CSci 242: Algorithms and Data Structures Spring, 2022

Instructor: Dr. M. E. Kim Date: February 24th, 2022

Please read the submission instructions carefully and comply with them.

**Home Assignment 5A: 90 points + 10 (optional)**

Due: by the end of the day, March 7th (Mon.), 2022.

Q1. [30] A hashing function *h* for a string key sums string ASCII character codes to make the hash code.

where *k* = with the characters s.

1. [15] Using the given ASCII table, compute a hash code for the following keys.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| character | ASCII value | character | ASCII value | character | ASCII value | character | ASCII value |
| A | 65 | H | 72 | O | 79 | V | 86 |
| B | 66 | I | 73 | P | 80 | W | 87 |
| C | 67 | J | 74 | Q | 81 | X | 88 |
| D | 68 | K | 75 | R | 82 | Y | 89 |
| E | 69 | L | 76 | S | 83 | Z | 90 |
| F | 70 | M | 77 | T | 84 |  |  |
| G | 71 | N | 78 | U | 85 |  |  |

1. [5] PEACE
   1. 80, 69, 65, 67, 69
   2. h(k) = 350
2. [5] VICTORY
   1. 86, 73, 67, 84, 79, 82, 89
   2. h(k) = 560
3. [5] WISDOM
   1. 87, 73, 83, 68, 79, 77
   2. h(k) = 467
4. [15] Using a hash function *h*(*k*) = *k* mod 19, compute a hash value for the above keys in (A) – (C).
   1. PEACE
      1. P = 4
      2. E = 12
      3. A = 8
      4. C = 10
      5. E = 12
      6. 350%19 = 8
   2. VICTORY
      1. V = 10
      2. I = 16
      3. C = 10
      4. T = 8
      5. O = 3
      6. R = 6
      7. Y = 13
      8. 560%19 = 9
   3. WISDOM
      1. W = 11
      2. I = 16
      3. S = 7
      4. D = 11
      5. O = 3
      6. M = 1
      7. 467%19 = 11

Q2 – Q6. Consider the keys, 3, 14, 18, 37, 9, 92, 21, 86, and 11. 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 methods below.

For each insertion of a key, give the initial hash value, a sequence of probed values, and the final hash value: e.g.) *k* = 10, h(10) = 3 🡪 h(10, 1) = 4 🡪 h(10, 2) = 5. Show the final hash table A[0 .. 10].

You have to show the essential computation steps.

Q2. [10] Collision is handled by ***chaining***. Draw the final hash table after insertion.

(2(3) + 5)%11 = 0 \* same math was applied to each number

h(k) = 0, 0, 8, 2, 1, 2, 3, 1, 0

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

Q3. [20] Collision handled by open addressing with the linear probing where probe function *f*(*j*) = *j.*

1. [10] Insert the given keys and show the result as it’s described above.
   1. Initial hashes h(k) = 0, 0, 8, 2, 1, 2, 3, 1, 0

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

h(3) = 0

h(14) = 0 → h(14, 1) = 1

h(18) = 8

h(37) = 2

h(9) = 1 → h(9, 1) = 2 → h(9, 2) = 3

h(92) = 2 → h(92, 1) = 3 → h(92, 2) = 4

h(21) = 3 → h(21, 1) = 4 → h(21, 2) = 5

h(86) = 1 → h(86, 1) = 2 → h(86, 2) = 3 → h(86, 3) = 4 → h(86, 4) = 5, h(87, 5) = 6

h(11) = 0 → h(11, 1) = 1 → h(11, 2) = 2 → h(11, 3) = 3 → h(11, 4) = 4, h(11, 5) = 5 → h(11,6) = 6 → h(11, 7) = 7

1. [10] Remove a key 14 from the final table in (A). Show the sequence of shifting and the final hash table A[0 .. 10].

e.g.) *k* = 14: h(14) = 0, A[0] = 14 ← A[3] = 18 ← A[4] = 21.

14 is removed from A[0], 18 is shifted from A[3] to A[0], and 21 is shifted from A[4] to A[3].

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

remove k = 14  
h(14) ← null  
h(37) ← A[1]  
h(9) ← A[2]  
h(92) ← A[3]  
h(21) ← A[4]  
h(86) ← A[5]  
h(11) ← A[6]  
h(18) ← A[7]

Q4. [10] Collision is handled by open addressing with the quadratic probing where the probe 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.

Keys = 3, 14, 18, 37, 9, 92, 21, 86, 11

Initial hashes h(k) = 0, 0, 8, 2, 1, 2, 3, 1, 0

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

h(2(3) + 5) mod 11 = 0 + 0^2 = A[0]  
h(2(14) + 5) mod 11 = 0 + 0^2 = A[0] → h(2(14) + 5) mod 11 = 0 + 1^2 = A[1]  
h(2(18) + 5) mod 11 = 8 + 0^2 = A[8]  
h(2(37) + 5) mod 11 = 2 + 0^2 = A[2]  
h(2(9) + 5) mod 11 = 1 + 0^2 = A[1] → h(2(9) + 5) mod 11 = 1 + 1^2 = A[2] → h(2(9) + 5) mod 11 = 1 + 2^2 = A[5]  
h(2(92) + 5) mod 11 = 2 + 0^2 = A[2] → h(2(92) + 5) mod 11 = 2 + 1^2 = A[3]  
h(2(21) + 5) mod 11 = 3 + 0^2 = A[3] → h(2(21) + 5) mod 11 = 3 + 1^2 = A[4]  
h(2(86) + 5) mod 11 = 1 + 0^2 = A[1] → h(2(86) + 5) mod 11 = 1 + 1^2 = A[2] → h(2(86) + 5) mod 11 = 1 + 2^2 = A[5] → out of table range

Q5. [10] Collision is handled by open addressing with the double hashing using a secondary hash function d(*k*) = 7 – (*k* mod 7).

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

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

Q6. [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. If the same pattern of eviction of keys is repeated 2nd time, declare ‘cycle occurs’ and stop.

|  |  |  |
| --- | --- | --- |
| K | Hash 1 | Hash 2 |
| 3 | 0 | 10 |
| 14 | 0 | 10 |
| 18 | 8 | 0 |
| 37 | 2 | 2 |
| 9 | 1 | 6 |
| 92 | 2 | 2 |
| 21 | 3 | 9 |
| 86 | 1 | 6 |
| 11 | 0 | 1 |

|  |  |  |
| --- | --- | --- |
|  | T1 | T2 |
| ***0*** | 3 |  |
| ***1*** | 9 | 11 |
| ***2*** | 37 | 92 |
| ***3*** | 21 |  |
| ***4*** |  |  |
| ***5*** |  |  |
| ***6*** |  | 86 |
| ***7*** |  |  |
| ***8*** | 18 |  |
| ***9*** |  |  |
| ***10*** |  | 14 |

Q7. [10, optional]

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

**Home Assignment 5B: 100 points + 20 (optional)**

Due: by the end of the day, March 12th (Sat.), 2022.

Q8. [15 \* 5 = 75] Implementation of Q2 – Q6

Implement the insertion algorithms with the given keys in Q2 – Q6 for each collision handling scheme.

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

Q9. [25] Implementation of removal of a key.

From the *final hash table* in Q3.1 *with the linear probing*, remove each key in the order of its insertion, i.e., 3, 14, 18, …., etc.

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 hash table A of (slot #, key)

Q10 [20, optional]

Implement your algorithm in Q7. Remove each key in the order of its insertion from the final hash table of Q4. After each key removal, print the hash table after filling the empty slots.