7.7. Source coding with Fixed length code

- Source S = {s1,s2,…,sq} with probability P(S) = {P(s1), P(s2),…, P(sq)}
- Codeword has fixed length l → L =l
- Each available combination must have length of l
  - Number of available combination (M) is r^l where r is base of code or number of different code symbols.
  - To satisfy uniquely decodable: minimum number of available combinations equals q where q is number of source symbols.
    - $r^l = q$ → l =$log_r q = H_r(S)max$ →efficient code
      - So, the fixed length code will be efficient code when q = r^l
  - If $r^l \neq q$, we need to choose number of available combination r^l > q
    - According to Shannon, $H_r(S) \leq l \leq H_r(S)$ +1
      - Thus, l = l2 +1 where l2 equals smallest integer greater than to $H_r(S)$
      - Efficiency of code = $\frac{H_r(S)}{l}$ = $\frac{H_r(S)}{l2+1}$ < 1

7.7. Source coding with Fixed length code (Cont.)

- To make the efficiency of code can progress to 1, the source needs to be extended
  - S→ S^n: $H_r(S^n)$ = n $H_r(S)$
  - Length of fixed length code for extension source = l_3 +1
    - l_3 smallest integer greater than to $H_r(S^n)$
  - Efficiency of code of extension source $\frac{H_r(S^n)}{l_3+1} < \frac{H_r(S^n)}{H_r(S^n)+1} = \frac{n\,H_r(S)}{n\,H_r(S)+1}$
    - When n→ ∞: efficiency of code → 1

7.8. Run-length coding

- Run-length coding is a simple and effective ways of compressing data in which it is frequently the case that the same character occurs many times in succession.
  - This may be true of some types of image data, but it is not generally true for text, where it is rare for a letter of the alphabet to occur more than twice in succession.
- To compress a sequence, one simply replaces a repeated character with one instance of the character followed by a count of the number of times it occurs.
- For example, the sequence AABBBBCCCDCCDDDBBBBBAAAA could be replaced by A2B4C3D1C2D3B5A4

7.9. Lempel Ziv coding

- Dictionary coding
  - Make a list of all the words and numbers appearing in the text, and then converting the text to a sequence of indices which point to the entries in the list. (The list of words is referred to as the dictionary)
  - It is suitable primarily for text
  - Both the encoder and the decoder need to have access to the dictionary
  - Lempel Ziv is a dictionary code and has different types. Only focus:
    - LZ77
    - LZ78

7.9. Lempel Ziv coding

- LZ77:
  - Build a word according to:
    - Each “new word” is the “old word” plus a new character in the text
    - Each word is represented by a triple (m,n,c)
      - n is the number of symbols of the string (found string) that matches the string that appears before it (n>1)
      - c is the next symbol of the found string
      - m is the number of positions that must be reversed to find the beginning of the found string
    - Text is converted into sequence of triples
  - In decoding process:
    - From the first triple, the text is restored

7.9. Lempel Ziv coding

- LZ77:
- Text: the_fat_cat_sat_on_the_mat.
- Build triples:
  - From beginning: (0,0,t), (0,0,h), (0,0,e),(0,0,_), (0,0,f), (0,0,a), (0,0,t), (0,0,_),(0,0,c), (4,3,s),(4,3,o), (0,0,n), (0,0,_), (19,4,m), (11,2,.)
- Decoding:
  - (0,0,t)->t, (0,0,h)->h (th), …..→ (0,0,c)->c(the_fat_c), (4,3,s)→(the_fat_cat_s)….

7.9. Lempel Ziv coding

- LZ78:
- Find word has only one symbol which is symbol from the beginning of the text
  - First found word is empty word in position 0
- Find word has only two symbols. The first symbol is the found symbol.
- Find word has only three symbols. The first two symbols are the found symbols
- …..
- Each word is represented by a pair (i,c) while
  - i is the position of found word (string of found symbols)
  - c is the next symbol
- Text is the string of pairs

7.9. Lempel Ziv coding

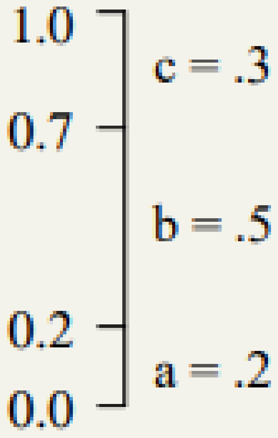
- Example:
- Text: the_fat_cat_sat_on_the_mat.

