CS200 Functional Data Structures
Assignment 2

Beep Beep: Anas-am00634 and Affan-sa00310

March 14, 2016

Problem 1. Hash Table

Please see the ass3.hs file in the same folder.

Problem 2. Hashing

1. Convert the input to radix-128 (for convenience) to compute k (if it is not ASCII already) h(x) = (k radix 128 (if not int)) mod m
   Where m is the size of the table which is chosen at rehash such that m is the next prime to 2^n where n is number of elements. This function fulfills the requirements as

   - Collisions are reduced because the function is cyclic and choosing m to be prime, we reduce the weight of getting even numbers to odd (which happens by choosing an even size)
   - Easy to compute because the most trivial algorithm for finding next prime (sieveo of Eratosthenes) is $O(nlog_n(loglog_n))$ and the rest is simple math.

   Our keys, no matter what data-type they belong to, are natural numbers (positive integers). So for all other data types, we convert them to integers and use the algorithm above.

2. If n is a character, take its ASCII including the quotations.

3. If n is a float, take its mantissa, and take the length of the mantissa minus 2, and store it in the variable x. product (10x, n) will be the ultimate integer.

4. If n is a string, loop through all characters and convert each of them to their integer and then radix-128 int representation including double quotations.

5. Treat false as an arbitrary large prime number, and treat true as another arbitrarily large prime number.

6. Pointer to n will pass the memory address of n, which can be converted to integer by a simple hexa to decimal conversion. Size may vary.

7. There are two ways that can be used to convert tuple to integer. Either take each element, convert them to integer, if they are integers, and add them while taking the radix. Or take the ASCII of the each character including parenthesis and comma. We prefer the first one for our implementation.

Problem 3. Insertion

1. Chaining

   Initially, we begin by choosing dimension 0, making the size of our table 1 (1 is prime).:

0	[]

- Then we Insert 1. 1mod1 = 0 therefore,

0	[(1,'Alpha')]

- Insert True. We increase our dimension to 1, which makes our table size 3 (3 is the first prime after 2). 1mod3 = 1 and we place 5 there. For this example, we take True to be 761. 761 mod 3 = 0 therefore,

0	[(True,5)]
1	[(1,'Alpha')]
2	[]

- Insert 'L'. ASCII value of L is 76 and 76mod3 = 1, therefore

0	[(True,5)]
1	[(1,'Alpha'),('L','Boss')]
2	[]

- Insert 3.14. We need to increase our dimension, and therefore, we resize to the next prime which becomes 5 since next prime after 2^2 is 5. We insert all other elements similarly. 1mod5 = 1, 761mod5 = 1, 76mod5 = 1, Mantissa is 2 digits behind, therefore we multiply it by 10^2, and get 314. 314mod5 = 4, therefore we get,

0	[]
1	[(1,'Alpha'),(True,5),('L','Boss')]
2	[]
3	[]
4	[(3.14,'pi')]

- Insert 'Habib'. We follow the compounding string algorithm. Radix 128 means $ASCII'H' \times 1 + ASCII'a' \times 128 + ASCII'b' \times 128^2$... This means $72 + 97 \times 128 + 98 \times 128^2$... The answer comes to be 26528493768 which mod5 = 3. Therefore,

0	[]
1	[(1,'Alpha'),(True,5),('L','Boss')]
2	[]
3	[('Habib','University')]
4	[(3.14,'pi')]

- Before next insert, we see that the number of elements == size of our table. Therefore we resize. 2^3 =8, and the next prime after that is 11. All calculations are done again. 1mod11 is 1, 761 mod 11 is 2, 76mod11 is 6, 314mod11 is 6, 26528493768mod11 is 7and therefore we get the table;

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[]
4	[]
5	[]
6	[('L','Boss'), (3.14,pi)]
7	[('Habib','University')]
8	[]
9	[]
10	[]

- Insert 'pie'. Following the same radix 128 strategy, ASCII value$\times 128^i$, we get $112 \times 1 + 105 \times 128 + 101 \times 128^2$ which comes out to be 1668336 which mod11 = 10

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[]
4	[]
5	[]
6	[('L','Boss'), (3.14,pi)]
7	[('Habib','University')]
8	[]
9	[]
10	[('pie','delicious')]

- Insert 'meaning'. Following the same radix 128 strategy, we get $109 + 101 \times 128 + 97 \times 128^2$... which comes out to be a very large number. We use the additive modulo identity ($amodc + bmodc \equiv (a+b)modc$) strategy and modulo 11 to get index 5. Therefore –

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[]
4	[]
5	[('meaning',42)]
6	[('L','Boss'), (3.14,pi)]
7	[('Habib','University')]
8	[]
9	[]
10	[('pie','delicious')]

- insert (1,'one'). According to our strategy, we should have 1 + radix128 'one' which becomes $1 + 111 + 110 \times 128 + 101 * 128^2$ which comes out to be 1668976, which modulo11 = 1.

