# VE 281 Homework 2 (Due 10/31 11:59PM)

Name: 周振鹏 ID: 518021911039

- **Exercise 1. Open Addressing**

  Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59$ into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h(k) = k$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \mod (m-1))$.

  **Solution:**

  We use $T_t$ to represent each time stamp $t$ starting with $i = 0$, and if encountering a collision, then we iterate $i$ from $i = 1$ to $i = m - 1 = 10$ until there is no collision.

  Linear probing:

| $h(k,i) = (k+i)$ mod 11 | $T_0$ | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 mod 11 | | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 |
| 1 mod 11 | | | | | | | | 88 | 88 |
| 2 mod 11 | | | | | | | | | |
| 3 mod 11 | | | | | | | | | |
| 4 mod 11 | | | | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 mod 11 | | | | | 15 | 15 | 15 | 15 | 15 |
| 6 mod 11 | | | | | | 28 | 28 | 28 | 28 |
| 7 mod 11 | | | | | | | 17 | 17 | 17 |
| 8 mod 11 | | | | | | | | | 59 |
| 9 mod 11 | | | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| 10 mod 11 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

  Quadradic probing:

| $h(k,i) = (k+i+3i^2)$ mod 11 | $T_0$ | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 mod 11 | | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 |
| 1 mod 11 | | | | | | | | | |
| 2 mod 11 | | | | | | | | 88 | 88 |
| 3 mod 11 | | | | | | | 17 | 17 | 17 |
| 4 mod 11 | | | | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 mod 11 | | | | | | | | | |
| 6 mod 11 | | | | | | 28 | 28 | 28 | 28 |
| 7 mod 11 | | | | | | | | | 59 |
| 8 mod 11 | | | | | 15 | 15 | 15 | 15 | 15 |
| 9 mod 11 | | | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| 10 mod 11 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

  Double hashing:

| $h(k,i) = (k+i(1+k \mod 10))$ mod 11 | $T_0$ | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 mod 11 | | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 |
| 1 mod 11 | | | | | | | | | |
| 2 mod 11 | | | | | | | | | 59 |
| 3 mod 11 | | | | | | | | 17 | 17 |
| 4 mod 11 | | | | | 4 | 4 | 4 | 4 | 4 |
| 5 mod 11 | | | | | | 15 | 15 | 15 | 15 |
| 6 mod 11 | | | | | | | 28 | 28 | 28 |
| 7 mod 11 | | | | | | | | 88 | 88 |
| 8 mod 11 | | | | | | | | | |
| 9 mod 11 | | | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| 10 mod 11 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

- **Exercise 2. Slot-Size Bound for Chaining**

  Suppose that we have a hash table with $n$ slots, with collisions resolved by chaining, and suppose that $n$ keys are inserted into the table. Each key is equally likely to be hashed to each slot. Let $M$ be the maximum number of keys in any slot after all the keys have been inserted. Your mission is to prove an $O(\lg n/\lg\lg n)$ upper bound on $\mathrm{E}[M]$, the expected value of $M$

  a. Argue that the probability $Q_k$ that exactly $k$ keys hash to a particular slot is given by

$$Q_k = \left(\frac{1}{n}\right)^k \left(1-\frac{1}{n}\right)^{n-k} \binom{n}{k}$$

  b. Let $P_k$ be the probability that $M = k$, that is, the probability that the slot containing the most keys contains $k$ keys. Show that $P_k \le nQ_k$.

  c. Use Stirling's approximation to show that $Q_k < e^k/k^k$.

  d. Show that there exists a constant $c > 1$ such that $Q_{k_0} < 1/n^3$ for $k_0 = c\lg n/\lg\lg n$. Conclude that $P_k < 1/n^2$ for $k \ge k_0 = c\lg n/\lg\lg n$

  e. Argue that

$$\mathrm{E}[M] \le \Pr\left\{M > \frac{c\lg n}{\lg\lg n}\right\}\cdot n + \Pr\left\{M \le \frac{c\lg n}{\lg\lg n}\right\}\cdot\frac{c\lg n}{\lg\lg n}$$

