ICS 311, Spring 2016, Problem Set 03, Topics 5 & 6 Solutions

#1. Peer Credit Assignment

You should have named your group partners and allocated 6 points total across them. If you did not, please email the TA with your points assignment.

#2. Expected Length of Coding Scheme

2 points - 1 for correct solution, 1 for explanation/derivation

An information source generates symbols at random from a five-letter alphabet {a, b, c, d, e} with probabilities Pr[a] =0.4 , Pr[b] = 0.2, Pr[c] = 0.2, Pr[d] = 0.1, and Pr[e] = 0.1. A coding scheme encodes these symbols into binary codes as follows:

a	**0**
b	**10**
c	**110**
d	**1110**
e	**1111**

Compute the expected length of the encoding of n letters generated by the information source. You may use any method you like, but must show your work/justify your answer.

Solution:

Let X_i be a random variable that defines the size of the encoding of the i-th symbol.

Then $E[X_i] = 1 * Pr[X_i=1] + 2 * Pr[X_i=2] + 3 * Pr[X_i=3] + 4 * Pr[X_i=4]$

$= 1 * 0.4 + 2 * 0.2 + 3 * 0.2 + 4 * (0.1 + 0.1) = 2.2$

And let X be the random variable that defines the size of encoding of n symbols.

Then $X = \sum_{i=1}^{n} X_i$

$$E[X] = E\left[\sum_{i=1}^{n} X_i\right]$$

$$= \sum_{i=1}^{n} E[X_i] \quad \text{// By Linearity of Expectations}$$

$$= \sum_{i=1}^{n} 2.2$$

$$= 2.2n$$

(Comment not required in student solutions: If we encoded each symbol using 3 bits the length would be 3n, so the encoding has saved space.)

#3. Hashing with Chaining

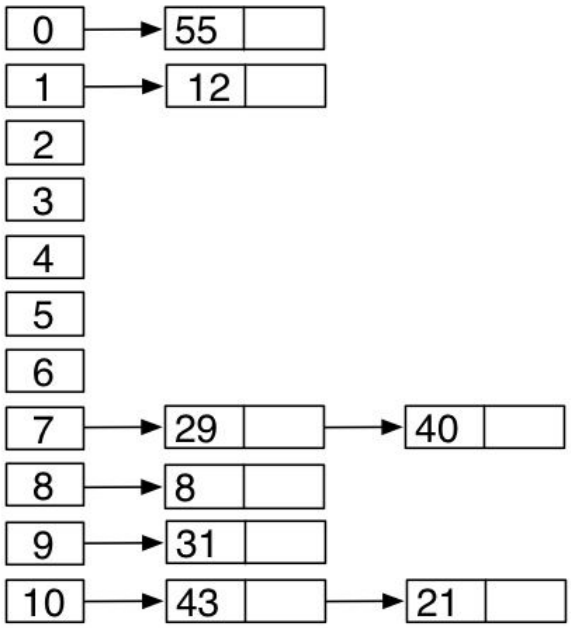
6 points

(a) (2 pts - 1 for correct solution, 1 for argument or proof) Consider a hash table with m slots that uses chaining for collision resolution. The table is initially empty. What is the probability that, after k keys are inserted, there is a chain of size k? Include an argument for or proof of your solution.

Answer: $1/m^{k-1}$
For a chain to have size k, all k keys must hash into the same location. We don't care where the first key hashes. Let's say it's in location *i*. Given this fact, we need to compute the probability that the remaining k-1 keys hash into the position *i* as well. Probability of a key hashing into a particular location is 1/m and is independent of any other value. Therefore, the probability of the remaining k-1 keys hashing into location *i* is $1/m^{k-1}$.

(b) (4 pts - 0.5 for correct placement of each) Show the table that results when 31, 40, 21, 8, 43, 12, 55, 29 are cumulatively inserted into an initially empty hash table of size 11 with chaining and $h(k) = k \bmod 11$. *Draw this with a vertical table indexed from 0 to 10, and linked lists going off to the right, as shown. You may want to print the table, draw with dark ink, and then scan or photograph it to include your solution in your document.*

Solution:



#4. Open Addressing Strategies

12 points

(a) (4 pts - 0.5 for correct placement of each): Show the table that results when 31, 40, 21, 8, 43, 12, 55, 29 are cumulatively inserted into an initially empty hash table of size 11 with <u>linear probing</u> and

h'(k) = k mod 11

h(k,i) = (h'(k) + i) mod 11, *where the first probe is probe i=0.*

Draw this and the next result as horizontal arrays indexed from 0 to 10 as shown below. Show your work to justify your answer to the next question!

