Assignment-2

Data Structures and Algorithms

Name : Adithya Rao Kalathur

Reg No.: 251100610012

Branch: M.E. Computer Science and Engineering (M.E.CSE)

Learner ID: adithya.msismpl2025@learner.manipal.edu

Signature : Adithya Rao Kalathur

Question 1-

A hash function h defined h(key) = key mod 6, by using Linear Probing insert the keys into a hash table indexed from 0 to 5 in the given order - 20, 10, 31, 19, 25, 28. What will be the location of key 19 & 25 in the hash table?

Note: Insert the keys as per the given order.

Try the same table for h(key) = key mod 7, use both Linear Probing & separate Chaining.

Answers -

Hash Function Implementation

→ Part A: h(key) = key mod 6 with Linear Probing

Hash Table Size: 0 to 5 (6 slots)

Insertion Process:

P.T.O.

Key	Hash value	collision?	Final Position	Explanation
20	20 mod 6 = 2	NO	2	Direct insertion
10	10 mod 6 = 4	NO	4	Direct insertion
31	31 mod 6 = 1	NO	1	Direct insertion
19	19 mod 6 = 1	yes (31 at 1)	3	Linear probe: 1 → 2 (occupied) → 3 (free)
25	25 mod 6 = 1	yes	5	Linear probe: 1→2→3→4 (all occupied) → 5 (free)
28	28 mod 6 = 4	yes	0	Linear probe: 4 → 5 (occupied) → 0 (free, wraps around)

Final Hash Table:

Index:	0	1	2	3	4	5
value:	28	31	20	19	10	25

Answer:

→ Key 19 is at location 3

→ Key 25 is at location 5

→ Part B: h(key) = key mod 7
with Linear Probing (Table Size 0 - 6)

Key	Hash Value	collision?	Final Position	Explanation
20	20 mod 7 = 6	No	6	Direct insertion
10	10 mod 7 = 3	No	3	Direct insertion
31	31 mod 7 = 3	yes (10 at 3)	4	Linear probe: 3 → 4 (free)
19	19 mod 7 = 5	NO	5	Direct insertion
25	25 mod 7 = 4	yes (31 at 4)	0	Linear probe: 4 → 5 (occupied) → 6 (occupied) → 0 (free)
28	28 mod 7 = 0	yes (25 at 0)	1	Linear probe: 0 → 1 (free)

Final Hash Table (Linear Probing):

Index:	0	1	2	3	4	5	6
Value:	25	28	-	10	31	19	20

→ With Separate Chaining (Table Size: 0-6)

Each index contains a linked list of elements that hash to that position.

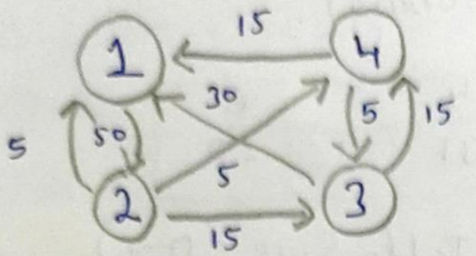
Key	Hash value	Action
20	20 mod 7 = 6	Insert at index 6
10	10 mod 7 = 3	Insert at index 3
31	31 mod 7 = 3	Append to chain at index 3
19	19 mod 7 = 5	Insert at index 5
25	25 mod 7 = 4	Insert at index 4
28	28 mod 7 = 0	~~Index a~~ Insert at index 0

Final Hash Table (Separate Chaining):

Index 0: 28 → Null
Index 1: Null
Index 2: Null
Index 3: 10 → 31 → Null
Index 4: 25 → Null
Index 5: 19 → Null
Index 6: 20 → Null

<u>Question 2</u> -

Obtain Matrix A with shortest paths using all pairs shortest path, for the given Graph:



Answers -

All pairs Shortest Path (Floyd - Warshall Algorithm)

Given Graph Analysis:

→ vertices: 1, 2, 3, 4

→ Edges with weights:

- ~~1→2: 50, 1→4: 15, 1→2: 5~~
- ~~2→1: 50, 2→3: 15~~
- ~~3→2: 15, 3→4: 5~~
- ~~4→1: 15, 4→3: 5, 4→3: 15~~
- ~~1→3: 30, 1→4: 5 (there~~

- 1→2: 50
- 2→1: 5
- 3→1: 30
- 4→1: 15
- 4→3: 5
- 3→4: 15
- 2→3: 15
- 2→4: 5

→ Initial distance matrix (no intermediates)

using ∞ for no direct edge. Rows = source i, columns = destination j.

