Introduction to Lab
Data Compression

José Nelson Amaral

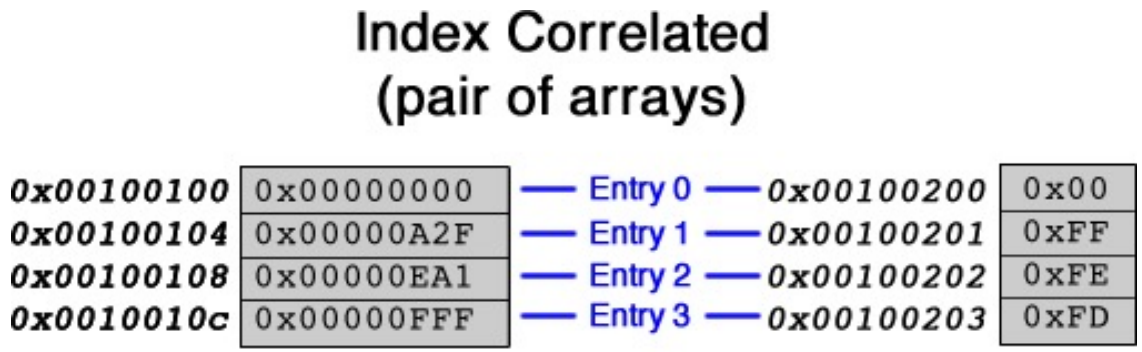
Data Compression

- Converting data into a smaller form for storage or transmission
  - Lossy Compression
  - Lossless Compression

Dictionary Compression

- Lossless compression
- Method of reducing redundancy in data
- Copies common patterns in a data structure called a **dictionary**
- Replaces occurrences of those patterns with a small reference to the dictionary element containing the full copy

Tables in Memory



Input

An array of words stored in memory

The end of the array is signaled by a sentinel word

`common.s`:

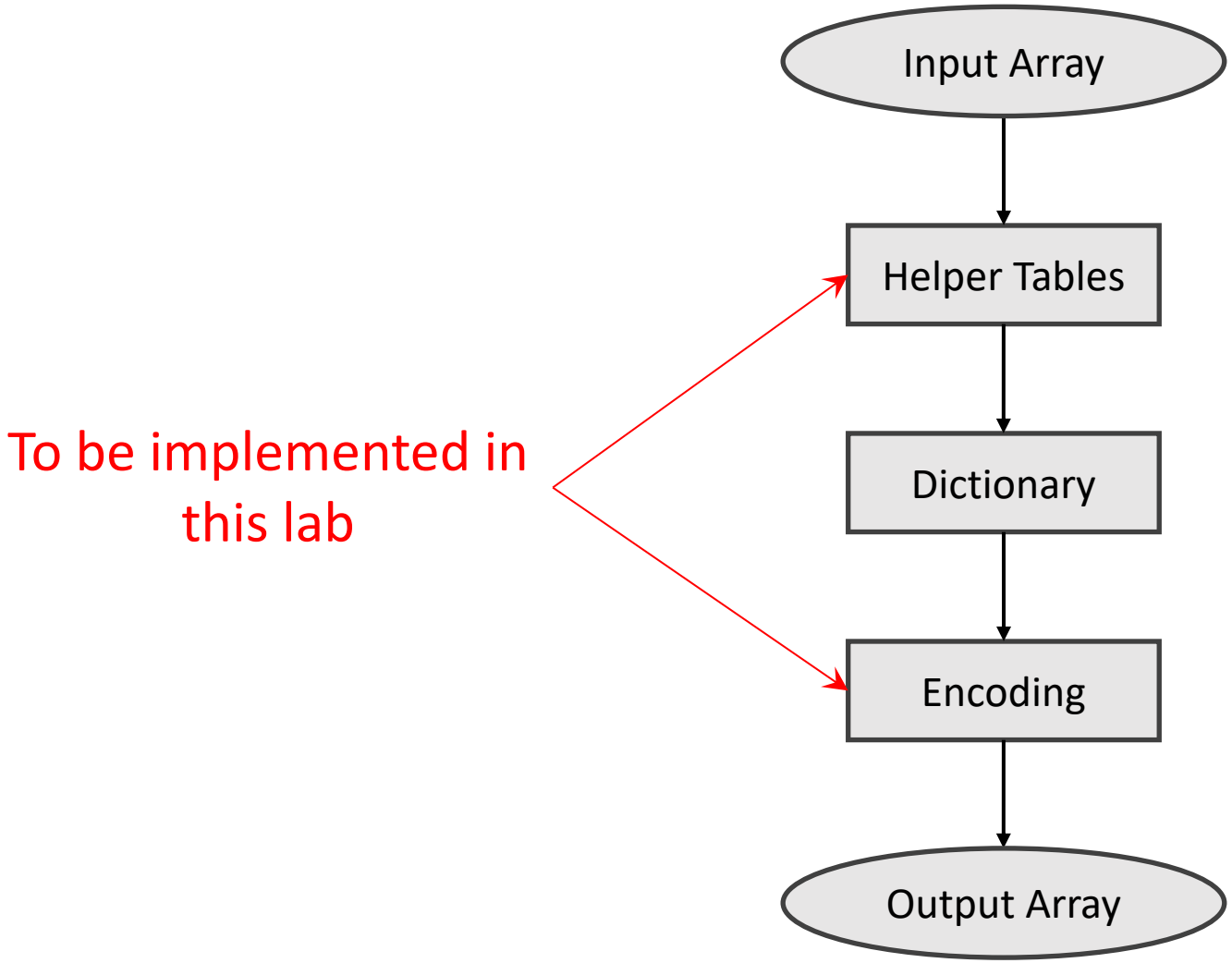
- reads a text file and stores in memory as an array of words
- passes address of first word to `buildTables` and `encode` functions

