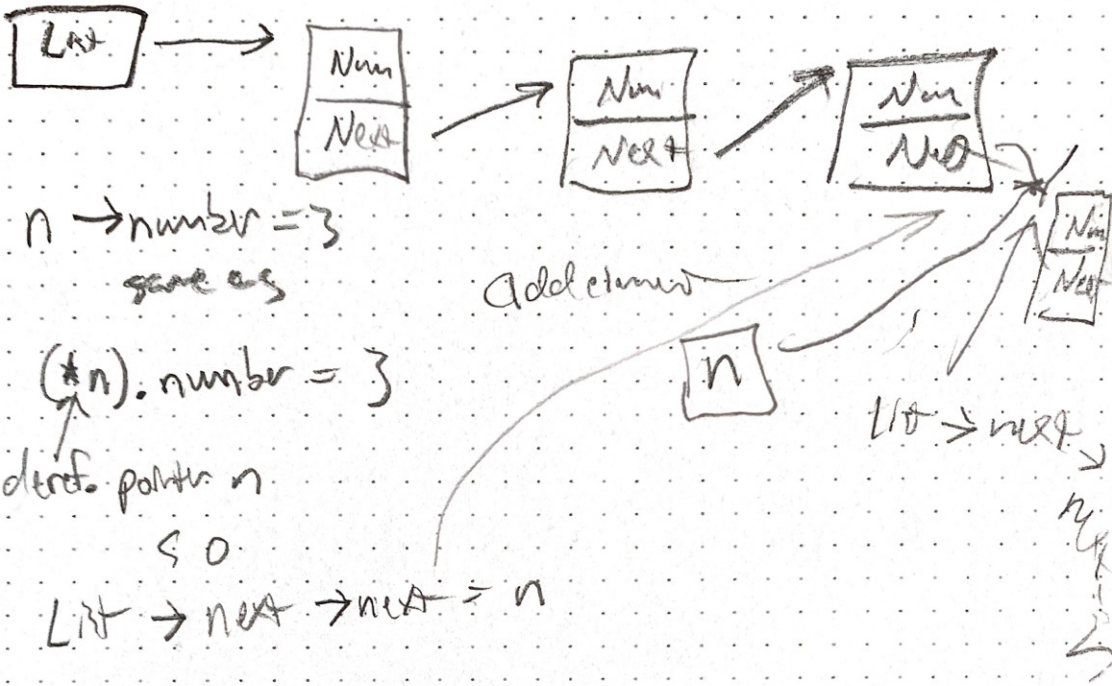
Linked list



n → number = 3

save as

(*n).number = 3

deref. pointer n

≤ 0

List → next → next = n

(*(*list).next).next = n

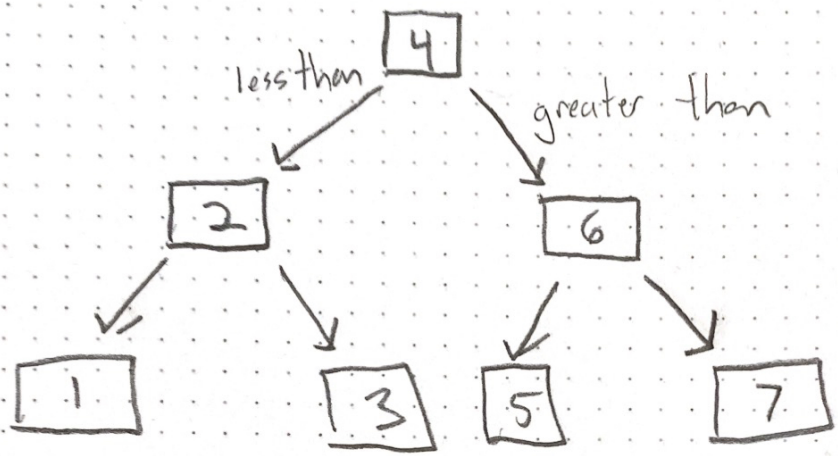
messy notation

Instead, List points to next, which points to next

list → next → next = [the address of] n

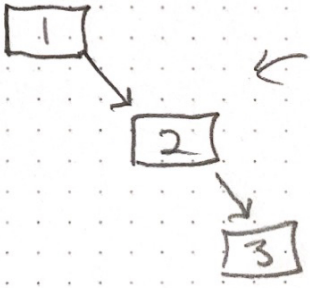
Introduce

Binary Trees



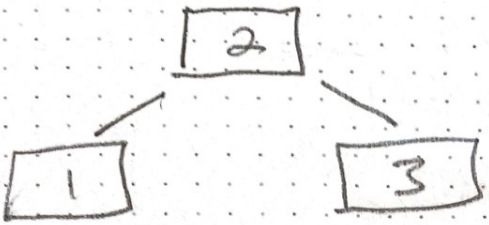
- recursive structure
  - each tree has 2 children
  - each child is a tree
- search is efficient
- recursive data structures lend themselves to recursive functions

* a devolution of tree into linked list solution



← technically binary tree but also linked list

Make 2 root node

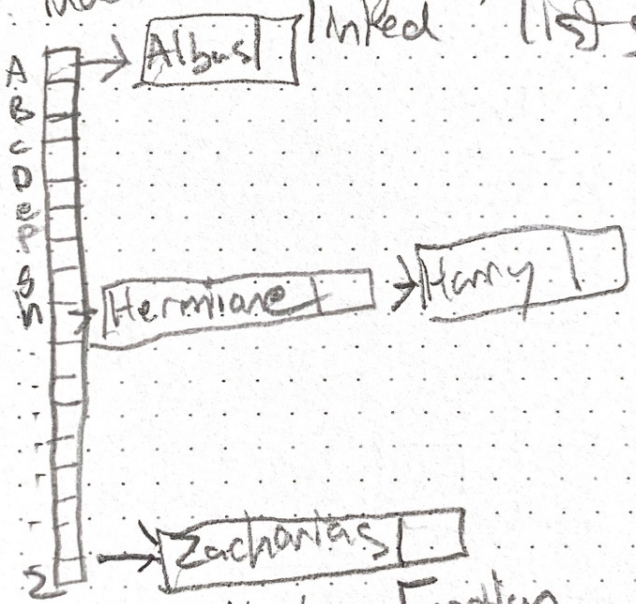


Can we search in constant time

Yes - with Hash tables

* we use ASCII to convert index

an array of linked lists



Hash Function

Input → Hash function → Output

char *

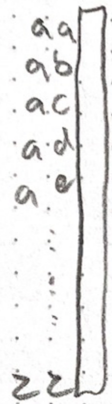
Zacharias →

returns int (hola)

→ 25

To solve inefficiency of Hash search

Solution 1 → grow Hash table



But Harry and Hagrid go to same location

we could use first 3 letters..... But we grow the table exponentially

$26 \rightarrow 26^2 \rightarrow 26^3$ etc

Tries (or prefix tree)

a tree made up of arrays