL	4460.4
	BST	4460.4
	LIST	4527.8

[Avg search time]

randomized	type	
False	AVL	3.514
	BST	4.514
	LIST	4243.600
True	AVL	3.972
	BST	4.198
	LIST	4676.482

[Avg removed]

randomized	type	
False	AVL	3851.6
	BST	3770.8
	LIST	3861.0
True	AVL	3990.8
	BST	3951.0
	LIST	3997.8

[Avg remove time]

randomized	type	
False	AVL	4140.280
	BST	8.504
	LIST	5979.704
True	AVL	5169.330
	BST	8.394
	LIST	7744.036

Results summary:

- For inserting, The most effective is BST tree (400x faster than second LIST)
- For inserting, The least effective is AVL tree (probably due to tree rotations
- For searching, The most effective is AVL tree
- For searching, The least effective is LIST (1000x slower than second BST)
- For removing, The most effective is BST tree (500x faster than second AVL)
- For removing, The leat effective is LIST

Conclusions:

- If we want to keep overall performance on high level, we should use BST tree
- If we want to focus mainly on searching in tree (without mutating it), we should use AVL tree
- If we don't care about performance but we focus on algorighm simplicity, we should use LIST