Problem Set 5, Data Structures

Jay R Bolton

December 15, 2011

Set 5: 5.1, 5.5, 5.14, 5.15

5.1 *(a)*

Index	**Key**
1	4371
2	
3	1323, 6173
4	4344
5	
6	
7	
8	
9	4199, 9679, 1989

(b)

Index	**Key**
1	4371
2	9679
3	1323
4	6173
5	4344
6	1989
7	
8	
9	4199

(c)

Index	**Key**
1	4371
2	9679
3	1323
4	6173
5	4344
6	
7	
8	9679
9	4199

(d) I guess I'll assume it's linear probing.

Index	Key
1	
2	
3	4344
4	1323
5	6173
6	4371
7	4199
8	9679
9	1989

5.5 I assume he's talking about the collision function for open addressing (probing with a set of random numbers?).

(a) If n is the length of our hash table, and i maps to 0..x where x is the parameter that resolved an open bucket, then f(i) can probe 0..n-1 and return indexes for n different buckets, so that every bucket in the hash table will be potentially tried.

(b) Probably.

(c) In my copy of the book, the load factor figures are red X boxes (missing image), so I could try to generalize. If the load factor is low, say 0.25, then the initial probe will have a 1/4 chance of colliding. If it was higher, such as .75, then there would be a 3/4 chance of colliding. The load factor of the table then plays a big part in probing time.

(e) You could start out with a list of integers, [1..n], and iterate n times. On each iteration, generate a random number modulo n-1, retrieve and remove that element from the list. I don't know what P refers to.

5.14

Index	Key
000	00000010, 00001011
001	00101011
010	01010001
011	01111111, 01100001
100	10011011, 10010110, 10011110
101	10111101,10111110,
110	11001111, 11011011
111	11110000

5.15

```
insert(key, hash)
index := f(key) where f is the hashing function
if (size(index(hash,index)) > hash->m)
hash := expand(hash)
insert(key, hash)
  else append bucket at index(hash,index) with key
```

```
expand(hash)
  allocate new hash of double size
create new hash function based on the new size
iterate through each element of the old hash,
reinserting each key into the new hash.
```

Its performance for lookups will me much better, because collision searches are likely to be much shorter. However, the act of extending the table will likely be very expensive.