Solution for Problem Set 6

Mianzhi Pan, 181240045

October 27, 2020

1 Problem 1

First randomly choose a bucket from the m buckets. Suppose the length of the chosen bucket is k, we use $RANDOM(1, L)$ and suppose the return value is a. If $a \leq k$, return the a^{th} element in that bucket. Otherwise, we continue this process until $a \leq k$. By this way, each element is chosen with probability $\frac{1}{mK}$.

The probability that we succeed in chosing an element in a particular bucket is $\frac{k}{L}$, so the expected chosing times are $\frac{L}{k}$. Together with a times for retriving the element, total time is $O(a + \frac{L}{k}) = O(L \cdot (a/L + 1/k))$, so the excepted time is $O(L \cdot (1 + 1/\alpha))$(excepted value of k is α and $a/L \leq 1$).

2 Problem 2

(a) Suppose string x of length $l + 1$: $x_l x_{l-1} \cdots x_0$, x has key value $x_l \times (m + 1)^l + \cdots + x_0 \times (m + 1)^0$. We have

$$\begin{aligned} h(x) &= x\ mod\ m \\ &= ((x_l \times (m+1)^l)\ mod\ m + \cdots + (x_0 \times (m+1)^0)\ mod\ m)\ mod\ m \\ &= (x_l\ mod\ m + \cdots + x_0\ mod\ m)\ mod\ m \end{aligned}$$

We can find the hash value of a string is determined by all its characters but is independent with the order of them. Hence, x and y hash to the same value.

(b) Linear probing, $h(k, i) = (h'(k) + i)\ mod\ 11$, let $h'(k) = k$

22
88
4
15
28
17
59
31
10

Quadratic probing: $h(k,i) = (k + i + 3i^2)\ mod\ 11$

22
88
17
4
28
59
15
31
10

Double hashing: $h(k,i) = h_1(k) + ih_2(k)\ mod\ 11$

22
59
17
4
15
28
88
31
10

3 Problem 3

Let $B_i = \{w | h(w) = i\}$, which respesents the i^{th} bucket in B. We have

$$\begin{aligned}
Pr[h(k) = h(l)] &= \frac{\sum_{i=1}^{|B|} C_{|B_i|}^2}{C_{|U|}^2} \\
&= \frac{\sum_{i=1}^{|B|} |B_i|(|B_i| - 1)}{|U|(|U| - 1)} \\
&= \frac{\sum_{i=1}^{|B|} (|B_i|^2 - |B_i|)}{|U|(|U| - 1)} \\
&= \frac{\sum_{i=1}^{|B|} |B_i|^2 - |U|}{|U|(|U| - 1)} \\
&= \frac{\sum_{i=1}^{|B|} |B_i|^2}{|U|(|U| - 1)} - \frac{1}{|U| - 1} \\
&= \frac{\frac{1}{|B|}(\sum_{i=1}^{|B|} |B_i|)^2}{|U|(|U| - 1)} - \frac{1}{|U| - 1} \\
&= \frac{|U|}{|B|(|U| - 1)} - \frac{1}{|U| - 1} \\
&\geq \frac{|U| - 1}{|U|}\left(\frac{|U|}{|B|(|U| - 1)} - \frac{1}{|U| - 1}\right) \\
&= \frac{1}{|B|} - \frac{1}{|U|}
\end{aligned}$$

Notice $Pr[h(k) = h(l)] \leq \epsilon$, hence $\epsilon \geq \frac{1}{|B|} - \frac{1}{|U|}$

4 Problem 4

Use **CircularArray** to implement this D.S. The $INSERT(S, x)$ operation is the same as that in class. When we do $DELLARGEHALF(S)$, we first use $QUICKSELECT$ to select the median, then go through all elements and copy those no larger than the median to another half-sized array.

The amortized cost of $INSERT(S, x)$ is $O(1)$ apparently. Notice the real cost c_i of $DELLARGEHALF(S)$ is $\Theta(|S|)$, i.e. $c_i = p|S|$. Suppose the potential function is linear to $|S|$, i.e. $\Phi(D_i) = q|S|$, then $\Phi(D_i) - \Phi(D_{i-1}) \leq -\frac{q}{2}|S|$. Therefore $\hat{c_i} = (p - \frac{q}{2})|S|$, we can always define a potential function such that $\hat{c_i} = O(1)$. Hence the amortized cost of $DELLARGEHALF(S)$ is alse $O(1)$. The m operations can run in $O(m)$ time in total.

When we want to output the elements, we just need to do $REMOVE$ like that in class and output them.

