Huffman Coding and Entropy Bounds

Vasiliki Velona

$\mu \prod \lambda \forall$

November, 2014

Outline

The problem

Transform a symbol string into a binary symbol string with the most economic way


```
01101010011010100010110110010010101100101010101001010101
01111000101011110001101110111000101010010101001101010100
01010100010010010110101000101001011100011001010100100110
00110101010111101011011110100100100010110101010100000101
00110101001101010001011011001001010110010101010100101010
10111100010101111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
00110101001101010001011011001001010110010101010100101010
10111100010101111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
00110101001101010001011011001001010110010101010100101010
10111100010101111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
00110101001101010001011011001001010110010101010100101010
10111100010101111000110111011100010101001010100110101010
00101010001001001011010100010100101110001100101010010011
00011010101011110101101111010010010001011010101010000010
00110101001101010001011011001001010110010101010100101010
10111100010101111000110111011100010101001010100110101010
```

Fixed-length Codes

- Each symbol from the alphabet X is maped into a codeword $C(x)$, and all the codewords are of the same length L.
- For example, if $X = \{a, b, c, d, e\}$then we could use L=3

$$C(a) = 000$$
$$C(b) = 001$$
$$C(c) = 010$$
$$C(d) = 011$$
$$C(e) = 100$$

- There are 2^L different L-tuples, thus for an alphabet of size M we need $L = \lceil \log M \rceil$ bits.

Variable-lengthed Codes

Our aim is to reduce the rate $\bar{L} = \frac{L}{n}$ of encoded bits per original source symbols.

- The idea is to map more probable symbols into shorter bit sequences, and less likely symbols into longer bit sequences.
- We need unique decodability.

  Example: If $X = \{a, b, c\}$ and
  $C(a) = 0$
  $C(b) = 1$
  $C(c) = 01$

- **Solution:** Prefix-free Codes

Prefix-free Codes

- A code is prefix-free (or just a prefix code) if no codeword is a prefix of any other codeword. For example, $\{0, 10, 11\}$ is prefix-free, but the code $\{0, 1, 01\}$ is not.
- Every prefix-free code is uniquely decodable. **Why?:** Every prefix-free code corresponds to a binary code tree, and each node on the tree is either a codeword or a proper prefix of a codeword.
- **Note:** The converse is not true.

Optimum Source Coding problem

Suppose that $X = \{a_1, a_2, ...a_M\}$, with probabilities $\{p(a_1), p(a_2), ...p(a_M)\}$ and lengths $\{l(a_1), l(a_2), ...l(a_M)\}$ respectively, where the lenghts correspond to a prefix-free code

Then the expected value of $\bar{L}$ for the given code is given by:

$$\bar{L} = E[L] = \sum_{j=1}^{M} l(a_j) p_X(a_j)$$

and we want to minimize this quantity.

Kraft's inequality

A prefix code with codeword lengths $l_1, l_2, ..., l_M$ exists if and only if:

$$\sum_{i=1}^{M} 2^{-l_i} \leq 1$$

Proof:

$$\sum_{i=1}^{M} 2^{l_{max}-l_i} \leq 2^{l_{max}} \Rightarrow \sum_{i=1}^{M} 2^{-l_i} \leq 1$$

For the converse:
Assume that the lengths are sorted in increasing order.
Start with a binary tree.Choose a free node for each l_i until all codewords are placed.
Note that in each i step there are free leaves at the maximum depth l_{max}:
The number of the remaining leaves is (using Kraft's inequality):
$2^{l_{max}} - \sum_{j=1}^{i-1} 2^{l_{max}-l_j} = 2^{l_{max}}(1 - \sum_{j=1}^{i-1} 2^{-l_j}) >$
$2^{l_{max}}(1 - \sum_{j=1}^{M} 2^{-l_j}) \geq 0$

Entropy, Lower and Upper Bounds

Entropy Definition:

$$H[X] = -\sum_j p_j \log p_j$$

We'll prove that if $\bar{L}_{min}$ is the minimum expected length over all prefix-free codes for X then:

$$H[X] \leq \bar{L}_{min} \leq H[X] + 1 \text{ bit per symbol}$$

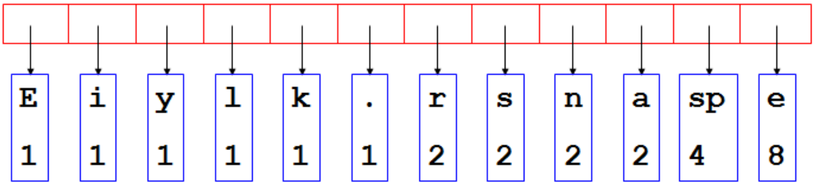
Entropy, Lower and Upper Bounds, cont.