Solution:

43	12	55	29				40	8	31	21
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*

(b) (1 pt - for correct count. Don't grade work but use it to debug problems): <u>How many re-hashes after collision are required for this set of keys?</u> *Show your work here so we can give partial credit or feedback if warranted.*

Solution:

31 % 11 = 9
40 % 11 = 7
21 % 11 = 10
8 % 11 = 8
43 % 11 = 10
(10 + 1) % 11 = 0
12 % 11 = 1
55 % 11 = 0
(0 + 1) % 11 = 1
(0 + 2) % 11 = 2
29 % 11 = 7
(7 + 1) % 11 = 8
(7 + 2) % 11 = 9
(7 + 3) % 11 = 10
(7 + 4) % 11 = 0
(7 + 5) % 11 = 1
(7 + 6) % 11 = 2
(97 + 7) % 11 = 3

10 rehashes/reprobes

(c) (4 pts - 0.5 for correct placements of each): Show the table that results when 31, 40, 21, 8, 43, 12, 55, 29 are cumulatively inserted into an initially empty hash table of size *m* = 11 with <u>double hashing</u> and

$h(k,i) = (h_1(k) + ih_2(k)) \bmod 11$

$h_1(k) = k \bmod 11$

$h_2(k) = 1 + (k \bmod 7)$

Refer to the code in the book for how i is incremented. Show your work to justify your answer to the next question!

Solution:

55	12	29		43			40	8	31	21
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*

(d) (1 pt - for correct count. Don't grade work but use it to debug problems)): <u>How many re-hashes after collision are required for this set of keys?</u> *Show your work here so we can give partial credit or feedback if warranted.*

Solution:

31 % 11 = 9
40 % 11 = 7

21 % 11 = 10
8 % 11 = 8
43 % 11 = 10
10 + (1 + (43 % 7)) = 10 + 5 = 15 % 11 = 4
12 % 11 = 1
55 % 11 = 0
29 % 11 = 7
(7 + 1(1 + (29 % 7))) % 11 = (7 + 1(1 + 5)) % 11 = 13 % 11 = 2

2 rehashes/reprobes

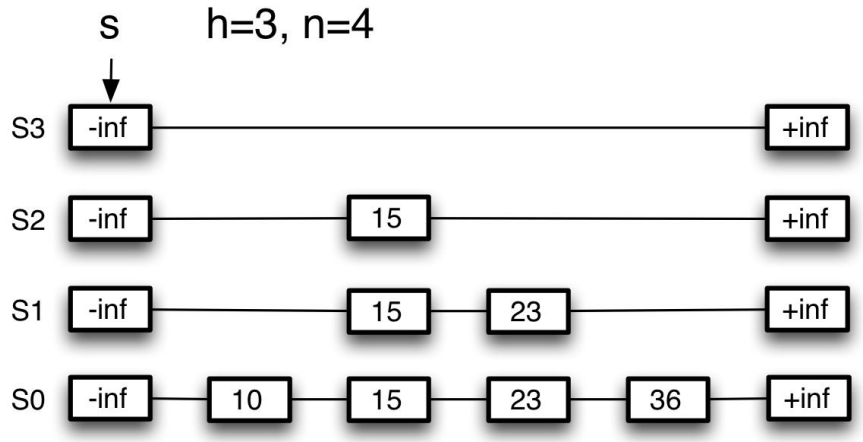
(f) (2 pts - 1 for correct answer, 1 for showing derivation from theorem): Open addressing insertion is like an unsuccessful search, as you need to find an empty cell, i.e., to *not* find the key you are looking for! If the open addressing hash functions above were uniform hashing, what is the expected number of probes at the time that the last key (29) was inserted? *Use the theorem for unsuccessful search in open addressing and show your work.* Answer with a specific number, not O or Theta.

Solution: Theoretical is the same as unsuccessful search: 1 / (1 - α) = 1 / (1 - *n*/*m*) = 1 / (1 - 7/11) = 1 / (4/11) = 11/4 or 2.75.

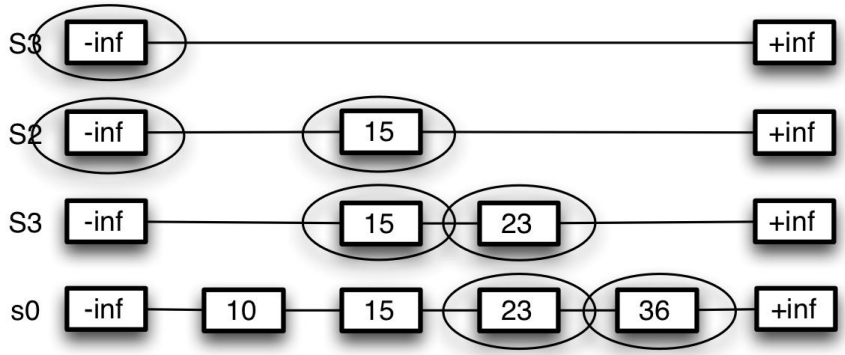
#5. Skip Lists

10 points

Here is a skip list, including instance variables **s** (the starting position), **h** (the height: we assume that h starts counting from 0), and **n** (the number of keys currently stored in the skip list). For simplicity, the graphic shows only horizontal lines, but keep in mind that these are doubly linked lists in both the horizontal and vertical directions. You may also use this simplified drawing in your responses.



(a) (1 pt) Trace the path that SkipSearch(36) takes, by circling every node that p is assigned to as the SkipSearch algorithm executes, starting with s. (You may want to print out the diagram, trace it with a dark pen, and scan or photograph it to include in your document.)

