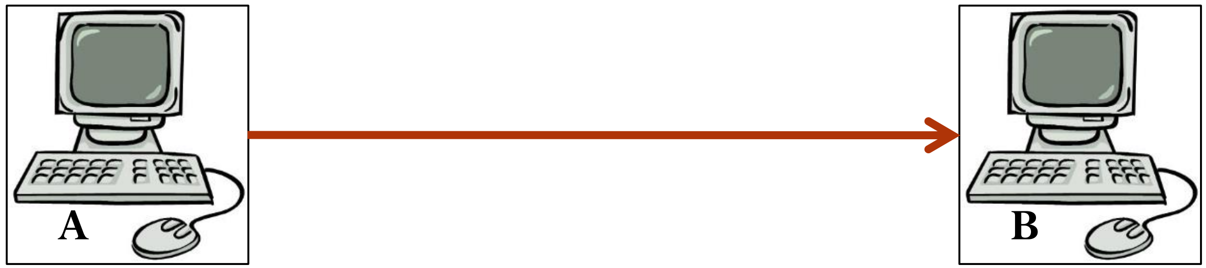
CSL 356: Analysis and Design of Algorithms

Ragesh Jaiswal

CSE, IIT Delhi

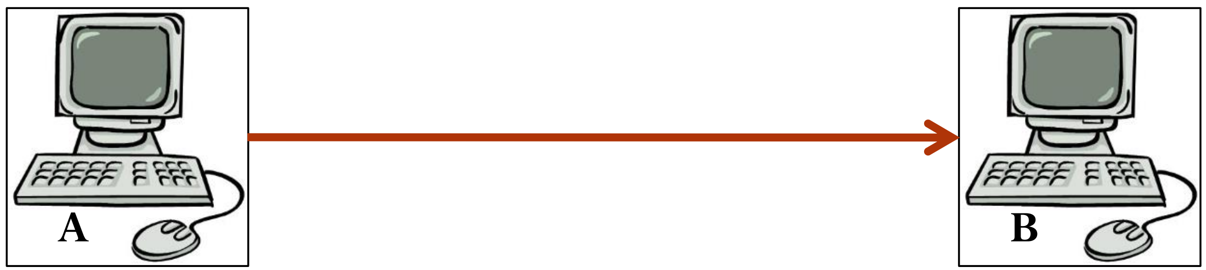
Greedy Algorithms: Huffman Coding

Greedy Algorithms: Huffman Coding



- **A** wants to send an email to **B** but wants to minimize the amount of communication (number of bits communicated).
- How do you encode an email into bits?
  - ASCII: (8 bits per character)
    - Is this the best way to encode the email given that the goal is to minimize the communication?

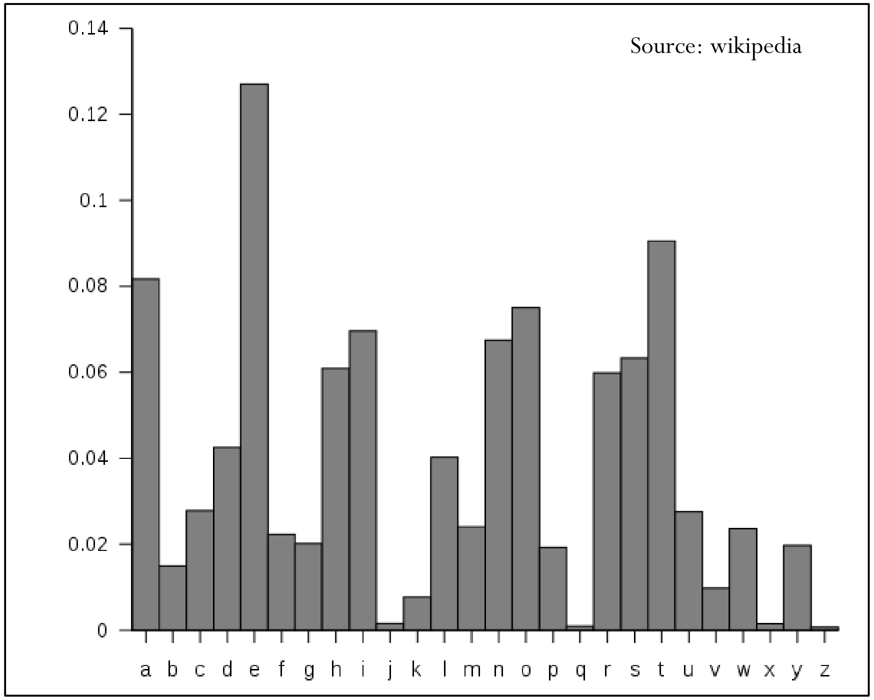
Greedy Algorithms: Huffman Coding



- **A** wants to send an email to **B** but wants to minimize the amount of communication (number of bits communicated).
- How do you encode an email into bits?
  - ASCII: (8 bits per character)
    - Is this the best way to encode the email given that the goal is to minimize the communication?
    - Different alphabets have different frequency of occurrence in a standard English document.

