CS010C

Lab9 (No.9 Lab)

Hash Lab

- See Canvas -> Lab6 -> zip file -> “hashlab.html”
- main.cpp
  - for 4 hash functions
- collisions.cpp
  - for open hashing collision
- Demo on Thursday

main.cpp

Include your ~~Main.java~~ for testing.

Rubric

- 20 points - Attendance
- 60 points - 4 Hash functions and main test method
- 20 points - Open Hashing Collision Testing

main.cpp

```
0:    nnnnn,
1:    ooooo,
2:    ppppp,
3:    qqqqq,
4:    hello, happy, heath, harps, rrrrr,
5:    iiiii, sssss,
6:    jjjjj, ttttt,
7:    kkkkk, uuuuu,
8:    lllll,
9:    mmmmm,
```

- separate chaining for collision resolution
  - A hash table containing M entries
  - each entry is a list that stores the elements
- How to "bring the result into the range of the hash table"
  - Use modulo operation by M
    - `hashCode %= M; // OR`
    - `hashCode = hashCode % M;`
- 3 hash functions that take string keys
  - Hash function 1
    - Correction: Use the ASCII value of the first character
    - See sample output for test1.txt
  - Hash function 2, 3
- Your own hash function that takes a string as the key

Use the 1st one

1. This hash function ~~adds up~~ the ASCII values of the characters in the key.

m	m
n	n
o	o
p	p

collisions.cpp

- open addressing hashing (with probing)
- Hash table size M, insert N random numbers
  - Randomly generate ints
    - Initialization

      ```
      srand(0);
      ```

    - Generate a batch of random numbers

      ```
      for (int i = 0; i < N; ++i)
      {
          int hashCode = rand() % M;
          // do sth …
      }
      ```

- probing
  - linear probing
  - quadratic probing
  - cubic probing
- Count the number of collisions during the process
  - Collisions increase as the load factor increases

```
❯ ./a.out 10007 9000
Here is the number of collisions for each type of probing.
linear = 18098
quadratic = 15487
cubic = 15160
❯ ./a.out 10007 7000
Here is the number of collisions for each type of probing.
linear = 6014
quadratic = 5587
cubic = 5530
❯ ./a.out 10007 5000
Here is the number of collisions for each type of probing.
linear = 2114
quadratic = 2061
cubic = 2033
~/UCR_CS010C_25U/Lab9  main
```

Probing

Linear hash table

H[0]	-1
H[1]	-1
H[2]	-1
H[3]	-1
H[4]	-1
H[5]	1
H[6]	-1
H[7]	-1
H[8]	-1
H[9]	-1
H[10]	-1

Insert at 5 (1st time)

Insert at 5 (2nd time)?

So insert at 6

- Initialize table to -1 or 0
  - (Indicates no item in table)
- Insert random number
  - Check the correspond value is not 1
  - Set corresponding value to 1
- Collision: $h[pos]$ is already 1
  - occupied, can't insert here
  - probe the next slot ($pos + 1$)
  - If another collision occurs, continue probing forward
    - Linear: $+1, +2, +3, \ldots$
    - Quadratic: $+1, +4, +9, \ldots$
    - Cubic: $+1, +8, +27, \ldots$
  - Always remember "%"