CSci 242: Algorithms and Data Structures Spring, 2020

Instructor: Dr. M. E. Kim Date: March 4th, 2020

Due: by the end of day, March 13th (Fri.), 2020.

Read the submission instructions carefully and comply with them.

**Home Assignment 4: 91/150 points + 20 (optional)**

Consider the keys, 3, 14, 18, 37, 9, 92, 21, 86, 11, 42, and 10. 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. [10/10] Collision handled by ***chaining***. Draw the final hash table after insertion.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 3 | 9 | 37 | 21 |  | 11 |  |  | 18 |  |  |
| 14 | 86 | 92 | 10 |  |  |  |  |  |  |  |
|  | 42 |  |  |  |  |  |  |  |  |  |

2(3) + 5 mod 11 = 0

2(14) + 5 mod 11= 0

2(18) + 5 mod 11 = 1

2(37) + 5 mod 11= 8

2(9)+ 5 mod 11 = 2

2(92) + 5 mod 11= 1

2(21) + 5 mod 11= 2

2(86)+ 5 mod 11 = 3

2(11) + 5 mod 11= 1

2(42) + 5 mod 11 = 5

2(10) + 5 mod 11 =1

Q2. [10/10] Collision handled by open addressing with the linear probing where probe function *f*(*j*) = *j.*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 3 | 14 | 37 | 9 | 92 | 21 | 86 | 11 | 18 | 42 | 10 |

2(3) + 5 mod 11 = 0

2(14) + 5 mod 11= 0

2(14) + 5 + 1 mod 11 = 1

2(18) + 5 mod 11 = 8

2(37) + 5 mod 11= 2

2(9)+ 5 mod 11 = 1

2(14) + 5 + 1 mod 11 = 2

2 (14) + 5 + 2 mod 11 = 3

2(92) + 5 mod 11= 2

2(92) + 5 + 1 mod 11 = 3

2(92) + 5 + 2 mod 11 = 4

2(21) + 5 mod 11= 3

2(21) + 5 + 1 mod 11 = 4

2(21) + 5+ 2 mod 11 = 5

2(86)+ 5 mod 11 = 1

2(86) + 5 + 1 mod 11= 2

2(86) + 5 + 2 mod 11= 3

2(86) + 5 + 3 mod 11= 4

2(86) + 5 + 4 mod 11= 5

2(86) + 5 + 5 mod 11= 6

2(11) + 5 mod 11= 5

2(11) + 5 + 1 mod 11 = 6

2(11) + 5 + 2 mod 11 = 7

2(42) + 5 mod 11 = 1

2(42) + 5 + 1 mod 11= 2

2(42) + 5 + 2 mod 11= 3

2(42) + 5 + 3 mod 11= 4

2(42) + 5 + 4 mod 11= 5

2(42) + 5 + 5 mod 11= 6

2(42) + 5 + 6 mod 11 = 7

2(42) + 5 + 7 mod 11 = 8

2(10) + 5 mod 11 = 3

2(10) + 5 + 1 mod 11= 4

2(10) + 5 + 2 mod 11= 5

2(10) + 5 + 3 mod 11= 6

2(10) + 5 + 4 mod 11= 7

2(10) + 5 + 5 mod 11= 8

2(10) + 5 + 6 mod 11 = 9

2(10) + 5 + 7 mod 11 = 10

Q3. [10/10] Collision handled by open addressing with the quadratic probing where the probe function *f*(*j*) = *j*2 + *j* 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 |
| 3 | 9 | 14 | 92 | 37 | 21 | 11 | 86 | 18 | 10 | 42 |

2(3) + 5 mod 11 = 0

2(14) + 5 mod 11= 0

33 + 2 mod 11 = 2

2(18) + 5 mod 11 = 8

2(37) + 5 mod 11= 2

79 + 2 mod 11 = 4

2(9)+ 5 mod 11 = 1

2(92) + 5 mod 11 = 2

189 + 2 mod 11 = 4

189 + 6 mod 11 = 8

189 + 12 mod 11 = 3

2(21) + 5 mod 11= 3

47 + 2 mod 11 = 5

2(86)+ 5 mod 11 = 1

177 + 2 mod 11 = 3

177 + mod 11 = 7

2(11) + 5 mod 11= 5

27 + 2 mod 11 = 7

27 + 6 mod 11 = 0

27 + 12 mod 11 = 6

2(42) + 5 mod 11 = 1

89 + 2 mod 11 = 3

89 + 6 mod 11 = 7

89 + 12 mod 11 = 2

89 + 20 mod 11 = 10

2(10) + 5 mod 11 = 3

25+ 2 mod 11 = 5

25+ 6 mod 11 = 9

Couldn’t add 10 or 42 because the modulo never lead to an open spot

Q4. [5/10] 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 |
| 3 |  |  |  |  |  |  | 14 |  |  |  |