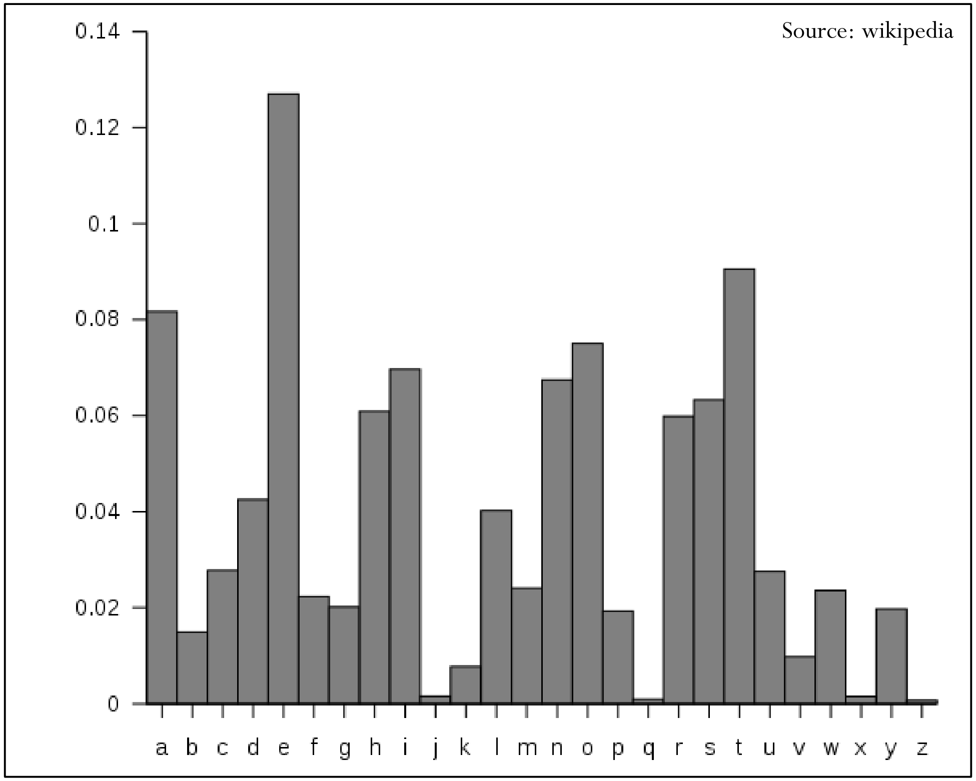
Greedy Algorithms: Huffman Coding

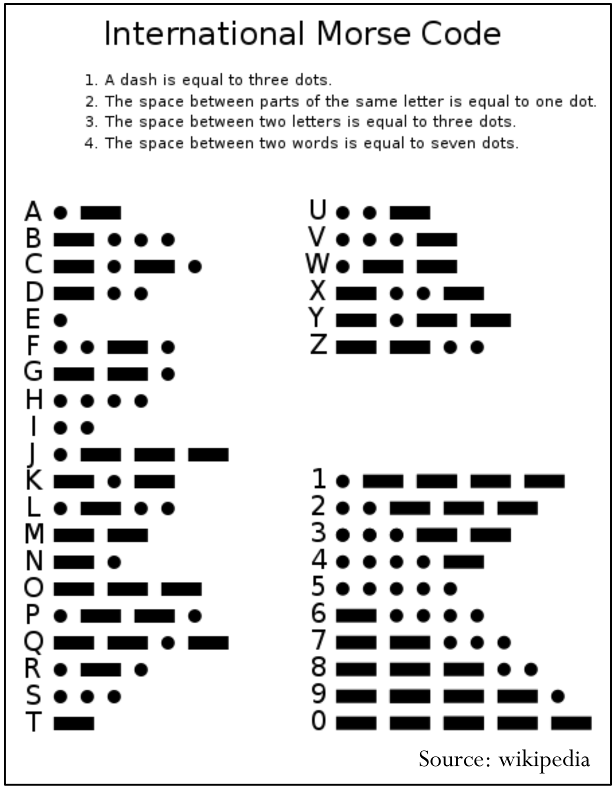
- **A** wants to send an email to **B** but wants to minimize the amount of communication (number of bits communicated).
- How do you encode an email into bits?
  - ASCII: (8 bits per character)
    - Is this the best way to encode the email given that the goal is to minimize the communication?
    - Different alphabets have different frequency of occurrence in a standard English document.



