DATA STRUCTURE

Chapter 9: Hashing

Problems and Solutions (Exam Oriented)

9.1 Hash Functions | 9.2 Deleting Items from Hash Tables

Formulas at a Glance

- **Division method:** h(k) = k mod m (m preferably prime).
- **Multiplication method:** h(k) = floor(m × (k·A mod 1)), A ≈ 0.6180339887.
- **Mid-square:** square the key and take the middle digits.
- **Folding:** split the key into parts, add them, ignore the carry. Boundary folding reverses alternate parts first.
- **Load factor:** $\alpha = n / m$.
- **Linear probing:** h(k, i) = (h(k) + i) mod m.
- **Quadratic probing:** h(k, i) = (h(k) + i^2) mod m (when c1 = 0, c2 = 1).
- **Double hashing:** h(k, i) = (h1(k) + i·h2(k)) mod m, h2(k) = R − (k mod R).
- **Expected probes, linear probing:** successful ≈ ½ (1 + 1/(1 − α)); unsuccessful ≈ ½ (1 + 1/$(1 - \alpha)^2$).
- **Deletion in open addressing:** mark the slot DELETED (tombstone); search skips DELETED slots and stops only at EMPTY.

Section A: Problems on 9.1 Hash Functions

Problem 1. Using the division method h(k) = k mod m with m = 13, find the hash address of the keys 45, 78, 123, 256 and 391. Identify any collision.

Solution:

- h(45) = 45 mod 13 = 6 (13 × 3 = 39, remainder 6)
- h(78) = 78 mod 13 = 0 (13 × 6 = 78, remainder 0)
- h(123) = 123 mod 13 = 6 (13 × 9 = 117, remainder 6)
- h(256) = 256 mod 13 = 9 (13 × 19 = 247, remainder 9)
- h(391) = 391 mod 13 = 1 (13 × 30 = 390, remainder 1)

Answer: the addresses are 6, 0, 6, 9 and 1. Keys **45 and 123** both hash to address 6, so a **collision** occurs and a collision resolution technique is needed.

Problem 2. The keys 20, 30, 40 and 50 are hashed using the division method. Compare the results for m = 10 and m = 11. Which is better and why?

Solution:

- For m = 10: 20 mod 10 = 0, 30 mod 10 = 0, 40 mod 10 = 0, 50 mod 10 = 0. All four keys go to address 0.
- For m = 11: 20 mod 11 = 9, 30 mod 11 = 8, 40 mod 11 = 7, 50 mod 11 = 6. All four addresses are different.

Answer: m = 11 is better. When m = 10 (having common factors with the keys), keys that are multiples of 10 all collide. A **prime table size** shares no factor with most keys, so the keys spread evenly over the table.

Problem 3. Using the multiplication method with A = 0.6180339887 and m = 10, find the hash addresses of the keys 15, 25, 35 and 45.

Solution:
Formula: h(k) = floor(m × (k·A mod 1)).

- k = 15: 15 × A = 9.2705, fractional part = 0.2705, 10 × 0.2705 = 2.705, so h = **2**.
- k = 25: 25 × A = 15.4508, fractional part = 0.4508, 10 × 0.4508 = 4.508, so h = **4**.
- k = 35: 35 × A = 21.6312, fractional part = 0.6312, 10 × 0.6312 = 6.312, so h = **6**.
- k = 45: 45 × A = 27.8115, fractional part = 0.8115, 10 × 0.8115 = 8.115, so h = **8**.

Answer: the addresses are 2, 4, 6 and 8. There is no collision.

Problem 4. Using the mid-square method, find the hash addresses (table size 100, take the middle two digits of the square) for the keys 3456, 5678 and 7891.

Solution:
- 3456^2 = 11943936. The digits are 1 1 9 **4 3** 9 3 6, so the middle two digits are 43. h = **43**.
- 5678^2 = 32239684. The digits are 3 2 2 **3 9** 6 8 4, so the middle two digits are 39. h = **39**.
- 7891^2 = 62267881. The digits are 6 2 2 **6 7** 8 8 1, so the middle two digits are 67. h = **67**.