Initial D (k = 0)

	1	2	3	4
1	0	50	∞	∞
2	5	0	15	5
3	30	∞	0	15
4.	15	∞	5	0

Floyd - Warshall iterations

We update $D[i][j] = \min(D[i][j], D[i][k] + D[k][j])$ for $k = 1..4$.

→ After $k=1$ (allow intermediate node 1)

Check paths that use node 1 in the middle:

- $D[4][2]$ via $4 \to 1 \to 2$: $15+50 = 65$ (was ∞) → set $D[4][2] = 6$
- $D[3][2]$ via $3 \to 1 \to 2$: $30+50 = 80$ (was ∞) → set $D[3][2] = 80$.

Matrix after $k=1$

After $k = 1$

	1	2	3	4
1	0	50	∞	∞
2	5	0	15	5
3	30	80	0	15
4	15	65	5	0

→ After $k=2$ (allow nodes $\{1,2\}$)

use node 2 as intermediate:

- $D[1][3]$ via $1 \to 2 \to 3$: $50+15 = 65$ (was ∞) → set $D[1][3] = 65$
- $D[1][4]$ via $1 \to 2 \to 4$: $50+5 = 55$ (was ∞) → set $D[1][4] = 55$

Matrix after $k=2$:

After $k=2$

	1	2	3	4
1	0	50	65	55
2	5	0	15	5
3	30	80	0	15
4	15	65	5	0

→ After $k=3$ (allow nodes $\{1,2,3\}$)

use node 3 as intermediate. check possible improvements:

- Almost all candidate paths through 3 are not better than existing values. No entry improves (e.g., 2→1 via 2→3→1 = 15+30 = 45 but 2→1 already 5, so no change).

Matrix after k = 3 (unchanged):

After k = 3

	1	2	3	4
1	0	50	65	55
2	5	0	15	5
3	30	80	0	15
4	15	65	5	0

→ After k = 4 (allow nodes {1, 2, 3, 4} - final relaxations use node 4 as intermediate:

- D[1][3] via 1→4→3 : D[1][4] + D[4][3] = 55 + 5 = 60 (was 65) → improve D[1][3] = 60.
- D[2][3] via 2→4→3 : D[2][4] + D[4][3] = 5+5 = 10 (was 15) → improve D[2][3] = 10
- Many other candidate updates are not improvements (e.g., 3→1 via 3→4→1 = 15+15 = 30 equals existing 30, so no change).

Final matrix after k = 4.

After k = 4 (Final)

	1	2	3	4
1	0	50	60	55
2	5	0	10	5
3	30	80	0	15
4	15	65	5	0

P.T.O.

→ Final All-pair shortest-path Matrix A

$$A = \begin{array}{c} \begin{bmatrix} 0 & 50 & 60 & 55 \\ 5 & 0 & 10 & 5 \\ 30 & 80 & 0 & 15 \\ 15 & 65 & 5 & 0 \end{bmatrix} \end{array} \begin{array}{l} \rightarrow \text{node } 1 \\ \rightarrow \text{node } 2 \\ \rightarrow \text{node } 3 \\ \rightarrow \text{node } 4 \end{array}$$

From\To	1	2	3	4
1	0	50	60	55
2	5	0	10	5
3	30	80	0	15
4	13	65	5	0

→ Shortest routes (interpretation)

- 1→3 = 60 via 1→4→3 (55+5)
  There was no direct 1→3 edge; the best route uses node 4.
- 1→4 = 55 via 1→2→4 (50+5).
  (There is no direct 1→4 edge).
- 2→3 = 10 via 2→4→3 (5+5) - better than direct 2→3 (15).
- 3→1 = 30 (direct edge 3→1).
- 3→2 = 80 via 3→1→2 (30+50) - there's no shorter route.
- 4→2 = 65 via 4→1→2 (15+50) - no shorter path exists.
- 4→3 = 5 (direct).