Parallel ZIP

Two encoding schemes:

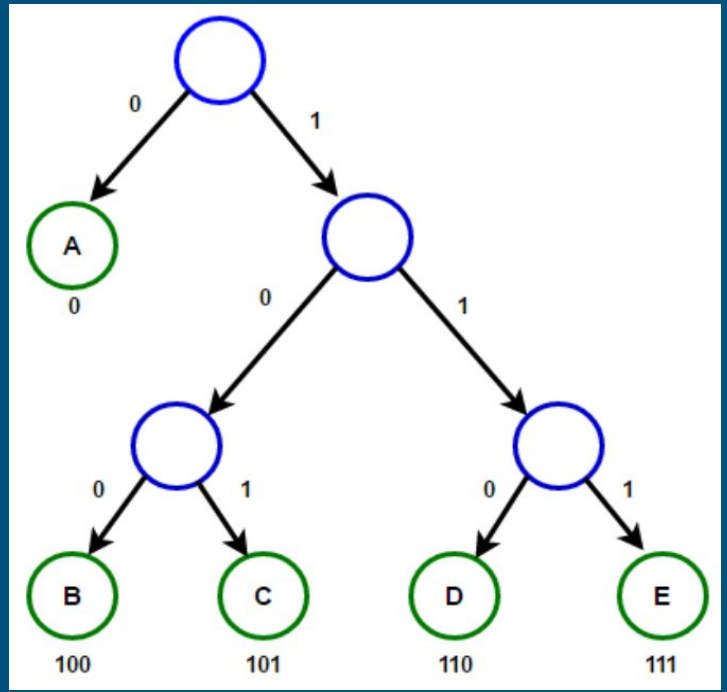
- Huffman Coding
- Run-length Coding

Team Members:

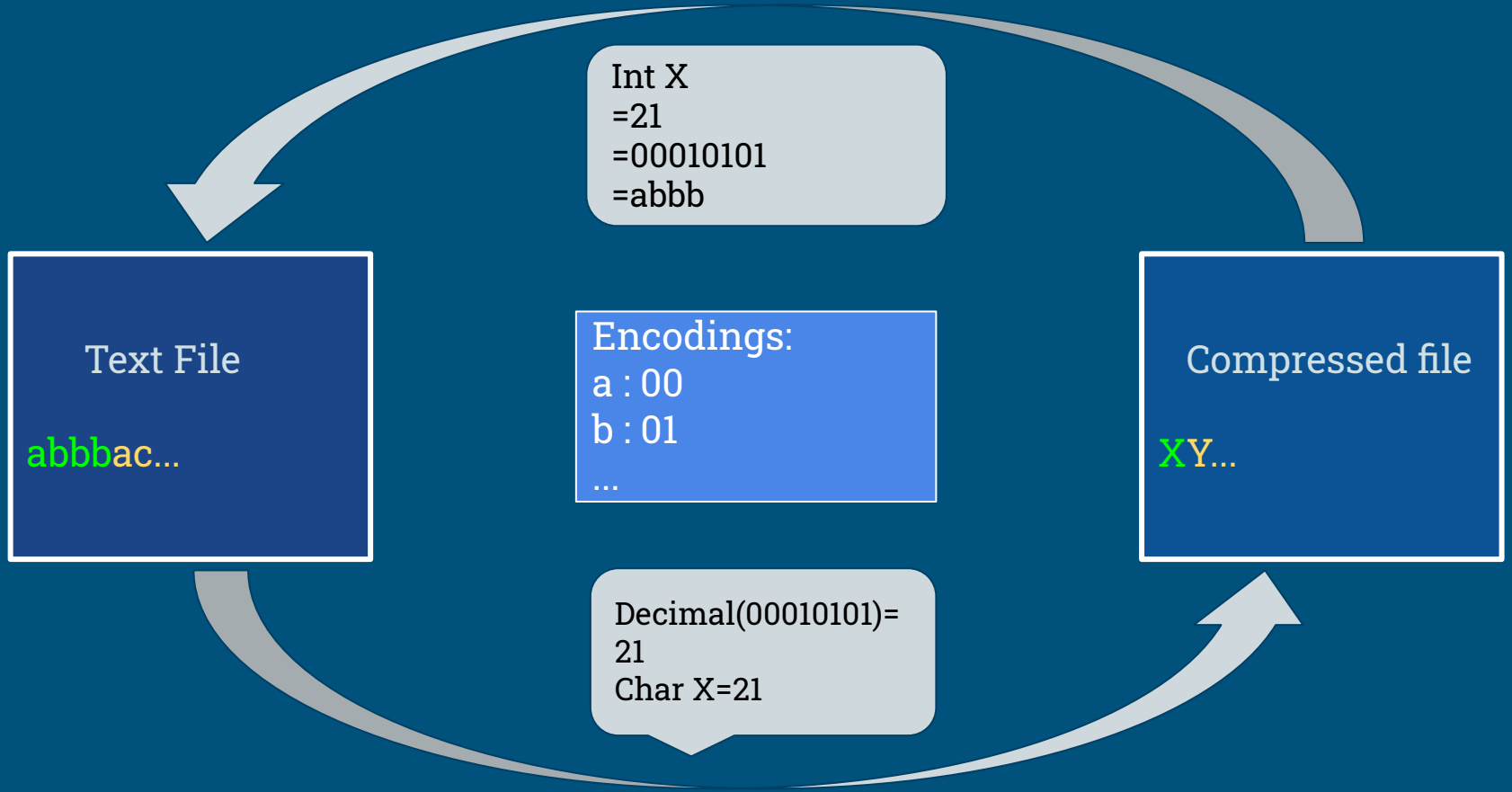
Ajinkya (18110013) | Jitender (18110075) | Palak (18110110) | Sagar (18110149)

