DATA STRUCTURE

Chapter 9: Hashing

Problem Set 5: Model Test Paper 2 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
- This is a second paper with new questions. Attempt it first without looking at the solutions. In tables, blank means an empty slot and D means a DELETED slot (tombstone).

Part A: Short Answer Questions (2 marks each)

Problem 1. What is a hash function? Write two properties of a good hash function. [2 marks]

Answer: A **hash function** h(k) converts a key k into an index in the range 0 to $m - 1$ of a hash table. A good hash function should (i) be **fast and easy to compute** and always give the same value for the same key, and (ii) **distribute the keys uniformly** over the table so that collisions are minimised.

Problem 2. Differentiate between open hashing and closed hashing. [2 marks]

Answer: Open hashing (separate chaining) stores colliding keys in linked lists outside the table, so α can exceed 1. **Closed hashing** (open addressing) stores every key inside the table itself, using a probe sequence, so α can never exceed 1.

Problem 3. Find the address of the key 200 using the division method with m = 13. [2 marks]

Answer: h(200) = 200 mod 13. Since $13 \times 15 = 195$, the remainder is **5**.

Problem 4. Find the address of the key 15 using the multiplication method with A = 0.6180339887 and m = 10. [2 marks]

Answer: $15 \times A = 9.2705$. The fractional part is 0.2705. Multiplying by m gives $10 \times 0.2705 = 2.705$, and its floor is **2**. So h(15) = 2.

Problem 5. Find the address of the key 48 by the mid-square method for a table of size 100 (middle two digits). [2 marks]

Answer: $48^2 = 2304$. The digits are 2 **3 0** 4, so the middle two digits are 30. The address is **30**.

Problem 6. Write the probe sequence formulas of quadratic probing and double hashing. [2 marks]

> **Answer: Quadratic probing:** $h(k, i) = (h(k) + c1 \cdot i + c2 \cdot i^2) \bmod m$, for example $(h(k) + i^2) \bmod m$. **Double hashing:** $h(k, i) = (h1(k) + i \cdot h2(k)) \bmod m$, where $h2(k)$ is never 0 (for example $h2(k) = R - (k \bmod R)$ with R a prime smaller than m). Here i = 0, 1, 2, ...

Problem 7. Why is the load factor of open addressing limited to 1, and what happens when it is close to 1? [2 marks]

> **Answer:** All the keys are stored inside the m slots, so $n \le m$ and $\alpha = n / m \le 1$. When α approaches 1, only a few empty slots remain, so the probe sequences become very long (a search may scan almost the whole table) and insertion may fail. Hence α is kept below about 0.5 to 0.7 in practice.

Problem 8. Write the steps for deleting a key from a chained hash table. [2 marks]

> **Answer:** (1) Compute the index i = h(k). (2) Traverse the chain at slot i to find the node with the key, keeping a pointer to the previous node. (3) If found, link the previous node to the next node (or change the head of the chain if it is the first node) and free the node. If not found, report that the key is absent.

Problem 9. Write two drawbacks of the tombstone method of deletion. [2 marks]

> **Answer:** (i) Tombstones are never removed by themselves, so after many deletions the table has few real records but few EMPTY slots, and searches get longer because they cannot stop at a tombstone. (ii) An unsuccessful search may scan the whole table. The table has to be **rehashed** from time to time to clear the tombstones.

Problem 10. What is backward shift deletion? [2 marks]

> **Answer:** It is a deletion method for linear probing that does not use tombstones. After a key is removed, the following slots up to the next EMPTY slot are examined, and every record whose home address is at or before the hole is moved back into the hole. This keeps every probe sequence unbroken and leaves no tombstones.

Part B: Numerical Problems (5 marks each)

Problem 11. A table of size 13 uses quadratic probing, $h(k, i) = (k \bmod 13 + i^2) \bmod 13$. Insert 13, 26, 39, 52 and 65. Show the table and the total number of probes. Also find the probes for an unsuccessful search of 78. [5 marks]

Solution:

All the keys have h(k) = 0. The offsets $i^2 \bmod 13$ for i = 0, 1, 2, 3, 4, 5 are 0, 1, 4, 9, 3, 12.

