Greedy Technique

Coding Problem

- Coding → assignment of bit strings to alphabet characters
- Code-words → bit strings assigned for characters of alphabet
- Prefix-free codes → no codeword is a prefix of another codeword

- Two types of codes:
  - fixed-length encoding (e.g., ASCII)
  - variable-length encoding (e,g., Morse code)

- Problem:

  If frequencies of the character occurrences are known, what is the best binary prefix-free code?

Codding Problem

- Fixed-length code-words:
- They will be prefix-free for sure
- No compression is there
- Characters with high hit and low hit rate have same code length

- Variable-length code-words:
- Use binary tree with edges labeled with 0's and 1's
- Code-word is 0's and 1's from root to leaves
- Each leaf can be a character

- How to create the optimal binary tree,
- minimizing the expected length of a codeword (average length)?

Huffman codes

- Initialize *n* one-node trees with alphabet characters
- Each with the tree weight of their frequencies.

- Repeat the following step *n*-1 times:
  - Join two binary trees with smallest weights into one
  - Smaller one as left child, bigger one as right child
  - Make its weight equal the sum of the weights of the two trees

- Mark left edges with 0's and right edges with 1's
- Move from root to each leaf and assign the code to it

Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:

Huffman codes

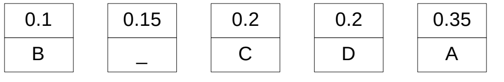
Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$
- Huffman codding:

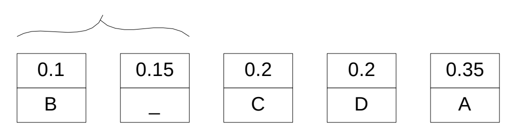


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:

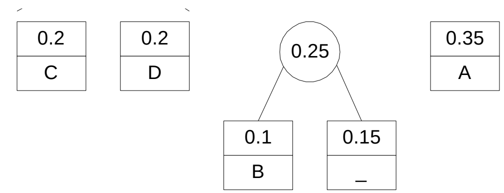


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:

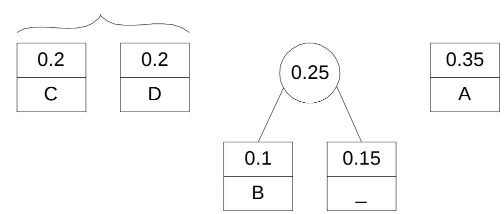


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:

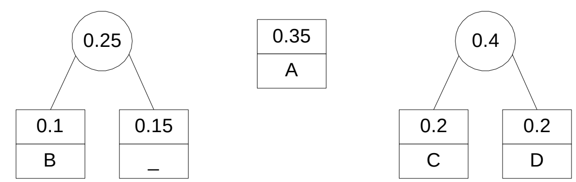


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:

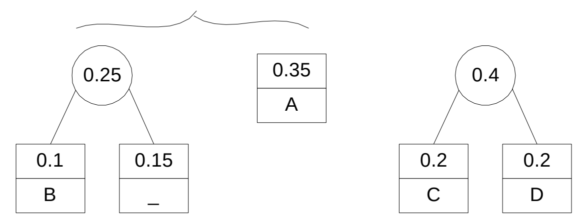


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:

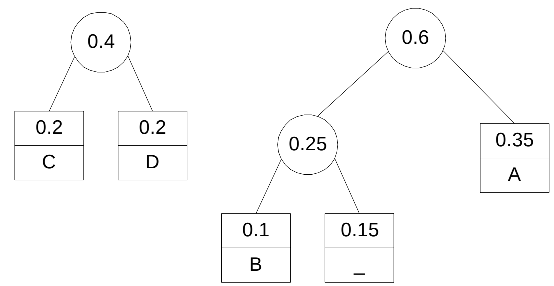


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:

