Binary Search Trees - Built Randomly

We want to insert n elements into a BST so that they will be stored in sorted order.

InOrder walk: 1 2 3 5 6 7 8

Insertion: nothing fancy. Just read elements and insert into current tree.
Given array of elements: 3 1 8 2 6 7 5

This is a sorting algorithm.
Given the very simple BST-sort/construction algorithm
- how long can it take to build the BST?
- what shape will it have? How balanced? What depth?

Depends on input sequence

1 2 3 4 5 \rightarrow \begin{tikzpicture}
  \node (1) at (0,0) {1};
  \node (2) at (1,1) {2} edge (1);
  \node (3) at (2,1) {3} edge (2);
  \node (4) at (3,1) {4} edge (3);
  \node (5) at (4,1) {5} edge (4);
\end{tikzpicture}

Can be bad: \( O(n) \) depth
\( O(n^2) \) time
\( \Omega(n \log n) \) worst-case time: sorting lower bound
Even for a balanced tree, \( \Theta(n) \approx \frac{n}{2} \) nodes have height = \( \Theta(\log n) \) so it must take \( \Omega(n \log n) \) time to build.

Any algorithm producing any tree shape: \( \Omega(n \log n) \) time

So ... if lucky, \( \Theta(n \log n) \) time
if unlucky, \( O(n^2) \) time

Sounds like ... quicksort
Stable quicksort

1 8 2 6 7 5

- Use first elt to partition
- Repeat on each side

quicksort round 1: compare all els to 3
BST sort: 3 = root; eventually all els pass through.
quicksort: partitions into 2 groups <3 & >3
each is independent

BST sort: same
exactly same comparisons but in different order

3rd round

4th round
Same tree as BST