Item sequence	Code	Item number	Current sequence
t	(0,t)	1	t
h	(0,h)	2	th
e	(0,e)	3	the
_	(0,_)	4	the_
f	(0,f)	5	the_f
a	(0,a)	6	the_fa
t_	(1,_)	7	the_fat_
c	(0,c)	8	the_fat_c
at	(6,t)	9	the_fat_cat
_s	(4,s)	10	the_fat_cat_s
at_	(9,_)	11	the_fat_cat_sat_
o	(0,o)	12	the_fat_cat_sat_o
n	(0,n)	13	the_fat_cat_sat_on
_t	(4,t)	14	the_fat_cat_sat_on_t
he	(2,e)	15	the_fat_cat_sat_on_the
_m	(4,m)	16	the_fat_cat_sat_on_the_m
at.	(9,.)	17	the_fat_cat_sat_on_the_mat.

5.2.4. Arithmetic Coding

Assign each symbol to an interval range from 0 (inclusive) to 1 (exclusive).

e.g.



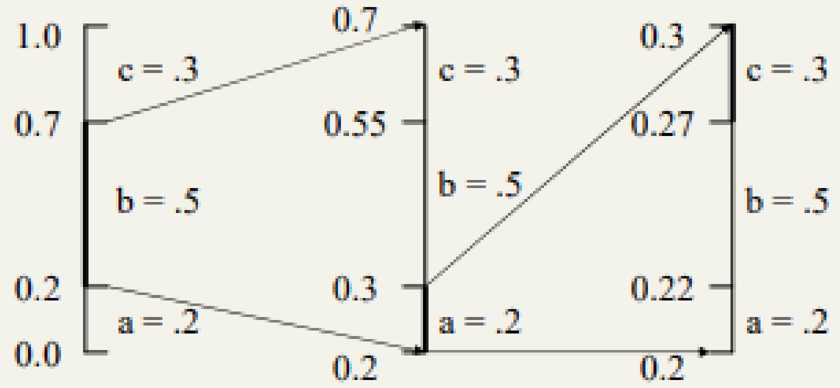
$$f(i) = \sum_{j=1}^{i-1} p(j)$$

$$f(a) = .0,\quad f(b) = .2,\quad f(c) = .7$$

The interval for a particular symbol will be called the **symbol interval** (e.g for b it is [.2,.7))

5.2.4. Arithmetic Coding (Cont)

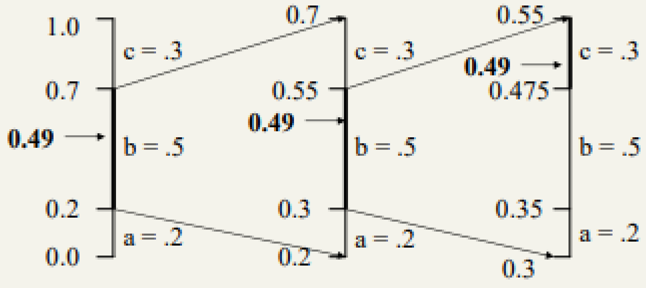
Coding the message sequence: **bac**



The final sequence interval is **[.27,.3)**

5.2.4. Arithmetic Coding (Cont)

Decoding the number .49, **knowing the message is of length 3**:



The message is **bbc.**

7.10. Continuous source coding

- Along with coding, we need to convert the source from analog to digital
  - Sampling
    - After each period of time T $\leq \frac{1}{2}$, one sample is extracted
  - Quantization
    - Each extracted sample value is approximated into one level. The level is integer times of the standard unit
- Obtained results are discrete source. Its output is sequence of levels at time nT
- For continuous source coding, 2 tasks need to be done:
  - Analog to Digital conversion (ADC)
  - Make the number of code symbol used as small as possible (compressing)

7.10. Continuous source coding

- Two tasks can be done in parallel or in any order
  - Compressing before ADC:
    - Amplitude compressing
    - Frequency compressing
  - Parallel:
    - Sampling
    - Calculate discrete derivation of samples (delta)
    - Quantize delta
  - ADC before Compressing
    - Discrete source coding