# Linear Probing

## Outline for Today

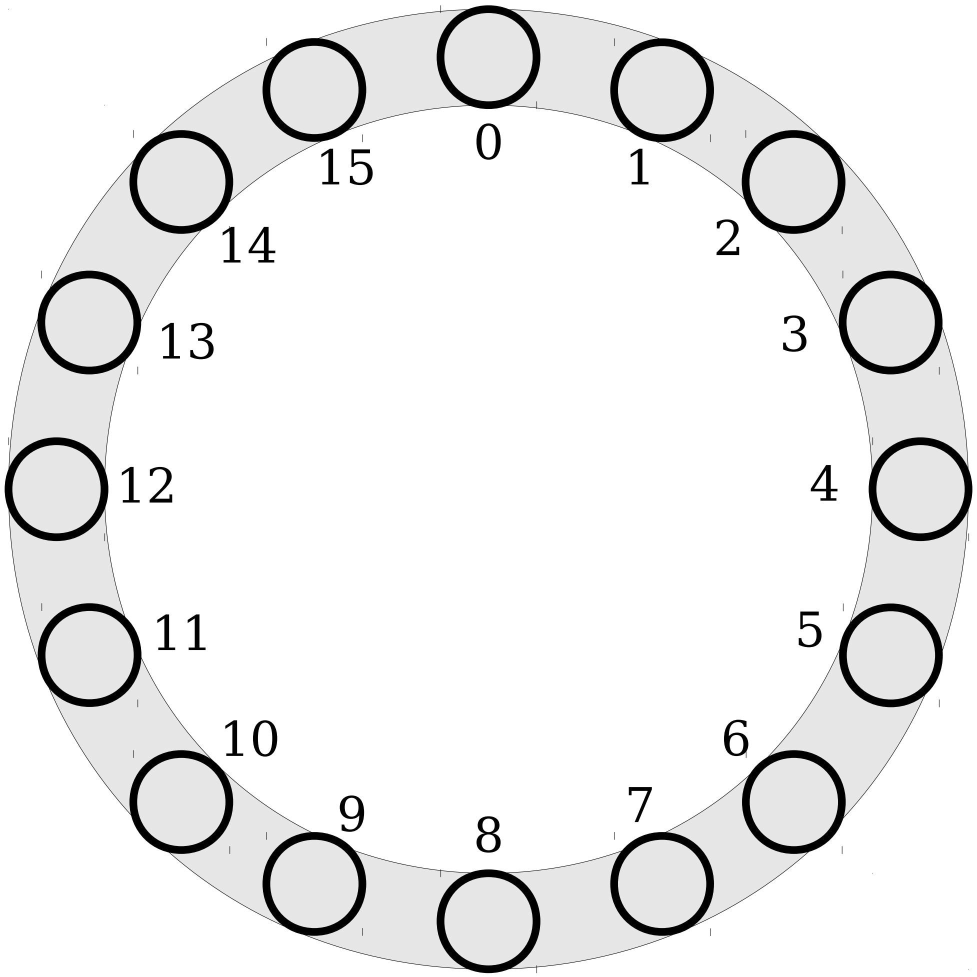
- ***Linear Probing Hashing***
  - A simple and lightning fast hash table implementation.
- ***Analyzing Linear Probing***
  - Why the degree of independence matters.
- ***Fourth Moment Bounds***
  - Another approach for estimating frequencies.

## Hashing Strategies

- All hash table implementations need to address what happens when collisions occur.
- Common strategies:
  - ***Closed addressing:*** Store all elements with hash collisions in a secondary data structure (linked list, BST, etc.)
  - ***Perfect hashing:*** Choose hash functions to ensure that collisions don't happen, and rehash or move elements when they do.
  - ***Open addressing:*** Allow elements to “leak out” from their preferred position and spill over into other positions.
- Linear probing is an example of open addressing.
- We'll see a type of perfect hashing (***cuckoo hashing***) on Thursday.