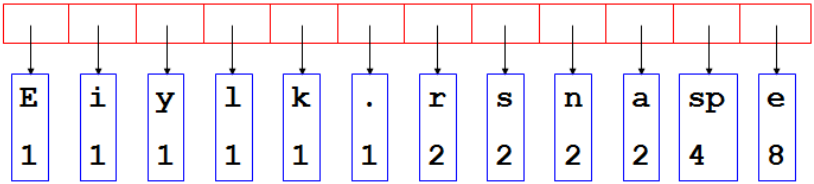
Proof:

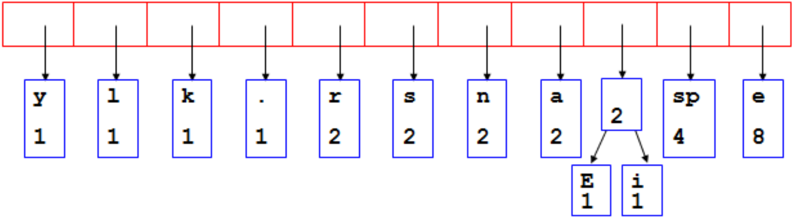
- *(First inequality)*
  $H[X] - \bar{L} = \sum_{j=1}^{M} p_j \log \frac{1}{p_j} - \sum_{j=1}^{M} p_j l_j = \sum_{j=1}^{M} p_j \log \frac{2^{-l_j}}{p_j}$
  Thus, $H[X] - \bar{L} \leq (\log e) \sum_{j=1}^{M} p_j (\frac{2^{-l_j}}{p_j} - 1) =$
  $(\log e)(\sum_{j=1}^{M} 2^{-l_j} - \sum_{j=1}^{M} p_j) \leq 0$
  where the inequality $lnx \leq x - 1$, the Kraft inequality, and $\sum_j p_j = 1$ have been used.
- *(Second Inequality)* We need to prove that there exist a prefix-free code such that $\bar{L} < H[X] + 1$.It suffices to choose $l_j = \lceil -\log p_j \rceil$. Then $-\log p_j \leq l_j < -\log p_j + 1$ which is equivalent (the left part) to $2^{-l_j} \leq p_j$, thus $\sum_j 2^{-l_j} \leq \sum_j p_j = 1$ and the Kraft inequality is satisfied.

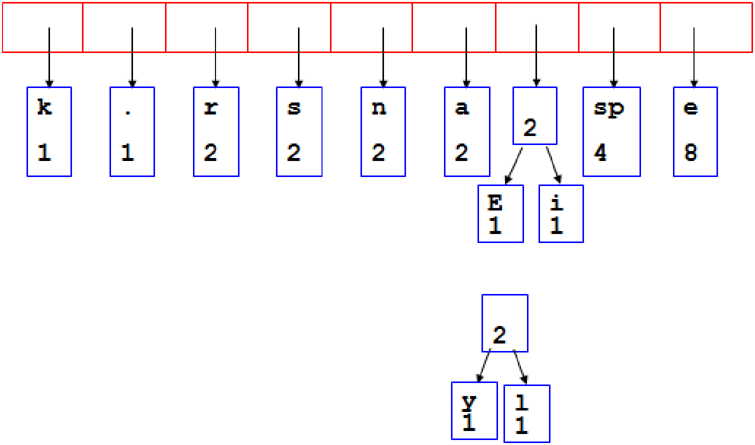
Huffman Encoding Algorithm

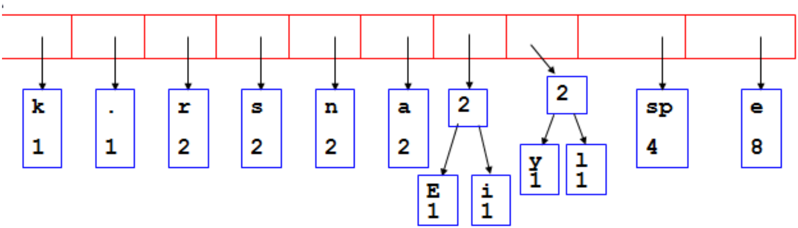
1. Pick two letters x, y from alphabet A with the smallest frequencies and create a subtree that has these two characters as leaves. Label the root of this subtree as z.
2. Set frequency $f(z) = f(x) + f(y)$. Removex, y and add z creating new alphabet $A' = A \cup \{z\} - \{x, y\}$. Then $|A'| = |A| - 1$.
3. Repeat this procedure with new alphabet A' until only one symbol is left.

