Week 13 Exercises Submit

Sandra Batista

1.1–1.2

1. Hash Table Insertion

b) Start with an empty hash table of size m = 10. Use the hash function h(k)=k%10 and draw the contents of a hash table after inserting keys 1, 11, 2, 21, 12, 31, 41 using linear probing.

1. Hash Table Insertion

c) Start with an empty hash table of size m = 9. Use the hash function h(k)=k%9 and draw the contents of a hash table after inserting keys 36, 27, 18, 9, 0 using quadratic probing.

2. Hash Table Find Algorithm

- Given the following hash table, trace the steps for find(3). The hash function is h(k)=k%7 (m=7) and quadratic probing was used to hash the items.

14	8	21	2	7	__	__
0	1	2	3	4	5	6

3. Hash Table Remove Algorithm

- Given the following hash table, trace the steps for remove(21) followed by find(7). The hash function is h(k)=k%7 (m=7) and linear probing was used to hash the items.

- 14 8 21 2 7 __ __
  0 1 2 3 4 5 6

4. Bloom Filter Exercise

- Problem 5 from practice final a:
- https://bytes.usc.edu/cs104/resources/final-a.pdf
- We have a Bloom Filter with an array of 10 elements (the elements of the set are integers), and using three hash functions:
- h1(x) = (7x + 4) mod 10
- h2(x) = (2x + 1) mod 10
- h3(x) = (5x + 3) mod 10.
- We execute the sequence of operations given below. What does the program output? Which of the answers are false positives (the Bloom filter says “Yes”, even though the correct answer is “No”)? Which are false negatives (the Bloom filter says “No”, even though the correct answer is “Yes”)? If your final answer is incorrect, you may get more partial credit if you show enough work for us to isolate the mistake.

```
BloomFilterSet bf (10);
 bf.add (0);
bf.add (1);
bf.add (2);
bf.add (8);
// Show us what the Bloom Filter’s Array looks like at this point.
if (bf.contains (2)) std::cout << "2\n";
if (bf.contains (3)) std::cout << "3\n";
if (bf.contains (4)) std::cout << "4\n";
if (bf.contains (9)) std::cout << "9\n";
```

5. Compressed Trie Exercise

a) Construct a compressed trie for the following set of keys: Heap, Helm, Hear, Help, He, In, Ink, Irk, She
b) Insert the key Hawk to the compressed trie

6. Backtracking Review Exercise

This is an extra credit challenge review exercise written by *Stephanie Yoshimoto*.

Starter code:
https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/maze_solver.cpp

Maze file: https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/maze.txt

Given a text file representing a maze, maze.txt, the driver code reads the data into a 2D array. +, -, and | represent walls that cannot be passed. Empty spaces, ‘ ‘, are valid steps for the algorithm to search on. This function to read the maze is given already.

Write a recursive function, solve, to determine if there is a solution to go from the start of the maze to the end. The start will always be at the top left (array index [1][0]) and the end will always be at the bottom right (array index [rows-2][columns-1]). Represent the resulting path with asterisks (‘*’). You may use helper functions if desired.

If a solution is found, the driver code will write it to an output file given as an argument.