CSE 13S Winter Quarter 2022
Assignment 6: Huffman Coding

Description of the Program:

The goal of this assignment is to emulate the Huffman coding algorithm in both encoding and decoding.

Files to be included in the “asgn1” directory

- encode.c
- decode.c
- defines.h
- header.h
- node.h
- node.c
- pq.h
- pq.c
- code.h
- code.c
- io.h
- io.c
- stack.h
- stack.c
- huffman.h
- huffman.c

Layout:

- encode.c
    - employs the encoding of the input
- decode.c
    - employs the decoding of the input
- huffman.c
    - utilizes a lot of the huffman algorithms using the adts we made
- adt’s
    - used throughout the rest of the code

Pseudocode:

- Node *node_create
    - malloc necessary space
    - initialize symbol and frequency
- void nod_delete
    - free malloced memory
    - clear pointer
- Node *node_join

    - frequency = left frequency + right frequency
    - symbol = $
- void node_print
    - print symbol
    - print frequency

- Priority Queue *pq_create
    - malloc necessary memory size by capacity value
- void pq_delete
    - free memory
    - clear pointer
- bool pq_empty
    - if head and tail are same return true
    - else return false
- bool pq_full
    - if the successor of head is tail return false
    - else return true
- uint32_t pq_size
    - return size
- bool enqueue
    - set pointer of i as q head
- bool dequeue
    - set pointer of i as tail
- void pq_print
    - loop through pq_size
        - print value from queue

- Code code_init
    - malloc enough space
    - set first block to 0
- uint32_t code_size
    - return code size
- bool code_empty
    - if code size == 0 return true
    - else return false
- bool code_full
    - if code size == Block return true
    - else return false
- bool code_full
    - if code size == 256 return true
    - else return false
- bool code_set_bit
    - bit twiddling
    - int 256 of 0 bits turn bit i into 1

        - pointer to c |= with new 256 length bit
- bool code_clr_bit
    - bit twiddling
    - int 256 of 1 bits turn bit i into 0
    - pointer to c &= with new 256 length bit
- bool code_get_bit
    - bit tiwddling
    - int 256 of 0 bits
    - pointer to c ^= with 0
    - if c != c return false
    - else return true
- bool code_push_bit
    - push to stack
- bool code_pop_bit
    - pop stack
- void code_print
    - loop through code size
        - print values

- int read_bytes
    - read until EOF
- int write_bytes
    - fwrite(with size block)
- read_bit
    - fwrite(with size 256)
- write_code
    - write until empty
- flush_code
    - write remaining code

- Stack *stack_creaate
    - malloc necessary space capacity
    - initialize capacity and top
- void stack_delete
    - free allocated memory
    - clear pointer
- bool stack_empty
    - if the top of stack is the capacity return true
    - else return false
- bool stack_full
    - if stack_empty return false
    - else return true

Credit:

- Eugene’s lab section for the general idea of the lab
- Dr. Long’s lecture for some help on the adt’s