The Australian National University Semester 2, 2024
School of Computing Tutorial 9

Convenor(s): Yun Kuen Cheung (Marco) & Ahad N. Zehmakan

Concrete version: Tuesday 1st October, 2024, 17:22 (Compilation date and time)

Algorithms (COMP3600/6466)

Problems marked with $(*)$ are challenge exercises. They will not be discussed in tutorials, and solutions will not be released. Once confident in your solution for a challenge exercise, you are welcome to discuss it with your tutor during the consultation period in the tutorial session, or schedule a time with Marco to present it.

Exercise 1 *Hash Tables*

1. Suppose $\mathcal{U} = \{0, 1, 2, \ldots, 2024\}$, $m = 23$.
   (a) If we use $h'(k) = 11k + 8 \pmod m$ as hash function, and we use chaining as the collision resolution scheme, what is the resultant hash table when we insert the following keys: 100, 2024, 331, 124, 253, 951?
   (b) If we use open addressing as the collision resolution scheme with the hash function $h(k, i) = h'(k) + i \pmod m$, where h' is the function given in part (a), what is the resultant hash table when we insert the following keys: 100, 2024, 331, 124, 253, 951?
   (c) If we use double hashing with the hash function $h(k, i) = h'(k) + i \cdot h^*(k) \pmod m$, where h' is the function given in part (a), and $h^*(k) = 1 + (k \pmod{(m-1)})$, what is the resultant hash table when we insert the following keys: 100, 2024, 331, 124, 253, 951?

Solution

First, we compute $h'(k)$ for $k = 100, 2024, 331, 124, 253, 951$.

- $h'(100) = 11 \cdot 100 + 8 \pmod{23} = 4$.
- $h'(2024) = 11 \cdot 2024 + 8 \pmod{23} = 8$.
- $h'(331) = 11 \cdot 331 + 8 \pmod{23} = 15$.
- $h'(124) = 11 \cdot 124 + 8 \pmod{23} = 15$.
- $h'(253) = 11 \cdot 253 + 8 \pmod{23} = 8$.
- $h'(951) = 11 \cdot 951 + 8 \pmod{23} = 4$.

(a) The hash table has 23 slots, indexed from 0 to 22. Except slots 4, 8 and 15, the others are empty (pointing to None). For the exceptional three slots, they point to linked lists as below:

- $4 ->$ [951, 100]
- $8 ->$ [253, 2024]
- $15 ->$ [124, 331]

Note that the keys which are inserted latter occupies the head position of the respective linked lists.

(b) Note that this is linear probing.
When inserting 100, it goes to slot 4.
Then insert 2024, it goes to slot 8.
Then insert 331, it goes to slot 15.
Then insert 124, it first checks slot 15, as the slot is occupied, it goes to slot 16.
Then insert 253, it first checks slot 8, as the slot is occupied, it goes to slot 9.
Finally, insert 951, it first checks slot 4, as the slot is occupied, it goes to slot 5.
Eventually, the hash table is

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
key	-	-	-	-	100	951	-	-	2024	253	-	-	-	-	-	331	124	-	-	-	-	-	-

(c) When inserting 100, it goes to slot 4.
Then insert 2024, it goes to slot 8.
Then insert 331, it goes to slot 15.
Then insert 124, it first checks slot 15, as the slot is occupied, we compute $h^*(124) = 15$, and goes to slot $15 + 1 \times 15 \pmod{23} = 7$.
Then insert 253, it first checks slot 8, as the slot is occupied, we compute $h^*(253) = 12$, and goes to slot $8 + 1 \times 12 \pmod{23} = 20$.
Finally, insert 951, it first checks slot 4, as the slot is occupied, we compute $h^*(951) = 6$, and goes to slot $4 + 1 \times 6 \pmod{23} = 10$.
Eventually, the hash table is

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
key	-	-	-	-	100	-	-	124	2024	-	951	-	-	-	-	331	-	-	-	-	253	-	-

2. Suppose we have a collection of hash functions $\mathcal{H} = \{h_1, h_2, h_3\}$ that map the key set $\{a, b, c, d\}$ into the range $\{0, 1, 2, 3, 4\}$ as follows:

- $h_1(a) = 1$, $h_1(b) = 0$, $h_1(c) = 3$, $h_1(d) = 2$
- $h_2(a) = 2$, $h_2(b) = 4$, $h_2(c) = 2$, $h_2(d) = 1$
- $h_3(a) = 3$, $h_3(b) = 1$, $h_3(c) = 0$, $h_3(d) = 4$

(a) If $m = 5$, explain why $\mathcal{H}$ is not a family of universal hash functions.
(b) Can you make changes to h_2 only so that $\mathcal{H}$ becomes a family of universal hash functions?