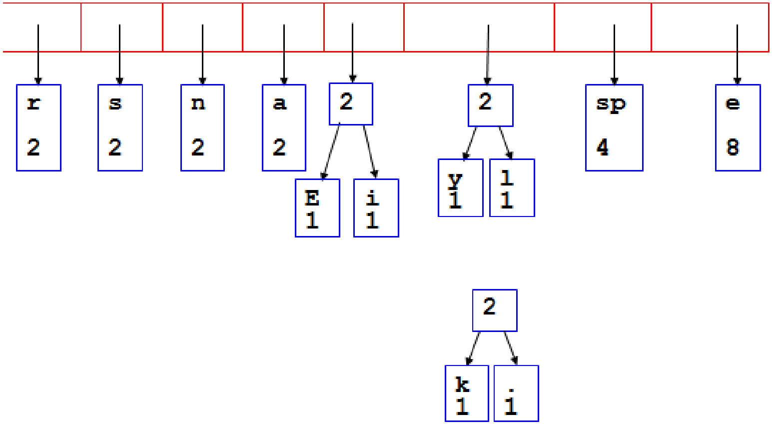

E
1
i
1
y
1
l
1
k
1
.
1
r
2
s
2
n
2
a
2
sp
4
e
8

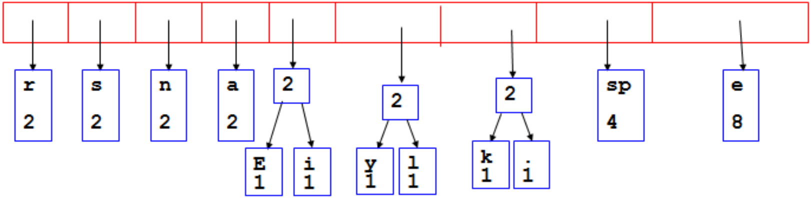

E 1
i 1
y 1
l 1
k 1
. 1
r 2
s 2
n 2
a 2
sp 4
e 8

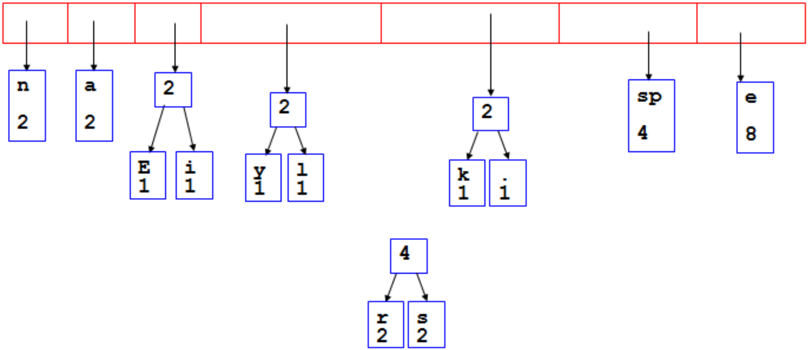

y
1
l
1
k
1
.
1
r
2
s
2
n
2
a
2
2
E
1
i
1
sp
4
e
8

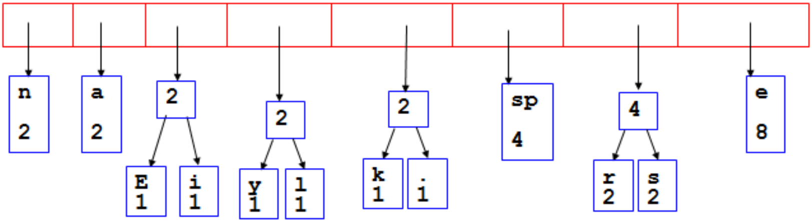

k
1
.
1
r
2
s
2
n
2
a
2
2
E
1
i
1
sp
4
e
8
2
y
1
l
1

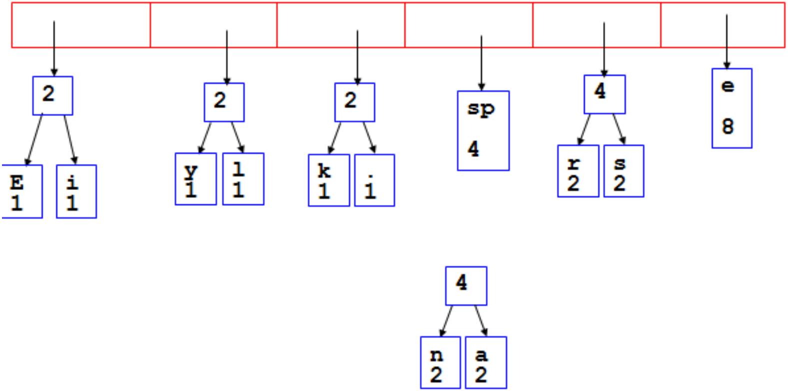

k
1
.
1
r
2
s
2
n
2
a
2
2
E
1
i
1
2
y
1
l
1
