LECTURE-28

Encoding – Huffman Coding

For Poll Ev

CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan
Department of Computer Science, SBASSE

Agenda

- Compression

- Huffman Coding

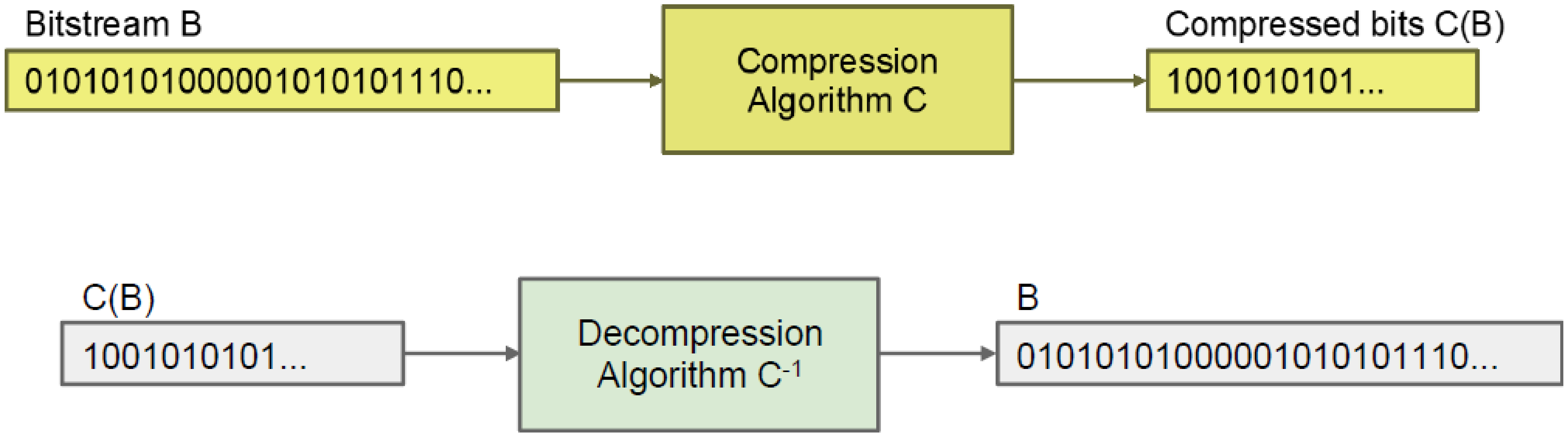
Data Compression

What do you think is the purpose of compression?

Compression means shrinking the file size of a text or audio/visual file.

Compression is the …

Process of encoding information in fewer bits



Compression

Compression can be …

Lossless OR

Lossy

Compression can be ...

Lossless OR

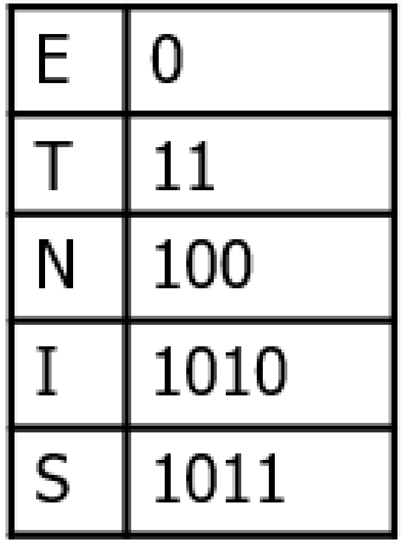
Lossy

Compression can be ...

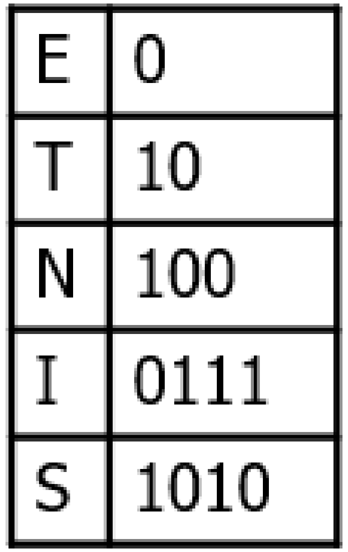
Lossless OR

Lossy

Decode the string on both sides according to the coding tables given in the picture.