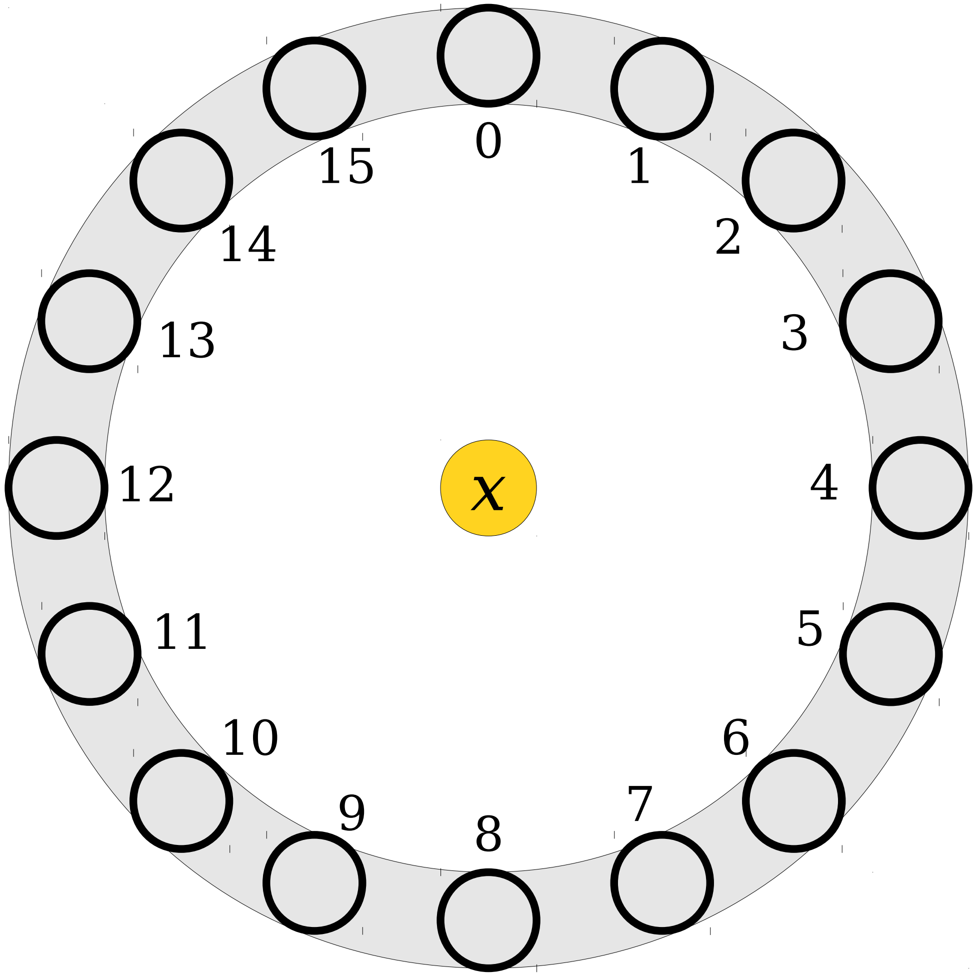
## Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



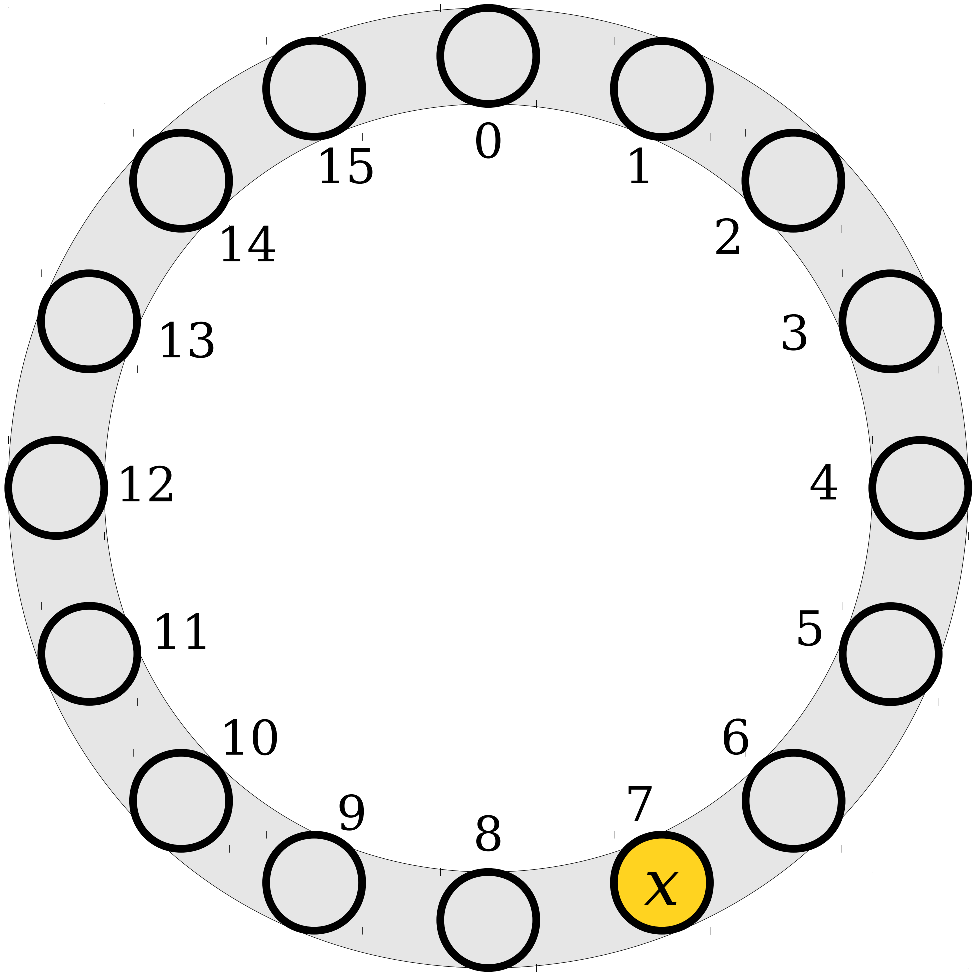
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