Answer: the addresses are 43, 39 and 67. (Each square has 8 digits, so the 4th and 5th digits are the middle ones.)

Problem 5. Find the hash address of the key 987654321 for a table of size 1000 using (a) shift folding and (b) boundary folding, with 3-digit parts. Also find the address of the key 12345678 for a table of size 100 using 2-digit parts with both methods.

Solution:
Key 987654321 (parts: 987, 654, 321):

- Shift folding: 987 + 654 + 321 = 1962. Ignoring the carry (keep 3 digits), h = **962**.
- Boundary folding: reverse the middle part, 654 becomes 456. 987 + 456 + 321 = 1764. Ignoring the carry, h = **764**.

Key 12345678 (parts: 12, 34, 56, 78):

- Shift folding: 12 + 34 + 56 + 78 = 180. Keeping 2 digits, h = **80**.
- Boundary folding: reverse the 2nd and 4th parts, 34 becomes 43 and 78 becomes 87. 12 + 43 + 56 + 87 = 198. Keeping 2 digits, h = **98**.

Problem 6. Using digit extraction, where the 2nd, 4th and 5th digits (from the left) form the address, find the hash address of the keys 62719 and 83452.

Solution:
- 62719: digits are 6, 2, 7, 1, 9. The 2nd, 4th and 5th digits are 2, 1, 9, so h = **219**.

- 83452: digits are 8, 3, 4, 5, 2. The 2nd, 4th and 5th digits are 3, 5, 2, so h = **352**.

Problem 7. Compute the hash value of the strings "ABC" and "CBA" for m = 11 using (a) the sum of ASCII values and (b) the polynomial method with base 31. (ASCII: A = 65, B = 66, C = 67.) What do you conclude?

Solution:

(a) Sum of ASCII values:

- "ABC": 65 + 66 + 67 = 198, and 198 mod 11 = 0.
- "CBA": 67 + 66 + 65 = 198, and 198 mod 11 = 0.

(b) Polynomial method: $h = (s0 \times 31^2 + s1 \times 31 + s2) \bmod 11$.

- "ABC": 65 × 961 + 66 × 31 + 67 = 62465 + 2046 + 67 = 64578, and 64578 mod 11 = **8**.
- "CBA": 67 × 961 + 66 × 31 + 65 = 64387 + 2046 + 65 = 66498, and 66498 mod 11 = **3**.

Conclusion: the simple sum gives the same address for all anagrams (a collision), since the position of the character is ignored. The polynomial method gives different addresses because it depends on the position of each character, so it is better for strings.

Problem 8. Insert the keys 72, 27, 36, 24, 63, 81, 92 and 101 in this order into a hash table of size 10 with h(k) = k mod 10, using linear probing. Show the final table, the total number of probes and the load factor.

Solution:

- 72 → h = 2. Slot 2 is free. Placed at 2 (1 probe).
- 27 → h = 7. Placed at 7 (1 probe).
- 36 → h = 6. Placed at 6 (1 probe).
- 24 → h = 4. Placed at 4 (1 probe).
- 63 → h = 3. Placed at 3 (1 probe).
- 81 → h = 1. Placed at 1 (1 probe).
- 92 → h = 2. Slot 2 is occupied, 3 occupied, 4 occupied, 5 is free. Placed at **5** (4 probes).
- 101 → h = 1. Slots 1, 2, 3, 4, 5, 6, 7 are occupied, slot 8 is free. Placed at **8** (8 probes).

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Key		81	72	63	24	92	36	27	101	

Total probes = 1 + 1 + 1 + 1 + 1 + 1 + 4 + 8 = **18** (average 18 / 8 = 2.25 per insertion).

Load factor α = 8 / 10 = **0.8**.

Note the long run of occupied slots from 1 to 8. This is **primary clustering**.

Problem 9. Insert the keys 22, 33, 44, 55 and 12 in this order into a table of size 11 using quadratic probing, $h(k, i) = (k \bmod 11 + i^2) \bmod 11$.

Solution:

- 22: 22 mod 11 = 0. Slot 0 is free. Placed at **0**.
- 33: 33 mod 11 = 0, occupied. i = 1: (0 + 1) mod 11 = 1, free. Placed at **1**.
- 44: 44 mod 11 = 0, occupied. i = 1: slot 1, occupied. i = 2: (0 + 4) = 4, free. Placed at **4**.

