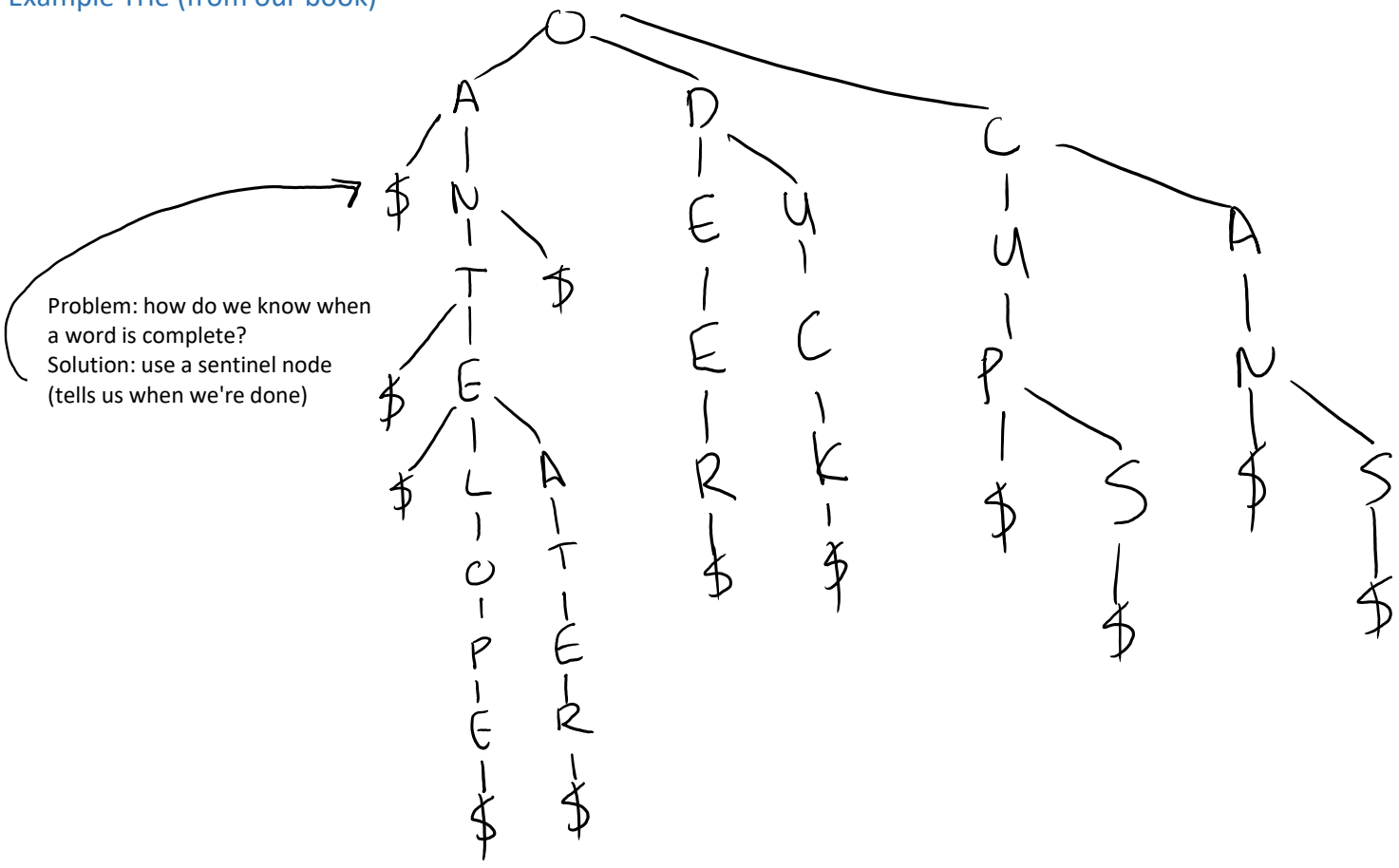
2018-02-15 Tries & AVL Trees

Thursday, February 15, 2018 8:59 AM

PA3

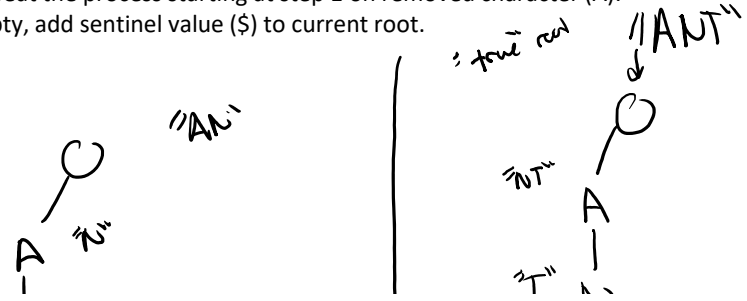
- Use a trie to loop up word spellings
- Given a dictionary, build a trie of this dictionary, then do some sort of word lookups
  - E.g. if we type "te" into our program, the trie should return "test, temp, tetris, etc."
- A sorted vector would work and allow for efficient lookups (LogN), but inserts are still slow (N)
- A trie is an N-way tree that supports efficient lookups (LogN) and inserts (LogN)
  - Each node in the trie represents a single character in a word
  - Performing a pre-order walk of the tree will provide a complete word

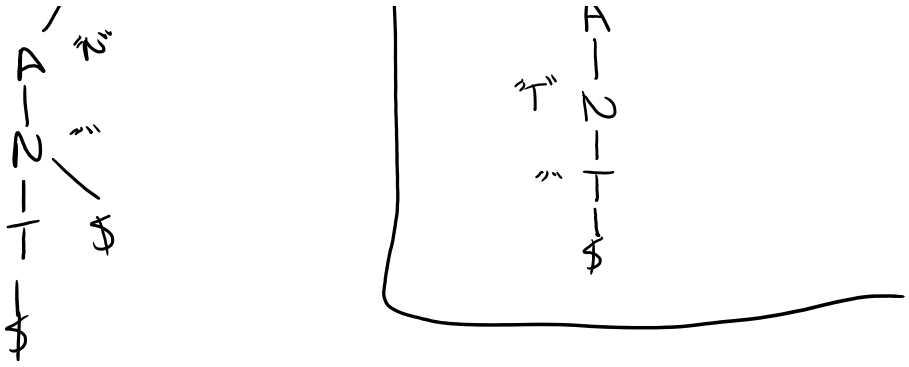
Example Trie (from our book)



How do we build a trie?

1. Given a word (e.g. ANT), look at front character. If the root does not have that character (A), add that value to the root as a child.
2. Remove the character from the string (NT). If string is not empty, recursively repeat the process starting at step 1 on removed character (A).
3. If string is empty, add sentinel value ($) to current root.





Removes in BST