Motivation

* A prefix tree or a Trie (pronounce as [tri:] or try) is a compact data structre for representing a set of strings, such as all the words in a text. (or a dictionary)
* A tries supports pattern matching queries in tim proportional to the pattern size. (length of searched string) O(d), d is the lenght of the pattern string.

Thought Tries are most commonly used for searching character strings, they don’t need to be.

Operations on Trie

* We mainly focus on the insertion and search.
  + We will not talk about deletion this time.
* Once we created the Trie, we can search and find whether a word in the Trie or not?
  + Most frequently used scenario
* Or we can search the Trie, to tell whether a string(partial word) we ssent in could possibly be a prefix of any valid English word in the Trie.

Search a String in Trie

If we search “stop”, start from root

1. First letter in prefix is ‘s’, we check whether there is a child of the root (symbolized as D) that is associated with ‘s’,
2. If there is not such child, return not found
3. If this is a base case, return true. (prefix to be searched contains one letter, and current tree node has a child associated with that letter.)
4. Otherwise, search the rest of the prefix → ‘top’ in the subtree rooted at **D.**

Look at Sample code in demo code folder on canvas. Makes alot of sense.