E	0
T	11
N	100
I	1010
S	1011



E	0
T	10
N	100
I	0111
S	1010

11010010010101011

100100101010

While the first string decodes to TENNIS the second string decodes to:

1. NNTS
2. TENST and
3. TETEST

Rules for Code Table

- No prefix of a codeword is a codeword
- Uniquely decodable

Suggestions for Coding Schemes

- **No prefix of a codeword is a codeword**
- **Uniquely decodable**

Rules for Code Table

Following are two popular schemes for designing codes for compression table (coding table):

- Keeping codes for frequently occurring letters short as opposed to less frequent occurring letters in text.
- Identifying repeating bit patterns in text file and replacing them with short codes in the table.

Huffman Coding

- **WinZip** uses Huffman Coding to compress a text file!
- **Compression Table** is generated for each file. For decoding the table is consulted to recover the file in its original form.
- Codes are generated based on **letter frequency** in the file.

How are Huffman Codes generated

1. Sort letters by frequency put these frequencies in an ordered collection. (At each step we need two least frequencies … DS of choice?)

2. Take two least frequencies make a binary tree with two letters and their frequencies and make root represent the combined frequency of both. Add new frequency of root to the ordered frequency collection.

3. Repeat step 1 and 2 till you do not have two entries to pull out of the the collection.

4. You will have landed with one binary tree. Leaves will be all letters from your text. Follow a path from root to leaf, adding a 0 to code if you took a left turn and a 1 to code if you took a right turn. When you get to a leaf you will have Huffman code for letter that is marked at the leaf.

Generating Huffman Codes

String X: **FACE A FACADE**

Find the frequency of all characters in string X.

A	**C**	**D**	**E**	**F**	**S(space)**
4	2	1	2	2	2

Generating Huffman Codes

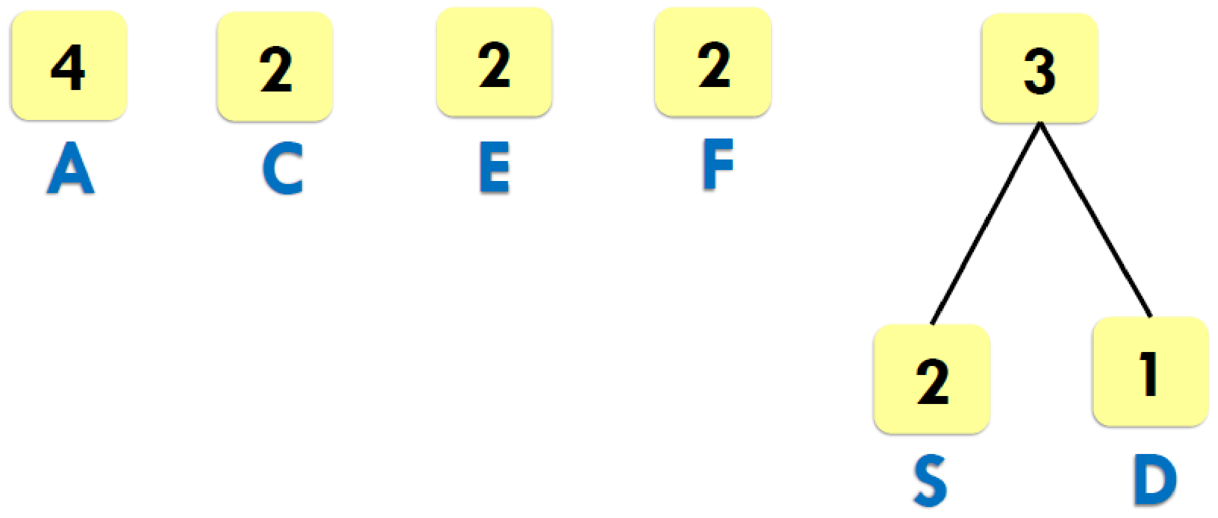
1. Place all frequencies in a Min Heap for quick smallest value retrieval!

