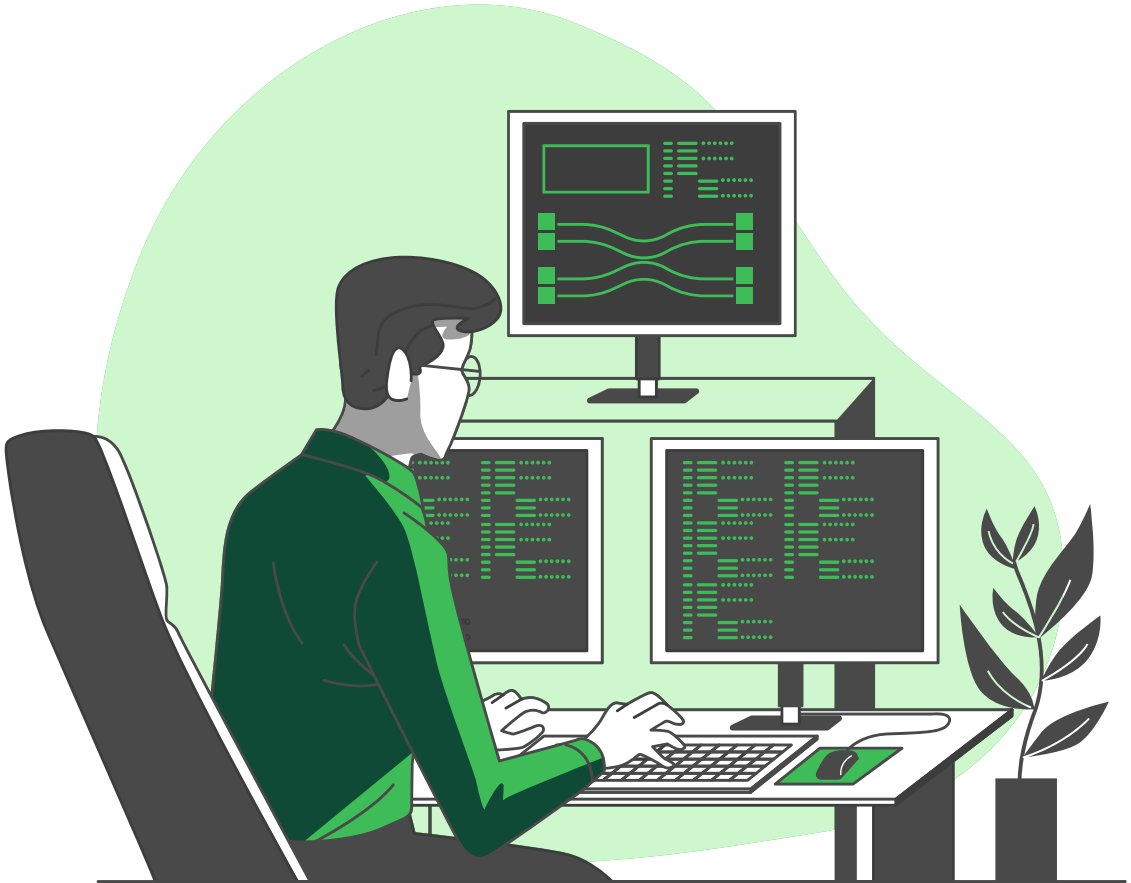


# Hash Table with Open Addressing

## CS 251 - Data Structures and Algorithms

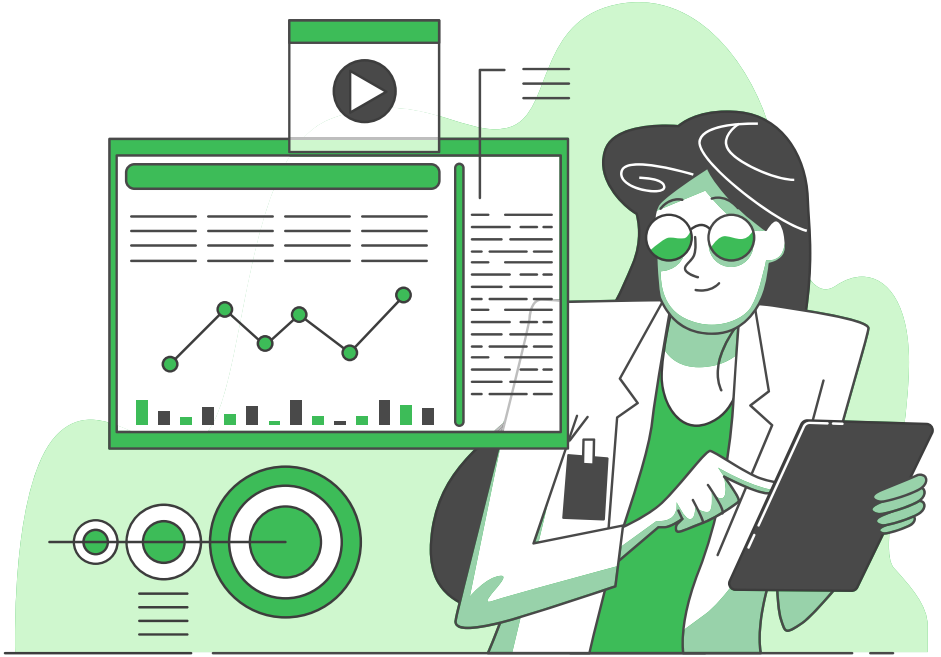
# Note:
# Slides complement the discussion in class

