Open Addressing: Linear Probing

- Why not use up the empty space in the table?
- Store directly in the array cell (no linked list)
- How to deal with collisions?
- If **`h(key)`** is already full,
  - try **`(h(key) + 1) % TableSize`**. If full,
  - try **`(h(key) + 2) % TableSize`**. If full,
  - try **`(h(key) + 3) % TableSize`**. If full…
- Example: insert 38, 19, 8, 109, 10

0	
1	
2	
3	
4	
5	
6	
7	
8	38
9	

Questions: Open Addressing: Linear Probing

How should **`find`** work? If value is in table? If not there?

Worst case scenario for **`find`**?

How should we implement **`delete`**?

How does **open addressing with linear probing** compare to **separate chaining**?

ith probe: `(h(key) + i²) % TableSize`

Quadratic Probing Example

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

TableSize=10
Insert:
89
18
49
58
79

ith probe: `(h(key) + i*g(key)) % TableSize`

Open Addressing: Double Hashing

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

T = 10 (TableSize)
Hash Functions:
h(key) = key mod T
g(key) = 1 + ((key/T) mod (T-1))

Insert these values into the hash table in this order. Resolve any collisions with double hashing:
13
28
33
147
43