2. To start with we will consider each character and its frequency as a single binary tree node. (Thereby at the start we have as many binary trees as the number of characters)

Generating Huffman Codes

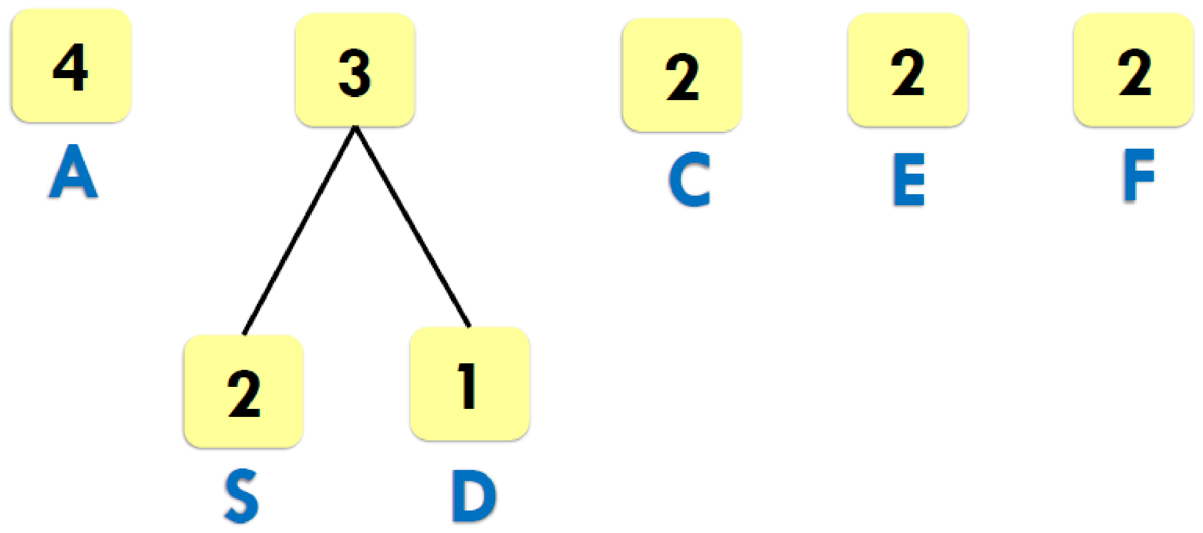
Our frequencies are in a **Min Heap** for quick min() removal! We call removeMin() twice from the min heap to get the smallest and then second smallest frequency.

- Take two binary trees with smallest frequency and merge them into a single binary tree



