UNIT 4: Greedy Technique

Greedy Technique:

Huffman Trees and codes

Code word:

(Communication and coding theory)

- A code word is an element of a standardized code or protocol.
- Each code word is assembled in accordance with the specific rules of the code and assigned a unique meaning.
- Code words are typically used for reasons of **reliability, clarity, brevity, or secrecy**.

Fixed length encoding:

- each character is given a binary code with the same number of bits
- Examples:

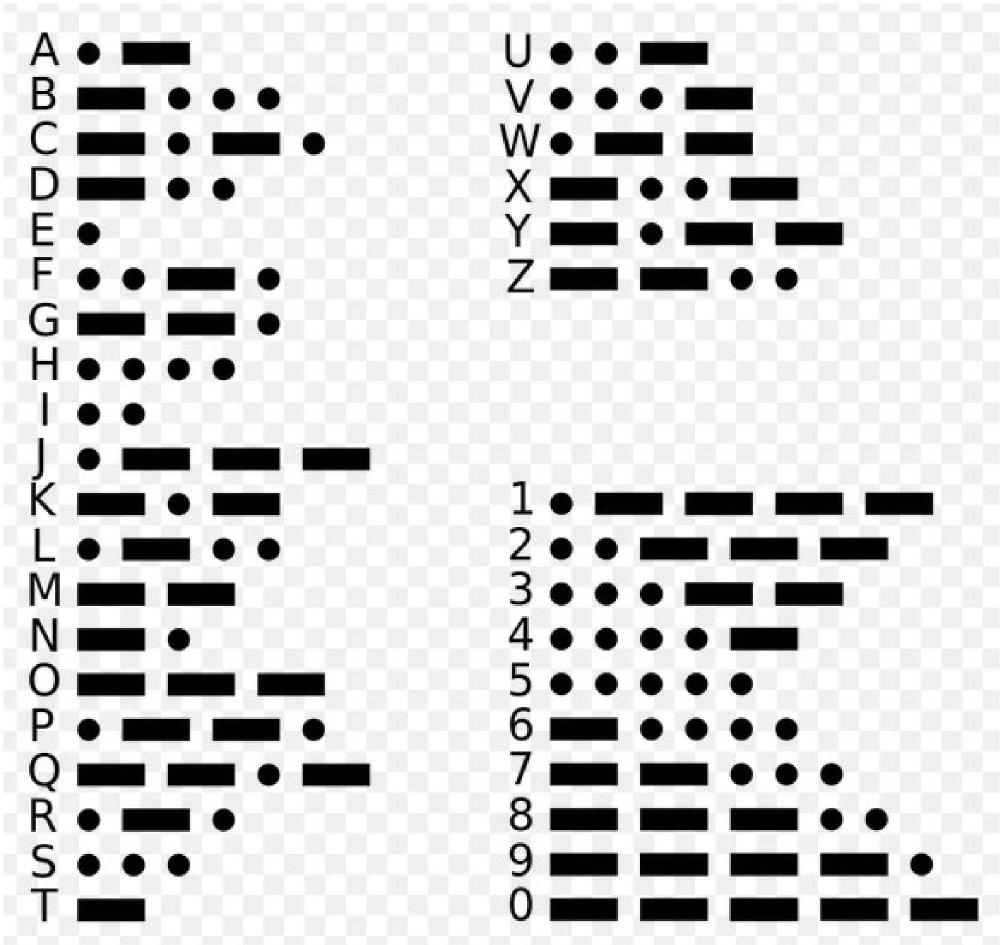
    Standard ASCII is a fixed width encoding scheme,

    where each character is encoded with 7 bits

Interesting fact!

- to get a coding scheme that yields a shorter bit string on the average, assign shorter codewords to more frequent characters and longer codewords to less frequent characters.

- idea was used in the telegraph code invented in the mid-19th century by Samuel Morse.
- frequent letters such as a, e, i are assigned short sequences of dots and dashes while infrequent letters such as q and z have longer ones



Guess this Morse code

... --- ...

Variable length encoding:

- variable-length code is a code which maps source symbols to a variable number of bits.
- Examples:

  Huffman coding,

  Lempel–Ziv coding,

  arithmetic coding,

  context-adaptive variable-length coding.