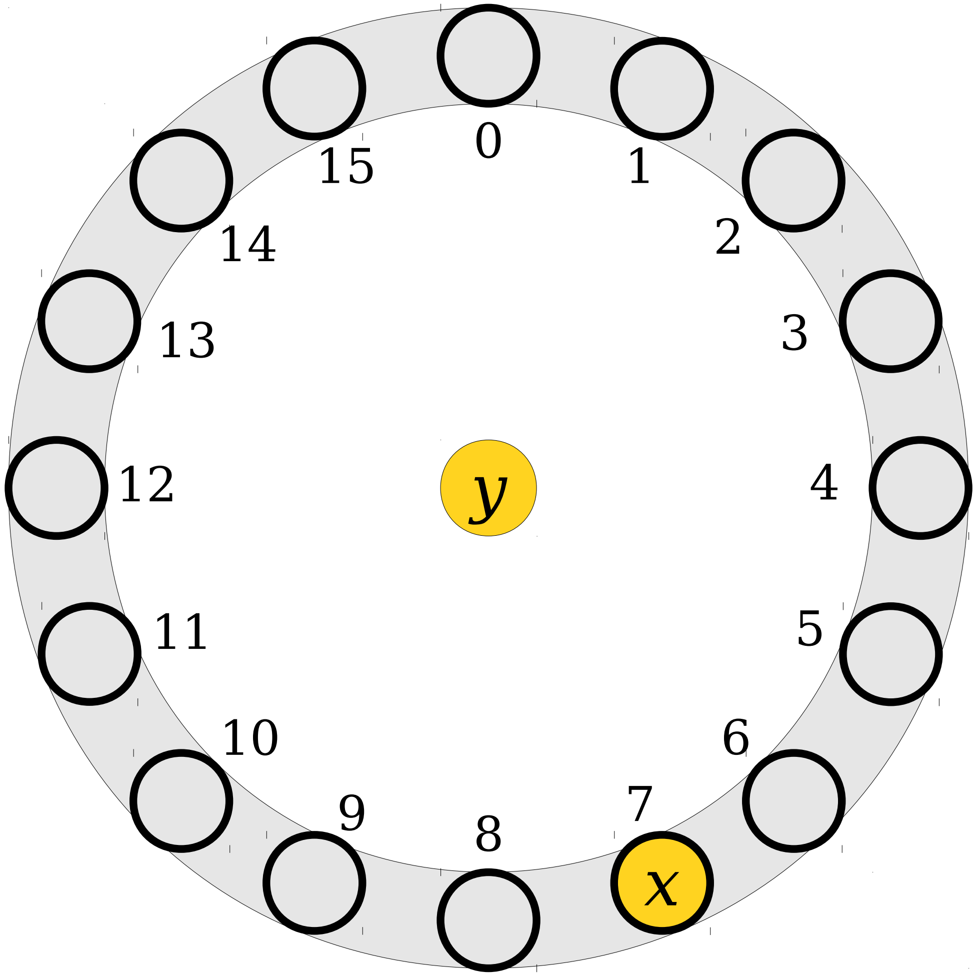
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