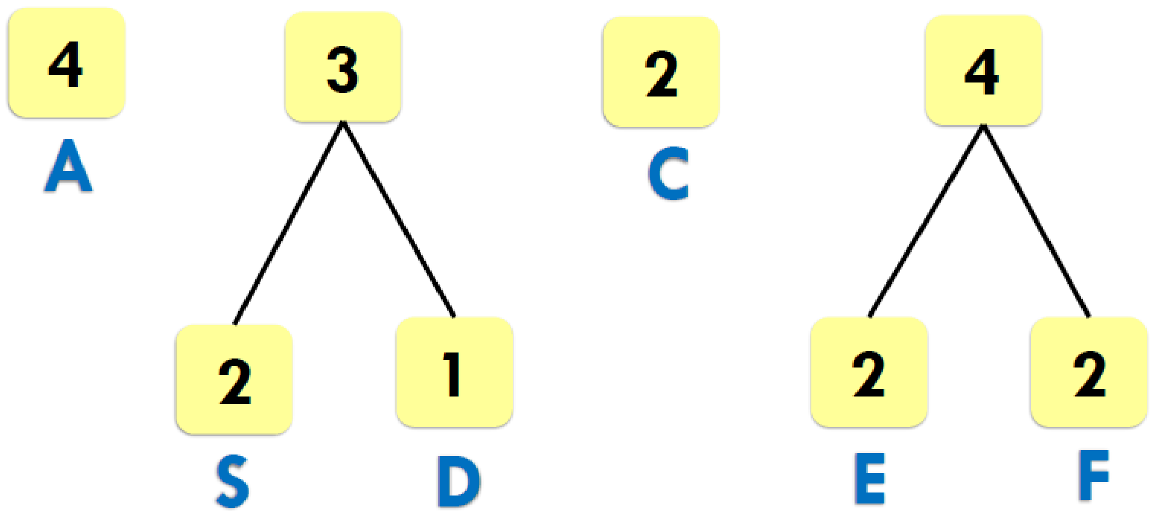
Generating Huffman Codes

- Take two binary trees with smallest frequency and merge them into a single binary tree.
- Repeat the process until we are left with single tree.



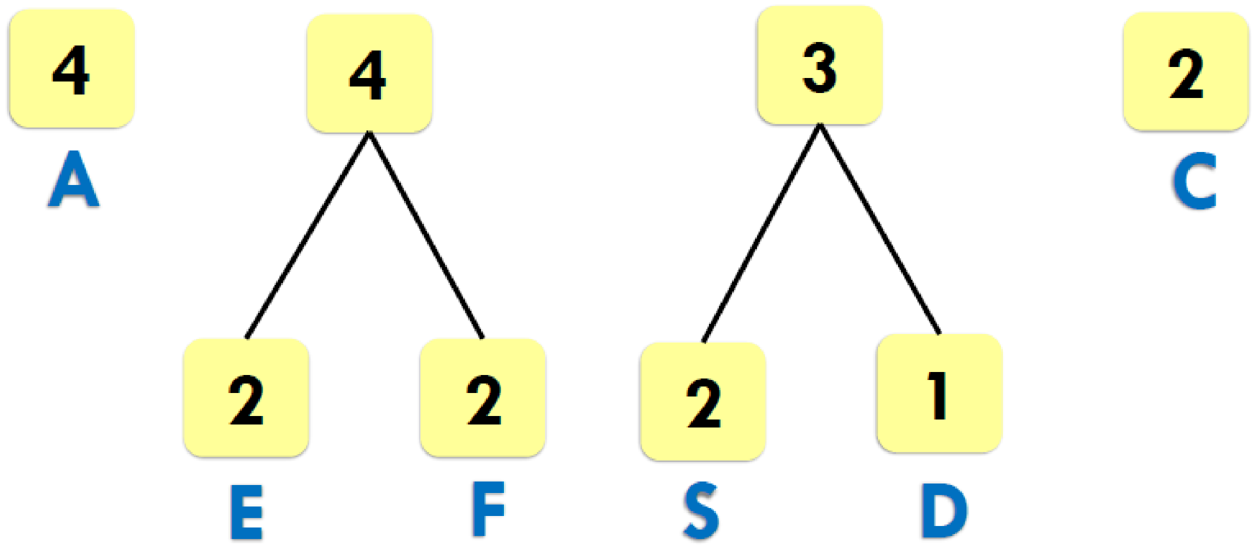
Generating Huffman Codes

- Take two binary trees with smallest frequency and merge them into a single binary tree.
- Repeat the process until we are left with single tree.



