论题 2-9 作业

姓名：陈劭源　　　　学号：161240004

1 [CS] Problem 5.8

a. When $n \leq k$, the probability that all n items hash to different locations is

$$P_1 = \frac{k^{\underline{n}}}{k^n} = \frac{k!}{(k-n)!k^n}$$

and when $n > k$, the probability is 0.

b. When $i \leq k+1$, the probability that the ith item is the first collision is

$$P_2(i) = \frac{k^{\underline{i-1}}}{k^(i-1)} \times \frac{i-1}{k} = \frac{k!(i-1)}{(k-i+1)!k^i}$$

and when $i > k+1$, the probability is 0.

c. Let X denote the number of items you hash until the first collision, then

$$E(X) = \sum_{i=2}^{k+1}(i-1)P_2(i) = \left(\sum_{i=2}^{k+1}\frac{k!(i-1)i}{(k-i+1)!k^i}\right) - 1$$

d. When $k = 20$, the expected number is 5.29358.
When $k = 100$, the expected number is 12.20996.

2 [CS] Problem 5.11

a. Since we must go through the whole list to ensure that the item is indeed not in the hash table, the running time for an unsuccessful search is $\Theta(1 + length)$, where the expectation of $length$ is n/k, thus the expected running time is $\Theta(1 + n/k)$.

b. Assume there are i elements inserted after the item you are searching for, then the expected running time is

$$t_i = \Theta(1 + \sum_{j=0}^{i} j\binom{i}{j}(1/k)^j(1-1/k)^{i-j}) = \Theta(1 + i/k)$$

And the expected running time for a successful search is

$$\Theta(\frac{1}{n}\sum_{i=0}^{n-1} t_i) = \Theta(\frac{1}{n}\sum_{i=0}^{n-1}(1 + i/k)) = \Theta(1 + (n-1)/2k)$$

Since there is no need to go through the whole list and we can stop searching immediately we've found the item, the expected running time of a successful search is about half of the unsuccessful one.

3 [CS] Problem 5.14

By Theorem 5.15, the expected number of empty slots when hashing $2k$ items into a hash table with k slots is

$$E(X) = k(1 - 1/k)^{2k}$$

and when k grows large, the expected fraction of empty slots is

$$\lim_{k\to\infty} E(X)/k = \lim_{k\to\infty} k(1 - 1/k)^{2k}/k = \lim_{k\to\infty}(1 - 1/k)^{2k} = e^{-2}$$

4 [TC] Problem 11.2-3

Since the linked list is not randomly accessible, binary search does not apply and we can only use sequential search, so the modification does not affect the running time for successful searches and unsuccessful searches.

We have to insert the item to the correct position in the list, so the average running time of insertions becomes $\Theta(1 + \alpha/2)$ where α is the load factor.

This modification does not affect the running time of deletions, which is still $O(1)$.

5 [CS] Problem 11.2-5

There are $|U| > nm$ elements (pigeons), but there are only m slots (holes). By the pigeonhole principle, there exists a subset of U, consisting of n keys that all hash to the same slot. If we want to search for the last element in the slot, the running time is $\Theta(n)$. Therefore, the worst-case searching time for hashing with chaining is $\Theta(n)$.

6 [CS] Problem 11.2-6

TODO

7 [CS] Problem 11.3-3

The string can be represented as

$$S = \sum_{i=0}^{n-1} S_i(2^p)^i$$

Consider $(2^p)^i \mod (2^p - 1)$, we have

$$(2^p)^i \bmod (2^p - 1) = (2^p \bmod (2^p - 1))^i = 1^i = 1$$

therefore

$$S \bmod (2^p - 1) = \sum_{i=0}^{n-1} S_i(2^p)^i \bmod (2^p - 1) = \sum_{i=0}^{n-1} S_i \bmod (2^p - 1)$$

The result is only in terms of the sum of the characters in the string and has nothing to do with the position of each character. That means, if string y is a permutation of string x, then they hash to the same value.

If there are many string pairs that one of them is a permutation of the other, such as 'POT' and 'TOP', in our application, then the hash function will lead to a lot of collisions, which is undesirable.

8 [CS] Problem 11.3-4

k	kA	$kA \bmod 1$	$m(kA \bmod 1)$	$h(k)$
61	37.70007	.70007	700.07	700
62	38.31811	.31811	318.11	318
63	38.93614	.93614	936.14	936
64	39.55418	.55418	554.18	554
65	40.17221	.17221	172.21	172