sp
4
e
8

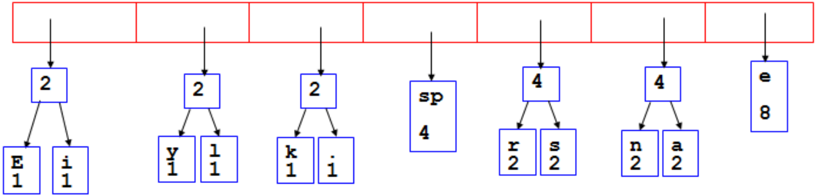

r
2
s
2
n
2
a
2
2
E
1
i
1
2
y
1
l
1
sp
4
e
8
2
k
1
.
1

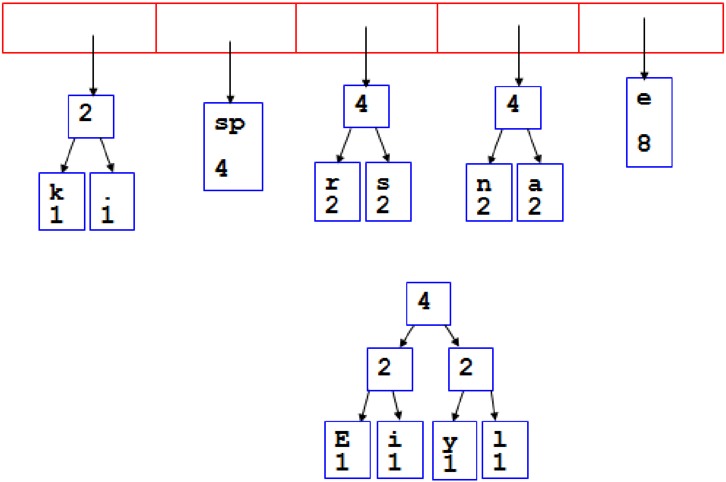

r
2
s
2
n
2
a
2
2
E
1
i
1
2
y
1
l
1
2
k
1
.
1
sp
4
e
8

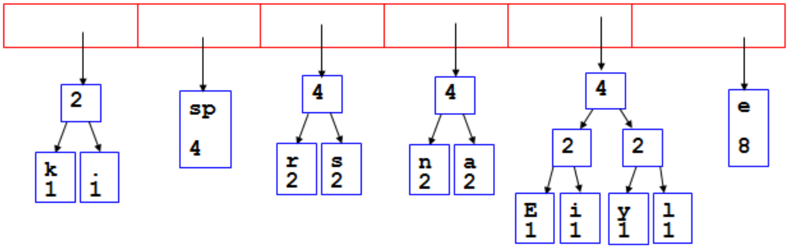

n
2
a
2
2
E
1
i
1
2
y
1
l
1
2
k
1
.
1
sp
4
e
8
4
r
2
s
2

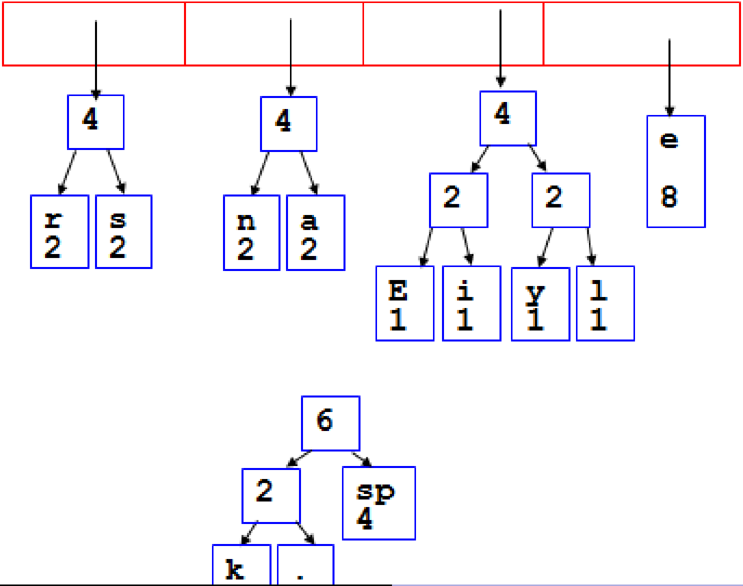

n
2
a
2
2
E
1
i
1
2
y
1
l
1
2
k
1
.
1
sp
4
4
r
2
s
2
e
8

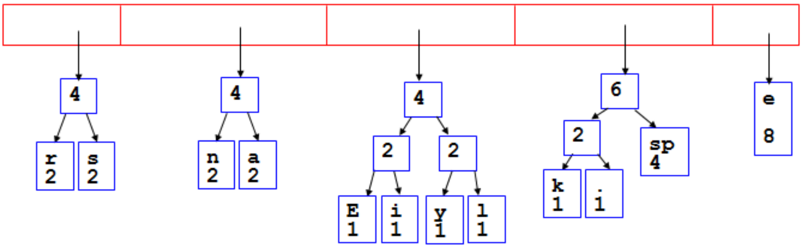

2
E
1
i
1
2