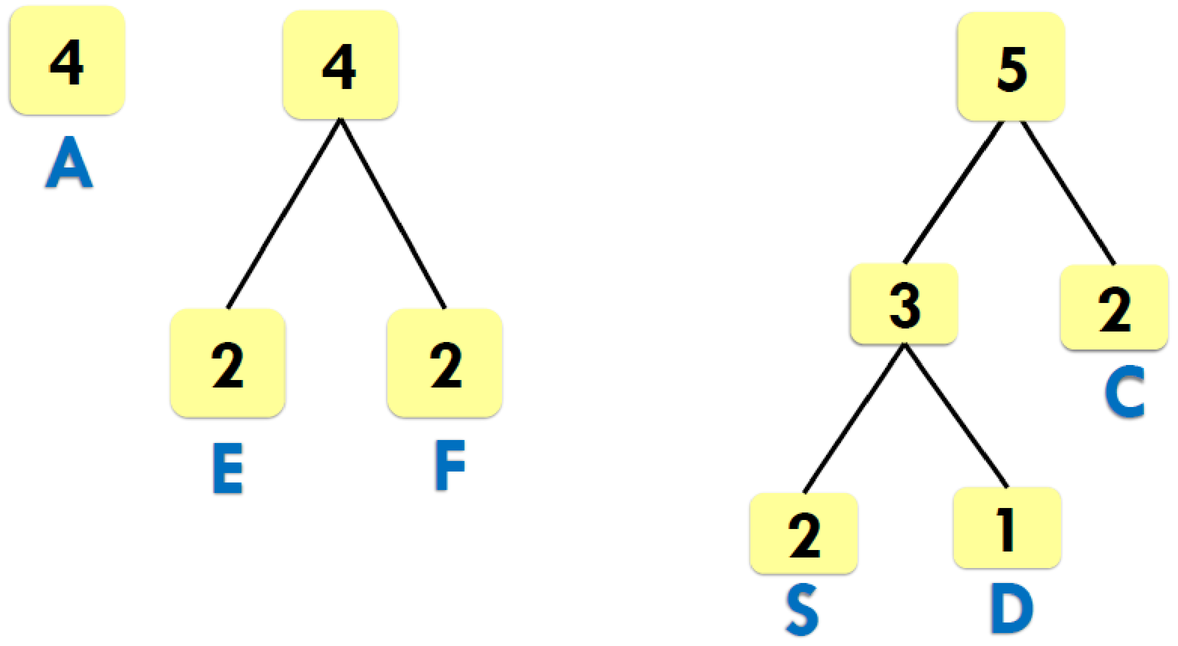
Generating Huffman Codes

- Take two binary trees with smallest frequency and merge them into a single binary tree.
- Repeat the process until we are left with single tree.