- 55: 55 mod 11 = 0, occupied. i = 1: slot 1, occupied. i = 2: slot 4, occupied. i = 3: (0 + 9) = 9, free. Placed at **9**.
- 12: 12 mod 11 = 1, occupied. i = 1: (1 + 1) = 2, free. Placed at **2**.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Key	22	33	12		44					55	

Problem 10. A table of size 11 uses double hashing with h1(k) = k mod 11 and h2(k) = 7 − (k mod 7). Insert 58, 14, 91 and 25 in this order.

Solution:

The probe sequence is (h1(k) + i × h2(k)) mod 11.

- 58: h1 = 58 mod 11 = 3. Slot 3 is free. Placed at **3**.
- 14: h1 = 3, occupied. h2 = 7 − (14 mod 7) = 7 − 0 = 7. i = 1: (3 + 7) mod 11 = 10, free. Placed at **10**.
- 91: h1 = 91 mod 11 = 3, occupied. h2 = 7 − (91 mod 7) = 7 − 0 = 7. i = 1: slot 10, occupied. i = 2: (3 + 14) mod 11 = 6, free. Placed at **6**.
- 25: h1 = 25 mod 11 = 3, occupied. h2 = 7 − (25 mod 7) = 7 − 4 = 3. i = 1: (3 + 3) = 6, occupied. i = 2: (3 + 6) = 9, free. Placed at **9**.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Key				58			91			25	14

Each key that collided at slot 3 followed a different path because h2 depends on the key. This avoids clustering.

Problem 11. Insert the keys 50, 700, 76, 85, 92, 73 and 101 into a hash table of size 7 using separate chaining, h(k) = k mod 7 (new keys are added at the end of the chain). Find the load factor and the number of comparisons needed to search for 92.

Solution:

- 50 mod 7 = 1; 700 mod 7 = 0; 76 mod 7 = 6; 85 mod 7 = 1; 92 mod 7 = 1; 73 mod 7 = 3; 101 mod 7 = 3.

Index	**Chain**
0	700
1	50 → 85 → 92
2	(empty)
3	73 → 101
4	(empty)
5	(empty)
6	76

- **Load factor** α = 7 / 7 = **1**. In chaining, α can be 1 or more.

- **Searching 92:** h(92) = 1. Go to chain 1 and compare with 50, 85 and then 92. It needs **3 comparisons**.

Problem 12. (a) A table using open addressing must store 200 keys with a load factor not exceeding 0.5. Choose a suitable table size. (b) A table of size 5 with h(k) = k mod 5 and linear probing holds the keys 12, 25, 37 and 44. Rehash it into a table of size 11.

Solution:

(a) Load factor $\alpha = n / m \le 0.5$, so $m \ge 200 / 0.5 = 400$. Since a prime table size is preferred, choose the next prime after 400, which is **401** (it is not divisible by any prime up to 20, so it is prime).

(b) The old table (m = 5): 12 → slot 2; 25 → slot 0; 37 → slot 2 is occupied, so slot 3; 44 → slot 4. Old table: slot 0 = 25, slot 2 = 12, slot 3 = 37, slot 4 = 44 ($\alpha = 0.8$, too high, so rehash).

New table size 11 (prime, about twice the old size). Take each key from the old table and recompute h(k) = k mod 11:

- 25 mod 11 = 3, so slot 3.
- 12 mod 11 = 1, so slot 1.
- 37 mod 11 = 4, so slot 4.
- 44 mod 11 = 0, so slot 0.

Index	0	1	2	3	4	5	6	7	8	9	10
Key	44	12		25	37						

The new load factor is 4 / 11 ≈ 0.36 and there are no collisions.

Section B: Problems on 9.2 Deleting Items from Hash Tables

Problem 13. From the chained hash table of Problem 11, delete the keys 85 and 101. Show the table after deletion and the comparisons now required to search for 92.