y
1
l
1
2
k
1
.
1
sp
4
4
r
2
s
2
e
8
4
n
2
a
2

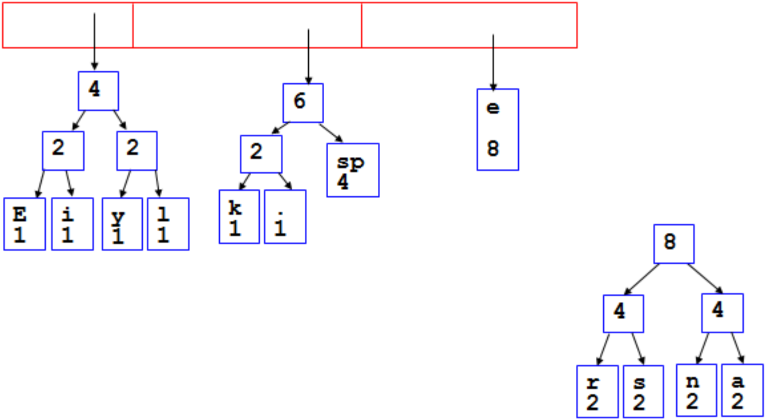

2
E 1
i 1
2
y 1
l 1
2
k 1
. 1
sp 4
4
r 2
s 2
4
n 2
a 2
e 8

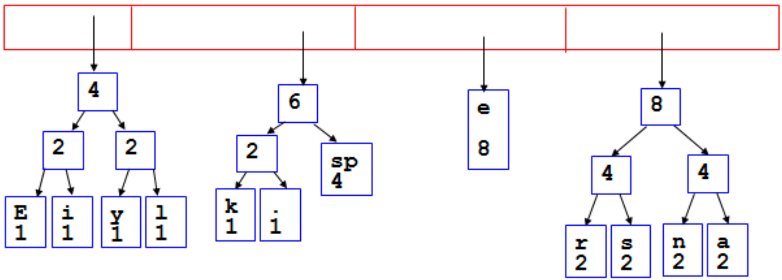

2
k 1
i 1
sp 4
4
r 2
s 2
4
n 2
a 2
e 8
4
2
2
E 1
i 1
y 1
l 1

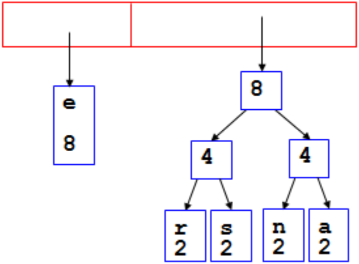

2
k 1
. 1
sp 4
4
r 2
s 2
4
n 2
a 2
4
2
2
E 1
i 1
y 1
l 1
e 8

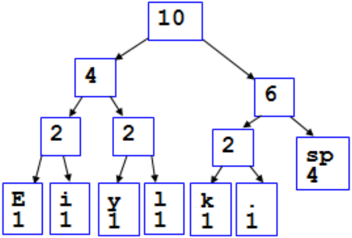

4
r 2
s 2
4
n 2
a 2
4
2
2
E 1
i 1
y 1
l 1
e 8
6
2
sp 4
k
.