- 13: slot 0 (1 probe).
- 26: slot 0 is occupied. i = 1: slot 1 (2 probes).
- 39: slots 0, 1 are occupied. i = 2: slot 4 (3 probes).
- 52: slots 0, 1, 4 are occupied. i = 3: slot 9 (4 probes).
- 65: slots 0, 1, 4, 9 are occupied. i = 4: $(0 + 16) \bmod 13 = 3$. Placed at slot 3 (5 probes).

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Key	13	26		65	39					52			

- **Total probes** = 1 + 2 + 3 + 4 + 5 = **15**.
- **Search 78:** h = 0. i = 0: slot 0 (13, no). i = 1: slot 1 (26, no). i = 2: slot 4 (39, no). i = 3: slot 9 (52, no). i = 4: slot 3 (65, no). i = 5: (0 + 25) mod 13 = 12, which is EMPTY. So **6 probes**, not found.

Problem 12. A chained hash table of size 8 uses h(k) = k mod 8. Insert 10, 18, 26, 5, 13, 21 and 7 (new keys are added at the end of the chain). Show the table, the load factor and the average comparisons for a successful search. Then delete 18, 5 and 7 and give the comparisons for each deletion. [5 marks]

Solution:

- 10, 18 and 26 have remainder 2. 5, 13 and 21 have remainder 5. 7 has remainder 7.

Index	Chain
0	(empty)
1	(empty)
2	10 → 18 → 26
3	(empty)
4	(empty)
5	5 → 13 → 21
6	(empty)
7	7

- **Load factor** = 7 / 8 = **0.875**.
- **Successful search:** chain 2 needs 1 + 2 + 3 = 6, chain 5 needs 1 + 2 + 3 = 6, and chain 7 needs 1. The total is 13 for 7 keys, so the average is 13 / 7 ≈ **1.86** comparisons.
- **Delete 18:** chain 2, compare with 10, 18. **2 comparisons.** Link 10 to 26.
- **Delete 5:** chain 5, 5 is the first node. **1 comparison.** The head of the chain now points to 13.
- **Delete 7:** chain 7, the only node. **1 comparison.** The chain becomes empty.

Index	Chain
0	(empty)
1	(empty)
2	10 → 26
3	(empty)
4	(empty)
5	13 → 21
6	(empty)
7	(empty)

The load factor is now 4 / 8 = 0.5.

Problem 13. A hash table of size 13 stores three-letter strings. The hash function is h(s) = (s[0] × 9 + s[1] × 3 + s[2]) mod 13, where the values are ASCII codes (a = 97, c = 99, t = 116). Find the addresses of "cat", "act" and "tac". Compare with the simple sum of ASCII codes. [5 marks]

Solution:

- **"cat":** 99 × 9 + 97 × 3 + 116 = 891 + 291 + 116 = 1298. 1298 mod 13 = **11** (13 × 99 = 1287).
- **"act":** 97 × 9 + 99 × 3 + 116 = 873 + 297 + 116 = 1286. 1286 mod 13 = **12** (13 × 98 = 1274).
- **"tac":** 116 × 9 + 97 × 3 + 99 = 1044 + 291 + 99 = 1434. 1434 mod 13 = **4** (13 × 110 = 1430).
- **Simple sum:** all three strings have the same letters, so the sum is 99 + 97 + 116 = 312 for each, and 312 mod 13 = 0 (13 × 24 = 312). All three would collide at address 0.

Conclusion: the weighted (polynomial) hash gives three different addresses, since the position of each letter matters. The plain sum cannot separate anagrams.

