DATA STRUCTURE

Chapter 9: Hashing

Problem Set 4: Model Test Paper with Solutions

9.1 Hash Functions | 9.2 Deleting Items from Hash Tables

- **Total marks:** 50 **Suggested time:** 90 minutes
- **Part A:** 10 short answer questions (2 marks each) = 20 marks
- **Part B:** 4 numerical problems (5 marks each) = 20 marks
- **Part C:** 2 long answer questions (5 marks each) = 10 marks
- Attempt the paper first without looking at the solutions, then check your answers and marks. In tables, blank means an empty slot and D means a DELETED slot (tombstone).

Part A: Short Answer Questions (2 marks each)

Problem 1. Define hashing and hash table. [2 marks]

Answer: Hashing is a technique in which a hash function converts a key into an index, so that a record can be stored and searched directly at that index in almost constant time. A **hash table** is the array (of m slots) in which the records are stored using this technique.

Problem 2. What is a collision? Name two techniques to resolve it. [2 marks]

Answer: A **collision** occurs when two different keys give the same hash address, i.e. h(k1) = h(k2) with k1 ≠ k2. Two techniques are **separate chaining** (open hashing) and **open addressing** (linear probing, quadratic probing, double hashing).

Problem 3. Define the load factor. Find it for 45 keys stored in a table of 60 slots. [2 marks]

Answer: The load factor is α = n / m, the ratio of the number of stored keys to the table size. It shows how full the table is. Here α = 45 / 60 = **0.75**.

Problem 4. Why is a prime number preferred as the table size in the division method? [2 marks]

Answer: A prime m has no common factors with most keys, so the remainders k mod m are spread evenly over the table. When m is a power of 2 or 10, only the lowest bits or digits of the key decide the address, and keys with a common pattern (for example multiples of 10) collide heavily.

Problem 5. Find the address of the key 5678 for a table of size 100 using (a) shift folding and (b) boundary folding with 2-digit parts. [2 marks]

Answer: The parts are 56 and 78. (a) Shift folding: 56 + 78 = 134, ignoring the carry gives **34**. (b) Boundary folding: reverse the second part, 78 becomes 87, and 56 + 87 = 143, ignoring the carry gives **43**.

Problem 6. Differentiate between primary clustering and secondary clustering. [2 marks]

Answer: Primary clustering occurs in linear probing: occupied slots form long runs, and any key that hashes into a run must go to its end, which makes the run longer. **Secondary clustering** occurs in quadratic probing: keys with the same home address follow the same probe sequence, so they compete for the same slots, although runs do not merge.

Problem 7. Why should a slot not simply be made empty when a key is deleted in open addressing? [2 marks]

Answer: A search stops at the first EMPTY slot. If a deleted slot is made EMPTY, the probe sequence of other keys that were placed after it (because of collisions) is broken, and a search for them wrongly reports "not found".

Problem 8. What is a tombstone? How does a search treat it? [2 marks]

Answer: A **tombstone** is a special marker (DELETED) placed in a slot when its key is deleted. A search **does not stop** at a tombstone; it continues along the probe sequence, and stops only at an EMPTY slot or when the key is found. An insertion may reuse a tombstone slot after confirming that the key is not already present.

Problem 9. State the time complexity of deletion in separate chaining. [2 marks]

Answer: The average time is **O(1 + α)**, since the chain of load factor α on average has to be traversed after computing the hash. The worst case is **O(n)**, when all keys are in one chain.

Problem 10. What is rehashing? When is it done? [2 marks]

Answer: Rehashing means creating a larger table (usually about twice the size and prime), recomputing the hash address of every live key, and inserting it in the new table. It is done when the load factor crosses a threshold, or when many tombstones have accumulated (which also clears the tombstones).

Part B: Numerical Problems (5 marks each)

Problem 11. A table of size 11 uses h(k) = k mod 11 and linear probing. Insert 31, 44, 17, 22, 55 and 28. Show the table, the total number of probes and the load factor. Also find the number of probes for an unsuccessful search of 66. [5 marks]

Solution:

- 31: 31 mod 11 = 9. Placed at slot 9 (1 probe).
- 44: 44 mod 11 = 0. Placed at slot 0 (1 probe).
- 17: 17 mod 11 = 6. Placed at slot 6 (1 probe).
- 22: 22 mod 11 = 0, occupied, so slot 1 (2 probes).
- 55: 55 mod 11 = 0, slots 0 and 1 are occupied, so slot 2 (3 probes).
- 28: 28 mod 11 = 6, occupied, so slot 7 (2 probes).