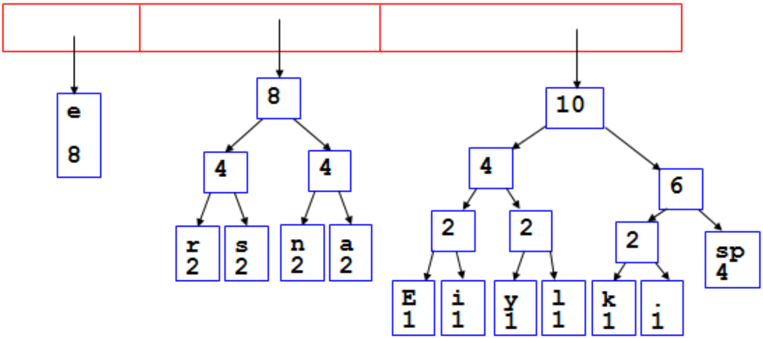

4
r 2
s 2
4
n 2
a 2
4
2
2
E 1
i 1
y 1
l 1
6
2
k 1
. 1
sp 4
e 8

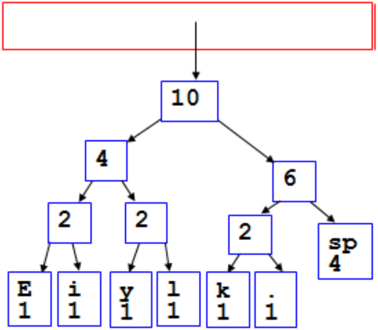

4
2
2
E
1
i
1
y
1
l
1
6
2
sp
4
k
1
.
1
e
8
8
4
4
r
2
s
2
n
2
a
2

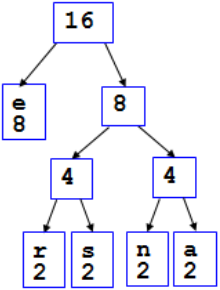

4
2
2
E
1
i
1
y
1
l
1
6
2
sp
4
k
1
.
1
e
8
8
4
4
r
2
s
2
n
2
a
2

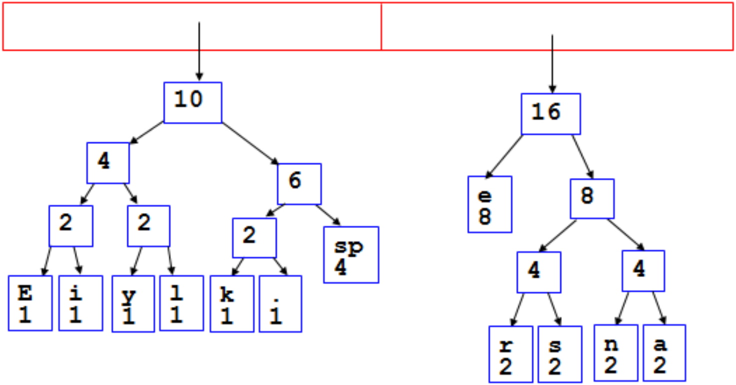

e
8
8
4
4
r
2
s
2
n
2
a
2

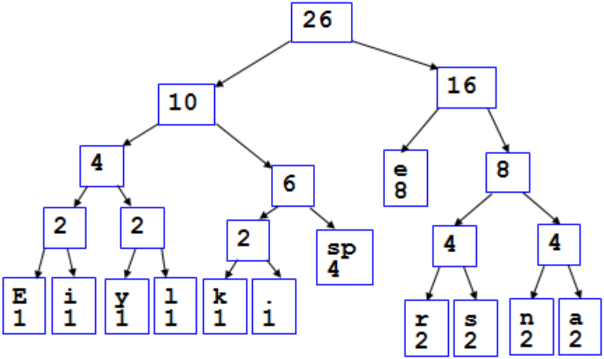

10
4
6
2
2
2
sp
4
E
1
i
1
y
1
l
1
k
1
.
1

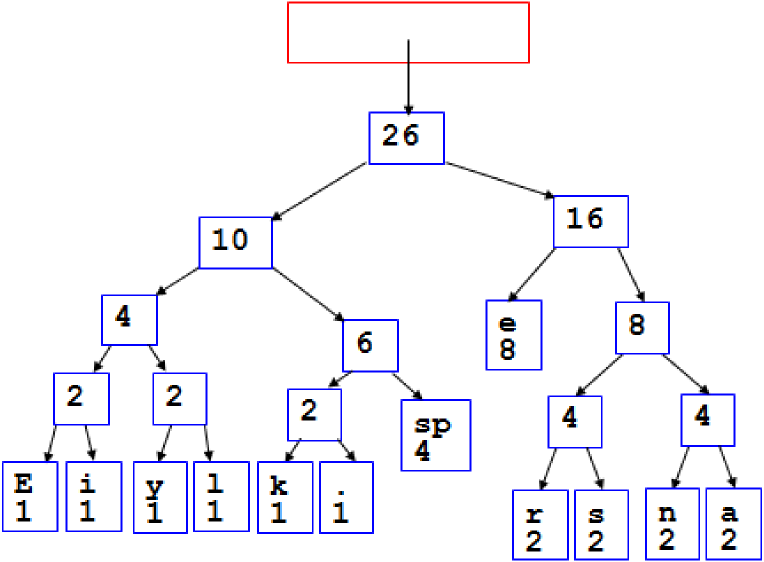

e
8
8
4
4
r
2
s
2
n
2
a
2
10
4
6
2
2
2
sp
4
E
1
i
1
y
1
l
1
k
1
.
1


10
4
6
2
2
2
sp
4
E
1
i
1
y
1
l
1
k
1
.
1


16
e
8
8
4
4
r
2
s
2
n
2
a
2


10
4
6
2
2
2
sp
4
E
1
i
1
y
1
l
1
k
1
.
1
16
e
8
8
4
4
r
2
s
2
n
2
a
2


26
10
16
4
6
e
8
8
2
2
2
sp
4
4
4
E
1
i
1
y
1
l
1
k
1
.
1
r
2
s
2
n
2
a
2


26
10
16
4
6
e
8
8
2
2
2
sp
4
4
4
E
1
i
1
y
1
l
1
k
1
.
1
r
2
s
2
n
2
a
2

Algorithm Revisited

- **For** $(i, 1 \text{ TO } n-1)$ **do**
- Merge last two subtrees;
- Rearrange subtrees in nonincreasing order of root - probability
- **End for**

Complexity: $\mathcal{O}(n \log n)$ - if a heap is used.

Huffman Coding is Optimal

1. Prefix-free Codes have the property that the associated code tree is full.
