Design Document Assignment 6

Abhinav Prasanna

December 6, 2021

Description

The goal of the assignment is to create a program to filter out content from the internet. We will create data structures such as bit vector, bloomfilter, node, binary search tree, and a hashtable.

Bit Vector

The bitvector represents a one dimensional array of bits which is used to denote true or false. We use n/8 uint8$_{t}insteadofntobeabletoaccess8indiceswithasingleintegermakingitextremely$

Pseudocode

```
Struct
BitVector bv
length
vector[]

BitVector bvcreate(size)
create bv
if(bv)
bytes = 1/8 + 1 remainder 8
create vector
set length
return bv

else
return 0


bvdelete(bv[][])
if(vector)
free vector
```

```
else
free bitvector
free vector

bvlength()
return length

bvsetbit()
if i in range
bv vector in 1/8 —= 0x1 set bit right of i mod 9
return true
return false

bvclrbit()
if i in range
bv vector in 1/8
return true
return false

bvgetbit()
return bv vector in 1/8 set bit right i mod 8 and 1

bvprint()
index
while( index ¡ len)
print bv vector
index ++
print end line
```

BloomFilter

Bloom Filter uses Bit Vector to set, check and clear bits. The Bloom Filter consists of our teritary, primary, and secondary values within salt,h and a BitVector data structure. Arrays will be used to store these values from salt.h. We need a delete, size, insert, probe, count, and print function.

Pseudocode

```
struct BF
arr tert
BV filter
arr prim
arr second

BF bfcreate(size)
create BF
check if bloom filter is not null
if so
set tert values
set prim values
set secondary values
if bf-¿fil
free fil
bf = NULL
return bf

bfsize(bf)
return bv length of fil

bfinsert(bf,oldspeak)
set bit of fil with hash prim and oldspeak mod bv length of bf fil
set bit of fil with hash second and oldspeak mode bv length of bf fil
set bit of fil with hash tert and oldspeak mode bv length of bf fil

bfprobe()
if ( bv get of bf fil with hash of prim second anmd tert with oldspeak mod bv
length)
return true
return false

bfcount()
count
for index in bf-¿fil
if bv get bit bf-¿ fil from index
count ++

    return count

bfprint()
bv print bf-¿fil
```

HashTable

Hash table is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index, also called a hash code, into an array of buckets or slots, from which the desired value can be found. Hash Table will be used to store several binary search trees.

Pseudocode

```
Struct HashTable
Node[][] trees
size
salt[]

htcreate()
create hashtable
if(hashtable)
set size
set salt values

htdelete(ht[][])
for index in ht size
bst delete ht-¿trees in index
free ht trees
free ht
ht = NULL

htsize(ht[])
return size

htlookup()
index = hash in ht salt and oldspeak mod ht size
if(ht-¿trees in index is NULL)
return NULL
else
return bst find if ht trees index in oldspeak
```

BST

Binary Search Tree is is a rooted binary tree data structure whose internal nodes each store a key greater than all the keys in the node's left subtree and less than those in its right subtree. We use this search tree to store oldspeak and newspeak values.

Pseudocode

```
bstcreate()
return NULL

bstheight(root)
if root is NULL
return 0
else if root left is NULL and root right is NULL
return 1
else
if bstheight(root left) ¿ bstheight(root right)
return 1 + bstheight(root left)
else
return 1 + bstheight(root right)
bstsize(root)
if root is NULL
return 0
else
return bstsize(root left) + 1 + bstsize(root right)

bstfind(root,oldspeak)
if root
if strcmp root-¿oldspeak ¿ oldspeak
return go root-¿left
else if strcmp root-¿oldspeak ¡ oldspeak
return go root-¿right
return root
return NULL

bstinsert(root,oldspeak,newspeak)
if root
strcmp root-¿oldspeak ¿ oldspeak
root-¿left = recursive go left
strcmp root-¿oldspeak ¡ oldspeak
root-¿right = recursive go right
return root
return node create oldspeak to newspeak

bstprint(root)
if root is NULL
return
bst print root left
node print root
bst print root right
bstdelete(root[][])
```

```
if(root[])
bstdelete(root-¿left)
bstdelete(root-¿right)
nodedelete(root)
```

Node

Nodes are devices or data points on a large network, devices such a PC, phone, or printer are considers nodes. In general, a node has a programmed or engineered capability that enables it to recognise, process, or forward transmissions to other nodes

Pseudocode

```
Node nodecreate(oldspeak,newspeak)
create node
if node
set right and left to NULL
if old speak
set node to oldspeak
else
oldspeak = NULL
repeat for newpseak
return node
nodedelete()
if new speak
delete newspeak
if oldspeak
delete oldspeak
if node
delete node

nodeprint(node)
if oldspeak and newspeak
print oldspeak and newspeak
else
print oldspeak
```

banhammer

Banhammer will read regex, badspeak, and newspeak and use the exisitng data structures to create mixpseak badspeak and goodspeak message.

Pseudocode

```
int main(void)
buff = calloc 4097
buff2 = calloc 4097
hashtable
bloomfilter
create nodes thoughtcrime and rightspeak
create opt functions
case h print help
case s statsverbose
case t
set sizeofht
case f
set sizeofbloomfilter
break
default printhelp break
set badspeak
set newspeak
read badspeak
set bf and ht
read newspeak
set bf and ht
while read word
set lowercase
if we find in bf
node = ht lookup from ht in word
if node
if node-¿newspeak and node-¿oldspeak
rightspeak bst insert oldspeak and newspeak
else if node oldspeak
thoughtcrimes bst insert oldspeak and null

if statsverbose
print bst size
print bst height
print average branches
print bloomfilter
else
thoughtcrimesize = bstsize(thoughtcrimes)
```

```
rightspeaksize = bstsize(rightspeak)
if thoughtcrimesize is greater than 0 and rightspeaksize greater than 0
print badspeak message
bst print thoughtcrimes and rightspeak
else if thoughtcrime size
print badspeak message
bst print thoughtcrimes
else
print goodspeak message
bst print rightspeak
free memory
```