  Conclude that $\mathrm{E}[M] = O(\lg n/\lg\lg n)$

  **Solution:**

a. The probability of going into one slot is $\frac{1}{n}$ and not going into it is $(1-\frac{1}{n})$. Since we want to choose $k$ keys to go into this slot, we have $Q_k = \binom{n}{k}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{n-k}$

b. $P_k$ = P[one slot has $k$ keys and others have no more than $k$ keys]
$\le$ P[one slot has $k$ keys] $= \sum_{i=1}^{n} Q_k = n\cdot Q_k$

c. Stirling's approximation: $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ when $n$ is large.

$$Q_k = \frac{1}{n^k}\left(1-\frac{1}{n}\right)^{n-k}\frac{n!}{k!(n-k)!} < \frac{n!}{n^k\,k!(n-k)!} < \frac{1}{k!} \approx \frac{e^k}{\sqrt{2\pi n}\;k^k} < \frac{e^k}{k^k}$$

d. Since $\lg\lg n = 0$ when $n = 2$, we should assume $n \ge 3$.
Since $Q_{k_0} < \frac{e^{k_0}}{k_0^{k_0}}$, we can simply prove $\frac{e^{k_0}}{k_0^{k_0}} < \frac{1}{n^3} \Rightarrow n^3 < \frac{k_0^{k_0}}{e^{k_0}}$

$\lg n^3 < \lg k_0^{k_0}/e^{k_0}$

$3\lg n < k_0(\lg k_0 - \lg e)$

$3 < k_0\left(\frac{\lg k_0}{\lg n} - \frac{\lg e}{\lg n}\right) = c\left(1 + \frac{\lg c - \lg e}{\lg\lg n} - \frac{\lg\lg\lg n}{\lg\lg n}\right)$

Let $f(n) = 1 + \frac{\lg c - \lg e}{\lg\lg n} - \frac{\lg\lg\lg n}{\lg\lg n}$, we need to show there exists $c > 1$ such that $3 \le cf(n) \Rightarrow c > \frac{3}{f(n)}$

Since $\lim_{n\to\infty} f(n) = 1$, we can find an $n_0$ such that $f(n) \geq \frac{1}{2}$ when $n \geq n_0$. Therefore $\frac{3}{f(n)} \leq b$, and we can find $c > b$ works for $n \geq n_0$.

When $3 \leq n < n_0$, we should choose the largest $c$ such that $3 < cf(n)$. Then, we have shown $Q_{k_0} < \frac{1}{n^3}$.

Since $P_k \leq nQ_k$, we have $P_{k_0} \leq nQ_{k_0} < n \cdot \frac{1}{n^3} = \frac{1}{n^2}$

For $k > k_0$, we should pick $c$ that is large enough such that $k_0 > 3 > e$. Then $\frac{e}{k} < 1$ for $k \geq k_0$. Also $\frac{e^k}{k^k}$ decreases as $k$ increases

$Q_k < \frac{e^k}{k^k} \leq \frac{e^{k_0}}{k_0^{k_0}} < \frac{1}{n^3}$ for $k \geq k_0$.

Therefore, $P_k < \frac{1}{n^2}$ for $k \geq k_0$.

e. $E[M] = \sum_{k=0}^{n} k \cdot Pr\{M = k\}$

$$= \sum_{k=0}^{k_0} k \cdot Pr\{M = k\} + \sum_{k=k_0+1}^{n} k \cdot Pr\{M = k\}.$$

$$\leq k_0 \sum_{k=0}^{k_0} Pr\{M = k\} + n \sum_{k=k_0+1}^{n} Pr\{M = k\}$$

$$= k_0 \cdot Pr\{M \leq k_0\} + n \cdot Pr\{M > k_0\}$$

Therefore, we have shown it is true with $k_0 = \frac{c \lg n}{\lg \lg n}$.

We know that $Pr\{M \leq k_0\} < 1$ and

$Pr\{M > k_0\} = \sum_{k=k_0+1}^{n} Pr\{M = k\} < \sum_{k=k_0+1}^{n} n \cdot \frac{1}{n^3} = \sum_{k=k_0+1}^{n} \frac{1}{n^2} < n \cdot \frac{1}{n^2} = \frac{1}{n}$

Then we have $E[M] \leq k_0 + n \cdot \frac{1}{n} = O\left(\frac{\lg n}{\lg \lg n}\right)$.