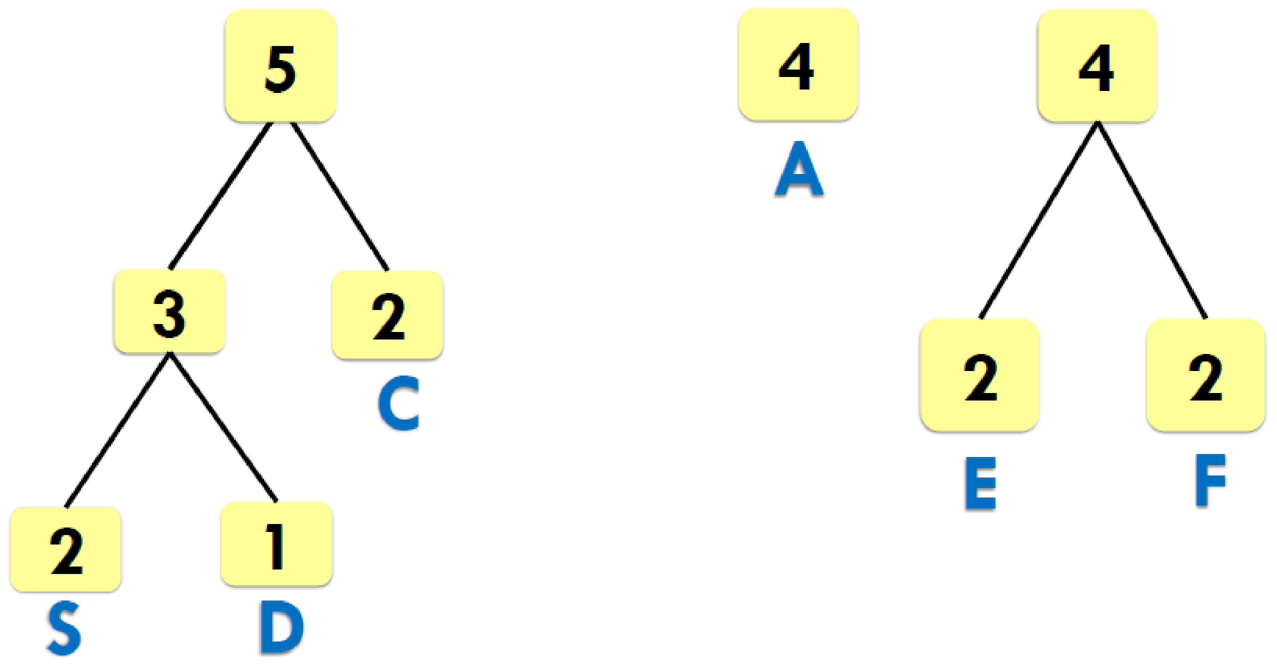
Generating Huffman Codes

- Take two binary trees with smallest frequency and merge them into a single binary tree.
- Repeat the process until we are left with single tree.



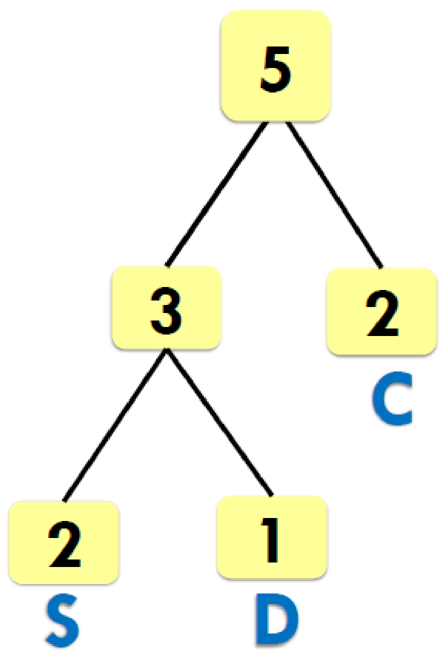
Generating Huffman Codes

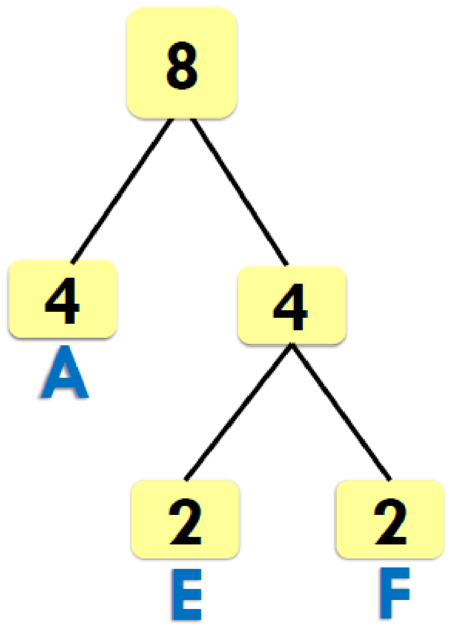
- Take two binary trees with smallest frequency and merge them into a single binary tree.
- Repeat the process until we are left with single tree.



