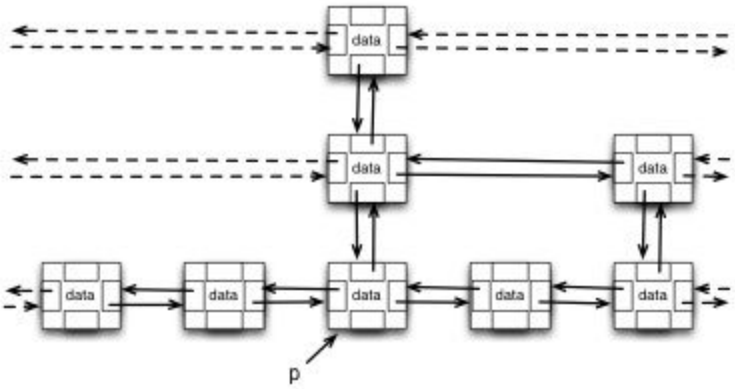
Class 02/03: Deletion from Skip Lists and Hash Tables

1. Deletion from Skip Lists



Here's a portion of a skip list. Assume SkipInsert has been given a uniform distribution of keys stored in random order, and built the skip list using random(0,1) with cutoff of 0.5.

In this problem we examine the expected runtime of two procedures for deleting from a skip list.

- **SkipDelete(k)** is given a key k and has to delete all positions (list cells) containing that key. This will be a "tower" of positions from the bottommost on up.
- **SkipDeleteTower(p)** is given a position p in the bottom most list of the data structure, and has to delete that position and the tower above it.

Before you continue you may want to trace out an example of how these work.

a. What is the expected runtime of SkipDelete(k) in terms of n, the number of keys stored in the skip list, and why?

> **Solution:** Θ(lg n), because this is the expected height of the list and like SkipSearch we must search to find all positions of k (including the bottommost one).

b. Define indicator random variable X_i = I{tower is exactly size i}. What is the probability $P_i = E[X_i]$ that a given call to SkipDeleteTower(p) would have to delete <u>exactly</u> i positions (list cells)? Write out some examples and then come up with a general expression for P_i.

> **Solution:** $P_i = (1/2)^i$

c. Let X be a random variable for the number of positions that have to be deleted by SkipDeleteTower(p). Write a formula for X in terms of the X_i, compute the expected number of positions E[X], and conclude with the expected runtime of SkipDeleteTower.

Solution:

$X = \sum_{i=0}^{n} iX_i$

that is, sum the cost to delete *i* elements times the indicator selecting which i has taken place.

$E[X] = E\left[\sum_{i=0}^{n} iX_i\right]$ taking expectation of both sides

$= \sum_{i=1}^{n} iE\left[X_i\right]$ by linearity of expectation

$= \sum_{i=1}^{n} i(\frac{1}{2})^i$ substituting P_i

$\leq \sum_{i=0}^{\infty} i(\frac{1}{2})^i$ Can include i=0 as the probability is 0. Extending series to ∞ so that formula A.8 will apply, this becomes an upper bound.

$= \frac{1}{2} / (1 - \frac{1}{2})^2$ Using formula A.8 for geometric series

$= \frac{1}{2} / (\frac{1}{2})^2 = 1/(\frac{1}{2}) = 2$ Simplifying.

Therefore the expected value of the number of nodes to delete is 2, so the expected runtime of SkipDeleteTower is Θ(1)!

Why not Θ(lg n)? Because the height of a tower for any given insertion is defined at insertion time and never changes with n!

2. Deletion from Hash Tables under Open Addressing

When deleting an element *k* from the hash table, the lecture notes mention that we cannot simply delete the element by writing NIL into the table entry that *k* occupied. Instead, when deleting *k*, we insert a special entry DELETED in place of *k*.

a. Since deletion requires first finding the item, we can write the code for Hash-Delete by modifying Hash-Search. Assuming unique keys, make the change.

```
Hash-Delete (T,k)
1    i = 0
2    repeat
3         j = h(k,i)
4         if T[j] == k
4.5            t[j] = DELETED
5              return SUCCESS // not required in student solutions
```

```
6       i = i + 1
7    until T[j] == NIL or i == m
8   return NIL
```

b. Write the new Hash-Insert to work with the DELETED value, again <u>assuming unique keys</u>.

```
Hash-Insert (T,k)  // modified to handle DELETED
1    i = 0
2    repeat
3        j = h(k,i)
4        if T[j] == NIL || T[j] == DELETED
5            T[j] = k
6            return j
7        else i = i + 1
8    until i == m
9    error "hash table overflow"
```

3. Extra Credit

a. Write the recursive pseudocode for SkipDeleteTower(p), given the hint below. A skip list node has fields p.next, p.prev, p.above and p.below.

```
SkipDeleteTower(p)
if p ≠ null
    // splice out of this doubly linked list
    p.prev.next = p.next
    p.next.prev = p.prev
    // recurse to splice out of list above
    SkipDelete(p.above)
```