Huffman Codes

Problem Definition

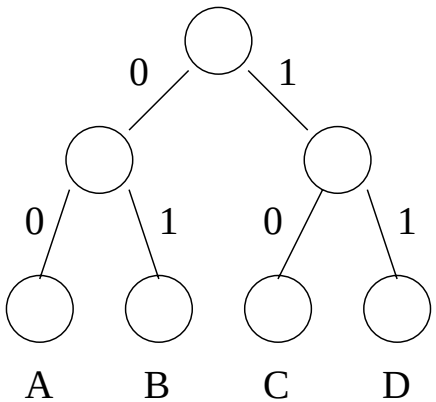
Algorithms: Design and Analysis, Part II

Codes as Trees

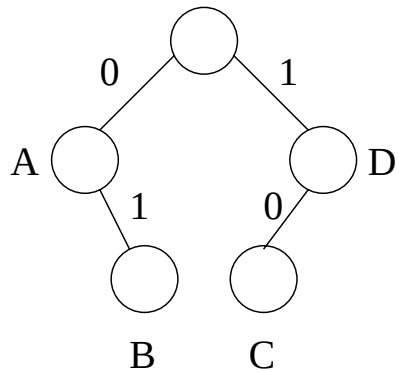
Goal: Best binary prefix-free encoding for a given set of character frequencies.

Useful fact: Binary codes $\leftrightarrow$ Binary trees

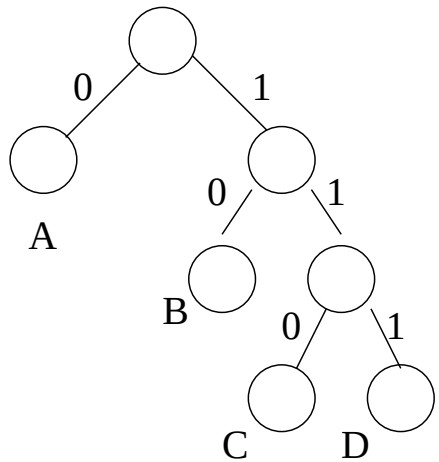
Examples: (Σ ={A,B,C,D})



{00,01,10,11}



{0,01,10,1}

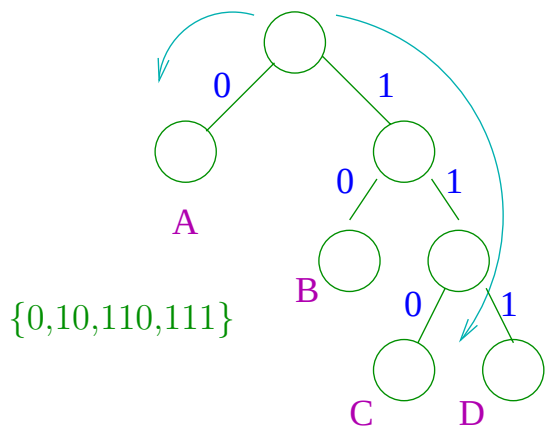


{0,10,110,111}

Prefix-Free Codes as Trees

In general: - Left child edges $\leftrightarrow$ "0", right child edges $\leftrightarrow$ "1"
- For each $i \in \Sigma$, exactly one node labeled "i"
- Encoding of $i \in \Sigma \leftrightarrow$ Bits along path from node to the node "i"
- Prefix-free $\leftrightarrow$ Labelled nodes = the leaves
  [since prefixes $\leftrightarrow$ one node an ancestor of another]

To decode: Repeadetly follow path from root until you hit a leaf.
[ex. $0110111 \mapsto$ ACD] (unambiguous since only leaves are labelled)



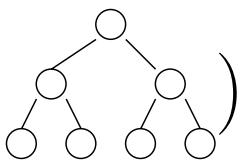
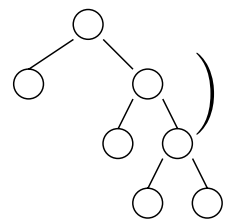
Note: Encoding length of $i \in \Sigma$ = depth of i in tree.

Problem Definition

Input: Probability p_i for each character $i \in \Sigma$.

Notation: If T = tree with leaves $\leftrightarrow$ symbols of Σ, then average encoding length $L(T) = \sum_{i \in \Sigma} p_i \cdot [\text{depth of } i \text{ in } T]$

Example: If $p_A = 60\%, p_B = 25\%, p_C = 10\%, p_D = 5\%$, then

$L($  $) = 2$ while $L($  $) = 1.55$

Output: A binary tree T minimizing the average encoding length $L(\cdot)$.