Greedy Algorithms: Huffman Coding

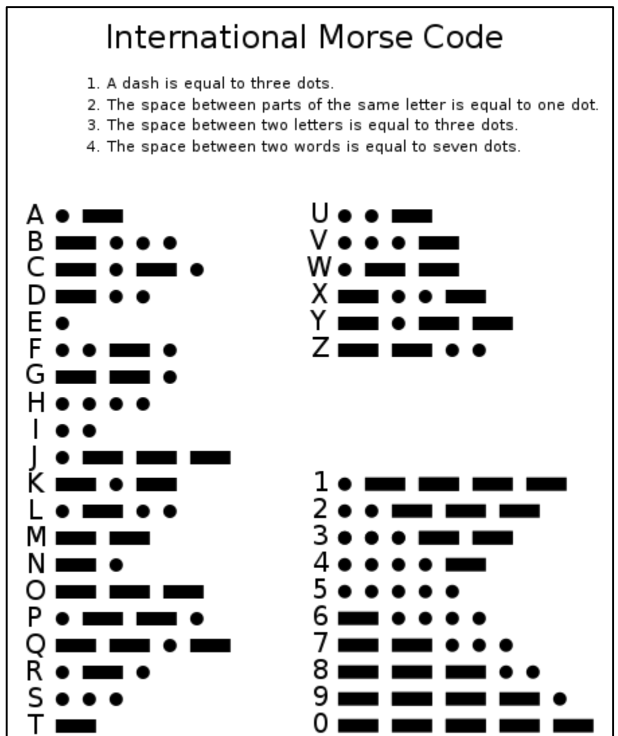
- The encoding of "e" should be shorter than the encoding of "x".
- In fact *Morse code* was designed with this in mind.





Greedy Algorithms: Huffman Coding

- Suppose you receive the following Morse code from your friend:
  - ● ● ● ▬
  - What is the message?



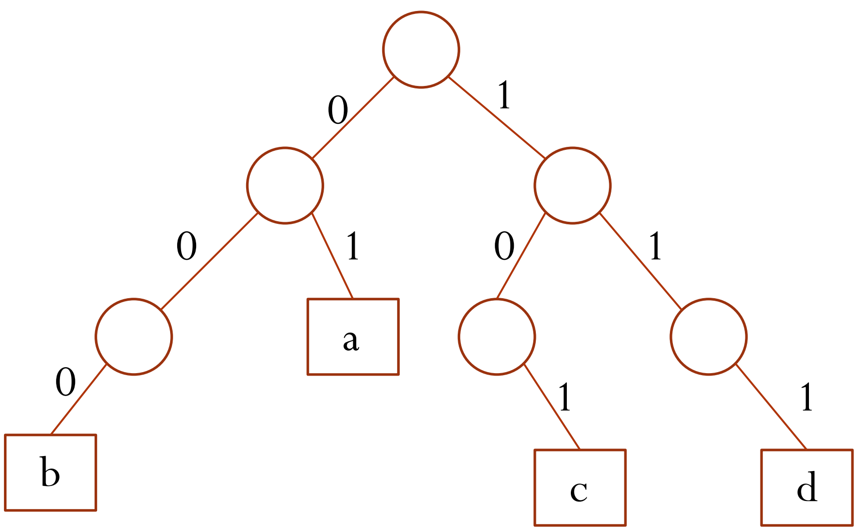
Source: wikipedia

Greedy Algorithms: Huffman Coding

- Prefix-free encoding: An encoding f is called prefix-free if for any pair of alphabets (a_1, a_2), $f(a_1)$ is not a prefix of $f(a_2)$.
- Morse code is clearly not prefix-free.
- Consider a *binary tree* with 26 leaves and associate each alphabet with a leaf in this tree.
  - Binary Tree: A rooted tree where each non-leaf node has at most two children.
- Label an edge 0 if this edge connects the parent to its left child and 1 otherwise.
- $f(x) =$ The label of edges connecting the root with x.