0	[]
1	[(1,'Alpha'),((1,'one'),False)]
2	[(True,5)]
3	[]
4	[]
5	[('meaning',42)]
6	[('L','Boss'), (3.14,pi)]
7	[('Habib','University')]
8	[]
9	[]
10	[('pie','delicious')]

- Insert 'One True Editor'. We follow the same radix 128 strategy but it would become too large, so we adopt the same additive property to get $79 mod 11 + 110 * 128 mod 11 + 101 * 128^2 mod 11$... this gives us the answer 4. Therefore,

0	[]
1	[(1,'Alpha'),((1,'one'),False)]
2	[(True,5)]
3	[]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss'), (3.14,pi)]
7	[('Habib','University')]
8	[]
9	[]
10	[('pie','delicious')]

2. Linear Probing

- Initially, we begin by choosing dimension 0, making the size of our table 1 (1 is prime).:

0	[]

- Then we Insert 1. 1mod1 = 0 therefore,

0	[(1,'Alpha')]

- Insert True. We increase our dimension to 1, which makes our table size 3 (3 is the first prime after 2). 1mod3 = 1 and we place 5 there. For this example, we take True to be 761. 761 mod 3 = 0 therefore,

0	[(True,5)]
1	[(1,'Alpha')]
2	[]

- Insert 'L'. ASCII value of L is 76 and 76mod3 = 1. Since 1 is occupied, we place it in next value. Therefore

0	[(True,5)]
1	[(1,'Alpha')]
2	[('L','Boss')]

- Insert 3.14. We need to increase our dimension, and therefore, we resize to the next prime which becomes 5 since next prime after 2^2 is 5. We insert all other elements similarly. 1mod5 = 1, 761mod5 = 1, 76mod5 = 1 (since there are collisions, they go to next index respectively.) Mantissa is 2 digits behind, therefore we multiply it by 10^2, and get 314. 314mod5 = 4, and that is not a collision, therefore we get,

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[('L','Boss')]
4	[(3.14,'pi')]

- Insert 'Habib'. We follow the compounding string algorithm. Radix 128 means $ASCII'H' \times 1 + ASCII'a' \times 128 + ASCII'b' \times 128^2$... This means $72 + 97 \times 128 + 98 \times 128^2$... The answer comes to be 26528493768 which mod5 = 3. Since there are collisions, the only empty space is at index 0. Therefore,

0	[('Habib','University')]
1	[(1,'Alpha')]
2	[(True,5)]
3	[('L','Boss')]
4	[(3.14,'pi')]

- Before next insert, we see that the number of elements == size of our table. Therefore we resize. 2^3 =8, and the next prime after that is 11. All calculations are done again. 1mod11 is 1, 761 mod 11 is 2, 76mod11 is 6, 314mod11 is 6 (there is a collision), 26528493768mod11 is 7 (which is already occupied)and therefore we get the table;

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[]
4	[]
5	[]
6	[('L','Boss')]
7	[(3.14,'pi')]
8	[('Habib','University')]
9	[]
10	[]

- Insert 'pie'. Following the same radix 128 strategy, ASCII value$\times 128^i$, we get $112\times 1+105\times 128+101\times 128^2$ which comes out to be 1668336 which mod11 = 10

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[]
4	[]
5	[]
6	[('L','Boss')]
7	[(3.14,'pi')]
8	[('Habib','University')]
9	[]
10	[('pie','delicious')]

- Insert 'meaning'. Following the same radix 128 strategy, we get $109+101\times 128+97\times 128^2$... which comes out to be a very large number. We use the additive modulo identity ($amodc+bmodc\equiv(a+b)modc$) strategy and modulo 11 to get index 5. Therefore

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[]
4	[]
5	[('meaning',42)]
6	[('L','Boss')]
7	[(3.14,'pi')]
8	[('Habib','University')]
9	[]
10	[('pie','delicious')]

- insert (1,'one'). According to our strategy, we should have 1 + radix128 'one' which becomes $1 + 111 + 110 \times 128 + 101 * 128^2$ which comes out to be 1668976, which modulo11 = 1, there is a collision, and the next unoccupied cell is 3.