* based on cslabcms.nju.edu.cn

- **Exercise 3. D-select**

  In the deterministic linear-time selection algorithm, instead of partitioning all the inputs into a number of groups of size 5, we partition the inputs into a number of groups of size 7. The rest steps of the algorithm are similar to the original version. Will the runtime of this new algorithm still be O(n), where n is the input size? Prove your claim.

  **Solution:**

Let $k = n/7$ be the number of groups. $x_i$ be the $i$-th smallest of the $k$ medians, the $x_{k/2}$ should be the pivot. Then, there are at least $\frac{4}{7} \times \frac{1}{2} = \frac{2}{7}$ elements smaller than $x_{k/2}$. Similarly, at least $\frac{2}{7}$ elements are larger than $x_{k/2}$. Then, the number of elements smaller than or larger than $x_{k/2}$ is between $\frac{2}{7}$ and $\frac{5}{7}$. There exists a positive constant $c$ such that $T(1) \leq c$ and $T(n) \leq cn + T(\frac{n}{7}) + T(\frac{5}{7}n)$. Assume there is a constant $a$ such than $T(n) \leq an$ for all $n > 1$. We choose $a = 7c$.

By induction: ① $T(1) \leq 7c$

② $T(k) \leq 7ck$, $\forall k < n$.

Then $T(n) \leq cn + T(\frac{n}{7}) + T(\frac{5}{7}n) \leq cn + cn + 5cn = 7cn$, proved.

Therefore, it is still linear time, $O(n)$.

- **Exercise 4. Silly Blue Tiger**

  1. Imagine that an algorithm requires us to hash strings containing English phrases. Knowing that strings are stored as sequences of characters, Blue Tiger decides to simply use the sum of those character values (modulo the size of its hash table) as the string's hash. Will the performance of the implementation match the expected value shown in lecture? (Yes/No and the brief reasoning are what we expect.)
  2. Blue Tiger decides to implement both collision resolution and rehashing for its hash table. However, it does not want to do more work than necessary, so it wonders if it needs both to maintain the correctness and performance it expects. After all, if it has rehashing, it can resize to avoid collisions; and if it has collision resolution, collisions don't cause correctness issues. Which statement about these two properties true? (Choose one or two of the statements and briefly explain your choice.)
  3. Blue Tiger wants to implement rehashing. Assuming it is enlarging a table of size $m$ into a table of size $m'$, and the table contains $n$ elements, what is the best time complexity to achieve this rehashing step (expanding a $m$-slot table with $n$ elements into a $m'$-slot table)? Answer in big theta notation in terms of the variables in the context. Intermediate results of your complexity analysis should be displayed.
  4. In lecture, we discussed approximately doubling the size of our hash table. Blue Tiger begins to implement this approach (that is, it lets $m'$ be a prime close to but greater than $2m$) but stops when it thinks that it might be able to avoid wasting half of the memory the table occupies on empty space by letting $m'$ be a prime number close to $(m + k)$ instead, where $k$ is some constant. Does this work equally well comparing to picking a prime number close to $2m$? Explain your answer.

**Solution:**

1. No. Different strings may have the same hash value, e.g. "post", "pots", "spot", "stop" and "tops". It will leads to a lot of collisions.

2. For rehashing, it is wrong. When two elements have the same hash value, rehashing will not work.
   For collision resolution, it is true if we use separate chaining, and there will not be correctness issue.

3. ① Create a hash table of size $m'$ $\Theta(1)$
   ② Scan the original hash table to find all the $n$ elements $\Theta(m)$
   ③ Rehash all the $n$ elements into the new hash table $\Theta(n)$
   Total time complexity: $\Theta(m+n+1) = \Theta(m+n)$

4. No. For the performance of hash table, we can use load factor to evaluate. The load factor $L = \frac{\#\text{ items}}{\text{hash table size}}$. When $m$ is large, since $k$ is a constant, for rehashing, the load factor $L$ will become small, indicating the low performance. If we choose $2m$, $L$ will not become small dramatically.

- **Exercise 5. Rehashing**

Given a sequence of inputs 4371, 1323, 6173, 4199, 4344, 9679, 1989, insert them into a hash table of size 10. Suppose that the hash function is h(x) = x mod 10. Hash table uses the double hashing with the second hash function as g(x) = (7 - x) mod 7.(Be careful with this mod calculation)
If any given input cannot be inserted correctly, do rehashing to guarantee a successful insertion, using a new table size as introduced in class (the first prime number after doubling the original hash table size) and a new hash function h(x) = x mod new_size. Note that the order in rehashing depends on the order stored in the old hash table, not their previous inserting order.
Solve this question step by step and show intermediate results.