Solution:

- Delete 85: h(85) = 1. Traverse chain 1 (50 → 85 → 92), find 85, and make the pointer of 50 point to 92. Free the node.
- Delete 101: h(101) = 3. Traverse chain 3 (73 → 101), find 101 at the end, and make the next pointer of 73 null. Free the node.

Index	Chain
0	700
1	50 → 92
2	(empty)
3	73
4	(empty)
5	(empty)
6	76

- Now n = 5, so α = 5 / 7 ≈ 0.71.
- Searching 92 needs **2 comparisons** (50, then 92), one less than before.

The other chains are not affected. Deletion in chaining is the same as deletion in a singly linked list.

Problem 14. In a table of size 10 with h(k) = k mod 10 and linear probing, the keys 15, 25, 35 and 45 are inserted. (a) Show what goes wrong if 25 is deleted by making its slot empty and then 45 is searched. (b) Show how tombstones solve the problem.

Solution:
Insertion: all four keys hash to 5. 15 goes to slot 5, 25 to slot 6, 35 to slot 7 and 45 to slot 8.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Key						15	25	35	45	

(a) Deletion by emptying the slot. Delete 25 and make slot 6 EMPTY. Now search for 45:

- h(45) = 5. Slot 5 holds 15, which is not 45. Continue.
- Slot 6 is EMPTY, so the search **stops** and reports that 45 is **not found**.
- This is wrong, since 45 is present in slot 8. The empty slot has broken the probe sequence.

(b) Deletion with a tombstone. Delete 25 and mark slot 6 as DELETED (D).

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Key						15	D	35	45	

- Search 45: slot 5 (15, no), slot 6 (DELETED, so continue), slot 7 (35, no), slot 8 (45, found). **Found in 4 probes.**

Conclusion: a deleted slot must be marked DELETED, and the search must continue past it. The search stops only at an EMPTY slot.

Problem 15. A table of size 7 uses h(k) = k mod 7 with linear probing and tombstones. Perform the following operations in order: insert 10, 17, 24, 31; delete 17; search 24; insert 38. Show the final table.

Solution:
Insertions: 10, 17, 24 and 31 all have h(k) = 3.

- 10 goes to slot 3.
- 17: slot 3 is occupied, so slot 4 is free. Placed at slot 4.
- 24: slots 3 and 4 are occupied, so slot 5. Placed at slot 5.
- 31: slots 3, 4, 5 are occupied, so slot 6. Placed at slot 6.

Delete 17: h = 3 (holds 10, no), slot 4 holds 17, found. Mark slot 4 as DELETED (D).

Search 24: slot 3 (10, no), slot 4 (DELETED, continue), slot 5 (24, found). It needs **3 probes**.

Insert 38: h(38) = 3. Slot 3 is occupied (10). Slot 4 is DELETED, so note it as the first reusable slot but continue to make sure 38 is not already present. Slot 5 holds 24 and slot 6 holds 31. Slot 0 is EMPTY, so 38 is not in the table. Now insert 38 in the noted slot, **slot 4**.

Index	0	1	2	3	4	5	6
Key				10	38	24	31

Problem 16. A table of size 10 with h(k) = k mod 10 and linear probing has the keys 15, 25, 35 and 16 inserted in this order. Delete 25 using the backward shift (cluster repair) method and show the final table.

Solution:

Insertion: 15 → slot 5; 25 → slot 6; 35 → slot 7; 16 → h = 6 is occupied, slot 7 is occupied, so slot 8.

Index	0	1	2	3	4	5	6	7	8	9
Key						15	25	35	16	

Deletion of 25: remove 25 from slot 6. This leaves a hole at slot 6. Examine the following slots until an EMPTY slot is reached. A record can be moved into the hole if its home address is at or before the hole.

- Slot 7 holds 35 (home = 5). Home 5 is before the hole (6), so move 35 to slot 6. The hole is now at slot 7.
- Slot 8 holds 16 (home = 6). Home 6 is before the hole (7), so move 16 to slot 7. The hole is now at slot 8.
- Slot 9 is EMPTY, so stop.

Index	0	1	2	3	4	5	6	7	8	9
Key						15	35	16		

Check: searching 16 goes slot 6 (35, no) and slot 7 (16, found). No tombstone is left in the table.
Note: if the record at the next slot has its home address exactly at that slot, it must **not** be moved, as that would put it before its home address and make it unreachable.

