Assignment 7 Design Document

Devasha Trivedi

March 11, 2022

This assignment has us implement a program that identifies the most likely authors of a given text using stylometry.

1 Files to Submit

1. bf.h: This header file was provided to us and contains the interface for the Bloom ADT.
2. bf.c: This file contains my implementation of the Bloom filter ADT.
3. bv.h: This header file was provided to us and contains the interface for the bit vector ADT.
4. bv.c: This file contains my implementation of the bit vector ADT.
5. ht.h: This header file was provided to us and contains the interface for the hash table ADT.
6. ht.c: This file contains my implementation of the hash table ADT.
7. identify.c: This file contains my implementation fo the author identificaiton program.
8. metric.h: This header file was provided to us and contains definitions for distance metrics.
9. node.h: This header file was provided to us and contains the interface for the node ADT.
10. node.c: This file contains my implementation of the node ADT.
11. parser.h: This header file was provided to us and contains the interface for the parsing module.
12. parser.c: This file contains the parsing module implementation.
13. pq.h: This header file was provided to us and contains the interface for the priority queue ADT.
14. pq.c: This file contains my implementation of the priority queue ADT.

15. salts.h: This header file was provided to us and contains the definitions for the salts to be used in bf.c, as well as in ht.c
16. speck.h: This header file was provided to us and contains the interface for the SPECK cipher.
17. speck.c: This file contains the SPECK cipher implementation.
18. text.h: This header file was provided to us and contains the interface for the text ADT.
19. text.c: This file contains my implementation of the text ADT.
20. Makefile: This file helps compile and run my program.
21. README.md: This file contains a summary of the program, as well as how to interact with the Makefile.
22. DESIGN.pdf: This file contains the pseudocode of my implementations.
23. WRITEUP.pdf: This file discusses my program's behavior. It also serves as a reflection of how I coded the program and how it works.

2 Pseudocode

2.1 bf.c

```
bf_create
    use salts to initialize bf
bf_delete
    bv_delete(bf)
    free(bf)
bf_size
    bv_length
bf_insert
    set bits for:
    primary
    secondary
    tertiary
```

2.2 bv.c

```
bv_create
    set vector
    set length
    for i <= length:
      clr_bit()
```

```
bv_delete
    free(bv)
bv_length
    return length(bv)
refer to code comments for:
set_bit
clr_bit
get_bit
```

2.3 ht.c

```
use salts to initialize

ht_delete
node_delete
free(ht)

ht_size
return length(ht)

ht_lookup
while node != NULL:
  index++
  index %= ht_size
  node = ht_slots
return node

ht_insert
same while loop from ht_lookup
if node = NULL
  node = node_create(word)
  node_count++
  ht_slots = node
return node


HTI:

table = ht
slot = 0

hti_delete
free(hti)
```

```
ht_iter
node = table->slots[slot]
slot++
return node
```

2.4 node.c

```
create a node
    -use malloc
    -set word
   -set a counter variable
delete a node
    -use free
    -set node to NULL
```

2.5 pq.c

```
create pq
    -set capacity
    -set counter variable
    -set nodes
delete pq
    -use free
    -set everything to NULL
is pq empty
    empty = true
    if size(pq) > 0:
        empty = false
    return empty
is pq full
    empty = true
    if size(pq) < pq(capacity):
        empty = false
    return empty
enqueue
    build heap from asgn3
    heap goes up
dequeue
    build heap from asgn3
    heap goes down
```

2.6 text.c

```
int ht_size, bf_size

create text->ht and text->bf
word_count = 0

use a while loop and a bool
    -insert ht and bf
    -return text

delete text
    ht_delete
    bf_delete

total_count
    use hti and for loop
    for i < ht_size:
      Node *n = ht_iter(hti);
      if n != NULL:
        total_count += n->count;
    return total_count;

text_distance
    calculate all metrics
    if statements to set them

text_frequency
    if text has a word:
      Node *slot = ht_lookup
        if slot != NULL
          freq = slot_size
    return freq
```

2.7 identify.c

```
main() is gonna have getopt() loop
getopt() loop has command-line parsing

create a noise text & anonymous text
create a pq

for i < author_count:
```

```
get data required

if author = NULL:
    print(File cant be opened)
else:
    enqueue author and distance

print matches, metric names
dequeue author and distance
```

3 (

Credit)ss:credit)

- I referred to the bv8.h file in the code comments repo for the set, clr, and get bit functions in bv.c
- I went to Eugene's section to understand the assignment and some pseudocode.
- I referred to various chapters from the textbook to refresh my memory/assist my coding.
- I took node.c and pq.c from my asgn6 code.
- I used the heapsort logic from asgn3 for pq.c