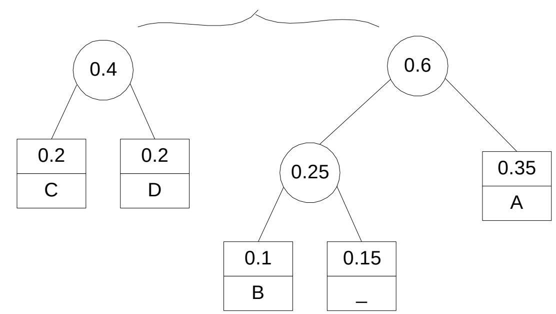


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

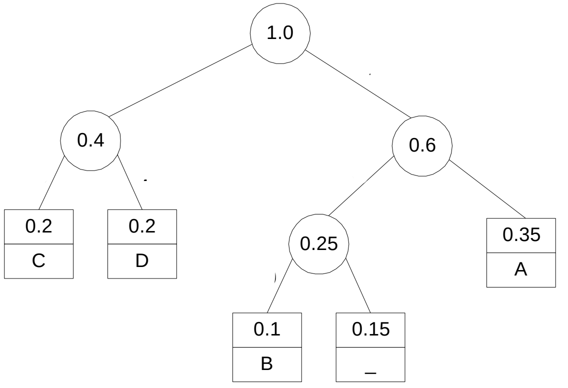
- Huffman codding:



Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$
- Huffman codding:

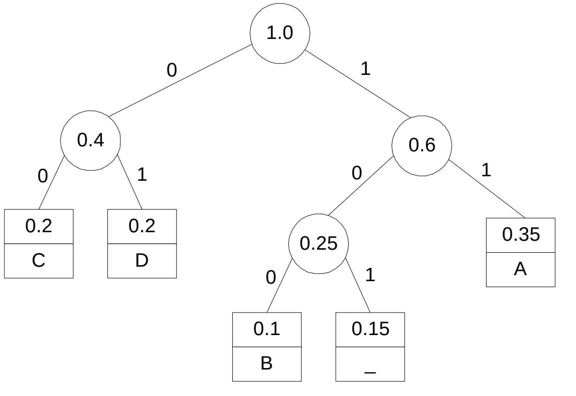


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:

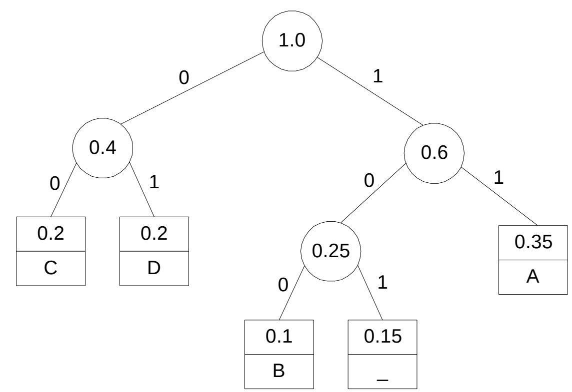


Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:



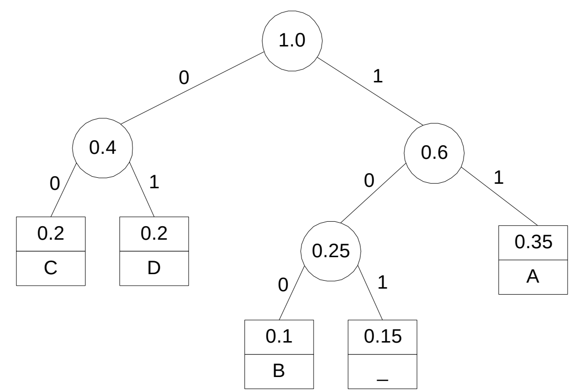
Char	A	B	C	D	-
Code					

Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:



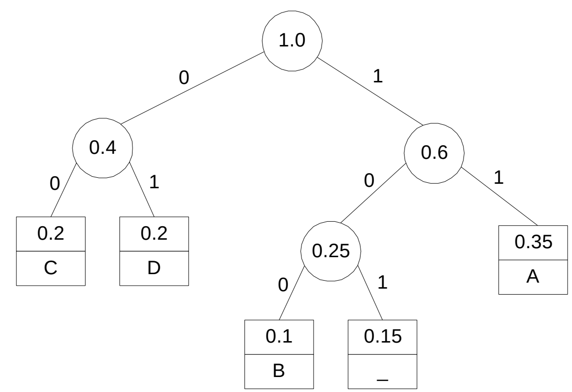
Char	A	B	C	D	-
Code	11	100	00	01	101

Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$

- Huffman codding:
  - Codewords:
  - 11 100 00 01 101
  - Average bits per character:
  - $\frac{2+3+2+2+3}{5} = 2.40$