Huffman Coding Implementation

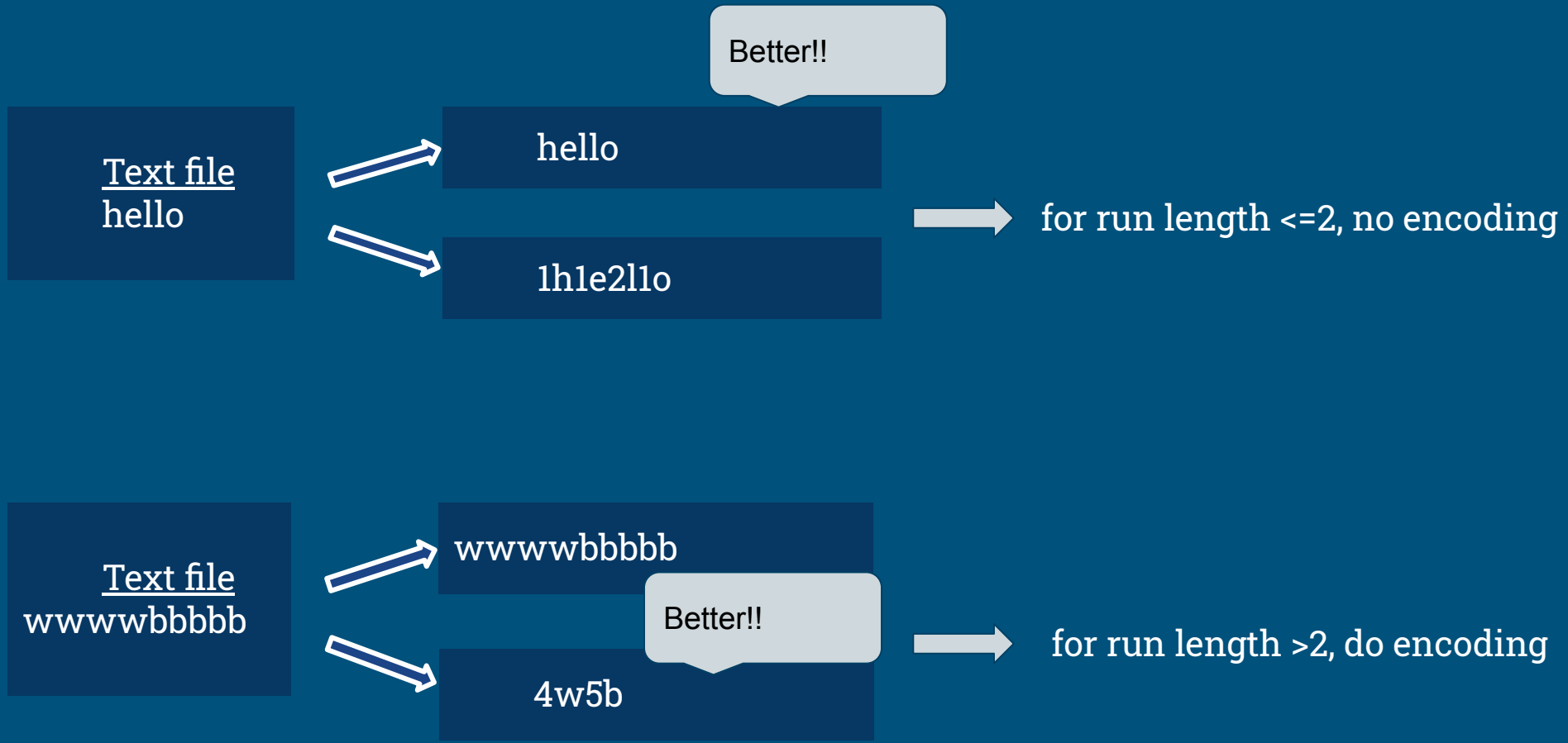
- Huffman Tree
  - Variable length encoding (greedy)
  - Prefix rule- no ambiguity in decoding
  - Time complexity -O(nlogn)



