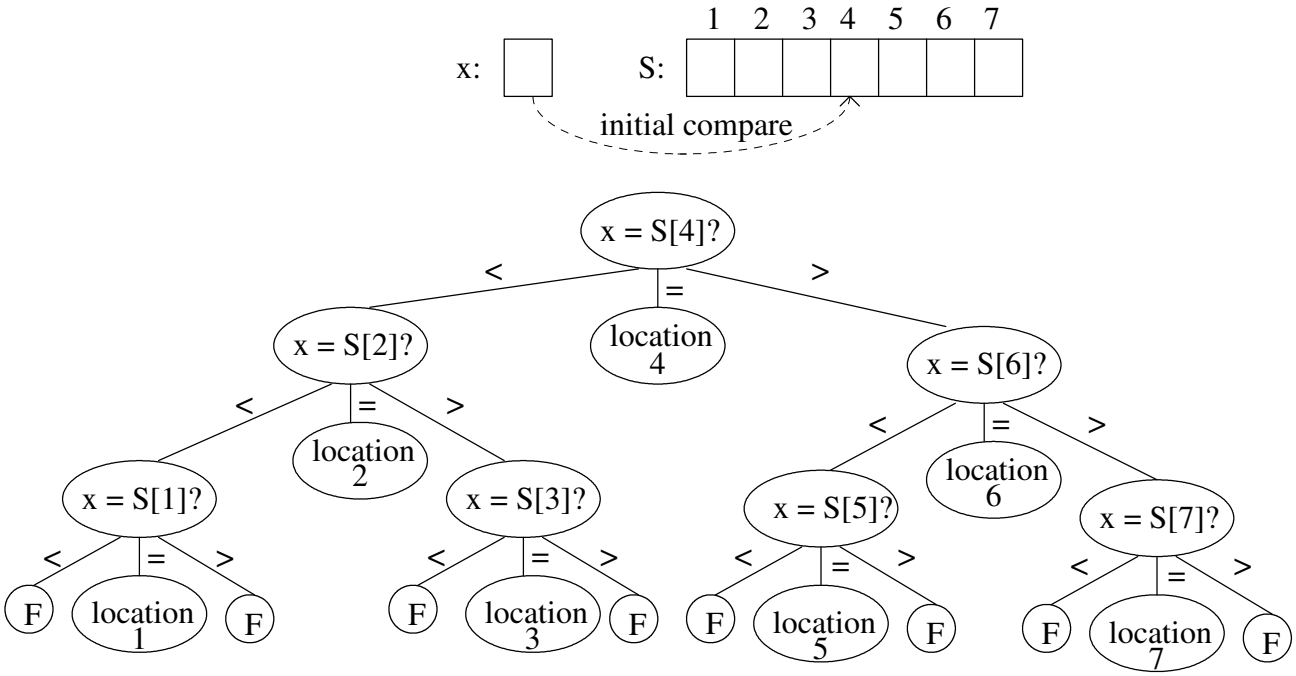
Chapter 8: Computational Complexity for Searching Problem

Goal: We what to determine a lower bound on the **best possible** searching algorithm (for the worst-case problem instance -- data arrangement) that compares items.

**Result**: **best possible** searching algorithm that compares items must do at least $\Theta(\log_2 n)$ comparisons.

For any searching algorithm, we can draw a decision tree. The following decision tree is for binary searching of an array S with 7 elements and a target of x.




1) Draw a similar decision tree for sequential search of unsorted array S with 7 elements and a target of x.

2) What are the general characteristics of decision trees for search algorithms that compare elements?

3) What part of the decision tree represents the worst-case behavior for searching?

4) To prove the result ("**best possible** searching algorithm that compares items must do at least $\Theta(\log_2 n)$ comparisons"), using the decision tree what are we going to have to show (proof)?

5) How might we get around this bound to do better than a $\Theta(\log_2 n)$ searching algorithm?