Hashing Homework - Strategies and Collision Resolution

Part A: Conceptual Questions

Define Hashing and Collision Resolution

1. What is a hash table and why is collision resolution necessary?
The hash table functions as a data structure containing key-value pairs while using hash functions for determining index placement. Multiple keys can collide on the same index so collision resolution must be implemented in this case.

2. Key differences between open hashing (separate chaining) and closed hashing (open addressing)
The Open Hashing method (Separate Chaining) enables boundless values to use the same index in the hash table by creating linked lists for each index to store matching hash results.
The whole hash table serves as storage for all elements under the Closed Hashing access method. The probing sequence searches for a suitable open slot after a collision occurs.

.

Collision Resolution Techniques

1. Two methods for resolving collisions in open addressing:
- Linear Probing: If a collision occurs, check the next available slot sequentially.
- Quadratic Probing: If a collision occurs, check the next available slot using a quadratic function (i.e., 1, 4, 9, 16, etc.).

2. Pros and Cons of Each Method:
- Linear Probing:
  - Pros: Simple and easy to implement.
  - Cons: Leads to primary clustering, where a sequence of filled slots forms, making future insertions slower.
- Quadratic Probing:
  - Pros: Reduces clustering compared to linear probing.
  - Cons: May not find an empty slot if the table size is not a prime number.

3. Which collision resolution technique can handle more values than table slots?
- Separate chaining (open hashing) can handle more values than there are slots because each slot can hold multiple elements using a linked list.

4. Worst-case performance (Big-O) for each technique:
- Separate chaining: \(O(n) \) (if all elements hash to the same slot, forming a long linked list).
- Open addressing: \(O(n) \) (if the table is nearly full, probing could take linear time).

Impact of Hash Table Size

1. How does table size affect key distribution?
   - A poor choice of table size (such as a power of 2) increases collisions and clustering, while a prime table size improves uniform key distribution.

2. Pitfalls of using a poor table size:
   - If the table size is a multiple of common factors in key distributions, it increases collisions (e.g., using size 10 with keys 10, 20, 30... all hashing to the same index).

Part B: Simulation and Diagram Exercises

Exercise 1: Open Hashing (Separate Chaining)

Given Keys:
5, 22, 17, 18, 35, 101, 16, 0, 8

Hash Function:
h(k) = k mod 10

Index	Keys Stored (Chained List)
0	0, 101
1	
2	22
3	
4	
5	5, 35
6	16
7	17
8	8
9	18

Exercise 2: Closed Hashing (Open Addressing with Linear Probing)

Index	Key Stored
0	0
1	101
2	22
3	17
4	18
5	5
6	35
7	16
8	8
9	

Clustering Observations:
Linear probing causes clustering where consecutive slots are occupied, reducing efficiency.

Exercise 3: Impact of Poor Table Sizes

Case 1: Table Size = 10 (Poor Choice)

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

Case 2: Table Size = 11 (Prime Number)

Index	Key Stored
0	
1	
2	
3	
4	
5	5
6	15
7	25
8	35
9	
10	

Comparison:
- Table size 10 caused excessive collisions.
- Table size 11 distributed keys more evenly, reducing clustering.

Conclusion:
Choosing a prime number as the hash table size improves key distribution and reduces collisions significantly.