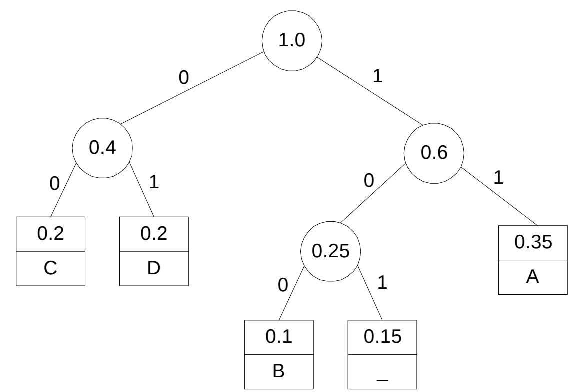


Char	A	B	C	D	-
Code	11	100	00	01	101

Huffman codes

Characters	A	B	C	D	-
Frequencies	0.35	0.1	0.2	0.2	0.15

- Fixed-length codding length:
  - $\lceil \log_2 n \rceil = \lceil \log_2 5 \rceil = 3$
- Huffman codding:
  - Codewords:
  - 11 100 00 01 101
  - Average bits per character:
  - $\frac{2+3+2+2+3}{5} = 2.40$
- compression ratio:
  - $\frac{3-2.40}{3} * 100 = 20\%$

