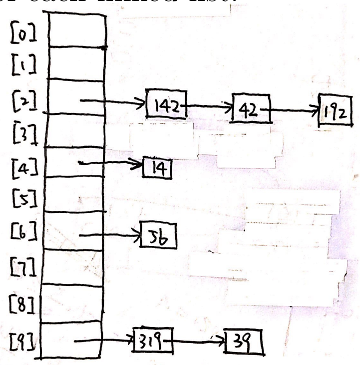
Lab04-Hashing

VE281 - Data Structures and Algorithms, Xiaofeng Gao, TA: Qingmin Liu, Autumn 2019

∗ Please upload your assignment to website. Contact webmaster for any questions.
∗ Name: Sun Yiwen   Student ID: 517370910213   Email: sunyw99@sjtu.edu.cn

1. Given a sequence of inputs 192, 42, 142, 56, 39, 319, 14, insert them into a hash table of size 10. Suppose that the hash function is $h(x) = x\%10$. Show the result for the following implementations:

   (a) Hash table using separate chaining. Assume that the insertion is always at the beginning of each linked list.



   (b) Hash table using linear probing.

319		192	42	142	14	56			39
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

   (c) Hash table using quadratic probing.

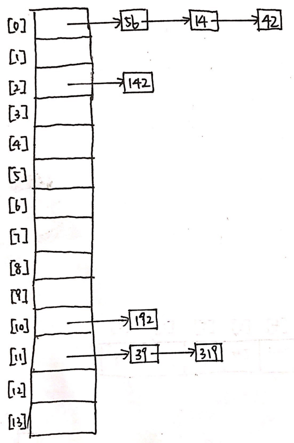
319		192	42	14		142	56		39
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

   (d) Hash table using double hashing, with the second hash function as $h_2(x) = (x + 4)\%7$.

56	319	192		14		42		142	39
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

2. Show the result of rehashing the four hash tables in the Problem 1. Rehash using a new table size of 14, and a new hash function $h(x) = x\%14$. (Hint: The order in rehashing depends on the order stored in the old hash table, not on their initial inserting order.)

   (a) Hash table using separate chaining.



(b) Hash table using linear probing.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
42	14	142	56							192	319	39	

(c) Hash table using quadratic probing.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
42	14	142		56						192	319	39	

(d) Hash table using double hashing, with the second hash function as $h_2(x) = (x + 4)\%7$.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
56		142		14				42		192	319	39	

3. Suppose we want to design a hash table containing at most 900 elements using linear probing. We require that an unsuccessful search needs no more than 8.5 compares and a successful search needs no more than 3 compares on average. Please determine a proper hash table size.

Solution.

$$U(L) = \frac{1}{2}\left[1 + \left(\frac{1}{1-L}\right)^2\right] \leq 8.5 \;\Rightarrow\; L \leq \frac{3}{4}$$

$$S(L) = \frac{1}{2}\left[1 + \frac{1}{1-L}\right] \leq 3 \;\Rightarrow\; L \leq \frac{4}{5}$$

$$\therefore L \leq \frac{3}{4}$$

$$L = \frac{|s|}{n} \leq \frac{3}{4} \;\Rightarrow\; n \geq \frac{4}{3} \cdot 900 = 1200$$

$\because$ We want to pick n as a prime number, $\therefore n = 1201$

□

4. Implement queues with two stacks. We know that stacks are first in last out (FILO) and queues are first in first out (FIFO). We can implement queues with two stacks. The method is as follows:

- For **enqueue** operation, push the element into stack S_1.
- For **dequeue** operation, there are two cases:
  - $S_2 = \emptyset$, pop all elements in S_1, push these elements into S_2, pop S_2
  - $S_2 \neq \emptyset$, pop S_2

Using amortized analysis to calculate the complexity of **enqueue** and **dequeue** step.

Solution.

Suppose popping an element, pushing an element and checking whether the stack is empty or not each has the complexity of $O(1)$.

Then, total cost of enqueueing n elements is $n \cdot O(1) = O(n)$. Average cost to enqueue an element is $O(1)$.

Suppose the initial state of the two stacks is $S1$ having n elements and $S2$ having no elements inside. Total cost of dequeueing the first n elements is calculated as $n \cdot O(1) + 2n \cdot O(1) + n \cdot O(1) = 4n \cdot O(1)$, $n \cdot O(1)$ for checking whether $S2$ is empty or not n times, $2n \cdot O(1)$ for popping the n elements out of $S1$ and pushing them into $S2$, $n \cdot O(1)$ for popping them out of $S2$ finally. For the $n+1$-th item, the state of $S1$ and $S2$ return to initial state. Therefore, total cost for dequeueing n items is $4n \cdot O(1) = 4O(n) = O(n)$. Average cost to dequeue an element is $O(1)$. $\square$