Huffman code:

(computer science and information theory)

- type of optimal prefix code (also known as **prefix free code**)
- commonly used for lossless data compression
- developed by **David A. Huffman** while he was a Sc.D. (Doctor of Science) student at MIT, and published in the 1952 paper **“A Method for the Construction of Minimum-Redundancy Codes”**

Huffman algorithm: Interesting facts

- idea is to use a frequency-sorted binary tree
- finds the most efficient binary code
- Building the tree from the bottom up guaranteed optimality, unlike the **top-down approach of Shannon–Fano coding**
- **prefix-free codes (Huffman code)**: the bit string representing a particular symbol is never a prefix of the bit string representing any other symbol

Huffman algorithm: Idea

- Uses greedy approach
- Is based on the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol
- associates the symbols with leaves of a binary tree in which all the left edges are labeled by 0 and all the right edges are labeled by 1 (or vice versa)
- codeword of a symbol can then be obtained by recording the labels on the simple path from the root to the character's leaf (**UNIQUE code**)

Huffman algorithm:

Step 1: Initialize n one-node trees and label them with the symbols. Record the frequency of each symbol in its tree's root to indicate the tree's weight. (weight of a tree will be equal to the sum of the frequencies in the tree's leaves.)

Step 2: Repeat the following operation until a single tree is obtained: (called as Huffman tree)

> Find two trees with the smallest weight (ties can be broken arbitrarily). Make them the left and right subtree of a new tree and record the sum of their weights in the root of the new tree as its weight.

Huffman trees and codes visualization

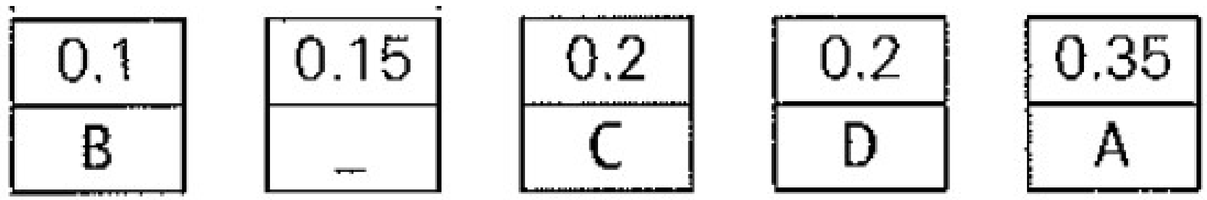
https://people.ok.ubc.ca/ylucet/DS/Huffman.html

Example:

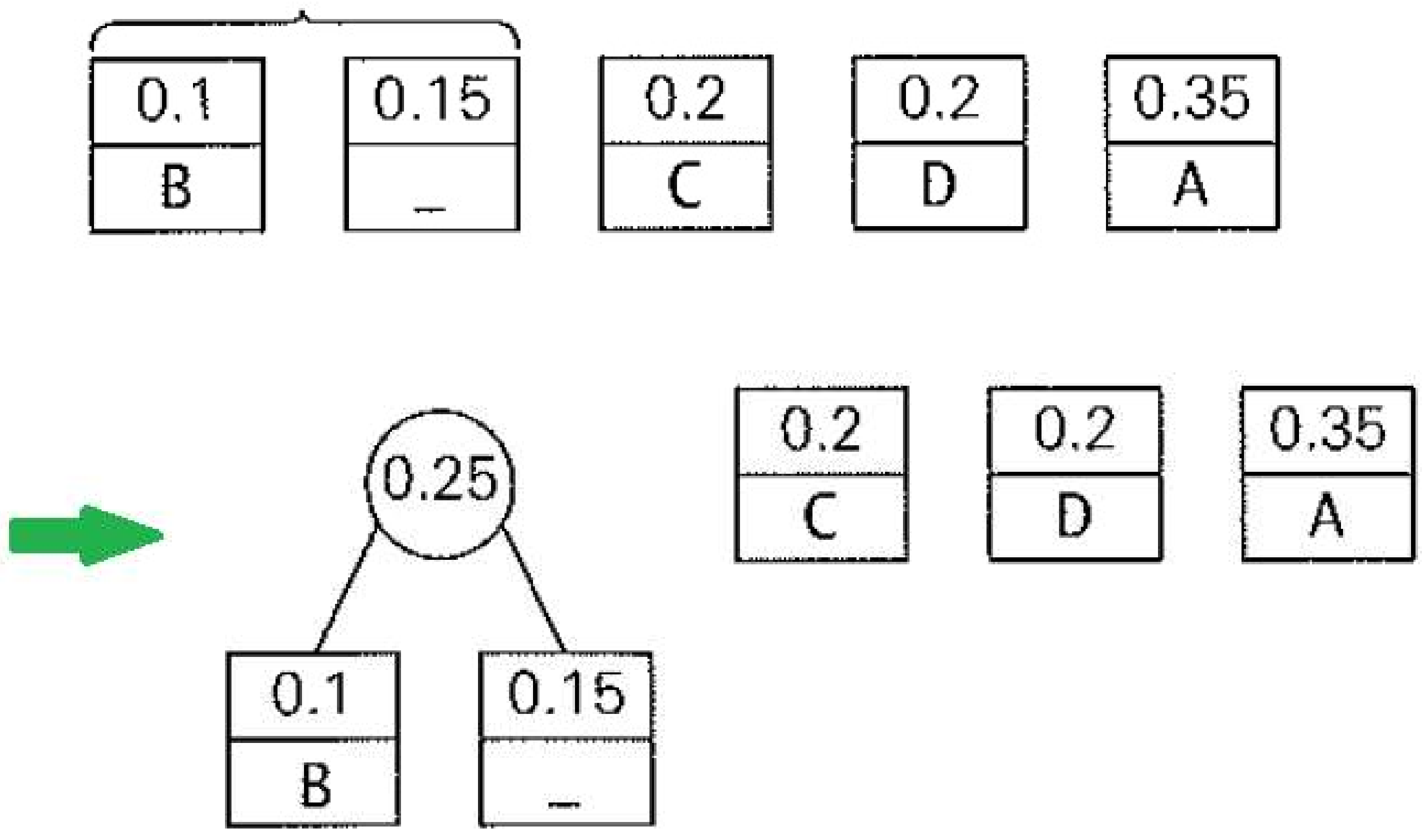
Compute the Huffman codes. Consider the five-character alphabet {A, B, C, D, _} with the following occurrence probabilities:

character	A	B	C	D	_
probability	0.35	0.1	0.2	0.2	0.15

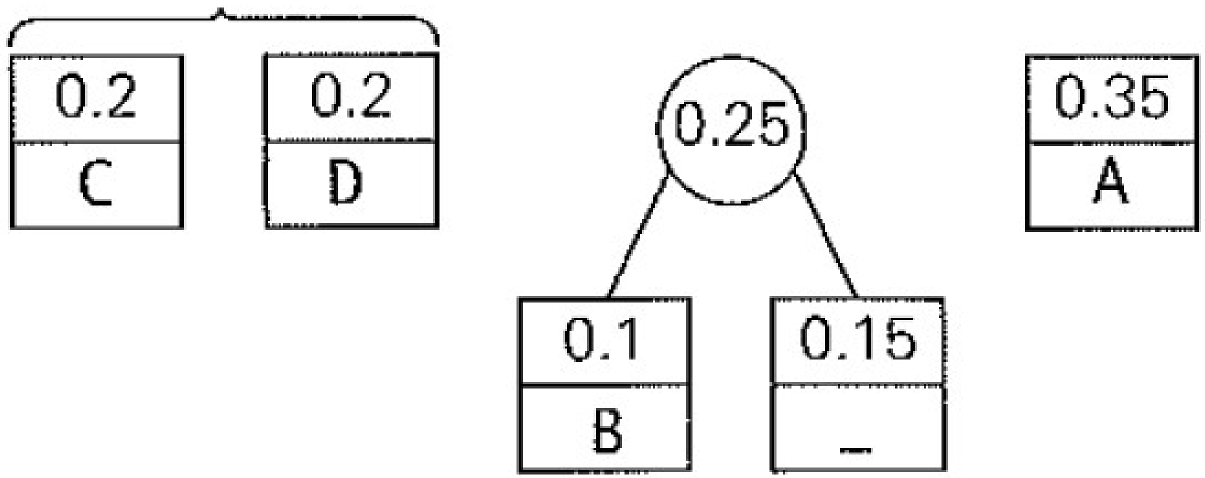
Step 1:


0.1
B
0.15
_
0.2
C
0.2
D
0.35
A

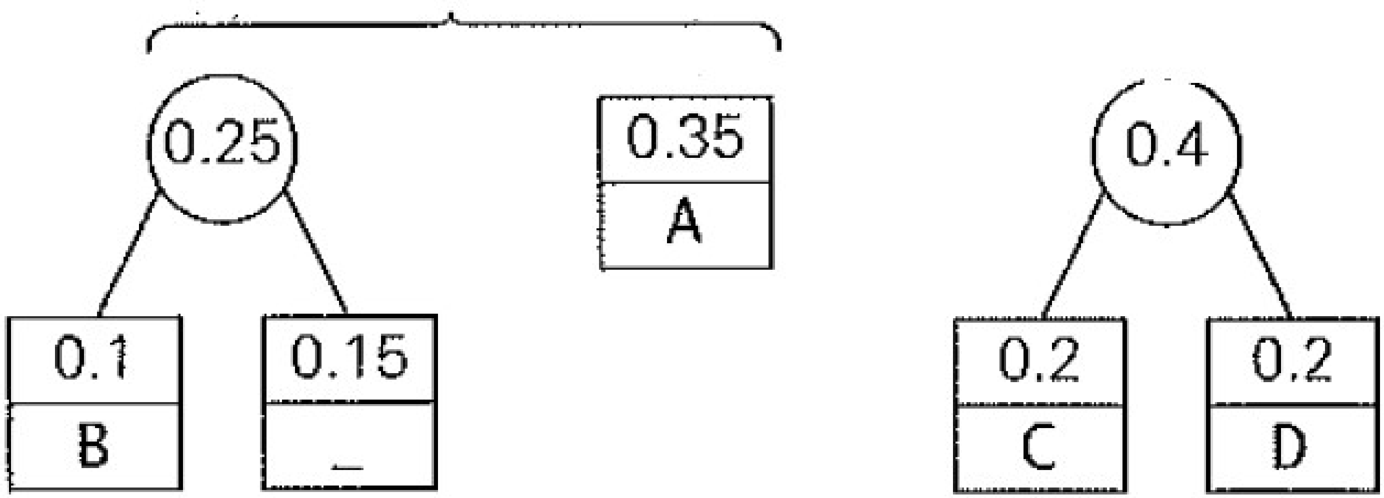
Step 2:


0.1
B
0.15
_
0.2
C
0.2
D
0.35
A
0.25
0.1
B
0.15
_
0.2
C
0.2
D
0.35
A

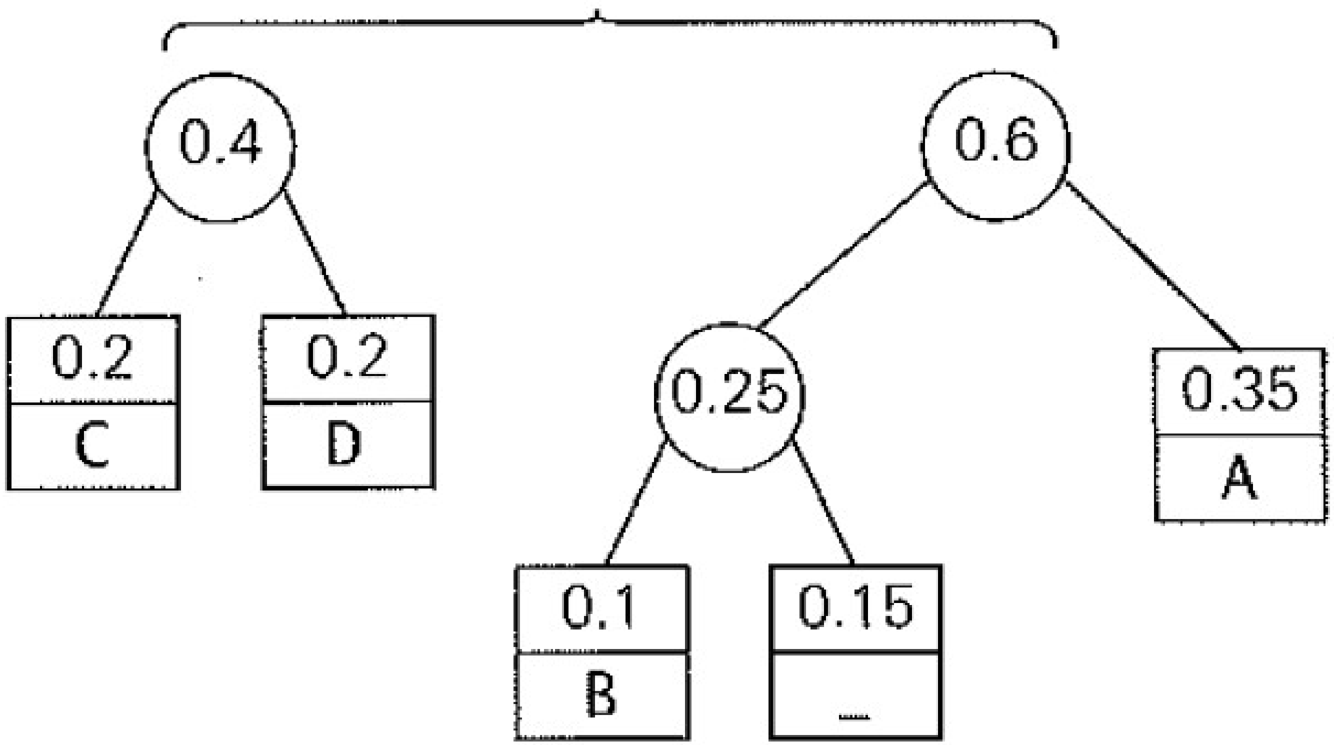
Step 3:



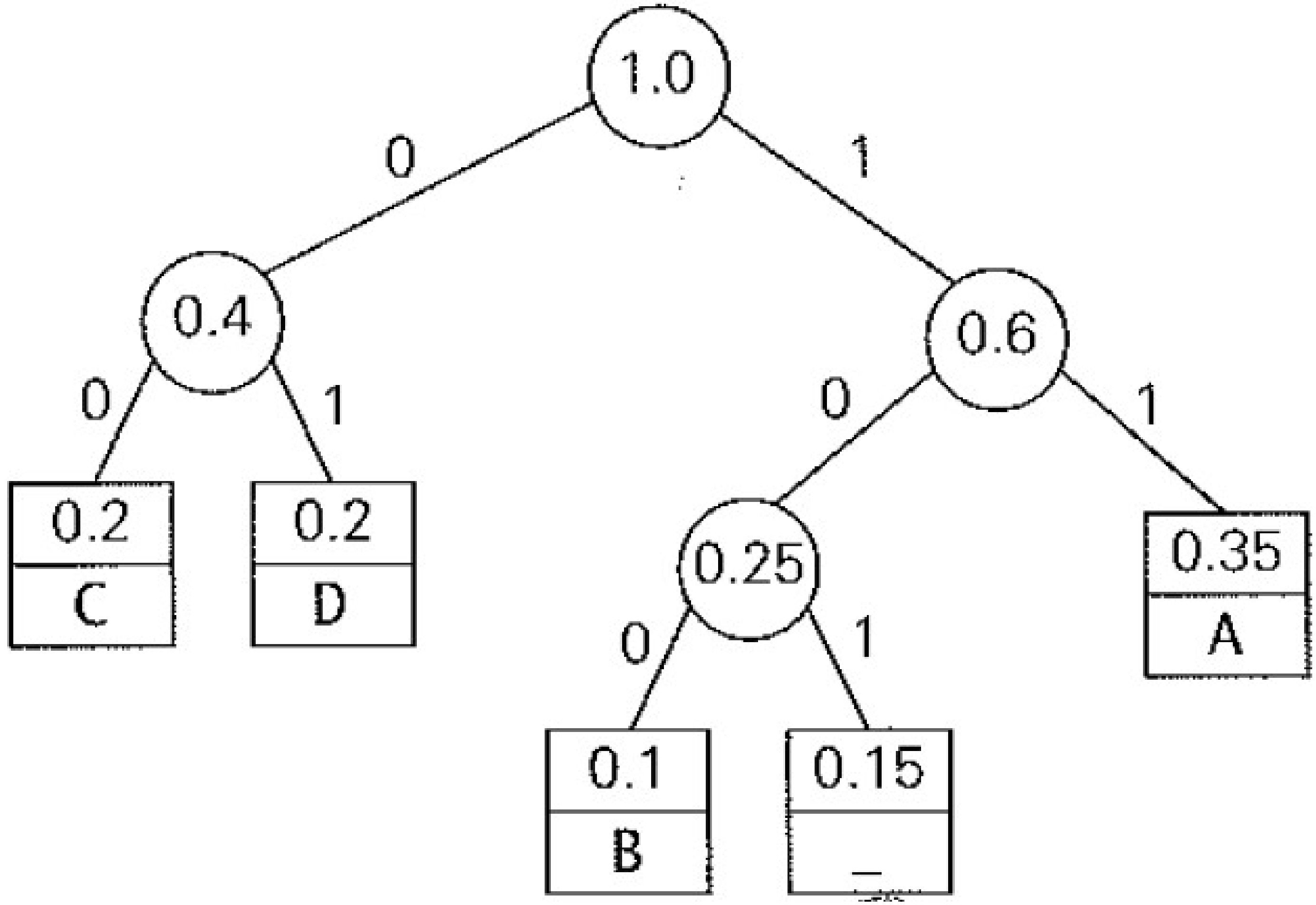
Step 4:



Step 5:



Step 6:



character	A	B	C	D	_
probability	0.35	0.1	0.2	0.2	0.15
codeword	11	100	00	01	101

character	A	B	C	D	_
probability	0.35	0.1	0.2	0.2	0.15
codeword	11	100	00	01	101

With the occurrence probabilities given and the codeword lengths obtained, the expected number of bits per character in the code is

2*0.35 + 3*0.1 + 2*0.2 + 2*0.2 + 3*0.15 = **2.25 bits/symbol**

compression ratio :

(a standard measure of a compression algorithm's effectiveness)

(3 - 2.25)/3 * 100% = **25%**

Let’s check our understanding

Construct a Huffman code for the following data:

character	A	B	C	D	_
probability	0.4	0.1	0.2	0.15	0.15

Encode the text ABACABAD using the code

Decode the text whose encoding is 100010111001010 in the code

Let’s check our understanding

For data transmission purposes, it is often desirable to have a code with a minimum variance of the codeword lengths (among codes of the same average length). Compute the average and variance of the codeword length in two Huffman codes that result from a different tie breaking during a Huffman code construction for the following data:

character	A	B	C	D	E
probability	0.1	0.1	0.2	0.2	0.4

Huffman code: Limitation

- makes it necessary to include the information about the coding tree into the encoded text to make its decoding possible.

Huffman code can be used to solve…

Suppose we have n positive numbers w1, w2, ... , wn that have to be assigned to n leaves of a binary tree, one per node. If we define the weighted path length as the sum

$$\sum_{i=1}^{n} l_i w_i$$

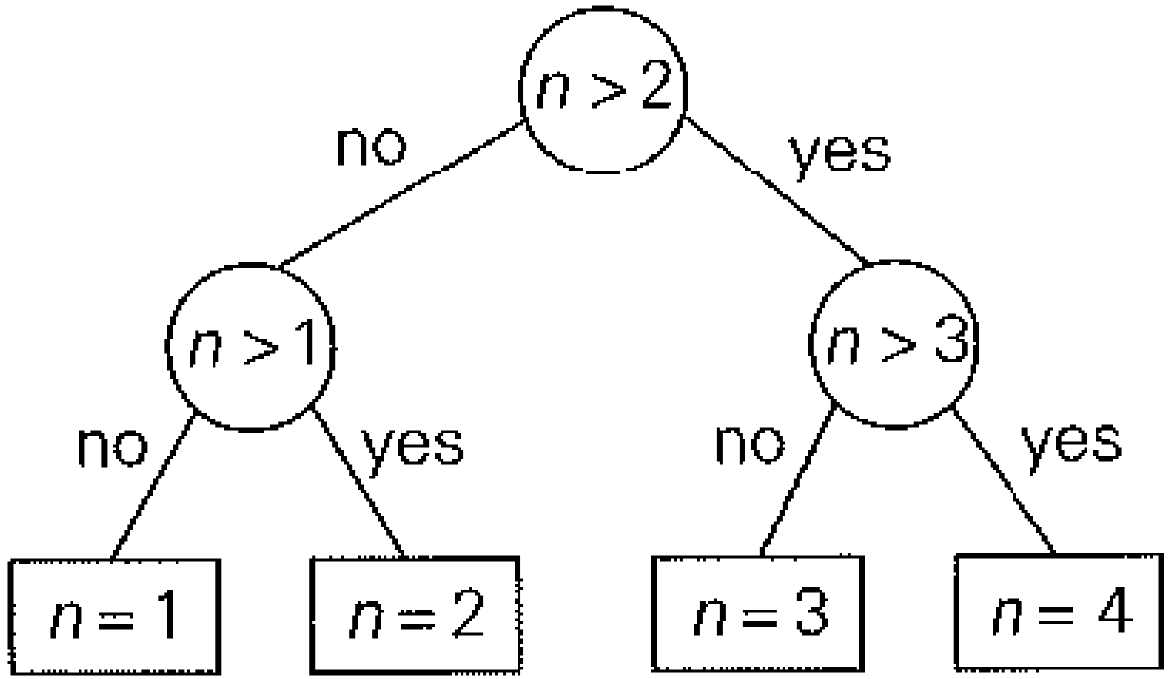
where li is the length of the simple path from the root to the ith leaf, how can we construct a binary tree with minimum weighted path length

