CSE 13S Assignment 6 - The Great Firewall of Santa Cruz

Jaren Kawai - jkawai

https://users.soe.ucsc.edu/~elm/classes/cse13s/banhammer-dist

- Bloom Filter
    - Variables for salts, keys, hits, misses, and bits examined
    - Bitvector filter
  - bf_create
    - Allocate memory for bloom filter
    - If bf do below, otherwise return pointer to bf
    - Keys and bits start at 0
    - Misses and bits examined start at 0
    - Create for loop that runs for about of hash functions
      - Set salt[i] to default_salts[i]
    - Create bit vector of size size
  - bf_delete
    - Free memory allocated by constructor
    - Null out parameter
  - bf_size
    - Take length of bit vector and return it
  - bf_insert
    - Take oldspeak and hash it with the five salts
    - Set bits at index calculated by hashing oldspeak
  - bf_probe
    - Check if each of the indices and return true if all of them are set
    - Otherwise return false
  - bf_count
    - Return number of set bits in the bloom filter
    - Create counter and iterate using a for loop
  - bf_print
    - Print out the contents of a bloom filter
  - bf_stats
    - Sets each passed pointer to the value of the original statistic that the variable is tracking
- Bit Vectors
    - Variables for length and *vector
  - bv_create
    - Allocate memory for bit vector
    - If bit vector do below, otherwise return pointer to bit vector
    - Otherwise return a bit vector or a pointer to a bit vector
  - bv_delete

        - Free memory allocated for the bit vector
        - Set passed in pointer to null
    - bv_length
        - Return the length of bit vector
    - bv_set_bit
        - Set the bit as passed in at index i
        - Location is equal to i / 64
        - Position of bit in the byte is i % 64
        - Bitwise or operation
    - bv_clear_bit
        - Clear the bit at index i passed in
        - Find location as before, and use bitwise and operation
    - bv_get_bit
        - Return the bit at index i
        - Find location as before and return bit at calculated location
    - bv_print
        - Print bit vector
        - Should be written after the constructor
- Hash table
        - Struct with variables for salt, size, keys, bits, misses, and bits examined
        - Bool for move to front
        - Linked list created here as well
    - ht_create
        - Allocate memory for hash table
        - If hash table, do below, otherwise return pointer to hash table
        - Initialize default values for each of the variables tracked by the hash table function
        - Crate linked list
    - ht_delete
        - Free all linked lists in lists
        - Null out passed pointer
    - ht_size
        - Return size of hash table
    - ht_lookup
        - Search for a node that contains oldspeak
        - Hash oldspeak to get index of linked list needed for look up
        - If node is found, return pointer to the found node
        - Otherwise null pointer is returned
        - Track number of lookups and links examined use ll_stats call at beginning of function and before return

  - ht_insert
    - Insert oldspeak and newspeak into hash table
    - Hash oldspeak to get location
    - Initialize linked list first if no linked list at location
    - Otherwise check if oldspeak exists in linked list
    - Don't key is already there
  - ht_count
    - Return number of null linked lists in the hash table
  - ht_print
    - Print out contents of hash table
  - ht_stats
    - Set tracked variables such as keys, hits, misses, and linked examined
- Node
    - Char pointers for oldspeak and newspeak
    - Node for next and prev
  - node_create
    - Make a copy of newspeak and oldspeak
    - Allocate memory and then copy contents in
  - node_delete
    - Free node at passed in value n
    - Free memory for newspeak and oldspeak at that node
    - Set pointer passed in to null
  - node_print
    - Use provided functions to print the contents of a node
    - Print oldspeak and newspeak if not equal to null, otherwise print oldspeak only
- Linked list
    - Variable for length
    - Nodes for head and tail
    - Bool for move to front
  - ll_create
    - Allocated memory for linked list
    - If linked list do below, otherwise return pointer to linked list
    - Set mtf to true or false depending on user input
    - Set sentinel nodes for the head and tail
  - ll_delete
    - Free each node in the linked list use node_delete
    - Set pointed passed in to null
  - ll_length
    - Return length of linked list

    - ll_lookup
        - Search for a node containing oldspeak, use loop to iterate over nodes
        - Return pointer to node if found
        - Move to front if move to front option is specified
        - Otherwise return null pointer
    - ll_insert
        - Use lookup to make sure oldspeak is already not in the linked list
        - Otherwise insert node at the head of the list
    - ll_print
        - Print each node of the linked list except sentinel nodes
    - ll_stats
        - Copy the number of lookups in n_seeks and number of linked traversed into n_links
- Parser
        - File pointer
        - Char for current line
        - Int for line_offset
    - parser_create
        - Allocate memory for parser
        - If parser, do below, otherwise return pointer to parser
        - constructor for parser, set default values for tracked variables
    - parser_delete
        - Set pointer to null
    - next_word
        - Check for invalid characters or spaces before
        - If fgets returns null return false
        - Find next valid word and save in buffer
        - Use f gets to get the current line
        - Iterate over the line and stop when invalid character or space is found
        - Increment offset value for every valid character that is found, reset offset if newline is reached
        - Use loop to copy found word into word buffer
        - Check for invalid characters again
        - Return true if can parse a given line
- banhammer.c
    - Initialize bloom filter and hash table
    - Read in list of old bad speak
    - Read in oldspeak
    - Create linked list for both goodspeak and and badspeak that correlates to any user input words

  - Read in user input from stdin, use parser and next word to get each word that the user inputs and do the below steps to determine if a person is guilty of thoughtcrime
  - Check if words have been added to bloom filter, check if probe returns true
    - Check if word has a newspeak translation, otherwise citizen is guilty of thoughtcrime and insert into badspeak. Variable for thoughtcrime is true.
    - If word has a newspeak translation, goodspeak variable is true.
    - Add words to linked lists created above accordingly
    - No action needs to be taken otherwise
  - Generate errors if there are any
  - Output mixspeak message if thoughtcrime and requires counseling
  - Print linked list for both thoughtcrime and goodspeak
  - If only thoughtcrime, then send badspeak message
  - Print linked list for thoughtcrime
  - If only counseling, then send goodspeak message
  - Print linked list for goodspeak
  - Get opt options
    - Take in options h for usage
    - t for size of hash tables
    - f for size of bloom filter
    - s statistics that are sent to stdout
      - Bits per miss is bits examined - hashes * hits all over misses
      - False positive is misses over hits
      - Filter load is count over size
      - Average seek length is examined over hits + misses
    - m to enable move to front rule
  - Free all created objects at the end of program
  - Return 0 at the end
- Plot.sh
  - Bloom filter size and bits examined per miss
    - Create loop that starts at default bloom filter size and adds 1000 each iteration
    - Run banhammer with same input every time to prevent confounding variables
    - Do same step above but add higher high values
    - Condense dat files
    - Add i value and bits examined per miss to a new dat file
    - Graph dat file with lines
  - Bloom filter size and hash table lookups

        - Create loop that starts at default bloom filter size and adds 1000 each iteration
        - Run banhammer with same input every time to prevent confounding variables
        - Do same step above but add higher high values
        - Condense dat files
        - Add i value and hash table probes to a new dat file
        - Graph dat file with lines
    - Hash table size and hash table probes
        - Create loop that starts at default hash table size and adds 10000 each iteration
        - Run banhammer with same input every time to prevent confounding variables
        - Do same step above but add higher high values
        - Condense dat files
        - Add i value and hash table probes to a new dat file
        - Graph dat file with lines