2(3) + 5 mod 11 = 0

2(14) + 5 mod 11= 0

2(18) + 5 mod 11 = 8

2(37) + 5 mod 11= 2

2(9)+ 5 mod 11 = 1

2(92) + 5 mod 11 = 2

2(21) + 5 mod 11= 3

2(86)+ 5 mod 11 = 1

2(11) + 5 mod 11= 5

2(42) + 5 mod 11 = 1

2(10) + 5 mod 11 = 3

7 – ( 3 mod 7 ) = 4

7 – 14 mod 7 = 7

7 – 18 mod 7 = 3

7 – 37 mod 7 = 5

7 – 9 mod 7 = 5

7 – 92 mod 7 = 6

7 – 21 mod 7 = 7

7 – 86 mod 7 = 5

7 – 11 mod 7 = 3

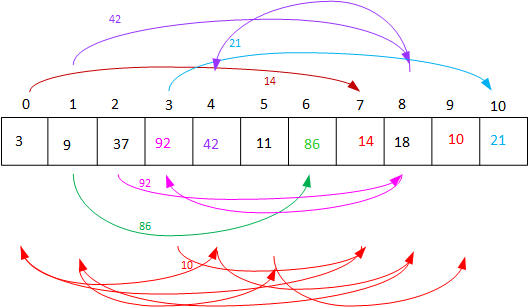
7 – 42mod 7 = 7

7 – 10 mod 7 = 4

Solution for the rest of the values except 3 and 14:

*h*(*k, j*) = (*i* + *j*⋅*d*(*k*)) mod 11 where *i* = *h*(*k*).

* h(18) = *i* = (2\*18+5)mod 11 = 8
* h(37) = *i* = (2\*37+5)mod 11 = 2
* h1(9) = *i* = (2\*9+5)mod 11 = 1
* h1(92) = *i* = (2\*92+5)mod 11 = 2 -- collision with 37
  + *d(92) = 7 – (92 mod 7) =* 7-1=6
  + *j*=1: h(i, f(j)) = (2 + 1⋅6) mod 11 = 8 -- collision with 18
  + *j=2:* h(i, f(j)) = (2 + 2⋅6) mod 11 = 3
* h1(21) = *i* = (2\*21+5)mod 11 = 3 -- collision with 92
  + *d(21) = 7 – (21 mod 7) =* 7-0=7
  + *j*=1: h(i, f(j)) = (3 + 1⋅7) mod 11 = 10
* h1(86) = *i* = (2\*86+5)mod 11 = 1 -- collision with 9
  + *d(86) = 7 – (86 mod 7) =* 7-2=5
  + *j*=1: h(i, f(j)) = (1 + 1⋅5) mod 11 = 6
* h1(11) = *i* = (2\*11+5)mod 11 = 5
* h1(42) = *i* = (2\*42+5)mod 11 = 1 -- collision with 9
  + *d(42) = 7 – (42 mod 7) =* 7- 0=7
  + *j*=1: h(i, f(j)) = (1 + 1⋅7) mod 11 = 8 -- collision with 18
  + *j*=2: h(i, f(j)) = (1 + 2⋅7) mod 11 = 4
* h1(10) = *i* = (2\*10+5)mod 11 = 3 -- collision with 92
  + *d(10) = 7 – (10 mod 7) =* 7-3=4
  + *j*=1: h(i, f(j)) = (3 + 1⋅4) mod 11 = 7 -- collision with 14
  + *j*=2: h(i, f(j)) = (3 + 2⋅4) mod 11 = 0 -- collision with 3
  + *j*=3: h(i, f(j)) = (3 + 3⋅4) mod 11 = 4 -- collision with 42
  + *j*=4: h(i, f(j)) = (3 + 4⋅4) mod 11 = 8 -- collision with 18
  + *j*=5: h(i, f(j)) = (3 + 5⋅4) mod 11 = 1 -- collision with 9
  + *j*=6: h(i, f(j)) = (3 + 6⋅4) mod 11 = 5 -- collision with 11
  + *j*=7: h(i, f(j)) = (3 + 7⋅4) mod 11 = 9



Q5. [9/10] 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.

T1 :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 14 | 86 | ~~37~~ 92 | 10 |  | 11 |  |  | 18 |  |  |

Infinity loop between 86 – 9 – 14

T2 :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  |  | ~~92~~ 37 |  |  |  | 9 |  |  | 21 | ~~10~~ 3 |

