Applications of Probability

Huffman Codes

for simplicity, assume just 26 letters plus space

suppose we want to encode text to transmit across a communication channel (noiseless) and we want to transmit as few bits as possible

trivial approach: 27 symbols to represent, need 5 bits to cover 27 symbols, give each symbol a different 5-bit number as its encoding

is this an efficient approach? why or why not?

should ~~need to~~ take frequencies of different letters into account, give shorter encodings to more frequently occurring letters and longer encodings ~~ones~~ to less frequent letters

Example - suppose we only have ~~7~~ letters plus space w/ following frequencies:

space - 25%	c - 7%
e - 25%	k - 5%
a - 15%	x - 2%
t - 20%	z - 1%

w/ basic encoding, each character uses 3 bits
and $E[\text{char. length}] = 3$

transmit 1,000,000 characters w/ distribution as given $\Rightarrow$ use 3,000,000 bits

instead, consider the following encoding:

space : 11	c : 0001
t : 10	k : 00001
e : 01	z : 000001
a : 001	x : 000000

$$E[\text{char length}] = 2 \times (0.25 + 0.25 + 0.2) + (3 \times 0.15) + (4 \times 0.07) + (5 \times 0.05) + 6 \times (0.02 + 0.01)$$

$$= 2.56$$

transmit 1,000,000 characters w/ given distribution $\Rightarrow$ use 2,560,000 bits

savings of 440,000 bits!!

(Aside: What is an alternative approach to saving bits? Compression - discuss and compare w/ Huffman codes - compression not as efficient but can use standard off-the-shelf software and applies to any type of information) the encoding I gave has a very special property - anyone see what it is?

no code word is a prefix of any other code word - explain this

why is this property important??

since this is a variable length code, the number of bits in a code word is not known beforehand - need this property to know when a code word ends

suppose we had e : 01 , a : 011

if we see 0110....

we don't know whether this is $\underbrace{01}_{e}\ \underbrace{10}_{t}\ 0....$

or $\underbrace{011}_{e}\ 00...$

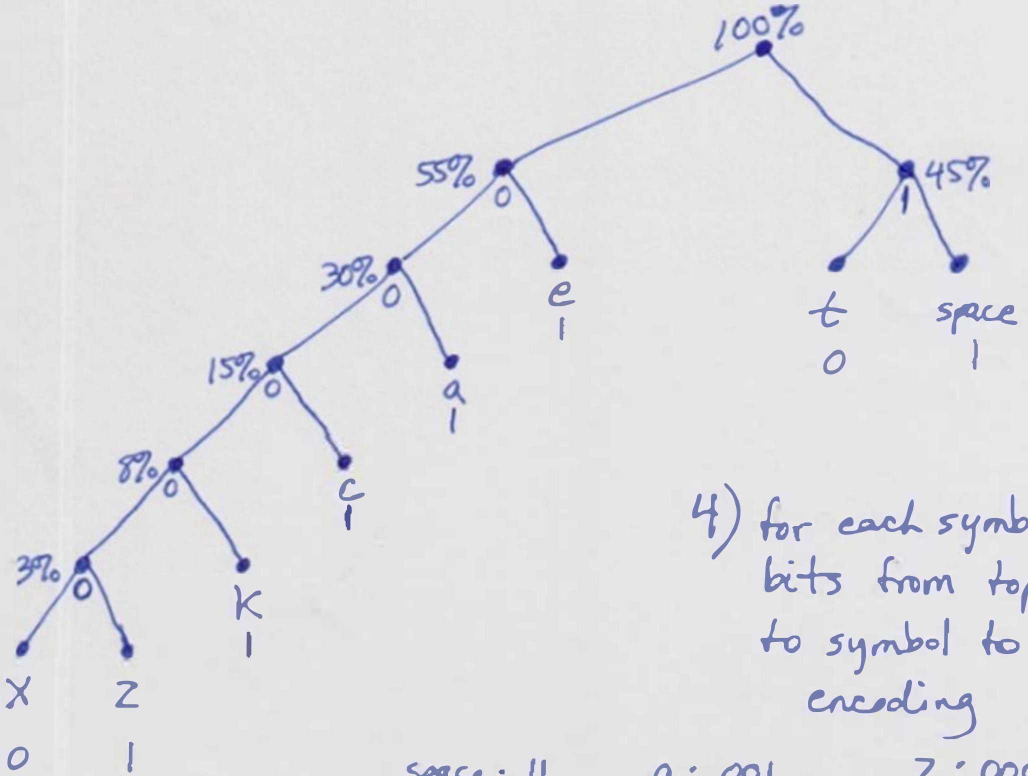
however, w/ stated property, the end of each code word is unambiguous (because it is not a prefix for any other code word)

the given encoding was constructed w/ a Huffman tree and is referred to as a Huffman code

Huffman codes are provably the most efficient codes (shortest expected length) w/ the property that no code word is the prefix of any other

Constructing a Huffman Code

1) take 2 least frequent symbols (real or logical) - assign their last bits to be 0 and 1, respectively
2) logically combine these 2 symbols into 1 symbol and add their frequencies together to get the frequency of this new logical symbol
3) repeat Step 1 and continue until all symbols are combined

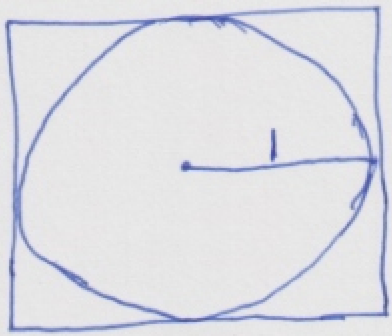


4) for each symbol, follow bits from top of tree to symbol to get its encoding

space : 11	a : 001	Z : 000001
t : 10	c : 0001	X : 000000
e : 01	k : 00001	

should be clear from the way the tree is constructed why no code word is a prefix of any other

~~Calculating~~ Estimating Pi using Random Numbers



Area of circle $= \pi r^2 = \pi$

Area of square $= 2 \times 2 = 4$

Throw darts randomly at square

$$P[\text{a dart lands in circle}] = \frac{\text{area of circle}}{\text{area of square}}$$

$$= \frac{\pi}{4}$$

Suppose you throw $n = 1{,}000{,}000$ darts at square and Z of them land in circle then, we estimate that

$$\frac{\pi}{4} = \frac{z}{n} \Rightarrow \pi = \frac{4z}{n}$$

we can do this w/ a program that generates random points in the unit square (random x value in [-1, 1], random y value in [-1, 1]) and calculates how many are inside unit circle (dist. from [0,0] ≤ 1)

(show random Pi estimator program in Eclipse)

this is an example of Monte Carlo simulation, which is the use of random experiments to numerically estimate values