Greedy Algorithms: Huffman Coding

- Consider a *binary tree* with 26 leaves and associate each alphabet with a leaf in this tree.
- Label an edge 0 if this edge connects the parent to its left child and 1 otherwise.
- $f(x) =$ The label of edges connecting the root with x.

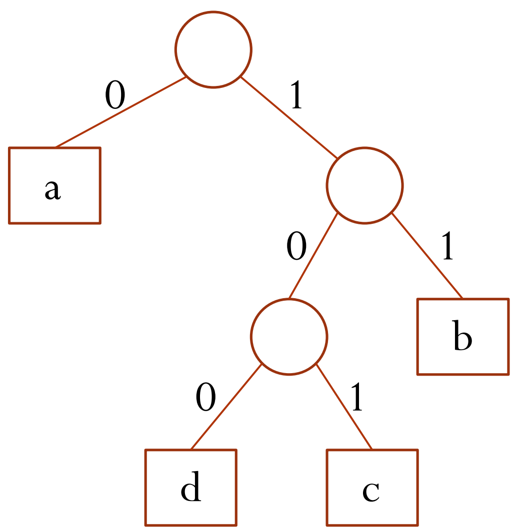


Simple example with 4 alphabets

- $f(a) = 01$
- $f(b) = 000$
- $f(c) = 101$
- $f(d) = 111$
- Is f prefix-free?

Greedy Algorithms: Huffman Coding

- Suppose you are given a prefix-free encoding g.
- Can you construct a binary tree with 26 leaves, associate each leaf with an alphabet, and label the edges as defined previously such that the for any alphabet, the label of edges connecting the root with $x = g(x)$?



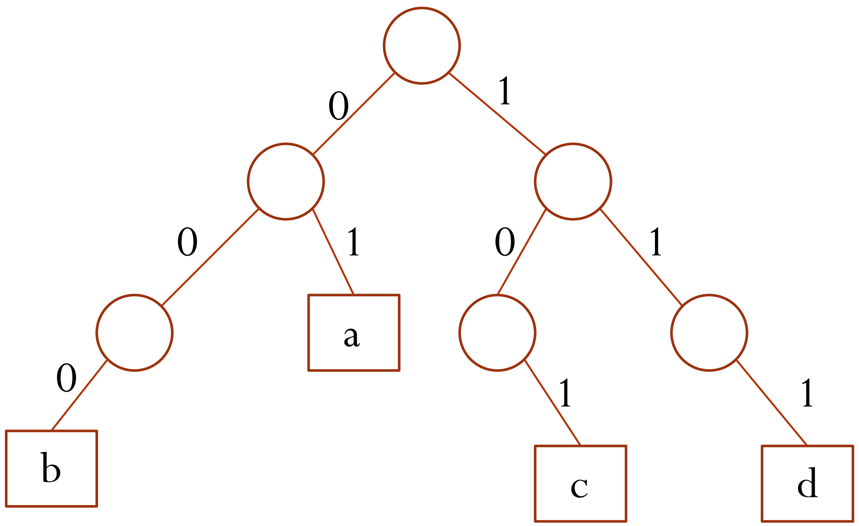
Simple example with 4 alphabets

- $g(a) = 0$
- $g(b) = 11$
- $g(c) = 101$
- $g(d) = 100$

Greedy Algorithms: Huffman Coding

- Problem: Given an alphabet set $\sum$ containing n alphabets and the frequency of occurrence of alphabets $(t(a_1), t(a_2), \ldots, t(a_n))$. Find the prefix-free encoding f that minimizes:
$$O_f = (|f(a_1)| \cdot t(a_1) + |f(a_2)| \cdot t(a_2) + \cdots + |f(a_n)| \cdot t(a_n))$$

- Example: $\sum = \{a, b, c, d\}, t(a) = 0.6, t(b) = 0.2, t(c) = 0.1, t(d) = 0.1$



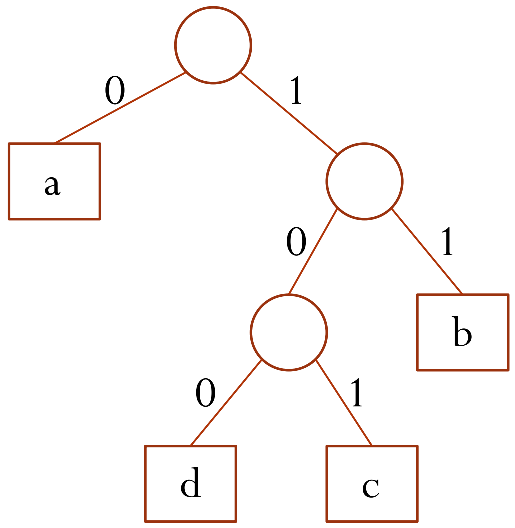
Simple example with 4 alphabets

- O_f?

