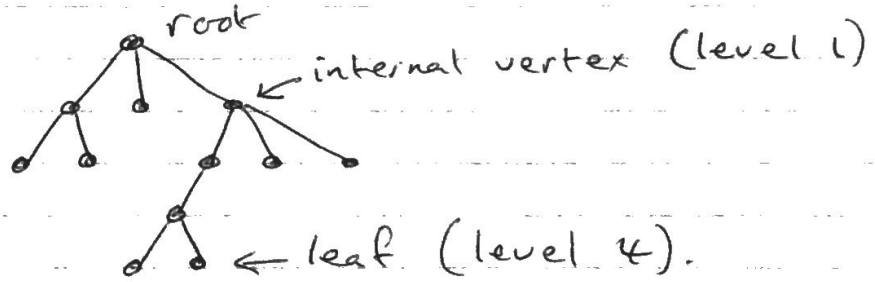
Review of Chapter 11 Trees.

A tree is a connected simple graph with no simple circuits.

A rooted tree is a tree where one vertex has been labelled the root. Put it at the top.

In a rooted tree we can have leaves, internal vertices, children etc.

Example ①



This rooted tree has 5 internal vertices and 7 leaves. It is a 3-ary tree since every vertex has at most 3 children. Its height is 4.

Formulas

- A tree with $n \geq 1$ vertices has $n-1$ edges.
- A full m-ary tree with i internal vertices has $mi+1$ vertices in total.

Example ② Suppose a full binary (2-ary) tree has 300 leaves. How many edges does it have?

Solution: Let i be the number of internal vertices and n the total number of vertices. Then

$$n - i = 300 \quad \text{(leaves)}$$

and $$n = 2i + 1 \qquad (m = 2).$$

So $$i + 300 = n = 2i + 1$$

and
$$\begin{array}{rrrrr} i & +300 & = & 2i & +1 \\ -i & & & -i & -1 \\ \hline & 299 & = & i & \end{array}$$

Means $$n = i + 300 = 599$$

and there are $n - 1 = \boxed{598 \text{ edges}}$.

Binary search trees

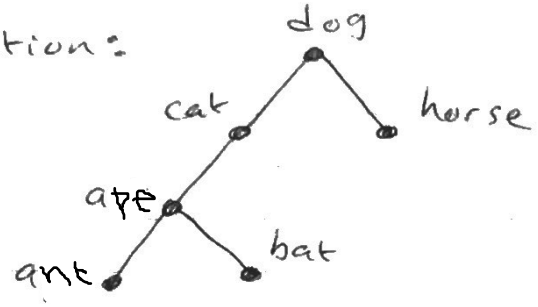
Given a list, for example words that can be ordered lexicographically, we can store the elements in a binary tree so that they are easy to find:

- Put the first element (word) at the root.
- For each next word go to the left child if it comes before and to the right child if it comes after. Put this word at the first open spot.

Example (3) Make a binary search tree for:

dog, cat, horse, ape, bat, ant, cow

Solution:

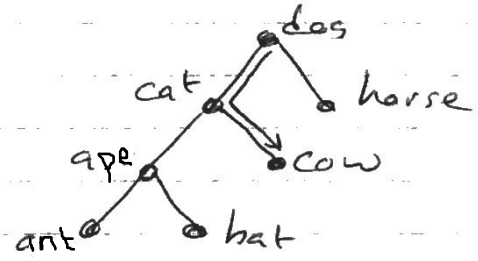


to add the last word cow we see

cow < dog

cat < cow

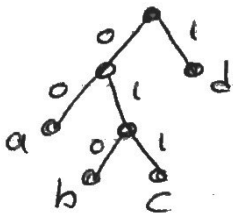
so



is the binary search tree for these words.

Another application of binary trees is to prefix codes. Label each leaf with a letter and left edges 0, right edges 1. Following a path from the root to the leaf gives each letter a bit string.