Problem 14. A table of size 10 uses h(k) = k mod 10 and linear probing. Insert 12, 22, 32, 13 and 23. Delete 22 by (a) the tombstone method and (b) backward shift deletion. Show both tables and compare the probes needed to search 23 afterwards. [5 marks]

Solution:

- 12: home 2, placed at slot 2.
- 22: home 2 is occupied, so slot 3.
- 32: home 2, slots 2, 3 are occupied, so slot 4.
- 13: home 3, slots 3, 4 are occupied, so slot 5.
- 23: home 3, slots 3, 4, 5 are occupied, so slot 6.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Key			12	22	32	13	23			

(a) Tombstone method: slot 3 (22) is marked D.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Key			12	D	32	13	23			

Search 23: h = 3. Slot 3 (D, continue), slot 4 (32), slot 5 (13), slot 6 (23, found). **4 probes.**

(b) Backward shift deletion: remove 22, so the hole is at slot 3. The homes are 32 → 2, 13 → 3, 23 → 3.

- Slot 4 holds 32 (home 2, before the hole). Move it to slot 3. The hole is now at slot 4.
- Slot 5 holds 13 (home 3, before the hole). Move it to slot 4. The hole is now at slot 5.
- Slot 6 holds 23 (home 3, before the hole). Move it to slot 5. The hole is now at slot 6.
- Slot 7 is EMPTY, so stop.

Index	0	1	2	3	4	5	6	7	8	9
Key			12	32	13	23				

Search 23: h = 3. Slot 3 (32), slot 4 (13), slot 5 (23, found). **3 probes.**

Comparison: backward shift needs 3 probes and leaves no dead slot, whereas the tombstone method needs 4 probes (before the deletion it was also 4). The tombstone deletion itself is faster to perform, but it leaves the table with a permanent dead slot.

Part C: Long Answer Questions (5 marks each)

Problem 15. Explain the division, multiplication, mid-square and folding methods of hash functions with one example each. [5 marks]

Solution:

- **Division method:** h(k) = k mod m. The remainder on dividing by the table size is the address. m should be a prime not close to a power of 2. **Example:** k = 100, m = 11, h = 100 mod 11 = 1.
- **Multiplication method:** h(k) = floor(m × (k·A mod 1)), with 0 < A < 1 (Knuth suggests A ≈ 0.6180339887). The fractional part of k·A is scaled by m. The value of m is not critical. **Example:** k = 123, m = 100: 123 × A = 76.0182, fractional part 0.0182, h = floor(1.82) = 1.
- **Mid-square method:** the key is squared and the middle digits are used as the address. All digits of the key affect the result. **Example:** k = 4567, k^2 = 20857489, middle two digits = 57, h = 57 (table size 100).
- **Folding method:** the key is split into equal parts, which are added, and the carry is ignored. In shift folding the parts are added directly; in boundary folding alternate parts are reversed first. **Example:** k = 123456789, 3-digit parts: shift folding 123 + 456 + 789 = 1368, h = 368; boundary folding 123 + 654 + 789 = 1566, h = 566 (table size 1000).

Problem 16. Describe the search, insert and delete operations of an open addressing table that uses tombstones. Give their steps and mention the drawbacks. [5 marks]

Solution:

Every slot is in one of three states: **EMPTY**, **OCCUPIED** or **DELETED**. Let the probe sequence be s(k, 0), s(k, 1), s(k, 2), ... for the key k.

Search(k):

- Examine the slots s(k, i) for i = 0, 1, 2, ... up to m probes.
- If the slot is EMPTY, stop and report **not found**.
- If the slot is OCCUPIED and holds k, report **found**.
- If the slot is DELETED (or holds another key), continue with the next probe.

Delete(k):

- Search for k as above.
- If found, mark the slot as DELETED (do not make it EMPTY). If not found, report that the key is absent.

Insert(k):

- Follow the probe sequence, and note the first DELETED slot met (if any). Continue until an EMPTY slot or the key itself is found.
- If k is found, it is already present, so do nothing.
- Otherwise, insert k in the first DELETED slot noted, or in the EMPTY slot if no DELETED slot was met.

Drawbacks:

- DELETED slots cannot end a search, so they behave like occupied slots, and the effective load factor is higher than the real one.
- After many deletions, unsuccessful searches become slow, and the table must be **rehashed** to clear the tombstones.
- Each search or insertion may have to examine extra slots.

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 or answer, and 1 mark for the probes, comparisons or comment.
- **Part C (5 marks):** marks are given for correct explanation, suitable examples and complete 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 earlier sets.