CSci 242: Algorithms and Data Structures **Fall, 2019**

Instructor: Dr. M. E. Kim Date: October 17th, 2019

Due: by the end of day, October 25th (Fri.), 2019.

**Home Assignment 4: 150 points + 20 (optional) 72/150**

Consider the keys, 12, 44, 13, 88, 23, 94, 11, 39, 20, 16 and 5. Insert them in a hash table of size 11 with the hash function, *h*(*k*) = (2*k* +5) mod 11. Collision is handled by each of the method in Q1 – Q4. Show the final content of hash table after insertion.

For each question, you have to show the proper computational steps.

Q1. [10] Collision handled by ***chaining***. Draw the final hash table after insertion.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 20 |  |  | 16 | 44 | 94 | 12 |  | 13 |  |
|  |  |  |  | 5 | 88 | 39 | 23 |  |  |  |
|  |  |  |  |  | 11 |  |  |  |  |  |

Q2. [10] Collision handled by open addressing with the linear probing.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 39 | 20 | 5 | 16 | 44 | 88 | 12 | 23 | 13 | 94 |

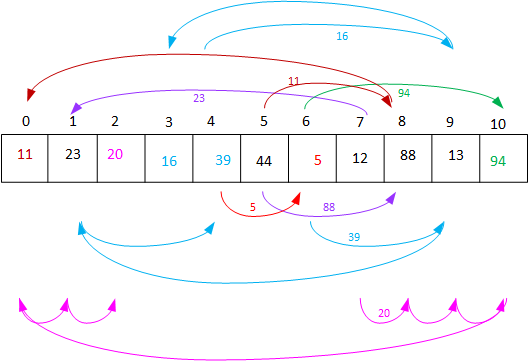
Q3. [10] Collision handled by open addressing with the quadratic probing where the auxiliary function is *f*(*j*) = *j*2. Show the final hash table up to the point where the method fails because no empty slot is found.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 20 | 16 | 11 | 39 | 44 | 88 | 12 | 23 | 13 | 94 |

Q4. [8] Collision handled by open addressing with the double hashing using a secondary hash function d(*k*) = 7 – (*k* mod 7).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 23 | 20 | 88 | 39 | 16 | 44 | 94 | 12 | 5 | 13 | 11 |

Correct answer is

**

Q5. [9] Collision handled by Cuckoo hashing using h1(k) = (2*k* +5) mod 11 and h2(k) = (3*k*+1) mod 11 the hash tables T1 and T2, respectively. **12, 44, 13, 88, 23, 94, 11, 39, 20, 16 and 5**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| T1 |  |  |  |  |  | 44 |  | 12 |  | 13 |  |
| T2 |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 12 | 44 | 13 | 88 | 23 | 94 | 11 | 39 | 20 | 16 | 5 |
| T1 | 7 | 5 | 9 | 5 | 7 | 6 | 6 | 6 | 1 | 4 | 4 |
| T2 | 4 | 1 | 7 | 1 | 4 | 8 | 1 | 8 | 6 | 5 | 5 |

* h1(12) = *i* = (2\*12+5)mod 11 = (29 mod 11) = 7
* h1(44) = *i* = (2\*44+5)mod 11 = (93 mod 11) = 5
* h1(13) = *i* = (2\*13+5)mod 11 = (31 mod 11) = 9
* h1(88) = *i* = (2\*88+5)mod 11 = (181 mod 11) = 5 -- collision with 44 🡪 evict 44 to T2.
* h2(44) = (3\*44+1) mod 11 = 1
* h1(23) = *i* = (2\*23+5)mod 11 = (51 mod 11) = 7 -- collision with 12 🡪 evict 12 to T2
* h2(12) = (3\*12+1) mod 11 = 4
* h1(94) = *i* = (2\*94+5)mod 11 = (193 mod 11) = 6
* h1(11) = *i* = (2\*11+5)mod 11 = (26 mod 11) = 5 -- collision with 88 🡪 evict 88 to T2
* h2(88) = (3\*88+1) mod 11 = 1 -- collision with 44 at T2 🡪 evict 44 to T1
* h1(44) = 5 🡪 cycle occurs. 11 🡪 88 🡪 44 🡪 11
* Stop hashing. The hash functions need to be redefined.

Q6. [25] Implementation of Q1 – Q5

Implement the insertion algorithms with the given keys in Q1 – Q5 for each collision handling scheme.

Print the final content of hash table: e.g.) (slot #, (a list of) key)

**Implementation missing**

Q7. [0] Implementation of removal of key.

From the final heap in Q2, remove each key in the order of its insertion, i.e. 12, 44, 13, ….

After each removal of a key, print the content of the hash table after filing the empty slots:

e.g.) After removal of 12: A table of (slot #, key)

Q8. [0]

In the hashing technique with linear probing, the ***remove*(*k*)** method is designed with ***Shift(i)*** method to fill the holes after the removal of a key in the slot *i*. In the hashing with quadratic probing, how would you handle the such a problem caused by the removal of a key? Explain your idea and design its algorithm. Suppose the auxiliary function is *f*(*j*) = *j*2 is used for the quadratic probing.

Q9 [0]

Implement your algorithm in Q8 by removing each from the final heap of Q3. Print the hash table after each removal.