Overflow handling – Open addressing

Open addressing

- **Collision Handling**: Since a hash function gets us a small number for a big key, there is possibility that two keys result in same value.

- The situation where a newly inserted key maps to an already occupied slot in hash table is called collision and must be handled using some collision handling technique.

- Following are the ways to handle collisions:

- 1.Open Addressing

- 2.Chaining

- **Open Addressing**
  Like separate chaining, open addressing is a method for handling collisions.

- In Open Addressing, all elements are stored in the hash table itself.

- So at any point, size of the table must be greater than or equal to the total number of keys

- Insert(k): Keep probing until an empty slot is found. Once an empty slot is found, insert k.

- Search(k): Keep probing until slot's key doesn't become equal to k or an empty slot is reached.

- Delete(k): If we simply delete a key, then search may fail. So slots of deleted keys are marked specially as "deleted".

- Insert can insert an item in a deleted slot, but the search doesn't stop at a deleted slot.

- Open Addressing is done following ways:

- *a) Linear Probing:* In linear probing, we linearly probe for next slot.

- For example, typical gap between two probes is 1 as taken in below example also.

- let **hash(x)** be the slot index computed using hash function and **S** be the table size

Steps:

- If slot hash(x) % S is full, then we try (hash(x) + 1) % S

- If (hash(x) + 1) % S is also full, then we try (hash(x) + 2) % S

- If (hash(x) + 2) % S is also full, then we try (hash(x) + 3) % S

Quadratic Probing :

- let hash(x) be the slot index computed using hash function.

- If slot hash(x) % S is full, then we try (hash(x) + 1*1) % S

- If (hash(x) + 1*1) % S is also full, then we try (hash(x) + 2*2) % S

- If (hash(x) + 2*2) % S is also full, then we try (hash(x) + 3*3) % S