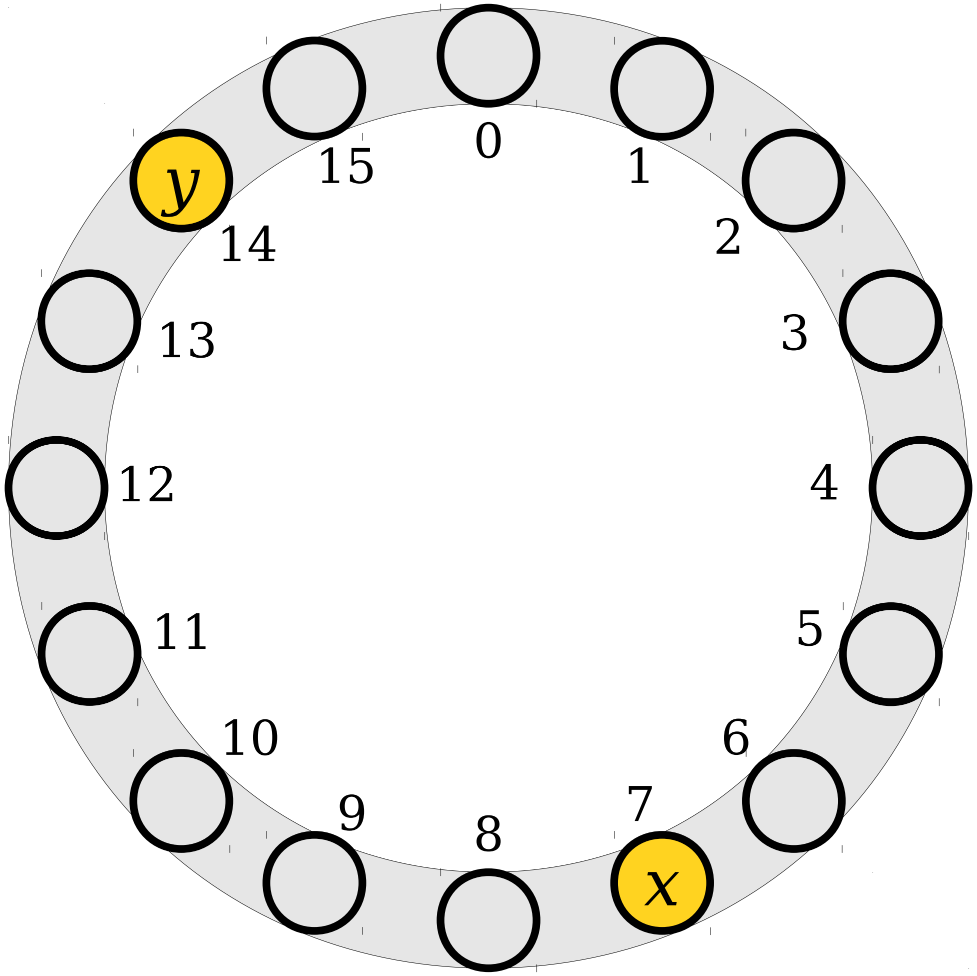
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



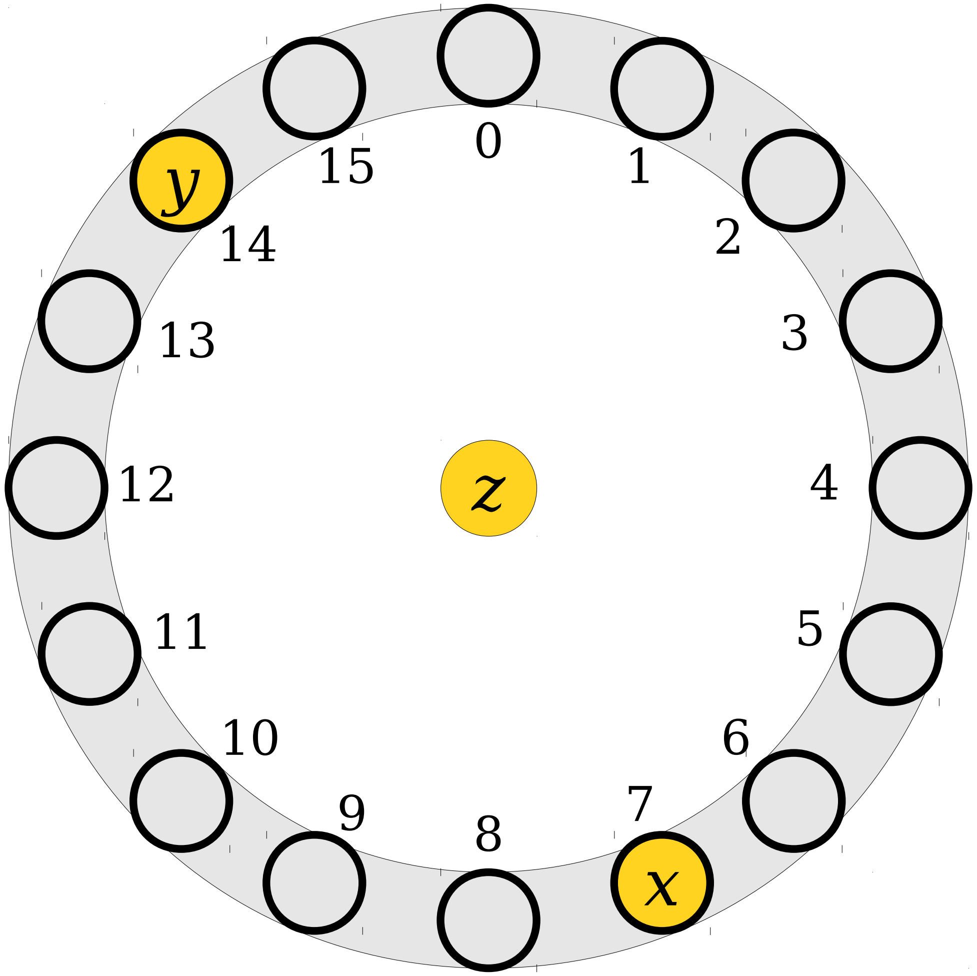
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