Compression through Huffman encoding:



Run Length Encoding(RLE)



Run Length Encoding(RLE)

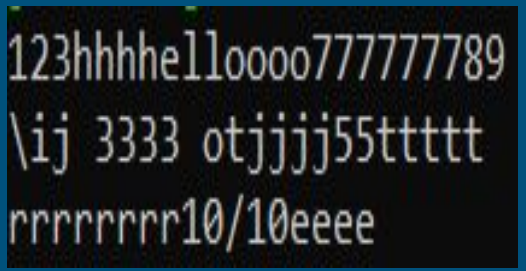
Based on run-length of each character write “count” only when needed!

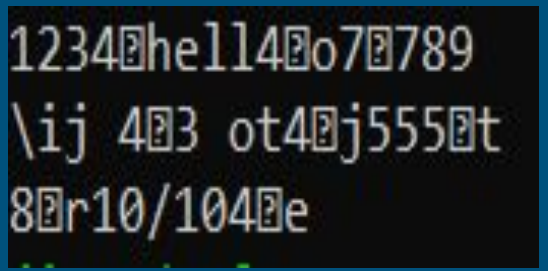
Text file
hellowwwbbb

hello3w3b

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

Final implementation on alphanumeric input



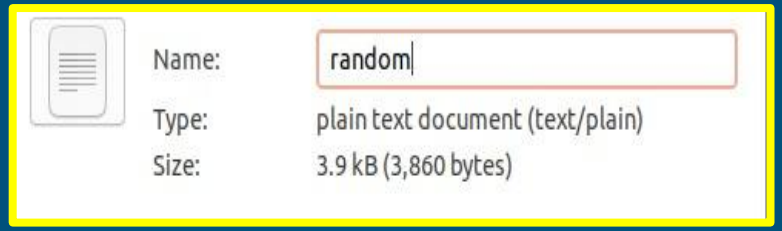


1. Extra character(ASCII 17) is used to separate count and character.
2. Extra character is used only when count>2, ensuring in **no case** we get a zip.txt > file.txt

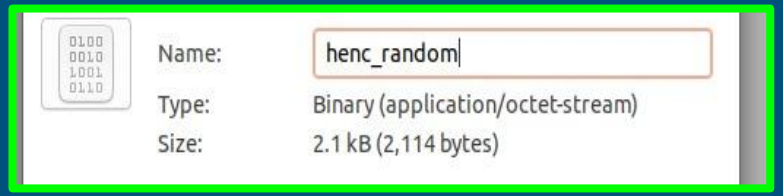
Analysis:

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

1.)



Original size



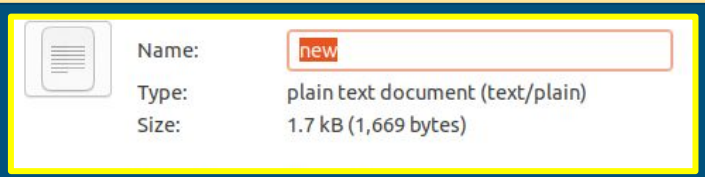
45%

Huffman

0%

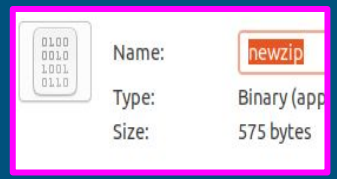
Run-Length

2.)

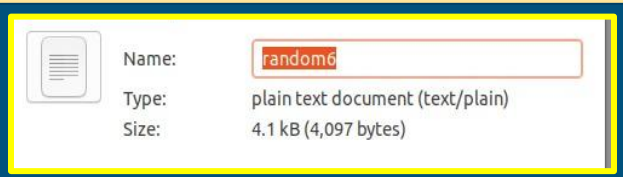


57%

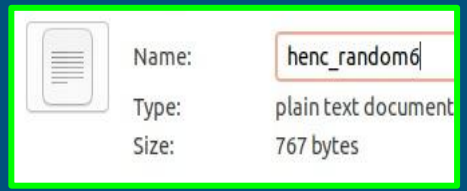
65%



3.)



81%



11%

Thank You!