Generating Huffman Codes

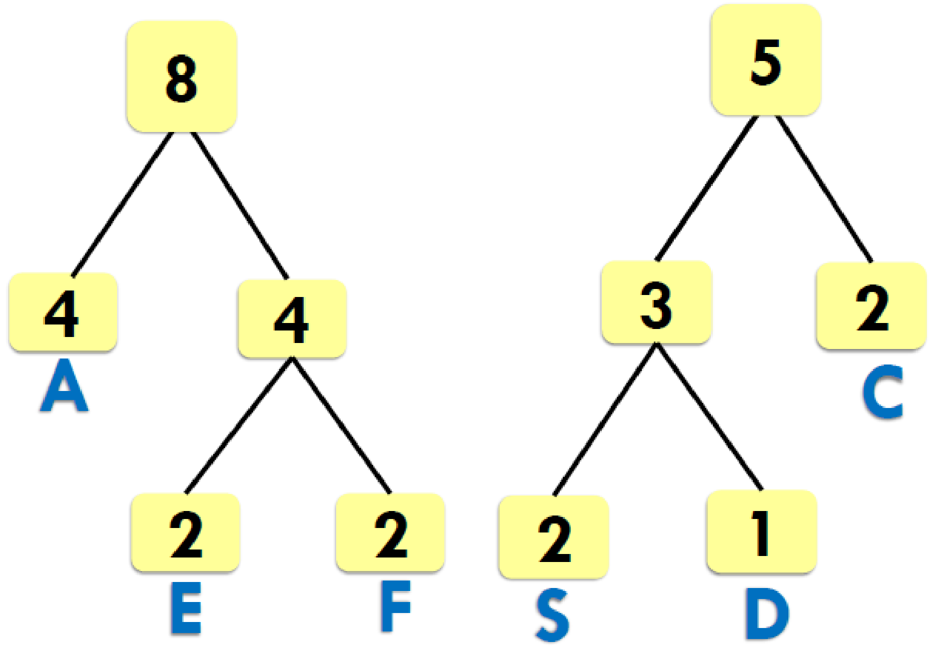
- Take two binary trees with smallest frequency and merge them into a single binary tree.
- Repeat the process until we are left with single tree.





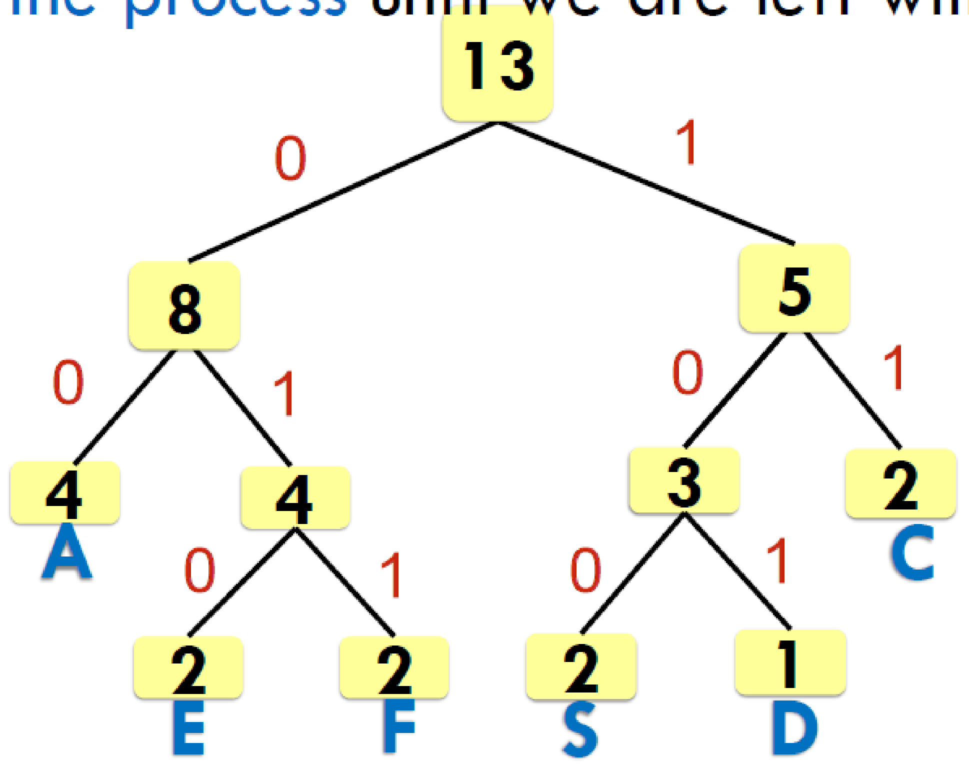
Generating Huffman Codes

- Take two binary trees with smallest frequency and merge them into a single binary tree.
- Repeat the process until we are left with single tree.



