Applications of Trees

There are many problems that are easily solved with trees. We'll examine two types of trees that are commonly used.

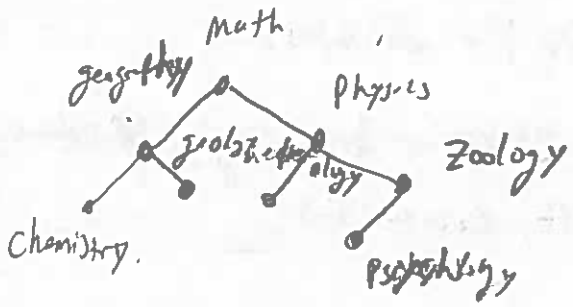
Binary Search trees:

One of the most common tasks in CS is searching for items in a list. We've discussed a binary search on lists, there is a tree version as well.

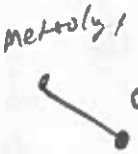
The procedure is to take a list & construct a Binary Search Tree that can be efficiently searched.

Ex: Form a binary search tree for math, physics, geography, zoology, meterology, geology, psychology, chemistry using Lexicographical ordering.

Phys > Math



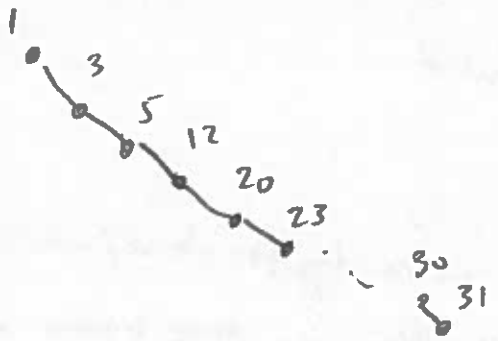
Ex: Add oceanography.



Note this is very different from our previous binary search where we had a sorted list that we could go halfway through etc. Here we construct a list in order. This can easily lead to an un balanced tree.

However the Search is the same! check root if smaller than target go right else go left.

Ex: Construct BST for [1, 3, 5, 12, 20, 23, 24, 29, 30, 31]



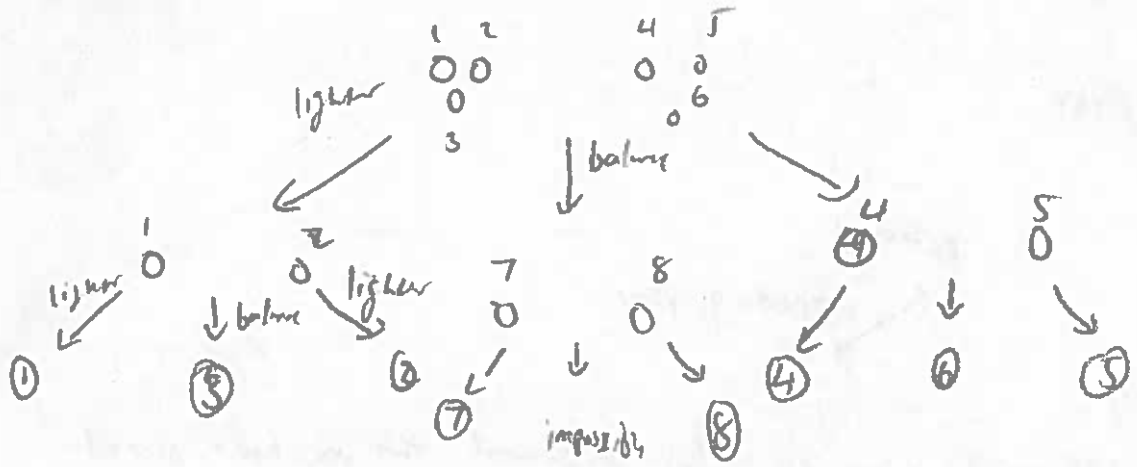
this is essentially just a linked list, not efficient for searching.

However there are algorithms to re-balance unbalanced trees, covered in data structures & Algorithms.

Decision trees:

Trees can also store info for decisions.

Ex: How many weighings are necessary for weighing 8 identical coins and one counterfeit to find the counterfeit.



depth 2 => 2 weighings total.

Prefix Codes

How do you represent English on Computers? ASCII or Unicode.

There are 26 letters => 5 bits per letter total of 32 options => 6 unused values.

What if you need to use less space? Can you?

Ex: Suppose we have 5 letters e, a, t, n, s => 3 bits (8) for each letter

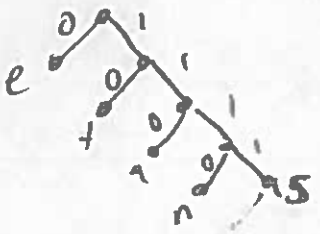
if our encoding is 000, 001, 010, 011, 100

to write eats we'd send 000 00 1 010 011 100 => 15 bits.

Can we do better? Yes! rank the letters in frequency e, t, a, n, s

Now use prefix codes essentially the more frequent the letter the shorter the representation. Our prefix codes will be 1s which mean go right 0 means go left (end of letter).

i.e.,



So e = 0
t : 10
a : 110
n : 1110
s : 1111

=> eats = 0 10 110 1110 1111

=> 14 bits

So much more efficient!

The larger the space can ruin efficiency, due to long strings however, if only the first couple letters are used, very efficient.

This encoding is unique

Ex: Decode 1111,110,1110,0

s a n e

Many more applications of trees!