Recitation 7:

Tries + Intro to Hashing

CIT 594

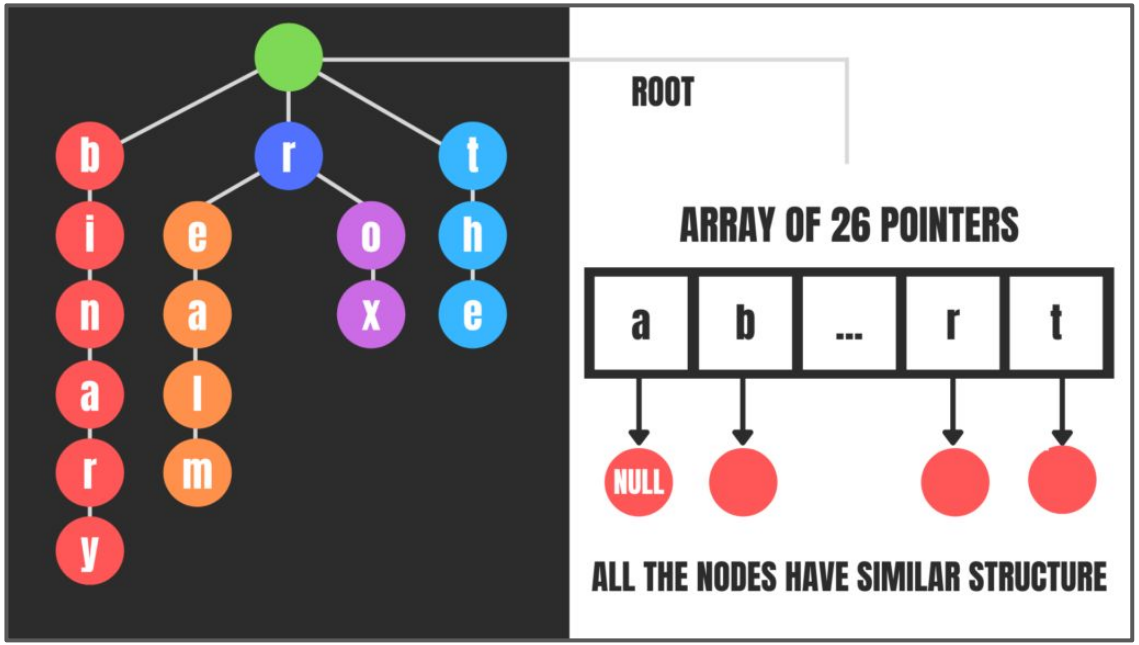
March 22, 2023

Announcements

- Recitation 6 Assignment
  - Weird due date last week, extended to **March 23rd (**if you haven't already submitted)
- Recitation 7 Assignment
  - Due Date: March 23rd @ 11:59 PM
- HW4 Reflections?
  - Comments, questions, concerns, etc.
- HW5
  - Due April 3rd @ 11:59PM ET
  - Group OH Date: March 26th @ 5:15 PM in Berger Auditorium

Tries

Tries



- Trie (“try”) is a tree where each vertex represents a single word or a prefix
- A vertex that are *k* edges of distance of the root have an associated prefix of length *k*

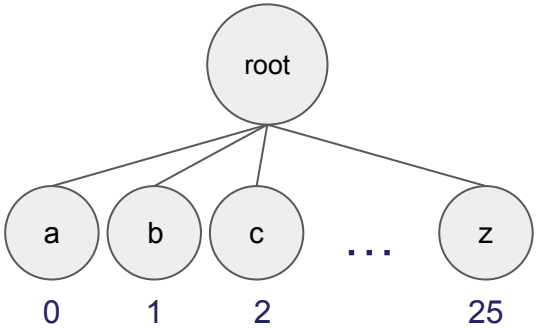
The Alphabet and Indexes

A `char` in java is represented by an `int` ASCII code
https://www.asciitable.com/

Since a trie node needs to fit 26 pointers to the alphabetical characters, we need to convert from ASCII to indices from 0-25