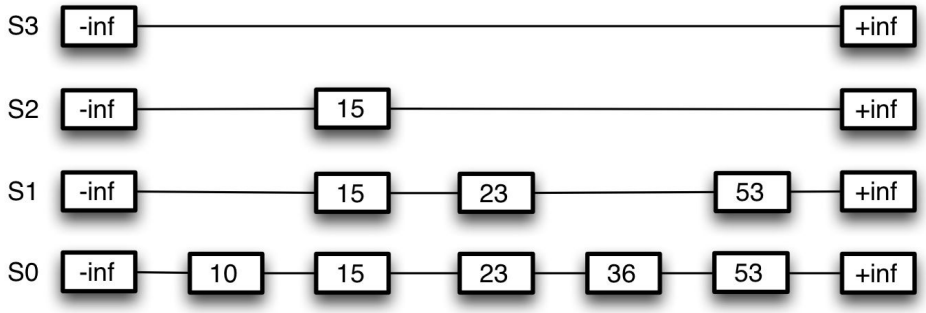


In the remaining questions, you will show what the skip list shown above looks like after the cumulative operations indicated below, using the pseudocode for SkipInsert and SkipSearch in the lecture notes, and your understanding of how SkipDelete works from the class activity.

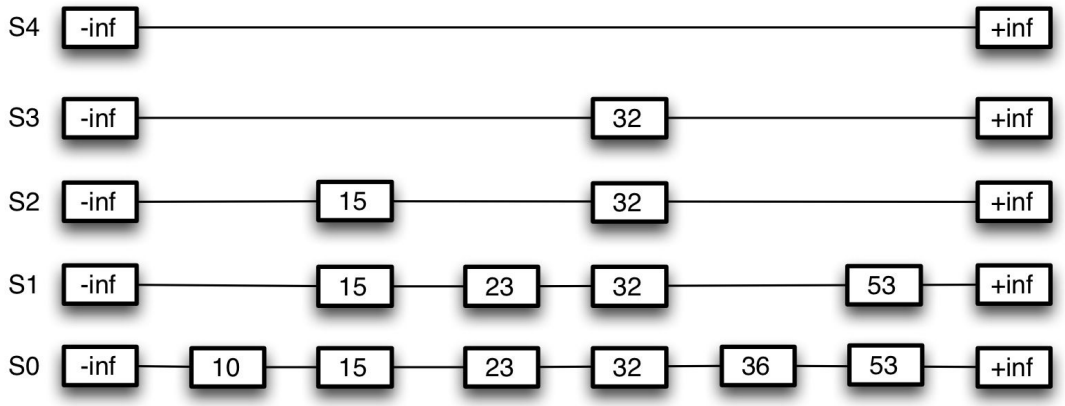
Since this is a random algorithm and we want everyone to have the same answer to facilitate grading, I give you sequences of random numbers in [0,1] that you should use in tracing the algorithm where the insertion code says `while random(0,1) ≤ 1/2 do…` (Not all of the numbers will be used, as I am testing your understanding of when and how the random numbers are used).

The operations are **cumulative:** each step builds on the result of the previous one. Redraw the entire data structure after each operation, and also update instance variables **s**, **h** and **n** as needed.

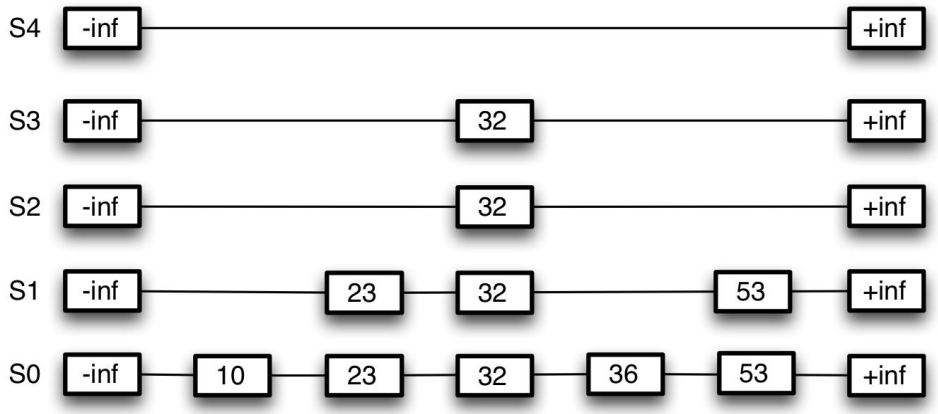
(b) (2 pts - placed correctly in both levels) Redraw after SkipInsert(53,data) where random(0,1) returns .14, .51, .22, .68, .45, ...



(c) (3 pts - 0.5 for placement of 32 in each level, 1 for correctly adding a new empty level) Redraw after SkipInsert(32,data) where random(0,1) returns .25, .39, .18, .97, .02, …



(d) (1 pt - tower removed) Redraw after SkipDeletePosition(SkipSearch(15)).



Something to think about (but not graded): What should the list look like if we now deleted 32? There is a choice to be made here that we have not discussed!

(e) (3 pts: 0.5 each for s, h, n, -inf, +inf and S0) Now draw what an *empty* skip list would look like, including s, h and n.