# Table of Contents

01

## Open Addressing

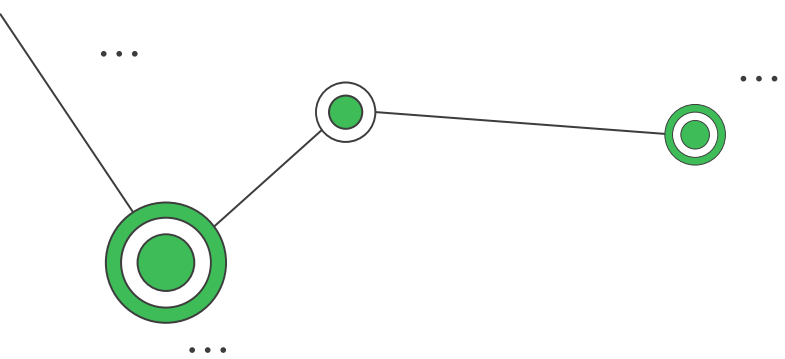
Store items directly in the table



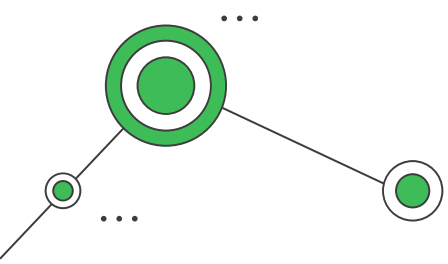
# 01

# Open Addressing

Store items directly in the table



# Remember: Collisions



Let $k_1$ and $k_2$ be two different keys. There is a collision in the hash table if $h(k_1) = h(k_2)$.

**Ideal**: Design a collision-free hash function.

**Reality**: Hard to design a $h(k)$ that creates random slot indices from non-random keys. **Assume collisions will occur**.

**Solution**: Implement **collision management strategies**.

# Collision Management Strategies

## 01

### Chaining

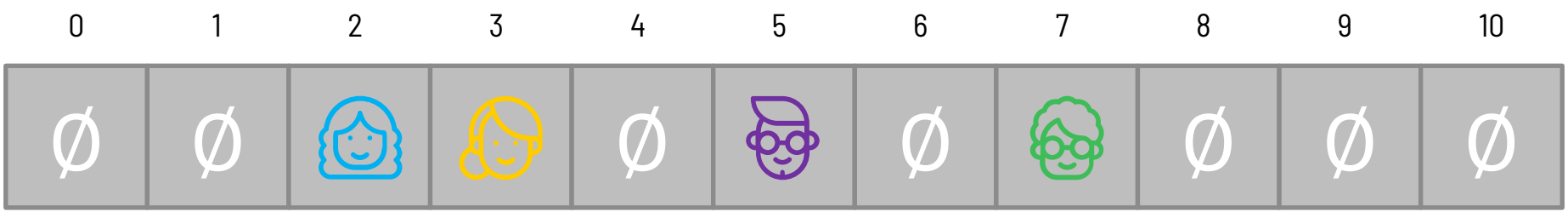
Multiple items in a slot? Store them in a Doubly Linked List.

## 02

### Open Addressing

Is the slot occupied? Search for the next one available.