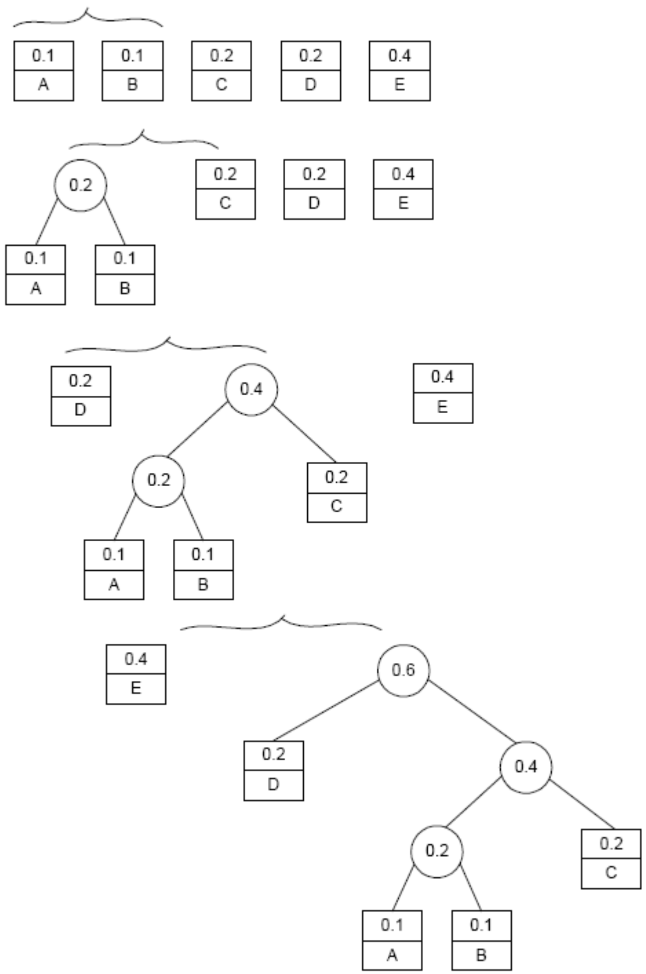


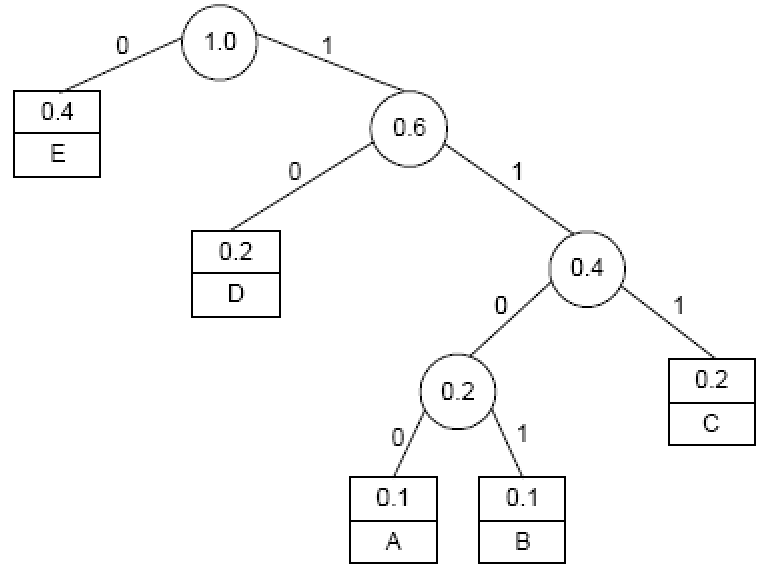
Char	A	B	C	D	-
Code	11	100	00	01	101

Class Discussion

2. 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





character	A	B	C	D	E
probability	0.1	0.1	0.2	0.2	0.4
codeword	1100	1101	111	10	0
length	4	4	3	2	1

Thus, the mean and variance of the codeword's length are, respectively,

$$\bar{l} = \sum_{i=1}^{5} l_i p_i = 4 \cdot 0.1 + 4 \cdot 0.1 + 3 \cdot 0.2 + 2 \cdot 0.2 + 1 \cdot 0.4 = 2.2 \quad \text{and}$$

$$Var = \sum_{i=1}^{5} (l_i - \bar{l})^2 p_i = (4\text{-}2.2)^2 0.1 + (4\text{-}2.2)^2 0.1 + (3\text{-}2.2)^2 0.2 + (2\text{-}2.2)^2 0.2 + (1\text{-}2.2)^2 0.4 = 1.36.$$

Applications of the Greedy Strategy

- Optimal solutions:
  - change making for "normal" coin denominations
  - minimum spanning tree (MST)
  - single-source shortest paths
  - simple scheduling problems
  - Huffman codes
- Approximations:
  - knapsack problem

Approximation Scheme for Knapsack Problem

Step 1: Order the items in decreasing order of relative values: $v_1/w_1 \geq \ldots \geq v_n/w_n$

Step 2: For a given integer parameter k, $0 \leq k \leq n$, generate all subsets of k items or less and for each of those that fit the knapsack, add the remaining items in decreasing order of their value to weight ratios

Step 3: Find the most valuable subset among the subsets generated in Step 2 and return it as the algorithm's output

Greedy Algorithm for Knapsack Problem

Step 1: Order the items in decreasing order of relative values: $v_1/w_1 \geq \ldots \geq v_n/w_n$

Step 2: Select the items in this order and skip those that don't fit into the knapsack

Example: The knapsack's capacity is 10

item	weight	value
1	7	$42
2	3	$12
3	4	$40
4	5	$25

Greedy Algorithm for Knapsack Problem

Step 1: Order the items in decreasing order of relative values: $v_1/w_1 \geq \ldots \geq v_n/w_n$

Step 2: Select the items in this order and skip those that don't fit into the knapsack

Example: The knapsack's capacity is 10

item	weight	value	v/w
1	7	\$42	6
2	3	\$12	4
3	4	\$40	10
4	5	\$25	5

Greedy Algorithm for Knapsack Problem

Step 1: Order the items in decreasing order of relative values:
$v_1/w_1 \geq \ldots \geq v_n/w_n$

Step 2: Select the items in this order and skip those that don't fit into the knapsack

Example: The knapsack's capacity is 10

item	weight	value	v/w
3	4	$40	10
1	7	$42	6
4	5	$25	5
2	3	$12	4

Enhanced Greedy Algorithm for Knapsack Problem

Step 1: Order the items in decreasing order of relative values: $v_1/w_1 \geq \ldots \geq v_n/w_n$

Step 2: Select the items in this order and skip those that don't fit into the knapsack

Step 3: Choose the better result of two alternatives:

- The one obtained in Step 2
- The one consisting of a single item of the largest value that fits into the knapsack

The value of an optimal subset will never be more than twice as large as the value of the subset obtained by this enhanced greedy algorithm