Andrew Taylor

Atayl136

Homework 10: Hashing

1. **If a hash table contains *tablesize* positions and *n* records currently occupy the table, the load factor *lf* is defined as *n/tablesize.* Suppose a hash function uniformly distributes *n* keys over the *tablesize* positions of the table and the table has load factor *lf*. Show that of new keys inserted into the table, (n-1)* *lf*/2 of them will collide with a previously entered key. Think about the accumulated collisions over a series of collisions.**

If a hash table has one element, it's probability of colliding with another element is zero. After inserting another element, we have the possibility of a collision. Collisions are equally likely per insert so the probability of collision for the next insert is $\frac{n-1}{tablesize}$:

Number of Keys	Probability of Collision
1	0
2	1/tablesize
3	2/tablesize
…	
N	(n-1)/tablesize

From this we can see the probability of n collisions is $\sum_{i=0}^{n-1} \frac{i}{tablesize}$. We obtain $\sum_{i=0}^{n-1} \frac{i}{tablesize} = \frac{n(n-1)}{2T}$, by the sum of natural numbers formula discovered by Gauss, where T is the number of unique hash values. The load factor (*lf*) being defined as $\frac{n}{T}$ we can see that the number of collisions for n keys is $\frac{(n-1)\times lf}{2}$.

2. **Assume that *n* random positions of a *tablesize*-element hash table are occupied, using hash and rehash functions that are equally likely to produce any index in the table. The hash and rehash functions themselves are not important. The only thing that is important is they will produce any index in the table with equal probability. Start by counting the number of insertions for each item as you go along. Use that to show that the average number of comparisons needed to insert a new element is *(tablesize* + 1*)/ (tablesize*-n+1). Explain why linear probing does not satisfy this condition.**

Since hash and rehash functions are equally likely to produce any index in the table with equal probability, the probability of finding an empty slot in the table during insertion is:

P(empty slot) = (tablesize - n) / tablesize

Now, let's analyze the insertion process. For the first comparison, there is a probability of P(empty slot) that we find an empty slot. If we don't find an empty slot in the first comparison, we move on to the second comparison, where again, there's a probability of P(empty slot) that we find an empty slot, and so on.

Therefore, the average number of comparisons (C_avg) needed to insert a new element can be represented as:

C_avg = ((tablesize - n) / tablesize) / (1 - ((tablesize - n) / tablesize))

Simplifying this expression, we get:

C_avg = (tablesize + 1) / (tablesize - n + 1)