Example (4)



→

prefix code

a = 00

b = 010

c = 011

d = 1

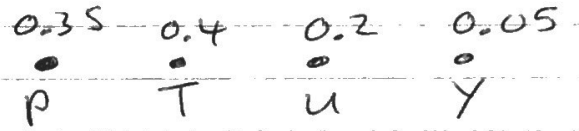
A Huffman code is an example of an efficient prefix code (uses fewest bits possible).

Example (5) Construct a Huffman code for these letters with frequencies

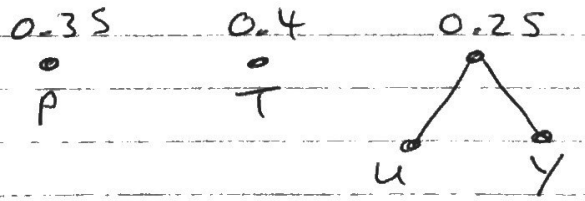
P: 0.35, T: 0.4, U: 0.2 Y: 0.05

Use your code to encode PUTTY and find the average number of bits used per letter.

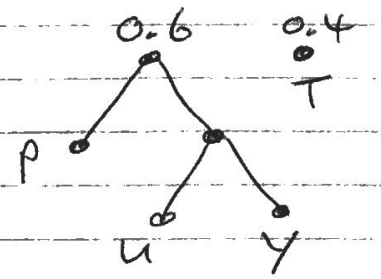
Solution: Start



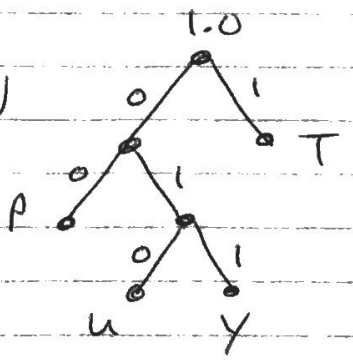
and at each stage combine the trees with the smallest weights and add the weights to get the new weight



then



and



so

P = 00, T = 1, U = 010, Y = 011

and PUTTY is 00 010 1 1 011

Average bits per letter is

$$2(0.35) + 1(0.4) + 3(0.2) + 3(0.05)$$

(2 = bits for p; 0.35 = freq. of p)

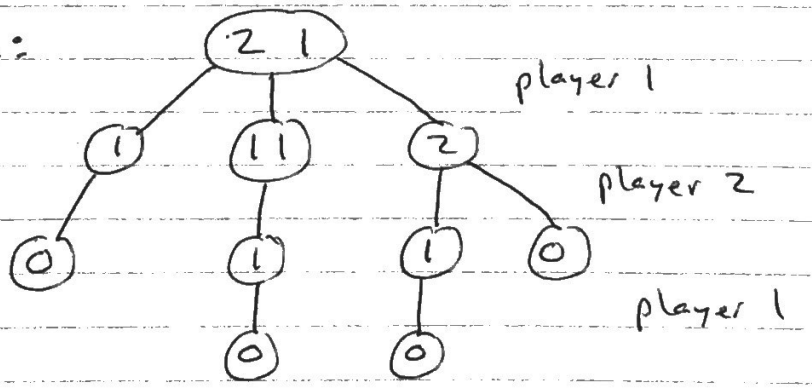
$$= \boxed{1.85}$$

Decision trees can be used to systematically analyze a problem. Game trees are an example of this.

In the game nim two players remove stones from piles and the player taking the last stone wins. You can only take stones from one pile.

Example (6) Draw the game tree for nim played with 2 stones in one pile and 1 stone in the second pile.

Solution:



As decribed in the notes we can label each vertex +1 for a winning position for p1 and -1 for a winning position for p2. Can you see the winning strategy for player 1?

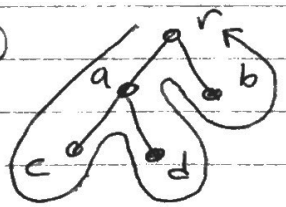
Tree traversals give different ways to order the vertices in a rooted tree. Make a curve around the tree, starting and finishing at the root.

Preorder: List a vertex the first time you pass it on the curve.

Inorder: List leaves the first time you pass them and internal vertices the second time.

