CS610 Semester 2024–2025-I
Assignment-5

Sandeep Nitharwal
(210921)

November 19, 2024

Compilation instruction and System information

Provided a make file for compilation.
$--> $ make
All the below result are generated on csews1 system

Problem.1

Comparison of different hash functions:
map_size = Maximum size of hash table.
It doubles after every resize operation.

LARGE_PRIME = 1e9 + 7

Hash Function 1

$$\text{hash}_1(k) = \langle k \bmod \text{map\_size}, (\text{LARGE\_PRIME} - (k) \bmod \text{LARGE\_PRIME}) \bmod \text{map\_size} \rangle \tag{1}$$

Hash Function 2

$$\text{hash}_2(k) = \big\langle k \bmod \text{map\_size}, \big((k^2 \bmod \text{LARGE\_PRIME}) + \text{LARGE\_PRIME}\big) \bmod \text{map\_size} \big\rangle \tag{2}$$

Hash Function 3 - Universal hash

$$\text{hash}_3(k, v_1, v_2) = \langle k \bmod \text{map\_size}, ((v_1 \cdot k + v_2) \bmod \text{LARGE\_PRIME} + \text{LARGE\_PRIME}) \bmod \text{map\_size} \rangle \tag{3}$$

Hash Function 4

$$\text{hash}_4(k, v_1, v_2) = \big\langle k \bmod \text{map\_size}, \big((v_1^2 \bmod \text{LARGE\_PRIME}) \cdot k + v_2\big) \big\rangle \tag{4}$$

Hash3 is the fastest.

Unit Test cases

I have also provided 3 unit test cases.

```
nsandeep@csews1:~/assign5$ ./problem1.out
NUM OPS: 1000 ADD: 600 REM: 300 FIND: 100
---------- hashTable -------------
---------- Testcases -------------
TEST 1: PASSED
TEST 2: PASSED
TEST 3: PASSED
---------- Time details -------------
---------- Time details for hash1 -------------
Time taken by insert kernel (ms): 6979
Time taken by delete kernel (ms): 0
Time taken by search kernel (ms): 0

---------- Time details for hash2 -------------
Time taken by insert kernel (ms): 6739.5
Time taken by delete kernel (ms): 0
Time taken by search kernel (ms): 0

---------- Time details for hash3 -------------
Time taken by insert kernel (ms): 3765
Time taken by delete kernel (ms): 0
Time taken by search kernel (ms): 0

---------- Time details for hash4 -------------
Time taken by insert kernel (ms): 5247
Time taken by delete kernel (ms): 0
Time taken by search kernel (ms): 0
```

Comparison with Intel TBB implementation

My implementation performed more poorly (almost 4 times slower) than I expected as compared to the Intel TBB hash table implementation.

Problem.2

Below are the results.
For 1e5

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 100000 ADD: 60000 REM: 40000
Time taken by(ms): 3 Threads : 1
```

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 100000 ADD: 60000 REM: 40000
Time taken by(ms): 6 Threads : 2
```

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 100000 ADD: 60000 REM: 40000
Time taken by(ms): 9 Threads : 4
```

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 100000 ADD: 60000 REM: 40000
Time taken by(ms): 9 Threads : 8
```

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 100000 ADD: 60000 REM: 40000
Time taken by(ms): 9 Threads : 16
```

For 1e6

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 1000000 ADD: 600000 REM: 400000
Time taken by(ms): 31 Threads : 1
```

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 1000000 ADD: 600000 REM: 400000
Time taken by(ms): 69.5 Threads : 2
```

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 1000000 ADD: 600000 REM: 400000
Time taken by(ms): 90 Threads : 4
```

```
nsandeep@csews1:~/assign5$ ./problem2.out
NUM OPS: 1000000 ADD: 600000 REM: 400000
Time taken by(ms): 99 Threads : 8
```