Advanced Data Structure and Algorithms Lab

Course Code: JCP7181

Assignment 1: IIT Delhi Student Directory System

Name: Himanshu Singh

Entry Number: 2025JCS2062

1. Introduction

The objective of this assignment is to implement a hash table-based student directory system for IIT Delhi. The system should be efficient in handling insertions, deletions, and searches for student records. Three collision resolution techniques are implemented:
1. Linear Probing
2. Quadratic Probing
3. Separate Chaining

2. Implementation Details

These programs are menu based programs so they are to be run on the terminal and operations can be chosen from the menu and records can be entered or other operations can be performed .

2.1 Linear Probing

Linear probing resolves collisions by checking the next available slot in a sequential manner. If the index calculated by the hash function is occupied, the search proceeds to index h+1, h+2, and so on, wrapping around when necessary. Lazy deletion is used to improve efficiency. Advantages: Simple implementation and good cache performance when load factor is low.
Disadvantages: Suffers from primary clustering, which increases collisions.

2.2 Quadratic Probing

Quadratic probing resolves collisions by checking indices $h+1^2$, $h+2^2$, $h+3^2$, etc. It reduces primary clustering compared to linear probing. However, it can suffer from secondary clustering and may enter a loop without finding an empty slot unless the table size is chosen carefully (preferably a prime number).

2.3 Separate Chaining

Separate chaining resolves collisions by maintaining a linked list (or vector) of records for each hash index. All elements mapping to the same index are stored in the list for that index.
Advantages: No clustering issues and table never gets 'full'.
Disadvantages: Extra memory overhead for pointers and potential performance hit if chains get long.

3. Comparison of Techniques

When comparing the three techniques:
- Linear Probing: Best for small load factors, fastest average access when not clustered.
- Quadratic Probing: Better than linear probing in avoiding primary clustering.
- Separate Chaining: Scales well with high load factor but requires extra memory.

4. Load Factor

Load factor (α) = Number of elements in table / Table size.
A low load factor reduces the likelihood of collisions, while a high load factor can degrade performance. For open addressing methods (linear and quadratic probing), it's recommended to keep α below 0.7.

5. Test Cases

Example test case for evaluating performance:
Insert records:
- alice@example.com Alice
- bob@example.com Bob
- charlie@example.com Charlie

Search for 'bob@example.com' → Found
Delete 'alice@example.com'
Search for 'alice@example.com' → Not Found
Print table → Displays all remaining records

6. Conclusion

In conclusion, each collision resolution strategy has its own strengths and weaknesses. Linear probing is simple and efficient at low load factors, quadratic probing reduces primary clustering, and separate chaining provides flexibility at higher load factors. The choice of method depends on the expected load and memory constraints.

Test Case Set

1. Insert without collision

1

3

a@iitd.ac.in Alice

b@iitd.ac.in Bob

c@iitd.ac.in Carol

Expected Output (example for LP/QP)

Number of collisions occurred while entering the record 0

Number of collisions occurred while entering the record 0

Number of collisions occurred while entering the record 0

---

2. Collision Handling

(Emails chosen so that they map to the same index in hash function)

1

2

ab@iitd.ac.in Adam

ba@iitd.ac.in Ben

Expected Output (LP)

Number of collisions occurred while entering the record 0

Number of collisions occurred while entering the record 1

Expected Output (QP)

Number of collisions occurred while entering the record 0

Number of collisions occurred while entering the record 1

Expected Output (SC)

Chain size after insertion 1

Chain size after insertion 2

---

3. Search for existing & non-existing records

2

a@iitd.ac.in

2

x@iitd.ac.in

Expected Output:

Found : a@iitd.ac.in Alice

Not Found

---

4. Update an existing record

1

1

a@iitd.ac.in Alicia

Expected Output:

Number of collisions occurred while entering the record 0

(*Now searching for a@iitd.ac.in should return Alicia*)

---

5. Lazy Deletion

4

b@iitd.ac.in

2

b@iitd.ac.in

Expected Output:

Deleted Successfully

Not Found

---

6. Re-insertion into deleted slot

1

1

bb@iitd.ac.in Benny

Expected Output (LP/QP)

Number of collisions occurred while entering the record 1

---

7. Table Full (LP/QP only)

Insert until capacity is reached (n=199 for LP/QP).
Expected final insert should say:

Table full can't enter the record

or for QP loop:

Table full — cannot insert record.

---

8. Multiple records in same chain (SC only)

Insert multiple keys that hash to the same index.
Expected Output:
Chain sizes increasing at that index:

Chain size after insertion 1

Chain size after insertion 2

Chain size after insertion 3