# IIT Gandhinagar

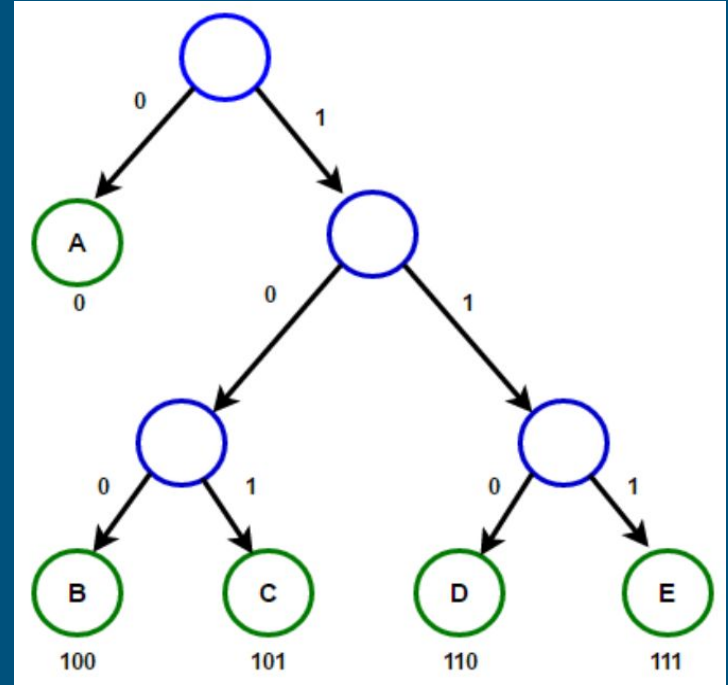## Project: Parallel ZIP

Two encoding schemes:

- Huffman Coding
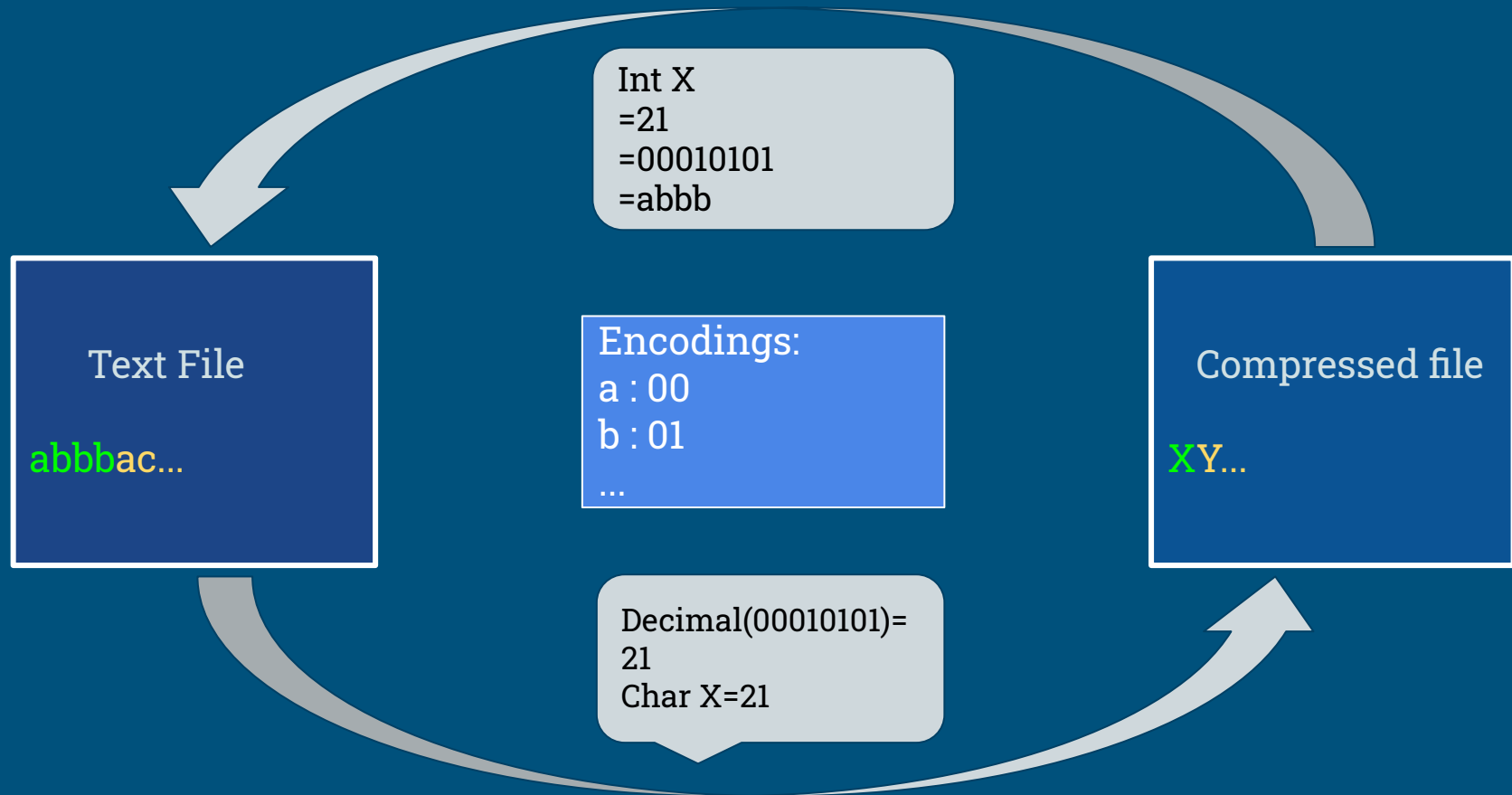- Run-length Coding

Team Members:

Ajinkya (18110013) | Jitender (18110075) | Palak (18110110) | Sagar (18110149)

# Huffman Coding Implementation

- Huffman Tree

  - Variable length encoding (greedy)
  - Prefix rule- no ambiguity in decoding
  - Time complexity -O(nlogn)

# Compression through Huffman encoding:

# Run Length Encoding(RLE)

Better!!

Text file
hello

hello

1h1e2l1o

for run length <=2, no encoding

Text file
wwwwbbbbb

wwwwbbbbb

Better!!

4w5b

for run length >2, do encoding

# Run Length Encoding(RLE)

Based on run-length of each character write "count" only when needed!

| Text file | |
|---|---|
| **Text file** | |
| hellowwbbb | → | hello3w3b |

**In worst case!!**
sizeof(zipped file) =
sizeof(original file)

Final implementation on alphanumeric input



1. Extra character(ASCII 17) is used to separate count and character.
2. Extra character is used only when count>2, ensuring in **no case** we get a zip.txt > file.txt

# Analysis:

Reduction-by % = (actual size - zipped size / actual size)*100

1.)

**random**
Type: plain text document (text/plain)
Size: 3.9 kB (3,860 bytes)

Original size

**henc_random**
Type: Binary (application/octet-stream)
Size: 2.1 kB (2,114 bytes)

45%
Huffman

**randomzip**
Type: plain text document (text/plain)
Size: 3.9 kB (3,860 bytes)

0%
Run-Length

2.)

**new**
Type: plain text document (text/plain)
Size: 1.7 kB (1,669 bytes)

3.)

**random6**
Type: plain text document (text/plain)
Size: 4.1 kB (4,097 bytes)

57%

**henc_new**
Type: plain text document
Size: 715 bytes

65%

**newzip**
Type: Binary (app
Size: 575 bytes

81%

**henc_random6**
Type: plain text document
Size: 767 bytes

11%

**random6zip**
Type: Binary (application/c
Size: 3.6 kB (3,641 bytes)

Thank You!