Postorder: List a vertex the last time you pass it.

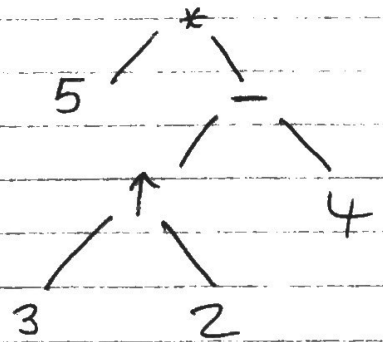
Example (7)



Preorder: r a c d b
Inorder: c a d r b
Postorder: c d a b r

Example (8) Write $5(3^2-4)$ in prefix notation and postfix notation.

Solution: First display as a binary tree



prefix notation is preorder

$* 5 - \uparrow 3\ 2\ 4$

postfix notation is postorder

$5\ 3\ 2 \uparrow 4 - *$

In prefix notation the operation is in front of the two numbers $\underline{- 9\ 4}$ is $9-4=5$

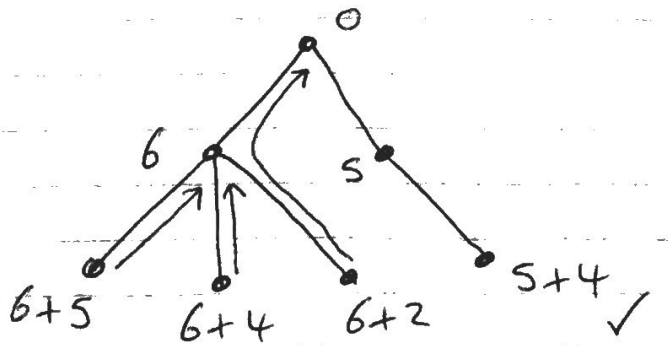
In postfix it's at the back $\underline{9\ 4\ -}$ is $9-4=5$.

A spanning tree of a graph G is a subgraph that is a tree and contains every vertex of G.

We saw that a useful way to make a spanning tree is with a depth-first search. You make a path as long as possible with no simple circuits. When you get stuck, backtrack until you can continue the path in a new direction.

Example (a) Use a depth-first search to find a subset of $\{6, 5, 4, 2\}$ that sums to 9.

Solution: Start with 0 at the root and add the numbers in order. If the sum gets too big then backtrack by removing the last number.

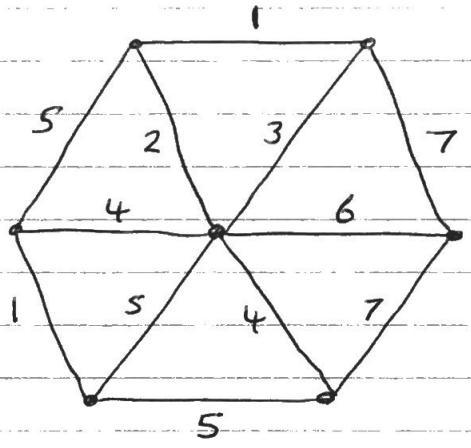


Answer is 5+4.

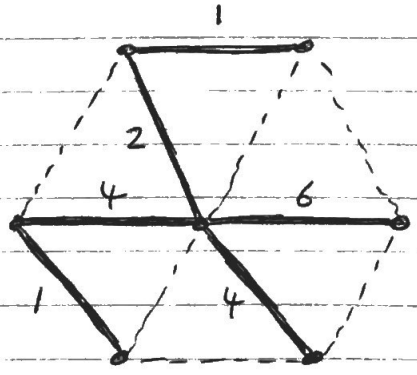
A weighted graph gives each edge a number.

A minimum spanning tree for a weighted graph is a spanning tree with smallest possible total weight. There are two simple algorithms by Prim and Kruskal to find them.

Example (10) Find a minimum spanning tree for this graph and give its total minimum weight.



Solution: We'll use Kruskal's algorithm. Just add the smallest weight edge at each step, avoiding simple circuits:



total weight of minimum spanning tree is 18.