# Open Addressing Idea




**Idea**: Search the hash table for an empty slot.

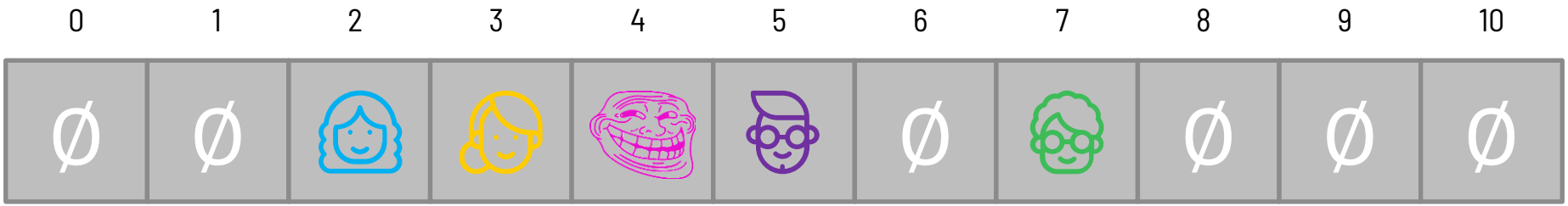
**How?**

- Try $h(k) \bmod m$.
- Taken? Then try $(h(k) + 1) \bmod m$.
- Taken? Then try $(h(k) + 2) \bmod m$.
- Repeat until you find an empty slot.

Insert(  ):

$13 \bmod 11 = 2$, it is taken.
$14 \bmod 11 = 3$, it is taken.
$15 \bmod 11 = 4$. It is available.

# Open Addressing Idea



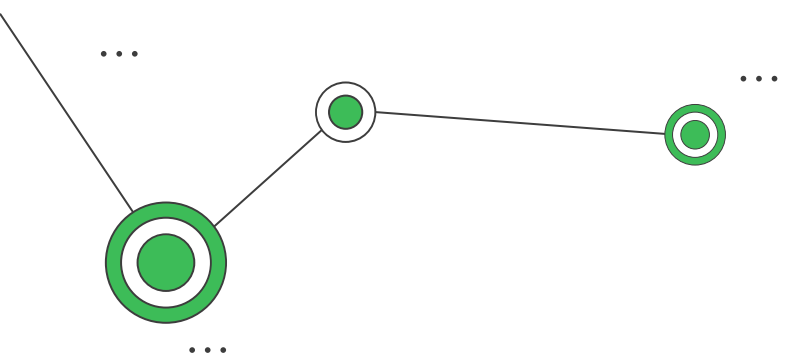

**Idea**: Search the hash table for an empty slot.

**How?**

- Try $h(k) \bmod m$.
- Taken? Then try $(h(k) + 1) \bmod m$.
- Taken? Then try $(h(k) + 2) \bmod m$.
- Repeat until you find an empty slot.

Insert(  ):

$13 \bmod 11 = 2$, it is taken.
$14 \bmod 11 = 3$, it is taken.
$15 \bmod 11 = 4$. It is available. **Solved!**



# Open Addressing: Probing

$h: U \times \{0,1, \ldots m-1\} \rightarrow \{0,1, \ldots, m-1\}$

**Probe sequence:**

$h(k,0), h(k,1), \ldots, h(k,m-1)$

**Linear Probing:**

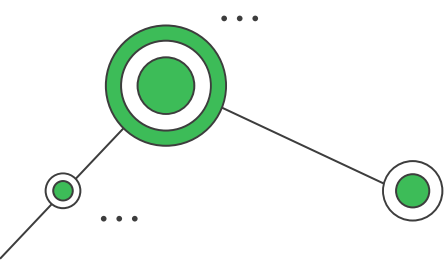
$h(k,i) = (h'(k) + i) \bmod m$

**Quadratic Probing:**

$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$

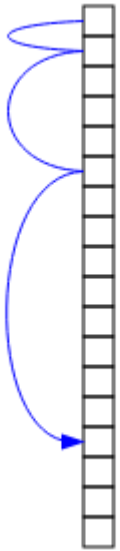
**Double Hashing:**

$h(k,i) = (h_1(k) + i h_2(k)) \bmod m$

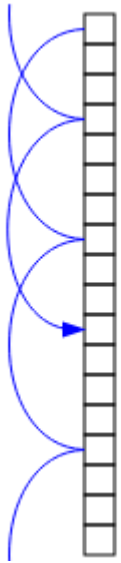


# Probing Patterns

Linear Probing



Quadratic Probing



Double Hashing

Images from https://programming.guide/hash-tables-open-addressing.html

# Example: Quadratic Probing

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| ∅ | ∅ |  |  | ∅ |  | ∅ |  | ∅ | ∅ | ∅ |