H1 =

2(3) + 5 mod 11 = 0

2(14) + 5 mod 11= 0

2(18) + 5 mod 11 = 8

2(37) + 5 mod 11= 2

2(9)+ 5 mod 11 = 1

2(92) + 5 mod 11 = 2

2(21) + 5 mod 11= 3

2(86)+ 5 mod 11 = 1

2(11) + 5 mod 11= 5

2(42) + 5 mod 11 = 1

2(10) + 5 mod 11 = 3

H2 =

3(3) + 1 mod 11 = 10 # a small typo in table

3(14) + 1 mod 11 = 9

3(18) + 1 mod 11 = 0

3(37) + 1 mod 11 = 2

3(9) + 1 mod 11 = 6

3(92) + 1 mod 11 = 2

3(21) + 1 mod 11 = 9

3(86) + 1 mod 11 = 6

3(11) + 1 mod 11 = 1

3(42) + 1 mod 11 = 6

3(10) + 1 mod 11 = 9

Q6. [7+15+0+15+5/15 \* 5 = 75] Implementation of Q1 – Q5

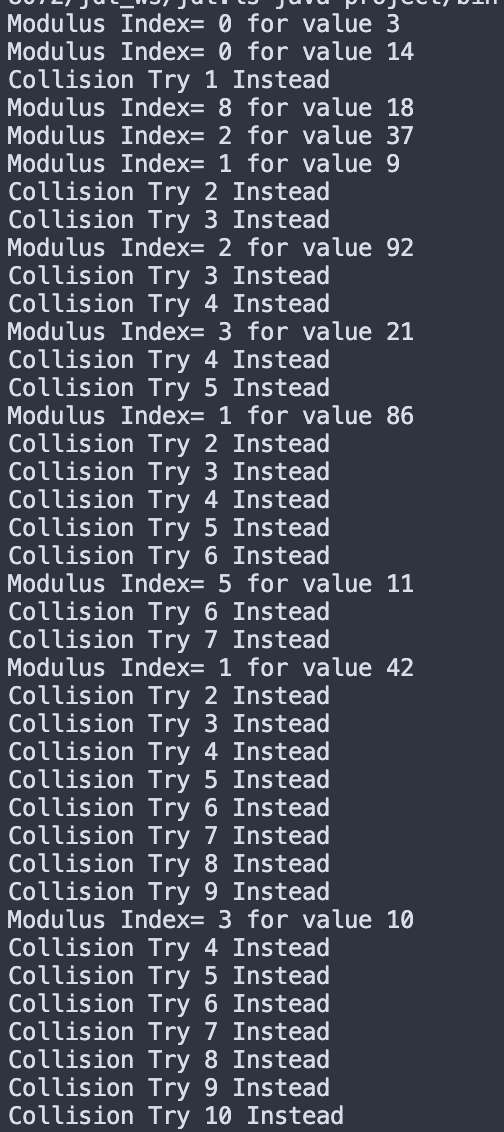
Implement the insertion algorithms with the given keys in Q1 – Q5 for each collision handling scheme.

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

Chaining :

No printout s during insertion or after insertion so I cannot check the work properly, giving half marks (7/15) by inspecting code.

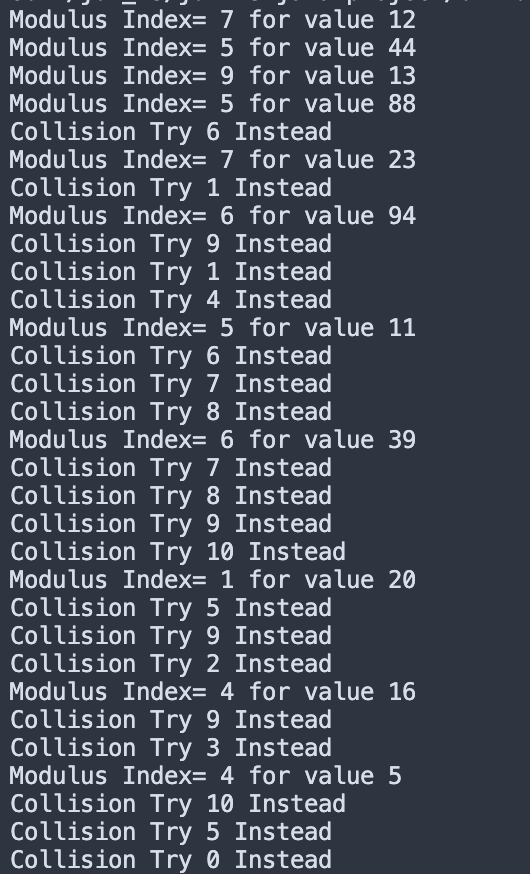
Linear :



Quadratic Hashing :

No code implementation found. (0/15)

Double Hashing :



Cuckoo Hashing :

No printouts during insertion or after insertion so I cannot check the work properly, giving 5 marks for attempt only since the code also does not look correct.

Q7. [0/25] Implementation of removal of key.

From the final heap in Q2, remove each key in the order of its insertion, i.e. 3, 14, 18, ….

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

e.g.) After removal of 3: A table of (slot #, key)

Q8. [5/10, optional]

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 probe function is *f*(*j*) = *j*2 is used for the quadratic probing.

**I would rehash the table using the original function and any clashes would use the auxiliary function.**

**This would allow the table to be shifted after the removal of a key.**

**Algorithm rehash**

**Input: array, k**

**Output: index**

**index = 2k + 5 mod 11**

**While array != null do**

**Index = (2k + 5 + k2 ) mod 11**

**Output index**

Q9 [0/10, optional]

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

**I hit an infinite loop and have been unable to figure out how to force it into completion, thus is why I did not include a screenshot of it.**