

ALGORITHMS AND DATA STRUCTURES II

Lecture 7

Greedy Algorithms,
File compression,
Huffman codes.

1/27

Lecturer: K. Markov
markov@u-aizu.ac.jp

GREEDY ALGORITHMS

- Algorithms for optimization problems typically go through a sequence of steps, with a set of choices at each step.
- A **greedy algorithm** always makes the choice that looks best at the moment. That is, it makes a **locally optimal** choice in the hope that this choice will lead to a globally optimal solution.

GREEDY ALGORITHMS

- Examples of greedy algorithms:
 - **Prim's** algorithm (for the minimum spanning tree problem).
 - **Kruskal's** algorithm (for the minimum spanning tree problem).
 - **Dijkstra's** algorithm (for the single source shortest path problem).

GREEDY ALGORITHMS

- A greedy algorithm obtains an optimal solution to a problem by making a **sequence** of choices.
- At each decision point, the algorithm makes a choice that **seems best** at the moment.
- This strategy **does not always** produce an optimal solution.

FILE COMPRESSION

- **Objective:** conserving of space (the memory space needed for storing the file)
- **Method:** use the redundancy (save space by exploiting the fact that most files have a relatively low "information content")

RUN-LENGTH ENCODING

- The simplest type of redundancy in a file is long runs (sequences) of repeated characters (no digits are allowed) e.g.:

**AAAABBBBAABBBBBBCCCCCCCCCDAB
CBAAABBBBCCCD**

- Above string can be encoded as :
4A3BAA5B8CDABCB3A4B3CD (every digit denotes the number of repetitions for the character that follows it)

RUN-LENGTH ENCODING

- If we want to use the same character set for encoding the lengths we need an "escape character" (*Q*) which signals the beginning of a (count, character) pair
- **AAAABBBBAABBBBBBCCCCCCCCCDABCBAABBBBCCCD** can be encoded as:
QDABBBBAAQEBQHCDABCBAAAQDBCCCD
- Repetitions: *D* = 4, *E* = 5, *F* = 6, *G* = 7, *H* = 8...
QQ is used for encoding a single *Q*

RUN-LENGTH ENCODING

- Bitmap encoding example

[illegible]

VARIABLE-LENGTH ENCODING

- Assume that the 26 letters A, B, ... Z are assigned the **5-bits binary codes** as follows:

1. **A** : 00001, 2. **B** : 00010,
3. **C** : 00011, , 26. **Z** : 11010.

- Given **ABRACADABRA**, using the 5-bits binary representation, it is encoded as the following:

VARIABLE-LENGTH ENCODING

ABRACADABRA:

000010001010010000010001100001

0010000001000101001000001

- To decode this message, read off 5 bits at a time and convert according to the above assignment.
- In the sequence, D appears once and A appears 5 times, while they are both assigned 5 bits in the binary codes.

VARIABLE-LENGTH ENCODING

- We may give an assignment like
A : 0, B : 1, R : 01, C : 10, D : 11,
and ABRACADABRA is encoded as
0_1_01_0_10_0_11_0_1_01_0
- This uses 15 bits compared with the 55 bits before. But, this code depends on the blanks to delimit the characters. Without the blanks, the string 010101001101010 could be decoded as RRRARBRRA or as several other strings.

VARIABLE-LENGTH ENCODING

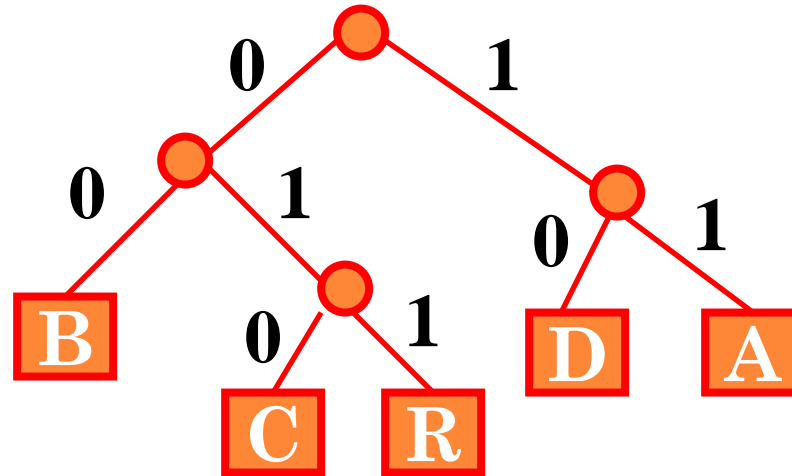
- How to encode the string compactly **without** using the delimiters?
- If the code of a character is **not** the prefix of any code of other characters, then delimiters are not needed.
 - 0, 00, 001, and 0011 are prefix of 00110.
 - In A : 0, B : 1, R : 01, C : 10, D : 11, the code of A is a prefix of R, the code of B is a prefix of C and D.