Conversion of Input Text File into Array of Words Memory

twinkle_twinkle_little_star!

Address	Word
0x10010800	twin
0x10010804	kle_
0x10010808	twin
0x1001080c	kle_
0x10010810	litt
0x10010814	le_s
0x10010818	tar!
0x1001081c	\0\0\0\0

Lab Compression Format Workflow



Word Table

- Stores each unique word from the input file exactly once, in the order that they first appeared in the file

twinkle_twinkle_little_star!

(Index)	Address	Word
00000000	0x10011000	twin
00000001	0x10011004	kle_
00000010	0x10011008	litt
00000011	0x1001100c	le_s
00000100	0x10011010	tar!

Count Table

- Contains the number of times that each word appears in the input
- Each count is stored as a byte
- Count Table is index correlated to the Word Table

twinkle_twinkle_little_star!

Word Table

Address	Word
0x10011000	twin
0x10011004	kle_
0x10011008	litt
0x1001100c	le_s
0x10011010	tar!

Index
00000000
00000001
00000010
00000011
00000100

Count Table

Address	Count
0x10021000	2
0x10021001	2
0x10021002	1
0x10021003	1
0x10021004	1

Dictionary

- Table containing up to 128 words from the Word Table
- Needs to be part of the output for decoding to be possible
  - To avoid taking too much space, only includes words that occur multiple times in the input
  - Words are picked using the Count Table
- A function to build a dictionary is provided to you

twinkle_twinkle_little_star!

(Index)	Address	Word
00000000	0x100120fc	twin
00000001	0x100120fd	kle_

Creating the Compressed Sequence

- For each word in the input
  - If the word does not have a copy in the dictionary, output the word as-is
  - If there is a copy in the dictionary, instead output a **dictionary reference byte**
- The decompressor can tell that a byte is a reference to the dictionary because the most-significant bit (MSB) of the byte is 1.

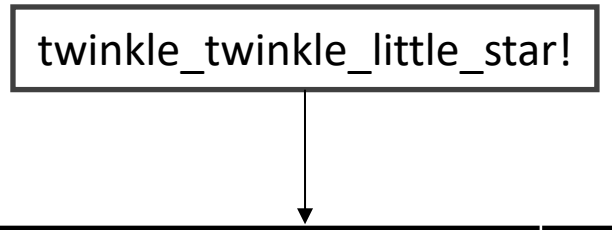
Dictionary reference byte to index 1: 10000001

bit [7]: 1

bits [0-6]: the index in the dictionary containing the word being referenced

Compressed Sequence

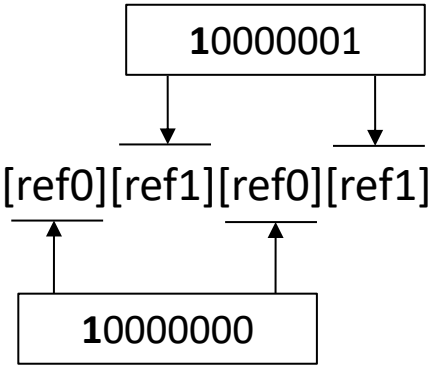
Input: twinkle_twinkle_little_star! (28 bytes)