5 Problem 5

We want to transform this problem to that in which the counter begins at a number with 0 1s. First, we assume $n \geq cb$. In order to transform the b 1s to 0s, we need to do b times $1 \to 0$ operations. Average them to n operations, each oeartion cost $\frac{b}{n} \leq \frac{1}{c}$ time. Together with the origin problem, the amortized cost of INC in this problem is no more than $2+\frac{1}{c}$, so the total cost is $n(2+\frac{1}{c}) = O(n)$.

6 Problem 6

(a) No. Consider a full array($stack.size == array.size$) and the following sequence of operations on it:

push, pop, pop, push, push, pop, pop, push, push$\cdots$

The first push makes the array size doubled, then the following two pop make the array half, the next two push make the array doubled again$\cdots$ Apparently, the amoritized time per stack operation is $O(N)$.

(b) Yes.

Notice the answer for problem (c) is yes(I have proved below), answer for this problem is yes as well. In this condition, we have to do more pop operations(without shrinking the array) than problem (c) before the pop operation with shrinking(except the first). So we can accumulate more account value than that in problem (c), and the amortized time of pop is $O(1)$, too.

(c) Yes.

The amortized time of push operation is 3, which has been proved in class.

To analyse the amortized time of pop operation, we first define potential function

$$\Phi(T) = \begin{cases} 2 \cdot num - size, & \alpha_i \geq 1/2 \\ size/2 - num, & \alpha_i < 1/2 \end{cases}$$

where num is the stack size, $size$ is the array size. And α_i is the load factor after the i^{th} operation, which is defined as $\alpha_i = num_i/size_i$ and has value in $[0,1]$. Apparently, $\Phi(T_0) = 0$ and $\Phi(T_i) \geq 0$.

If $\alpha_{i-1} < 1/2$ and there is no shrink. We have $size_i = size_{i-1}$ and $num_{i-1} = num_i + 1$, so

$$\hat{c_i} = c_i + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) = 2$$

If $\alpha_{i-1} < 1/2$ and the array is shrink during the i^{th} op, $size_i = size_{i-1}/2$. Notice $\alpha_i = \frac{num_i}{2(num_i+1)} < 1/2$, we have

$$\begin{aligned} \hat{c_i} &= c_i + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) \\ &= num_{i-1} - size_i/2 + 1 \\ &= size_i/2 - size_i/2 + 1 \\ &= 1 \end{aligned}$$

If $\alpha_{i-1} \geq 1/2$, notice the array will be shrunk if and only if $size_{i-1} == 2$ and $num_{i-1} == 1$, this condition takes constant time obviously. Considering other conditions, if $\alpha_i \geq 1/2$, we have $size_i = size_{i-1}$, therefore

$$\hat{c_i} = 1 + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1}) = -1$$

Otherwise, if $\alpha_i < 1/2$, we have

$$\begin{aligned}\hat{c_i} &= 1 + (size_i/2 - num_i) - (2 \cdot num_{i-1} - size_{i-1}) \\ &= 2 - 3 \cdot num_{i-1} + 3 \cdot size_{i-1}/2\end{aligned}$$

Remind that $\alpha_{i-1} \geq 1/2$, so $\frac{num_{i-1}}{size_{i-1}} \geq \frac{1}{2}$, then $size_{i-1} \leq 2 \cdot num_{i-1}$. Therefore

$$\hat{c_i} \leq 2$$

In conclusion, the amortized time of pop operation is $O(1)$.

7 Problem 7

(a) Notice the amortized time of each INC depends on the running time of $SOMETHINGELSE$, we only consider $T(i)$. If $T(i) = 4$, the amortized time of INC if $O(1) + 4$, which is still $O(1)$.

(b) Considering the value of i before $SOMETHINGELSE$ every time we do INC, it is clear that the index of the first 0 in $j - 1$ equals to i after j^{th} INC. We enumerate some i:

0 1
0 2
0 1 0 3
0 1 0 2 0 1 0 4
...

So the corresponding $T(i)$ is

1 2
1 4
1 2 1 8
1 2 1 4 1 2 1 16
...

Besides lots of 1s, we can always distribute 2^m(at the end of each line) to all numbers before equally with constant value 1. So the amortized time is $O(lgn)$.

(c) Enumerate some $T(i)$ as the above

1 4
1 16
1 4 1 64
1 4 1 16 1 4 1 256
...

Analyse like problem (b), we can't distribute the number at the end of each line to all numbers before with a constant value. The average value grows double, i.e. $2,\ 4,\ 8,\ 16 \cdots$. Add them all and the amortized time is $O(n)$.