Index	0	1	2	3	4	5	6	7	8	9	10
Key	44	22	55				17	28		31	

- **Total probes** = 1 + 1 + 1 + 2 + 3 + 2 = **10**.
- **Load factor** = 6 / 11 ≈ **0.55**.
- **Search 66:** h = 0. Slots 0, 1, 2 are occupied and do not match, and slot 3 is EMPTY. So **4 probes**, not found.

Problem 12. A table of size 7 uses double hashing with h1(k) = k mod 7 and h2(k) = 1 + (k mod 5). Insert 15, 22, 29 and 36 in this order. Show the table and the total number of probes. [5 marks]

Solution:

All four keys have h1 = 1 (15, 22, 29 and 36 all leave remainder 1 when divided by 7).

- 15: placed at slot 1 (1 probe).
- 22: slot 1 is occupied. h2 = 1 + (22 mod 5) = 1 + 2 = 3. i = 1: (1 + 3) = 4. Placed at slot 4 (2 probes).
- 29: slot 1 is occupied. h2 = 1 + (29 mod 5) = 1 + 4 = 5. i = 1: (1 + 5) = 6. Placed at slot 6 (2 probes).
- 36: slot 1 is occupied. h2 = 1 + (36 mod 5) = 1 + 1 = 2. i = 1: (1 + 2) = 3. Placed at slot 3 (2 probes).

Index	0	1	2	3	4	5	6
Key		15		36	22		29

Total probes = 1 + 2 + 2 + 2 = **7**. Every key that collided at slot 1 used its own step size, so no cluster was formed.

Problem 13. A chained hash table of size 9 uses h(k) = k mod 9. Insert 12, 21, 30, 39, 4, 13, 22 and 31 (new keys are added at the end of the chain). Show the table and find the average number of comparisons for a successful search. Then delete 30 and 13 and state the comparisons needed for each deletion. [5 marks]

Solution:

- 12, 21, 30 and 39 give 12 mod 9 = 21 mod 9 = 30 mod 9 = 39 mod 9 = 3, so they go to chain 3.
- 4, 13, 22 and 31 give remainder 4, so they go to chain 4.

Index	Chain
0	(empty)
1	(empty)
2	(empty)
3	12 → 21 → 30 → 39
4	4 → 13 → 22 → 31
5	(empty)
6	(empty)
7	(empty)
8	(empty)

- **Successful search:** each chain of 4 keys needs 1, 2, 3 and 4 comparisons, i.e. 10 per chain. The total is 20 for 8 keys, so the average is 20 / 8 = **2.5** comparisons.
- **Delete 30:** chain 3, compare with 12, 21, 30. **3 comparisons.** Link 21 directly to 39.
- **Delete 13:** chain 4, compare with 4, 13. **2 comparisons.** Link 4 directly to 22.

Index	Chain
0	(empty)
1	(empty)
2	(empty)
3	12 → 21 → 39
4	4 → 22 → 31
5	(empty)
6	(empty)
7	(empty)
8	(empty)

The load factor changes from 8 / 9 to 6 / 9 ≈ 0.67.

Problem 14. A table of size 7 uses h(k) = k mod 7, linear probing and tombstones. Insert 8, 15, 22 and 29. Then delete 15, delete 22, search 29 and insert 36. Show the table after the deletions and at the end, and state the number of probes for each operation after the insertions. [5 marks]

Solution:
All the keys have h(k) = 1, so 8, 15, 22 and 29 go to slots 1, 2, 3 and 4.