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[((1,'one'),False)]
4	[]
5	[('meaning',42)]
6	[('L','Boss')]
7	[(3.14,'pi')]
8	[('Habib','University')]
9	[]
10	[('pie','delicious')]

- Insert 'One True Editor'. We follow the same radix 128 strategy but it would become too large, so we adopt the same additive property to get $79 mod 11 + 110 * 128 mod 11 + 101 * 128^2 mod 11$... this gives us the answer 4. Therefore,

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[((1,'one'),False)]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[(3.14,'pi')]
8	[('Habib','University')]
9	[]
10	[('pie','delicious')]

Problem 4. Deletion

1. Chaining. The table we had achieved was;

0	[]
1	[(1,'Alpha'),((1,'one'),False)]
2	[(True,5)]
3	[]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss'), (3.14,pi)]
7	[('Habib','University')]
8	[]
9	[]
10	[('pie','delicious')]

- Delete True. The random prime attributed to True would be generated, giving us 761. 761mod11 = 2 and since there is only 1 item in the list, we remove it.

0	[]
1	[(1,'Alpha'),((1,'one'),False)]
2	[]
3	[]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss'), (3.14,pi)]
7	[('Habib','University')]
8	[]
9	[]
10	[('pie','delicious')]

- Delete 3.14. We follow the same strategy and get 314mod11 which is 6 and we access the value. Since there are two different values, we check each value with the key and then remove the actual value.

0	[]
1	[(1,'Alpha'),((1,'one'),False)]
2	[]
3	[]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[('Habib','University')]
8	[]
9	[]
10	[('pie','delicious')]

- Delete 'pie'. Since the number of resulting items would be ¿ 2^2, we do not resize. Following same strategy,

0	[]
1	[(1,'Alpha'),((1,'one'),False)]
2	[]
3	[]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[('Habib','University')]
8	[]
9	[]
10	[]

- Delete (1,'one'). Number of items after remove would be 5, so we still do not resize.

0	[]
1	[(1,'Alpha')]
2	[]
3	[]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[('Habib','University')]
8	[]
9	[]
10	[]

- delete 1. The number of items after delete will be ≤ 4, so we resize to the next prime after 4 which is 5. The hashes of each element are re-calculated. 'Habib'(26528493768) mod5 = 3, 'One True Editor'mod5 = $(79 + 110 * 128 + 101 * 128^2 + 32^1 28^3 + ...)$ mod 5 = 0, 'L' mod 5 = 76mod5 = 1, 'meaning'mod5 = 109mod5+101*128mod5+... = 3.

0	[('One True Editor','Emacs')]
1	[('L','Boss')]
2	[]
3	[('Habib','University'), ('meaning',42)]
4	[]

2. Linear Probing; Assume [] contains nil. The table we had achieved was;

0	[]
1	[(1,'Alpha')]
2	[(True,5)]
3	[((1,'one'),False)]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[(3.14,'pi')]
8	[('Habib','University')]
9	[]
10	[('pie','delicious')]

- Delete True. Hash True to 761, 761mod11 =2, we find it there and delete it.

0	[]
1	[(1,'Alpha')]
2	[del]
3	[((1,'one'),False)]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[(3.14,'pi')]
8	[('Habib','University')]
9	[]
10	[('pie','delicious')]

- Delete 3.14. 314mod11 is 6. We do not find it there, so we go to the next value, where we find it and insert del instead of it.

0	[]
1	[(1,'Alpha')]
2	[del]
3	[((1,'one'),False)]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[del]
8	[('Habib','University')]
9	[]
10	[('pie','delicious')]

- Delete 'pie'. 'pie' mod 11 is 10, number of remaining elements would be greater than 4, therefore we do not resize.

0	[]
1	[(1,'Alpha')]
2	[del]
3	[((1,'one'),False)]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[del]
8	[('Habib','University')]
9	[]
10	[del]

- Delete (1,'one'). This mod 11 = 1 (see insert). We do not find our item there and we move next. Since next is del, we move next again. Then we find our item and replace it with del.

0	[]
1	[(1,'Alpha')]
2	[del]
3	[del]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[del]
8	[('Habib','University')]
9	[]
10	[del]

- Delete 1. 1mod11 ==1, and it will be simply removed.

0	[]
1	[del]
2	[del]
3	[del]
4	[('One True Editor','Emacs')]
5	[('meaning',42)]
6	[('L','Boss')]
7	[del]
8	[('Habib','University')]
9	[]
10	[del]

- Since the number of elements is $\leq 2^2$, we resize to the prime number after 4, which is 5. The hashes of each element are recalculated. 'Habib'(26528493768) mod5 = 3, 'One True Editor'mod5 = $(79 + 110 * 128 + 101 * 128^2 + 32^1 28^3 + ...)$ mod 5 = 0, 'L' mod 5 = 76mod5 = 1, 'meaning'mod5 = 109mod5+101*128mod5+... = 3 and this leads to a collision so we store it at the next empty node.

0	[('One True Editor','Emacs')]
1	[('L','Boss')]
2	[]
3	[('Habib','University')]
4	[('meaning',42)]