Huffman Codes

A Greedy Algorithm

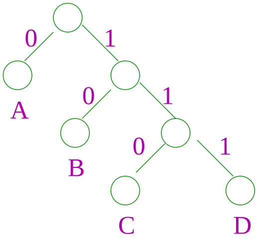
Algorithms: Design
and Analysis, Part II

Codes as Trees

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

Output: Binary tree (with leaves $\leftrightarrow$ symbols of Σ) minimizing the average encoding length:

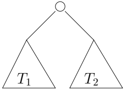
$$L(T) = \sum_{i \in \Sigma} p_i[\text{depth of } i \text{ in } T]$$



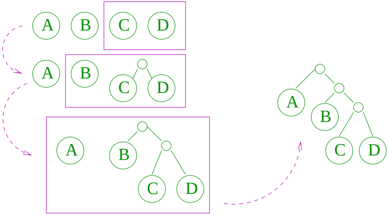
Building a Tree

Question: What's a principled approach for building a tree with leaves $\leftrightarrow$ symbols of Σ?

Natural but suboptimal idea: Top-down/divide+conquer.
- Partition Σ into Σ_1, Σ_2 each with $\approx$50% of total frequency.
- Recursively compute T_1 for Σ_1, T_2 for Σ_2, return:



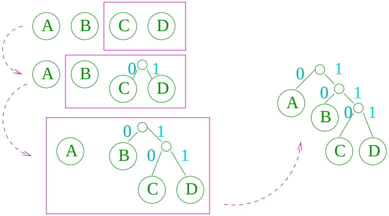
Huffman's (optimal) idea: Build tree bottom-up using successive mergers.



A Greedy Approach

Question: Which pair of symbols is "safe" to merge?

Observation: Final encoding length of $i \in \Sigma = \#$ of mergers its subtree endures.
[Each merger increases encoding length of participating symbols by 1]



Greedy heuristic: In first iteration, merge the two symbols with the smallest frequencies.

How to Recurse?

Suppose: 1st iteration of algorithm merges symbols a & b.

Idea: Replace symbols a, b by a new "meta-symbol" ab.

Question: What should be the frequency p_{ab} of this meta-symbol?

A) $\max\{p_a, p_b\}$

B) $\min\{p_a, p_b\}$

C) $\boxed{p_a + p_b}$ since ab is a proxy for "a or b" (intuitively)

D) $p_a - p_b$

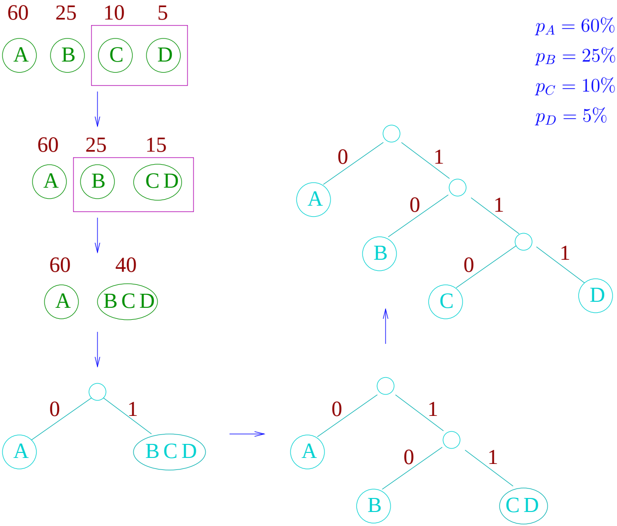
Example

$p_A = 60\%$

$p_B = 25\%$

$p_C = 10\%$

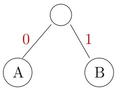
$p_D = 5\%$



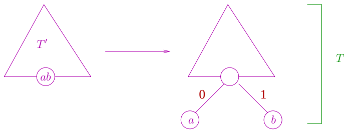
Huffman's Algorithm

(Given frequencies p_i as input)

If $|\Sigma| = 2$ return



Let $a, b \in \Sigma$ have the smallest frequencies.
Let $\Sigma' = \Sigma$ with a, b replaced by new symbol ab.
Define $p_{ab} = p_a + p_b$.
Recursively compute T' (for the alphabet Σ')
Extend T' (with leaves $\leftrightarrow \Sigma'$) to a tree T with leaves $\leftrightarrow \Sigma$ by splitting leaf ab into two leaves a & b.



Return T