Problem 17. Compare the expected number of probes for a successful and an unsuccessful search when the load factor is 0.5 and 0.8, using linear probing. Also give the unsuccessful search cost for uniform open addressing ($1 / (1 - \alpha)$) and for chaining ($1 + \alpha$). Comment on the effect of tombstones.

Solution:

Linear probing: successful = $\frac{1}{2}(1 + 1 / (1 - \alpha))$; unsuccessful = $\frac{1}{2}(1 + 1 / (1 - \alpha)^2)$.

- $\alpha = 0.5$: successful = $\frac{1}{2}(1 + 2)$ = **1.5**; unsuccessful = $\frac{1}{2}(1 + 4)$ = **2.5**.
- $\alpha = 0.8$: successful = $\frac{1}{2}(1 + 5)$ = **3**; unsuccessful = $\frac{1}{2}(1 + 25)$ = **13**.

Other techniques (unsuccessful search):

- Uniform open addressing (ideal double hashing): $1 / (1 - \alpha) = 2$ for $\alpha = 0.5$, and 5 for $\alpha = 0.8$.
- Separate chaining: $1 + \alpha = 1.5$ for $\alpha = 0.5$, and 1.8 for $\alpha = 0.8$.

Comment: the cost of linear probing rises sharply as α approaches 1 (13 probes at $\alpha = 0.8$). Tombstones behave like occupied slots for searching, since a search cannot stop at them. So many deletions raise the effective load, and searches slow down even though the number of real records is small. The remedy is to **rehash** the table to clear the tombstones.

Section C: Practice Problems (with Answers)

Solve the following problems yourself and check your results with the answers given below.

Problem 18. Find h(k) = k mod 13 for the keys 100, 200 and 300.

> **Answer:** 9, 5 and 1 ($100 = 13 \times 7 + 9$; $200 = 13 \times 15 + 5$; $300 = 13 \times 23 + 1$).

Problem 19. Find the shift folding address of the key 234567891 for a table of size 1000 (3-digit parts).

> **Answer:** 234 + 567 + 891 = 1692; ignoring the carry gives 692.

Problem 20. Find the mid-square address of the key 94 for a table of size 100 (middle two digits of the square).

> **Answer:** $94^2 = 8836$; the middle two digits are 83, so the address is 83.

Problem 21. Insert 8, 15 and 22 into a table of size 7 with h(k) = k mod 7 using linear probing.

> **Answer:** All three hash to 1. Slots: 8 at 1, 15 at 2, 22 at 3.

Problem 22. Insert 8, 15 and 22 into a table of size 7 with h(k) = k mod 7 using quadratic probing, $(h + i^2) \bmod 7$.

> **Answer:** 8 at 1; 15 at (1 + 1) = 2; 22: slots 1 and 2 are occupied, (1 + 4) = 5, so 22 at 5.

Problem 23. Insert 8 and 15 into a table of size 7 using double hashing with h1 = k mod 7 and h2 = 5 − (k mod 5).

> **Answer:** 8 at slot 1. For 15: h1 = 1 (occupied), h2 = 5 − 0 = 5, so (1 + 5) mod 7 = 6. 15 at slot 6.

Problem 24. Insert 5, 10, 15, 3 and 8 into a table of size 5 using separate chaining with h(k) = k mod 5. Give the load factor and the longest chain.

> **Answer:** Chain 0: 5 → 10 → 15; chain 3: 3 → 8. Load factor = 5 / 5 = 1; the longest chain has length 3.

Problem 25. Table size 5, h(k) = k mod 5, linear probing with tombstones. Insert 5, 10, 15; delete 10; search 15; then insert 20. Give the probes for the search and the slot where 20 is stored.

> **Answer:** 5, 10, 15 go to slots 0, 1, 2. After deleting 10, slot 1 is DELETED. Searching 15 takes 3 probes (slots 0, 1, 2). 20 (home 0) is placed in the first DELETED slot, slot 1, after slots 2 and 3 confirm that it is absent (slot 3 is EMPTY).