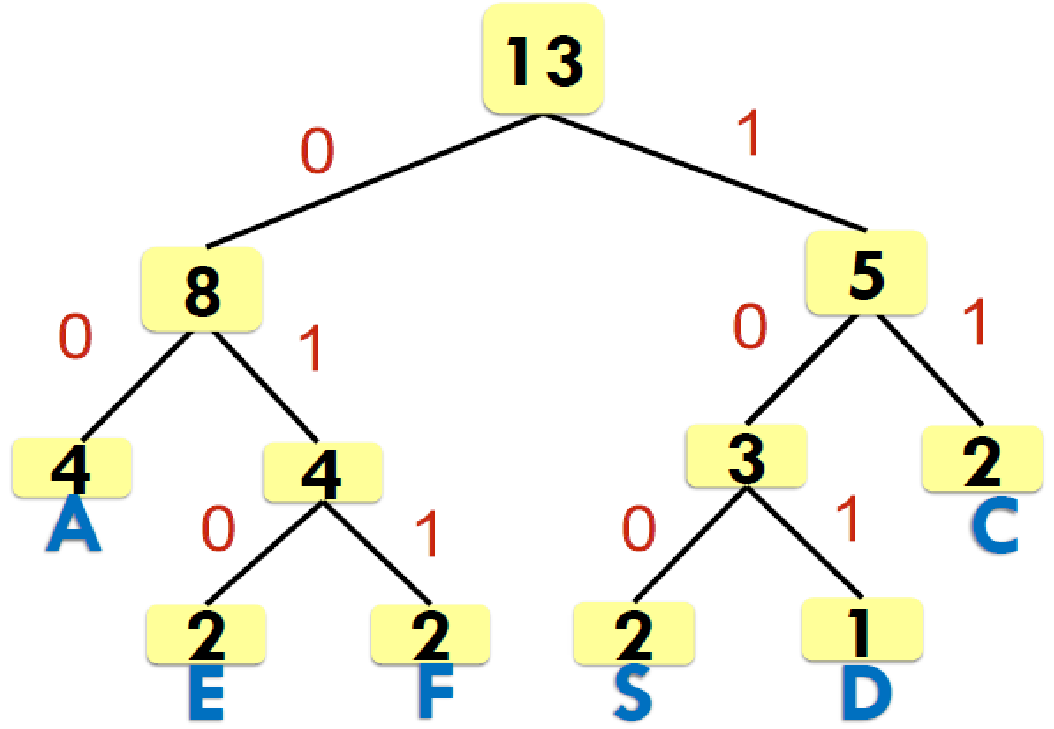
Generating Huffman Codes

- Take two binary trees with smallest frequency and merge them into a single binary tree.
- Repeat the process until we are left with single tree.



Generating Huffman Codes

- External node stores the character
- Internal nodes store the frequency of all external nodes rooted at v
- Trace a path from root to leaf for code construction for each character



Final encoding

A: 00
E: 010
F: 011
S: 100
D: 101
C: 11

Total bits: 33

When is Huffman Coding ineffective?

Ineffective Encoding:

If the input characters have equal weights (or frequencies), the algorithm is unable to build a coding tree that significantly shortens the codes for any particular character. The resulting codes assigned to the characters would be of nearly equal length, leading to minimal or no compression.

Feedback Form

https://evaluation.lums.edu.pk