Solution

(a) Recall that a hash family $\mathcal{H}$ is universal if for any two distinct keys k_1, k_2, if we draw $h \in \mathcal{H}$ uniformly randomly, the probability that $h(k_1) = h(k_2)$ is at most $\frac{1}{m}$. This is violated for the key pair a, c, because $h_2(a) = h_2(c)$, so if we draw $h \in \mathcal{H}$ uniformly randomly, $\mathbb{P}[h(a) = h(c)] = \frac{1}{3} > \frac{1}{5}$.
(b) To make $\mathcal{H}$ become a universal hash family, it suffices to make sure that for each hash function, there is no collision between two different keys. One way is to change $h_2(a)$ from 2 to 3.

3. Consider a hash table T with m slots. Suppose T contains a single key k. Someone searches for r keys that are different from k. Assume T uses simple uniform hashing, what is the probability that at least one of the r searches collides with the slot containing key k? Explain your answer.

Solution

Since T uses simple uniform hashing, for each of the r keys, denoted by k', $\mathbb{P}[h(k) = h(k')] = \frac{1}{m}$, and hence $\mathbb{P}[h(k) = h(k')] = 1 - \frac{1}{m}$. The probability that none of the r keys collide with k is $(1 - \frac{1}{m})^r$. Thus, the probability that at least one of the r keys collide with k is $1 - (1 - \frac{1}{m})^r$.

4. In lecture, we discuss a theorem of Carter and Wegman, which states that:

Let $\mathcal{U} = \{0, 1, 2, \ldots, u - 1\}$. Suppose p is a prime number satisfying $p > u$.
Define the family

$$\mathcal{H}_{p,m} = \{h_{a,b} \mid 1 \le a \le p - 1,\ 0 \le b \le p - 1\}\ ,$$

where $h_{a,b}(k) = (ak + b \pmod p) \pmod m$. Then $\mathcal{H}_{p,m}$ is universal.

If we use a prime number $p < u$ instead, does $\mathcal{H}_{p,m}$ remain universal? Explain your answer.

Solution

If $p < u$, this means the keys 0 and p are both in $\mathcal{U}$. Then observe that for any a, b,

$$h_{a,b}(p) = (ap + b \pmod p) \pmod m = (b \pmod p) \pmod m = h_{a,b}(0).$$

The second equality above holds because $ap = 0 \pmod p$. In other words, if $h_{a,b}$ is chosen uniformly

randomly from $\mathcal{H}_{p,m}$, $\mathbb{P}[h_{a,b}(0) = h_{a,b}(p)] = 1 > \frac{1}{m}$, indicating $\mathcal{H}_{p,m}$ is not a universal hash family.

5. In lecture, we discuss how to construct perfect hashing with total hash table size at most $4n$.

   (a) Analyze the expected running time for constructing the perfect hashing.

   (b) (*) If you are allowed to use arbitrarily long (expected) running time, show that it is possible to construct perfect hashing with total hash table size at most $2.01n$. Can you do even better?

Solution

To construct perfect hashing as described in the lecture, there are some main steps, listed below:

(1) Given $\mathcal{U}$, find a prime number p which is larger than $|\mathcal{U}|$. Set $m = n$.

(2) Repeat:

- sample a between 1 and $p-1$, and b between 0 and $p-1$;
- compute the hash values $h_{a,b}(k)$ for all keys $k \in S$, where $h_{a,b}(k) = (ak + b \pmod p) \pmod m$;
- compute the pairwise collision count C, quit this repeat loop if $C < n$.

(3) For each group S_i with $n_i = |S_i|$, set $m_i = (n_i)^2$, and repeat:

- sample a between 1 and $p-1$, and b between 0 and $p-1$;
- compute the hash values $h'_{a,b}(k)$ for all keys $k \in S_i$, where $h'_{a,b}(k) = (ak + b \pmod p) \pmod{m_i}$;
- quit this repeat loop if there is no collision.

In this analysis, we assume finding the prime number p and sampling a, b each time takes $\mathcal{O}(1)$ time, although depending on the computational model and algorithms used, their running times may depend on $|\mathcal{U}|$.

In Step 2, computing the hash values $h_{a,b}(k)$ and grouping the keys according to hash values take $\mathcal{O}(n)$ time. Recall that $C = \sum_i \binom{n_i}{2}$, so once we have the values of n_i, computing C takes $\mathcal{O}(n)$ time too. Thus, each iteration of Step 2 takes $\mathcal{O}(n)$ running time. In the lecture, we showed that Step 2 has success (thus quitting the loop) probability of at least $\frac{1}{2}$. Thus, the expected running time of Step 2 is $\mathcal{O}(n)$.

In Step 3, by following the analysis of Step 2, we can show the expected running time of Step 3 is $\mathcal{O}(n_i)$ for each group S_i. The total expected running time for all groups is $\sum_i \mathcal{O}(n_i) = \mathcal{O}(\sum_i n_i) = \mathcal{O}(n)$. Overall, the expected running times for Steps 1, 2 and 3 combined are still $\mathcal{O}(n)$.