Huffman code can be used to solve decision making problems…

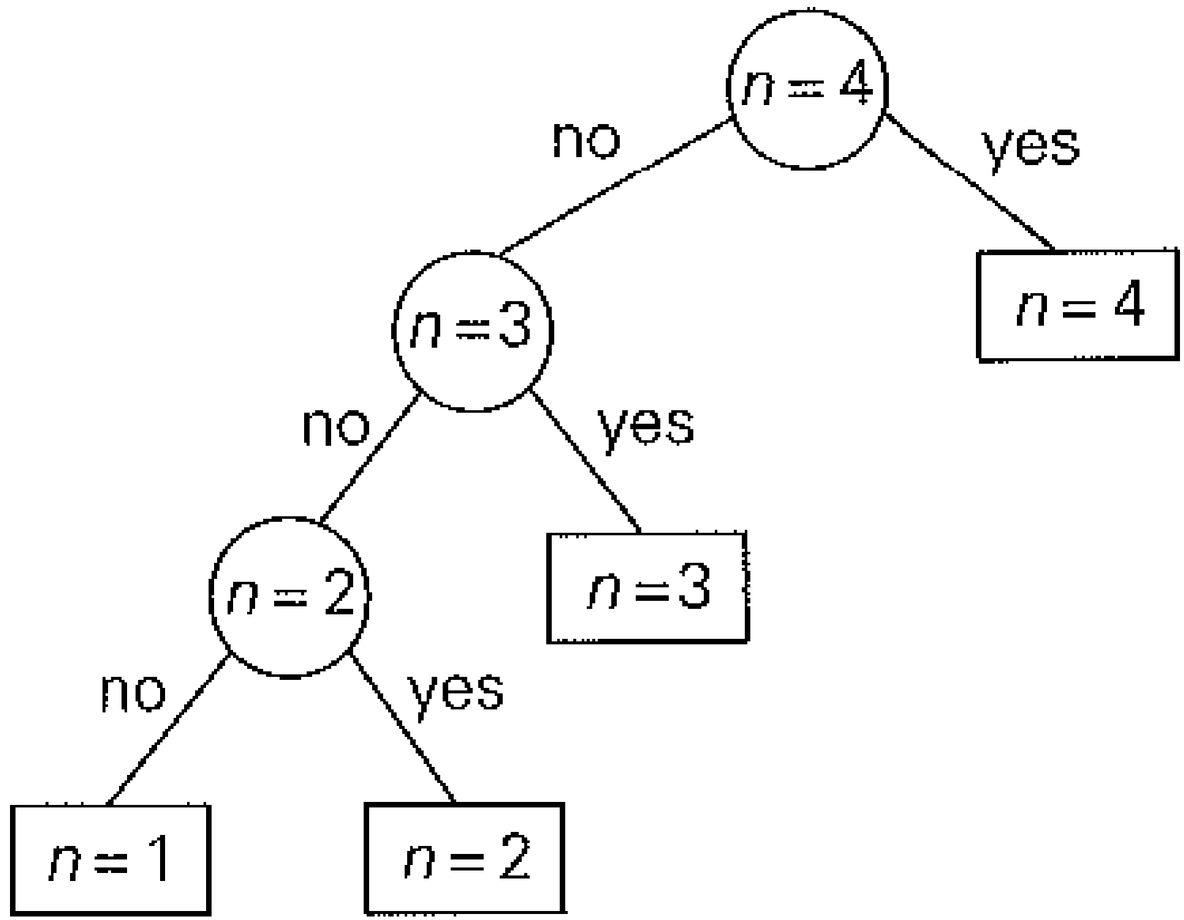
Example:

The game of guessing a chosen object from n possibilities (say, an integer between 1 and n) by asking questions answerable by yes or no.

Different strategies for playing this game can be modeled by **decision trees.** The length of the simple path from the root to a leaf in such a tree is equal to the number of questions needed to get to the chosen number represented by the leaf.



Decision tree for guessing an integer between 1 and 4



Decision tree for guessing an integer between 1 and 4