**Idea**: Search the hash table for an empty slot.

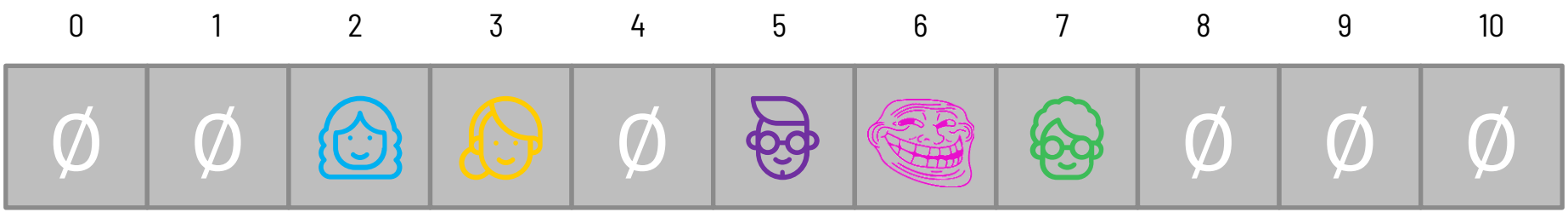
**How?**

- Try $h(k) \bmod m$.
- Taken? Then try $(h(k) + 1^2) \bmod m$.
- Taken? Then try $(h(k) + 2^2) \bmod m$.
- Repeat until you find an empty slot.

Insert(  ):

$13 \bmod 11 = 2$, it is taken.
$14 \bmod 11 = 3$, it is taken.
$17 \bmod 11 = 6$. It is available.

# Example: Quadratic Probing




**Idea**: Search the hash table for an empty slot.

**How?**

- Try $h(k) \bmod m$.
- Taken? Then try $(h(k) + 1^2) \bmod m$.
- Taken? Then try $(h(k) + 2^2) \bmod m$.
- Repeat until you find an empty slot.

Insert(  ):

$13 \bmod 11 = 2$, it is taken.
$14 \bmod 11 = 3$, it is taken.
$17 \bmod 11 = 6$. It is available. **Solved!**

# Open Addressing Algorithms

```
algorithm Insert(T:array, x:item)
let m be the capacity of T
i ← 0

repeat
   j ← h(x.key, i)

   if T[j] is null then
      T[j] ← x
      return j
   end if

   i ← i + 1
until i = m

end algorithm
error: “table overflow”
```

```
algorithm Search(T:array, key:ℤ)
let m be the capacity of T
i ← 0

repeat
   j ← h(key, i)

   if T[j] is not null and T[j].key = key then
      return j
   end if

   i ← i + 1
until T[j] is null or i = m

return -1
end algorithm
```

```
algorithm Delete(T:array, x:item)
j ← Search(T, x.key)

if j ≠ -1 then
   T[j] ← null
end if

end algorithm
```

That’s it?

An item x has attributes x.key (integer), and x.data.

# Delete when using Open Addressing

**Do not delete! Do Something else.**

# Why Not? Let's Delete Something

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| ∅ | ∅ |  |  |  |  | ∅ |  | ∅ | ∅ | ∅ |

Delete(  ):

$25 \bmod 11 = 3$, and the key of T[3] is 25, so, delete!

# Why Not? Let's Delete Something

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| ∅ | ∅ |  | ∅ |  |  | ∅ |  | ∅ | ∅ | ∅ |

Now Search(  ):

$13 \bmod 11 = 2$, but the key of T[2] is 2.
$14 \bmod 11 = 3$, but T[3] is null. So...

Is  in the hash table?! **IT IS!** But we cannot reach it now.

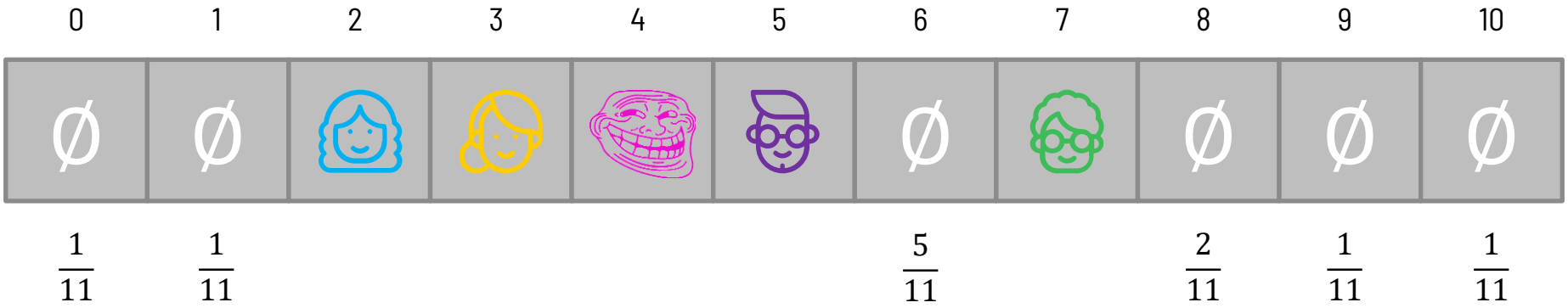
**Solutions:**
1) Label item as deleted.
2) Use a dummy object (aka. Tombstone).