VARIABLE-LENGTH ENCODING

- A binary tree with M leaves can be used to encode any message with M different characters (one leaf holds one character).
- The code for each character is determined by the path from the root to the leaf that holds that character, with 0 for go 'left' and 1 for go 'right'.

VARIABLE-LENGTH ENCODING

- Example: The following binary tree encodes A as 11, B as 00, C as 010, D as 10, and R as 011.



ABRACADABRA is encoded as

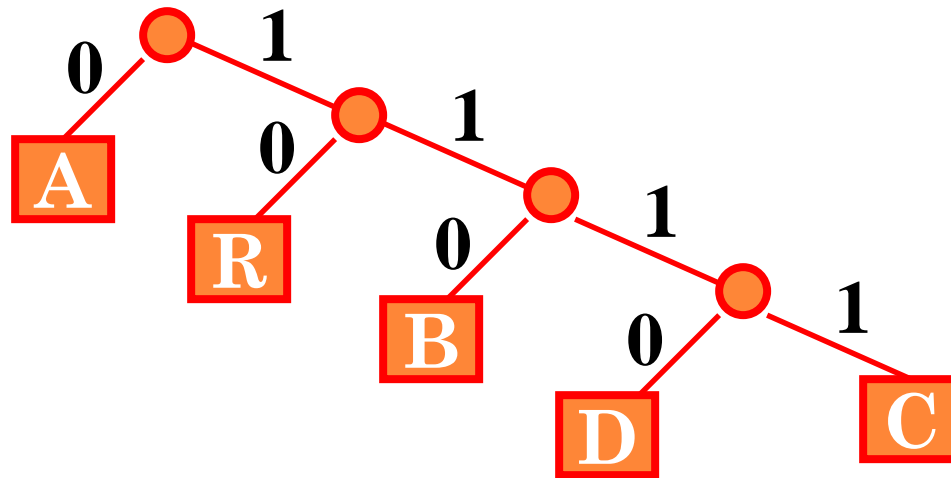
110001111010111011000111

VARIABLE-LENGTH ENCODING

- The binary tree representation **guarantees** that no character code is a prefix of another, so the string is uniquely decodable from the tree.
- **Decoding:** Starting at the root, proceed down the tree according to the bits of the message: each time a leaf is encountered, output the character at that leaf and restart at the root.

VARIABLE-LENGTH ENCODING

- The following is another binary tree for encoding **ABRACADABRA**.



VARIABLE-LENGTH ENCODING

- **Question:** which binary tree is the best one to use (gives the shortest binary sequence for a given string of characters)?
- D. Huffman gave an elegant method (called **Huffman encoding**) to compute a binary tree which leads to a bit string of minimal length for any given string of characters.

HUFFMAN ENCODING

- **Algorithm:** Let S be the set of characters that appear in a given string.

Step 1: Count the frequency of each character of S in the given string. Consider S as a set of roots of binary trees, each tree has one node.

Step 2: Find two roots with the minimum frequencies and generate a new node as the root of the two roots. The new root has the frequency of the sum of frequencies of its two children. Delete the two children from S and add the new root to S .

Step 3: If $|S| \neq 1$, go to Step 2.

HUFFMAN ENCODING

- Given the string **ABRACADABRA** we have:

$$S = \{A, B, C, D, R\},$$

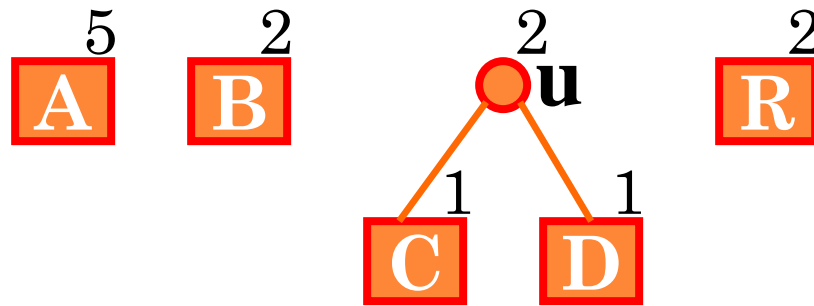
$$w(A) = 5, w(B) = 2, w(C) = 1,$$

$$w(D) = 1, w(R) = 2.$$

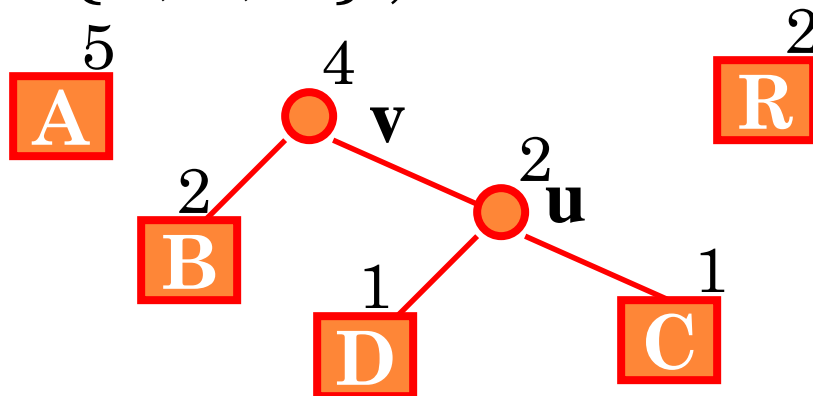
⁵
A ²
B ¹
C ¹
D ²
R

HUFFMAN ENCODING

$$S = \{A, B, u, R\},$$

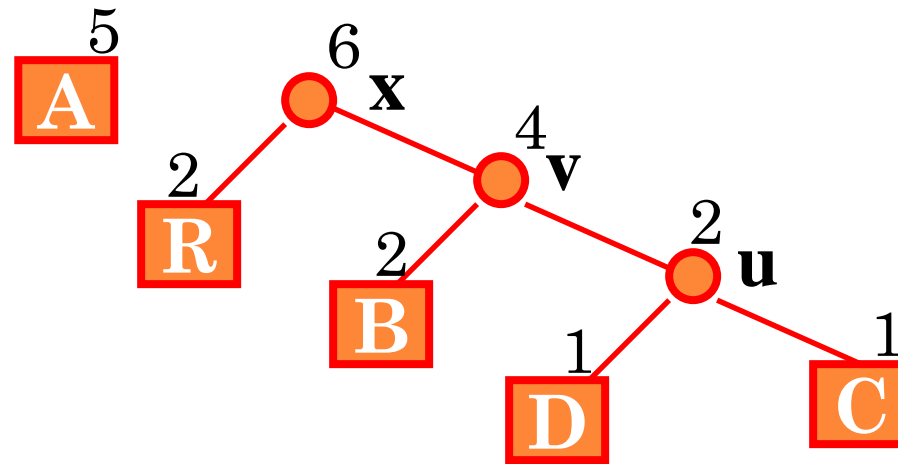


$$S = \{A, v, R\},$$

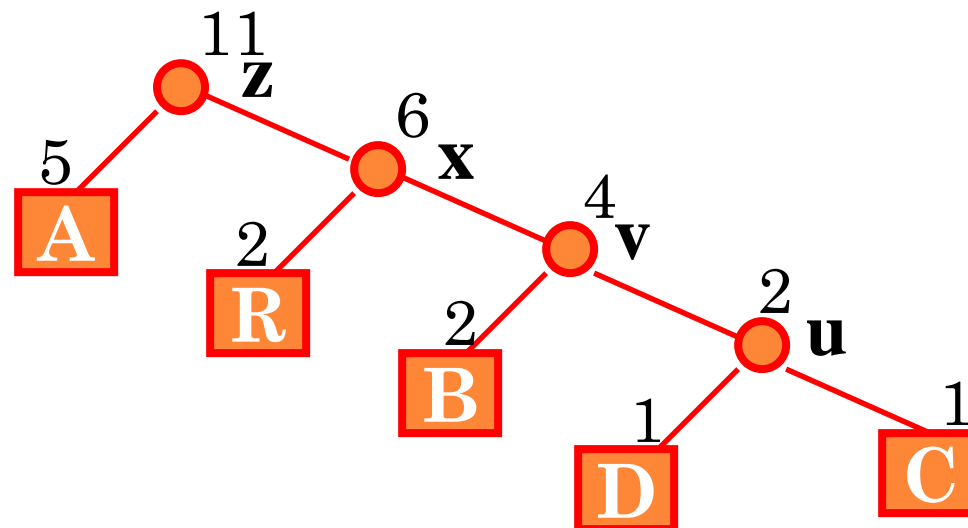


HUFFMAN ENCODING

$S = \{A, x\},$

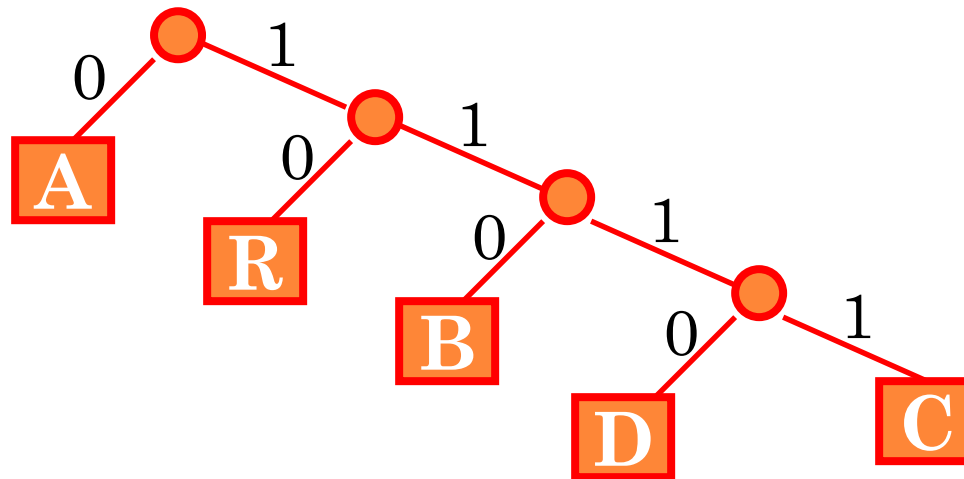


$S = \{z\},$



HUFFMAN ENCODING

- Encoding tree:



- We assign the binary codes to the characters of S as: A : 0, B : 110, C : 1111, D : 1110, and R : 10.

HUFFMAN ENCODING

- Algorithm implementation using minimum priority queue:

```
def HUFFMAN (S)
// Input: S – set of  $n$  letters with attribute freq giving their
// frequency. It is also the key for Q.
   $n = |S|$ 
   $Q = \text{MIN-PRIORITY-QUEUE}(S)$ 
  for  $i = 1$  to  $n - 1$ 
    allocate new node  $z$ 