Greedy Algorithms: Huffman Coding

- Problem: Given an alphabet set $\sum$ containing n alphabets and the frequency of occurrence of alphabets $(t(a_1), t(a_2), \ldots, t(a_n))$. Find the prefix-free encoding f that minimizes:
  $O_f = (|f(a_1)| \cdot t(a_1) + |f(a_2)| \cdot t(a_2) + \cdots + |f(a_n)| \cdot t(a_n))$

- Example: $\sum = \{a, b, c, d\}, t(a) = 0.6, t(b) = 0.2, t(c) = 0.1, t(d) = 0.1$



Simple example with 4 alphabets

- O_f?

Greedy Algorithms: Huffman Coding

- Definition: Given a binary tree, the depth of a vertex v, denoted by $d(v)$ is the length of the path from the root to v.

- Every binary tree gives a prefix-free encoding and every prefix-free encoding gives a binary tree. We will now use these properties to rephrase the previous problem in terms of binary trees and depths of leaves.

Greedy Algorithms: Huffman Coding

- Problem: Given an alphabet set Σ containing n alphabets and the frequency of occurrence of alphabets $(t(a_1), t(a_2), \ldots, t(a_n))$. Find a binary tree T with n leaves (one leaf labeled with one alphabet) such that:

  $$O_T = (d(a_1) * t(a_1) + d(a_2) * t(a_2) + \cdots + d(a_n) * t(a_n))$$

- $d(a_i)$ above is the depth of the leaf labeled with alphabet a_i

- What are the properties of the optimal tree T?
  1. Claim: T is a complete binary tree.
     - Complete binary tree: Every non-leaf node has exactly two children.

Greedy Algorithms: Huffman Coding

- Problem: Given an alphabet set Σ containing n alphabets and the frequency of occurrence of alphabets $(t(a_1), t(a_2), \ldots, t(a_n))$. Find a binary tree T with n leaves (one leaf labeled with one alphabet) such that:

$$O_T = (d(a_1) * t(a_1) + d(a_2) * t(a_2) + \cdots + d(a_n) * t(a_n))$$

- $d(a_i)$ above is the depth of the leaf labeled with alphabet a_i

- What are the properties of the optimal tree T?
  1. Claim: T is a complete binary tree.
     - Complete binary tree: Every non-leaf node has exactly two children.
  2. Claim: Consider the two alphabets x, y with least frequency. Then x and y have maximum depth in any optimal T and there is an optimal T where x and y are siblings.

Greedy Algorithms: Huffman Coding

- Let Ω be a new symbol not present in Σ. Consider the following (smaller) problem:
  - $\Sigma' = \Sigma - \{x, y\} \cup \{\Omega\}$
  - For all z in $\Sigma' \setminus \{\Omega\}, t'(z) = t(z)$
  - $t'(\Omega) = t(x) + t(y)$

  Find the optimal binary tree for the new alphabet Σ' and the new frequencies given by t'.
- Let T' be the optimal binary tree for the above problem.
- Consider the leaf v labeled with Ω in T'. Consider a new tree T where v has two children that are leaves and are labeled with x and y.
- Claim: T is the optimal tree for the original problem.

Greedy Algorithms: Huffman Coding

Huffman(Σ)

- Let $v_1, \dots, v_n$ be nodes. Each node denoting an alphabet
- $S = \{v_1, \dots, v_n\}$
- While ($|S| > 1$)
    - Pick two nodes x and y with the least value of $t(x)$ and $t(y)$
    - Create a new node z and set $t(z) = t(x) + t(y)$
    - Set x as the left child of z and y as the right child
    - Remove x and y from S and add z to S
- When $|S| = 1$, return the only node in S as the root node of the Huffman Tree

- Running time?

Greedy Algorithms: Huffman Coding

- A DNA sequence has four characters A, C, T, G and these characters appear with frequency 30%, 20%, 10%, and 40% respectively.
- We have to encode a sequence of length 1 million(10^6) in bits.
- If we use two bits for each character, then the size of the encoding will be 2 million bits.
- Huffman coding:
  - $f(A) = 10, f(C) = 110, f(T) = 111, f(G) = 0$
  - We will need 1.9 million bits.

End