Remember to update the Insert, Search, and Delete functions adequately.

# Issue: Clustering!

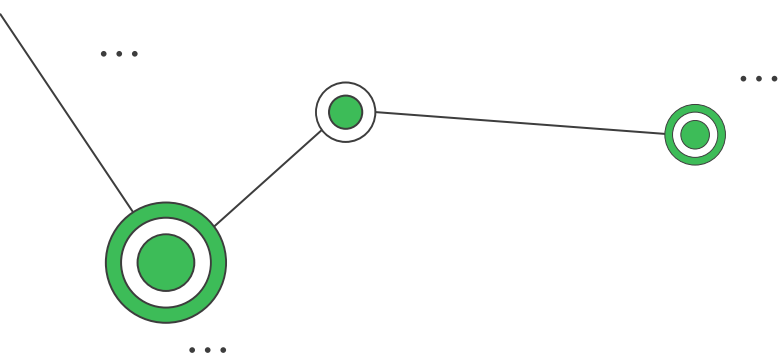
**Occupying empty slots breaks uniformity**

# Issue: Clustering

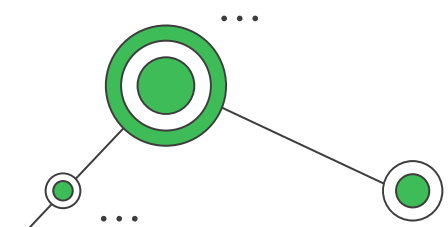



Also, longer search times if falling on long clusters.

**Goal:** Adjust $m$ such that $\alpha = \frac{n}{m}$ is less than some threshold.



# Open Addressing: Analysis



**Load Factor:** $\alpha = \frac{n}{m} < 1$.

- **Uniform Hashing Assumption:** the probe sequence of each key is equally likely to be any of the $m!$ permutations of $\{0,1,\ldots,m-1\}$.
- An insert requires at most $1/(1-\alpha)$ probes.
- A successful search requires at most $\frac{1}{\alpha}\ln\left(1/(1-\alpha)\right)$ probes.
- An unsuccessful search requires at most $1/(1-\alpha)$ expected probes.

Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to Algorithms (The MIT Press)(p. 298-299). The MIT Press.

# Table Resizing

## 01

### Chaining

Goal: $\alpha$ constant

Double $m$ when $\alpha \geq 8$

Halve $m$ when $\alpha \leq 2$

## 02

### Open Addressing

Goal: $\alpha \leq \frac{1}{2}$

Double $m$ when $\alpha \geq \frac{1}{2}$

Halve $m$ when $\alpha \leq \frac{1}{8}$

**Note:** Do not forget to rehash the values after resizing the table.

# Rehashing Example

```
      0       1   2      3           4
+-----------+---+---+----------+-----------+
| (5, dolor) |   |   | (3, sit) | (4, amet) |
+-----------+---+---+----------+-----------+
```

Load factor: 3/5 = 0.6. Resize required.

New table capacity set to 11.

Tuple (5, dolor) is now at index 5.
Tuple (3, sit) is now at index 3.
Tuple (4, amet) is now at index 4.

```
  0   1   2      3           4            5       6   7   8   9   10
+---+---+---+----------+-----------+------------+---+---+---+---+---+
|   |   |   | (3, sit) | (4, amet) | (5, dolor) |   |   |   |   |   |
+---+---+---+----------+-----------+------------+---+---+---+---+---+
```

# Hash Table Limitations

## Range Queries

Previous($k$)?
Next($k$)?
Between($k_1$,$k_2$)?
MaxKey()?
MinKey()?

## Memory Consumption

We need extra capacity than the number of keys we have.

## Real Data

Keys may not be independent.
There could be bias.

# $h(\text{last slide}) = \text{End}$

Do you have any questions?