     $z.\text{left} = x = \text{EXTRACT-MIN}(Q)$ 
     $z.\text{right} = y = \text{EXTRACT-MIN}(Q)$ 
     $z.\text{freq} = x.\text{freq} + y.\text{freq}$ 
    INSERT ( $Q, z$ )
  return EXTRACT-MIN ( $Q$ )
```

HUFFMAN ENCODING

- *Animated example:*

"j'aime aller sur le bord de l'eau les jeudis
ou les jours impairs"

File :

b	p	'	m	j	o	d	a	i	r	u	l	s	e	
1	1	2	2	3	3	3	4	4	5	5	6	6	8	12

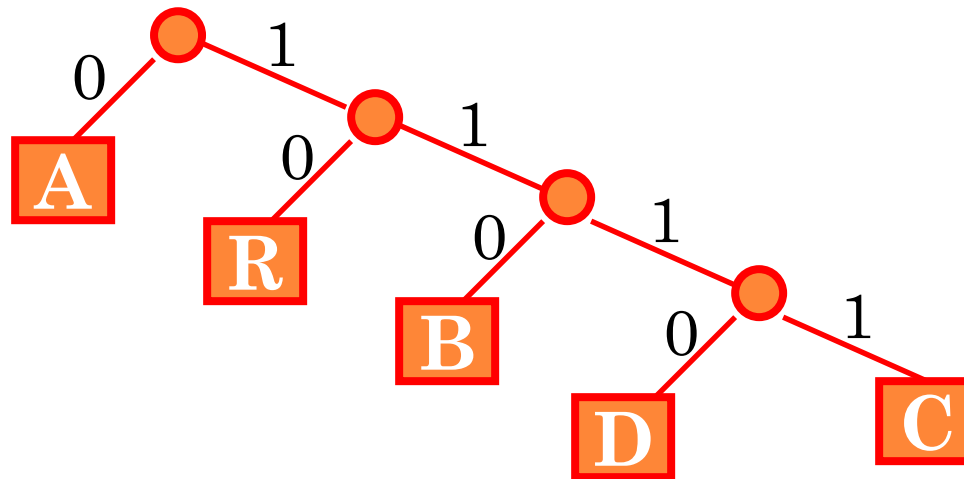
HUFFMAN ENCODING

◦ Decoding:

- Starting with the first bit in the stream, one then uses successive bits from the stream to determine whether to go left or right in the decoding tree.
- When we reach a leaf (character) of the tree we place that character to the output (decoded) stream.
- **Note:** Compressed file must contain the table of codes.

HUFFMAN ENCODING

- Decoding example:



01101001111011100110100

A B R A C A D A B R A

THAT'S ALL FOR TODAY!