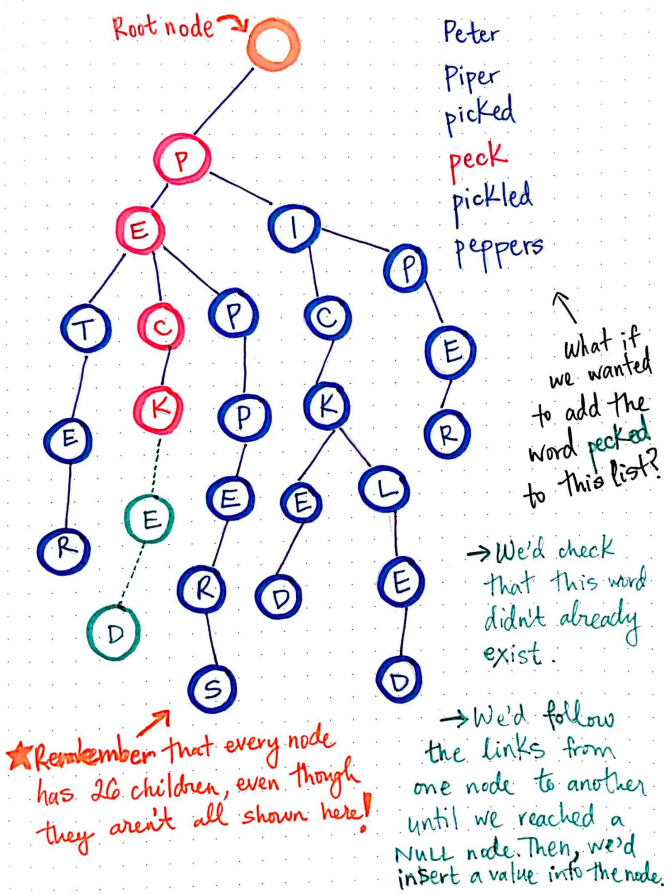
a
ASCII: 97 → a
Trie index: 0

z
ASCII: 122 → z
Trie index: 25

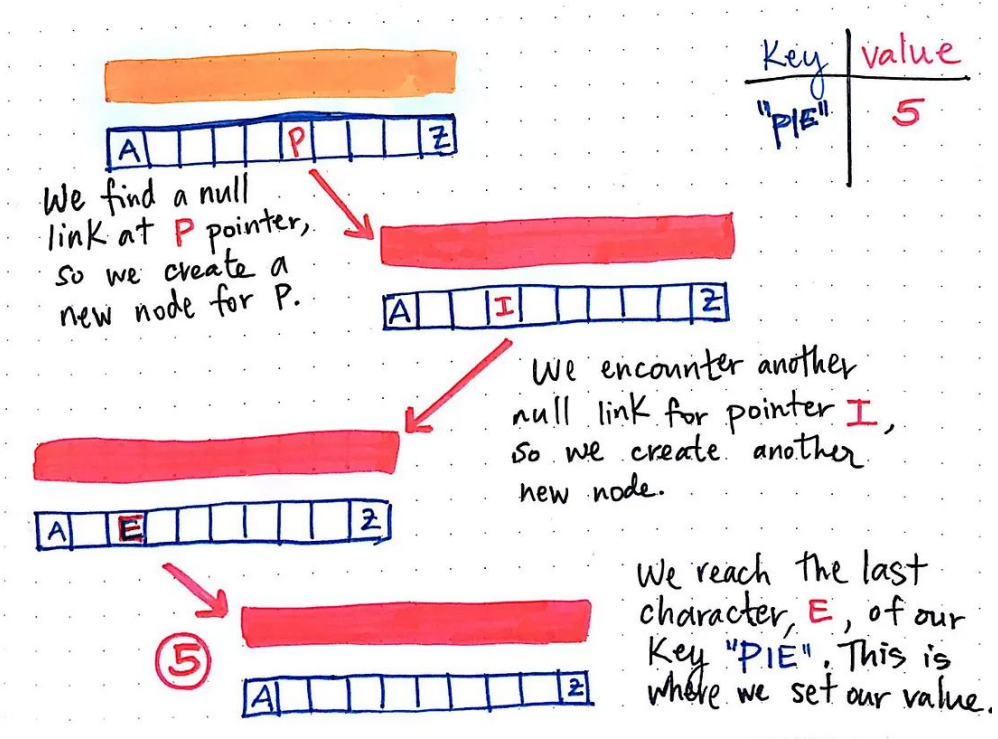


Why do we use Tries?

- Space Complexity
  - A trie greatly reduces the amount of space needed to store keys
  - Ex: P-E prefix has 3 specific branches that could be utilized to form words



Visual Representation: Lists



Ex: Inserting “Pie” into a Trie

- Check if ‘P’ is null, create a new node
- Continue on until the last character is reached
- Indicate at E that there is a full word from the previous letters

Fields + Methods

Fields:

- Words (int): # of words that match the given string
- Prefixes (int): # of words that have the prefixes of the vertex
- Edges (arr[26]): references to all the possible sons (letters in the alphabet)

Methods:

- addWord: Adds a single word to the trie
- countPrefixes: Counts the number of words in the trie that have the given prefix
- countWords: Counts the number of words in the trie that have the exact given word

Introduction to Hashing

Hash System, Collision, and Load Factor

Hash System:

- Consist of the hash table, which is for storing data, and hash function, which finds the position for a record to be stored in hash table

Collision:

- Two keys are mapped to the same position in HT

Load Factor:

- n/M (n = # of records in HT; M = # of slots in HT)

Use:

- Exact-match search query applications, but not range search

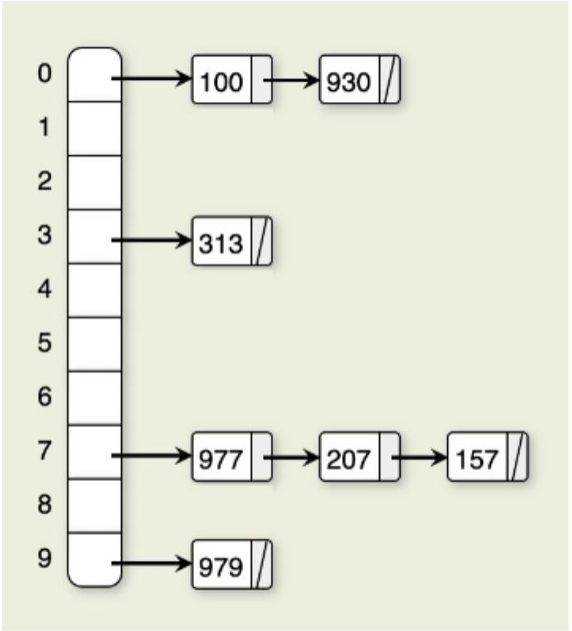
Runtime:

- Insertion, deletion and search all O(1)

Open Hashing

Open Hashing

- It is okay to hash two keys to the same position, just link it up
- Uses array of linked lists to resolve the collision
- Insert at head of the linked list to save time



Homework 5:

Autocomplete Review + GUI Demo

https://www.cis.upenn.edu/~cit5940/current/assignments/hw05/

Recitation Activity:
Implement Insert and Search in a Trie!