Dictionary:

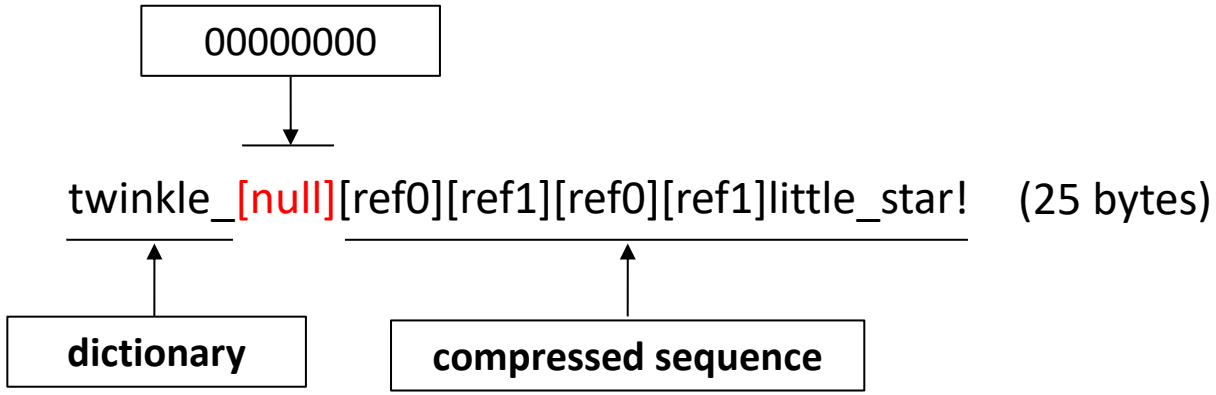
(Index)	Address	Word
00000000	0x100120fc	twin
00000001	0x100120fd	kle_

Compressed Sequence: [ref0][ref1][ref0][ref1]little_star! (16 bytes)



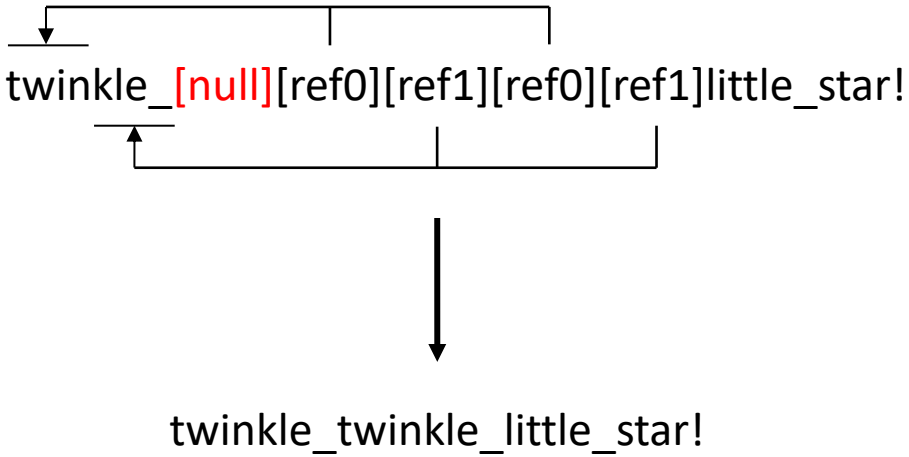
Output Format

- Place entire dictionary at the beginning of the output
- Place a null character byte (0) to signal end of dictionary
- Place the compressed sequence
- Place the end-of-array sentinel word



Decoding

- Replace dictionary reference bytes with the referenced word in the dictionary



Lab Assignment

- `buildTables`: generate word table and index-correlated count table
- `buildDictionary` – provided to students: generates a dictionary based on the word and count tables generated in `buildTables`
- `encode`: generate the output

Testing

- `print-tables.o`: prints tables generated by student solution column by column
- `print-encoding.o`: prints encoding generated by student solution in a human-readable format
- `decode.o`: decodes the encoding and prints it
- CheckMyLab