**Solution:**

The initial hash table:

| | $T_0$ | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|---|
| $entry0$ | | | | | | | |
| $entry1$ | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 | |
| $entry2$ | | | | | | | |
| $entry3$ | | 1323 | 1323 | 1323 | 1323 | 1323 | |
| $entry4$ | | | 6173 | 6173 | 6173 | 6173 | |
| $entry5$ | | | | | | 9679 | |
| $entry6$ | | | | | | | |
| $entry7$ | | | | | 4344 | 4344 | |
| $entry8$ | | | | | | | |
| $entry9$ | | | | 4199 | 4199 | 4199 | |

Does the hash table need rehashing? If it does, when and why (answer in 1-2 sentences)? Also finish the rehashing process by drawing your own progress table that is similar to the one above:

It needs rehashing. For 1989, 1989 mod 10 = 9, occupied. (7-1989) mod 7 = 6
For double hashing, $h_1 = 5$, $h_2 = 1$, $h_3 = 7$, $h_4 = 3$, $h_5 = 9$, $h_6 = 5$. We can see that there is a loop and we can never find a slot to put 1989. So we need rehashing.
The new hash table size is 23.

| entry | $T_0$ | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | | |
| 1 | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 | 4371 |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | 6173 | 6173 | 6173 | 6173 | 6173 |
| 10 | | | | | | | |
| 11 | | | | | | | 1989 |
| 12 | | 1323 | 1323 | 1323 | 1323 | 1323 | 1323 |
| 13 | | | | | | 4199 | 4199 |
| 14 | | | | | | | |
| 15 | | | | | | | |
| 16 | | | | | | | |
| 17 | | | | | | | |
| 18 | | | | | | | |
| 19 | | | | 9679 | 9679 | 9679 | 9679 |
| 20 | | | | | 4344 | 4344 | 4344 |
| 21 | | | | | | | |
| 22 | | | | | | | |

- **Exercise 6**

  Suppose we want to design a hash table containing at most 1162 elements using quadratic probing. We require it to guarantee successful insertions, S(L) < 2 and U(L) < 4. Please determine a proper hash table size and show all intermediate steps.

$$\begin{cases} S(L)=\frac{1}{L}\ln\frac{1}{1-L}<2 \\ U(L)=\frac{1}{1-L}<4 \\ L\le 0.5 \end{cases} \Rightarrow \begin{cases} L<0.797 \\ L<\frac{3}{4} \\ L\le 0.5 \end{cases} \Rightarrow L\le 0.5$$

We can choose $\frac{1162}{n}\le 0.5 \Rightarrow n\ge 2324$.

The closet prime number is 2333.

- **Exercise 7**

  In a few sentences (in an itemized fashion), explain why a hash table with open addressing and rehashing (doubling the size every time when the load factor exceeds a threshold) achieves insertion in $O(1)$ time.

  Answer the following questions: 1) When rehashing is not triggered, is the insertion time $O(1)$ (does it have an upper bound)? Why? 2) Show that at any given point, the amortized (distributed to all items in the hash table) cost of rehashing is $O(1)$ for each item in the hash table.

For open addressing, S(L) and U(L) are constants, so it is O(1). For rehashing, we can use amortized analysis to get O(1)

(1). For successful search, we can use S(L) to express the expected number of comparisons, which is a constant. Therefore, the time complexity is O(1).

(2). Initially, we insert M elements to hash table, which needs O(M) time. For (M+1)-th, we need rehashing. Creating new hash table needs O(1). Then, We should rehash all M elements and insert the new one, which needs O(M)+O(1).
Overall, we need O(M). And for each insertion, we need O(1).

  What if during rehashing, the hash table is expanded by a factor $x$ with $x > 2$? Is the amortized cost of rehashing still $O(1)$? If no, explain. If yes, show your work. If it depends, show both.

Yes, it is still O(1).

For inserting M elements: O(M)
Create new hash table: O(1)
Rehash M elements: O(M)
Insert the (M+1)-th element: O(1)
Totally, 2O(M) + 2O(1) = O(M)
Average cost for each insertion: O(1).