2. Optimal prefix-free Codes have the property that, for each of the longest codewords in the code, the sibling of the codeword is another longest codeword
3. There is an optimal prefix-free code for X in which the codewords for $M-1$and M are siblings and have maximal length within the code.
4. An optimal code for the reduced alphabet $X' = X - \{M, M-1\} \cup z$ yields an optimal code for X. (Note that $\bar{L} = \bar{L}' + p_{M-1} + p_M$)

General Comments

- Huffman Code is usefull in finding an optimal code, while the entropy bounds provide insightful performance bounds.
- Huffman Coding is generally close to the entropy.
- By Coding in Large k-blocks we can find codings that approximate as much as we want the lower entropy bounds (for large k). Not practical though, due to the size of $|X|^k$.

Sources used

1. Robert Gallager, course materials for 6.450 Principles of Digital Communications I, Fall 2006. MIT OpenCourseWare (http://ocw.mit.edu/)
2. Notes from 2005 Design and Analysis of Algorithms (Hong Kong University)
3. Stathis Zachos 2014, NTUA
4. Anadolu University, Notes from 2010 Algorithm Analysis and Complexity
5. Linkopings University, 2008 Data Compression Notes

Thank you!


© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com
search ID: hbrn1222
"I CAN'T READ THEIR SMOKE SIGNAL. IT'S ENCRYPTED."