Index	0	1	2	3	4	5	6
Key		8	15	22	29		

- **Delete 15:** slot 1 (8, no), slot 2 (15, found). Mark slot 2 as D. **2 probes.**
- **Delete 22:** slot 1 (8), slot 2 (D, continue), slot 3 (22, found). Mark slot 3 as D. **3 probes.**

Index	0	1	2	3	4	5	6
Key		8	D	D	29		

- **Search 29:** slot 1 (8), slot 2 (D, continue), slot 3 (D, continue), slot 4 (29, found). **4 probes.**
- **Insert 36:** h = 1. Slot 1 (8), slot 2 (D, note as reusable), slot 3 (D), slot 4 (29), slot 5 is EMPTY. So 36 is not present, and it is inserted at the first tombstone, **slot 2**. (5 probes for the check.)

Index	0	1	2	3	4	5	6
Key		8	36	D	29		

Part C: Long Answer Questions (5 marks each)

Problem 15. Explain why deletion is difficult in open addressing. Describe the two solutions with an example. [5 marks]

Solution:
In open addressing a search follows the probe sequence of the key and stops at the first EMPTY slot. If a record is removed by making its slot EMPTY, the chain of probes is cut, and keys that were placed beyond it are no longer found.

Example: table size 10, h(k) = k mod 10, linear probing. The keys 15, 25 and 35 all have home address 5 and are stored in slots 5, 6 and 7. If 25 is deleted by emptying slot 6, a search for 35 examines slot 5 (15, no), then finds slot 6 EMPTY and wrongly stops, although 35 is in slot 7.

Solution 1: lazy deletion with tombstones.

- Each slot has three states: EMPTY, OCCUPIED and DELETED.
- Deleting 25 marks slot 6 as DELETED. A search continues past DELETED slots, so 35 is found at slot 7.
- An insertion may reuse a DELETED slot, after checking that the key is not present further along.
- **Drawback:** tombstones lengthen the searches, so the table is rehashed after many deletions.

Solution 2: backward shift (cluster rehash), for linear probing.

- After removing 25, the records after the hole are examined until an EMPTY slot. A record is moved into the hole if its home address is at or before the hole.
- Here 35 (home 5) moves from slot 7 to slot 6, and slot 7 becomes EMPTY. A search for 35 then finds it at slot 6.
- **Advantage:** no tombstones remain. **Drawback:** deletion is more expensive and more complex, especially with wrap-around, and it is hard to use with quadratic probing and double hashing.

Problem 16. Compare separate chaining with open addressing, and compare linear probing, quadratic probing and double hashing. [5 marks]

Solution:

Separate chaining vs open addressing:

- **Storage:** chaining keeps colliding keys in linked lists outside the table, and needs extra memory for pointers. Open addressing keeps all keys inside the table.
- **Load factor:** in chaining α can exceed 1. In open addressing α ≤ 1, and performance falls sharply when α is close to 1.
- **Deletion:** in chaining it is simple (unlink a node). In open addressing it needs tombstones or cluster repair.
- **Cache behaviour:** open addressing uses contiguous memory and is usually faster for small keys. Chaining needs no probing sequence and is easier to implement.
- **Average cost:** chaining $O(1 + \alpha)$; open addressing depends on the probing method and α.

Linear vs quadratic vs double hashing:

- **Linear probing:** $h(k, i) = (h(k) + i) \bmod m$. Simple and cache friendly, but causes primary clustering.
- **Quadratic probing:** $h(k, i) = (h(k) + i^2) \bmod m$. Avoids primary clustering but has secondary clustering, and needs a prime m and α ≤ 0.5 to guarantee that an empty slot is found.
- **Double hashing:** $h(k, i) = (h1(k) + i \times h2(k)) \bmod m$. The step depends on the key, so clustering is the least and the behaviour is closest to ideal hashing. It needs a second hash function that is never 0.

Marks Guide and Self-Evaluation

- **Part A (2 marks):** 1 mark for a correct definition or statement, 1 mark for the example or complete reason.
- **Part B (5 marks):** about 1 mark for each correct step (hash values), 2 marks for the correct final table, and 1 mark for the probes, comparisons or the load factor.
- **Part C (5 marks):** marks are given for the correct explanation, a suitable example and a complete list of points.
- **Score of 40 or above:** the chapter is well prepared. **30 to 39:** revise the problem types you missed. **Below 30:** read the notes again and practise the problems of the earlier sets.