Solution no. 6

2. Many solutions are possible here. One is to keep a sorted list of the elements, in addition to putting them in an AVL tree. This can be done simply by adding 'next' and 'previous' pointers to each element in the tree. It is easy to maintain the list structure withought affecting the complexity of the tree operations. By maintaining a pointer to the smallest element, it is also easy to print the first k elements in time $\Theta(k)$.

3. This is proven via induction. The recursive formula of the best-case compexity is given by

$$T(n) = n + \min_k \{T(k) + T(n-k-1)\}$$

   Assuming that $T(k) \geq Ck \log k$ for every $k < n$ and doing some calculus, one obtains that $T(n) \geq Cn \log n$.

4. Think of the a_i's as bolts (umim) and and of the b_i's as nuts (bragim). You can only compare nuts to bolts. Use a random nut a_i to partition the bolts, and them use the bolt which matches a_i to partition the nuts. The smaller bolts in the bolt-partitionmatch the smaller nuts, and the larger bolts match the larger nut, so it is possible to continue by recursion.

   This sorts the a_i's as well as the b_i's, in time equivalent to that of the quick-sort algorithm, namely $\Theta(n \log n)$ in the expected case.

5. A node is called a left-child if it is the left child of its parent. As long as the number of left-children is smaller than $n-1$, it is possible to increase that number by applying a rotation (prove!). After at most $n-1$ rotations, it is therefore possible to obtain a left-bound string. The 'rotation-distance' of every tree with n nodes from a left-bound string is therefore at most $n-1$. Hence the distance between every two trees is at most $2n-2$.

6. Use the idea from class of dividing into bins, only divide into bins of length $\frac{1}{n^2}$ instead of $\frac{1}{n}$. To keep the running time linear in n. Divide these bins into mega-bins, each containing n consecutive bins. For each element write to which mega-bin it belongs, and to which bin in this mega-bin. It is now possible to sort the elements in linear time according to their respective bins, using radix sort.