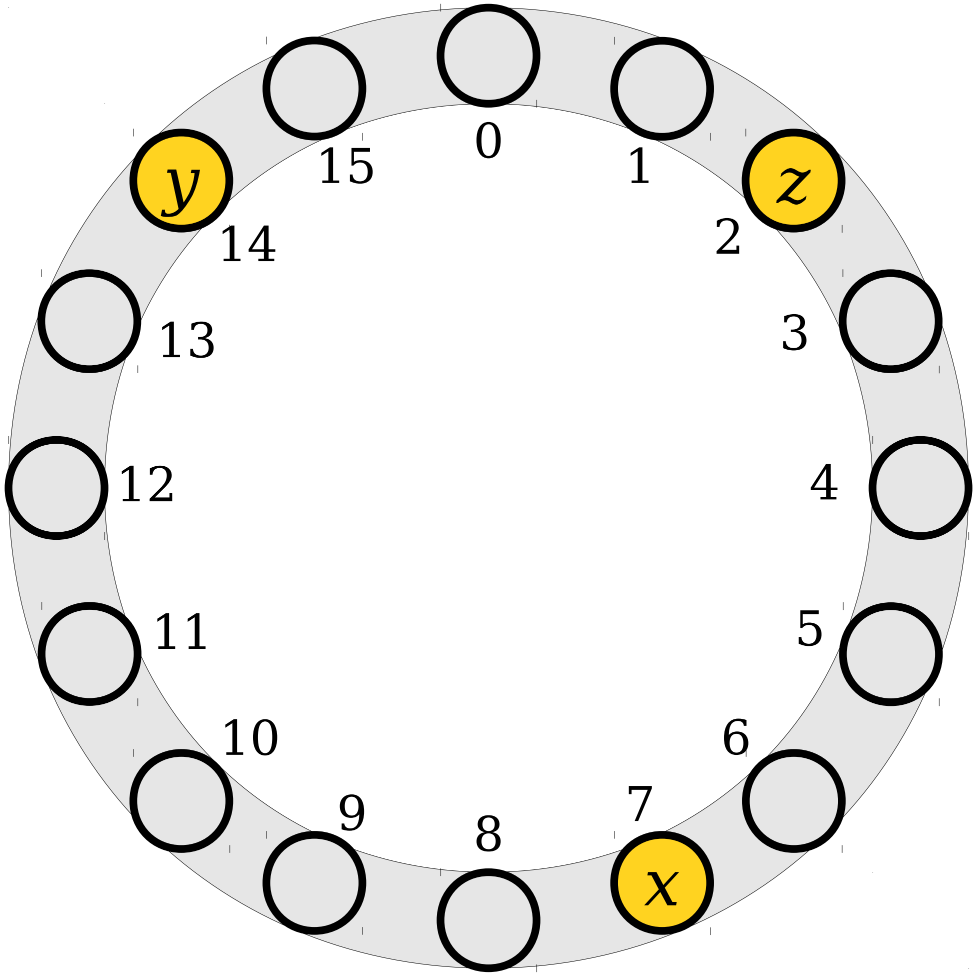
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



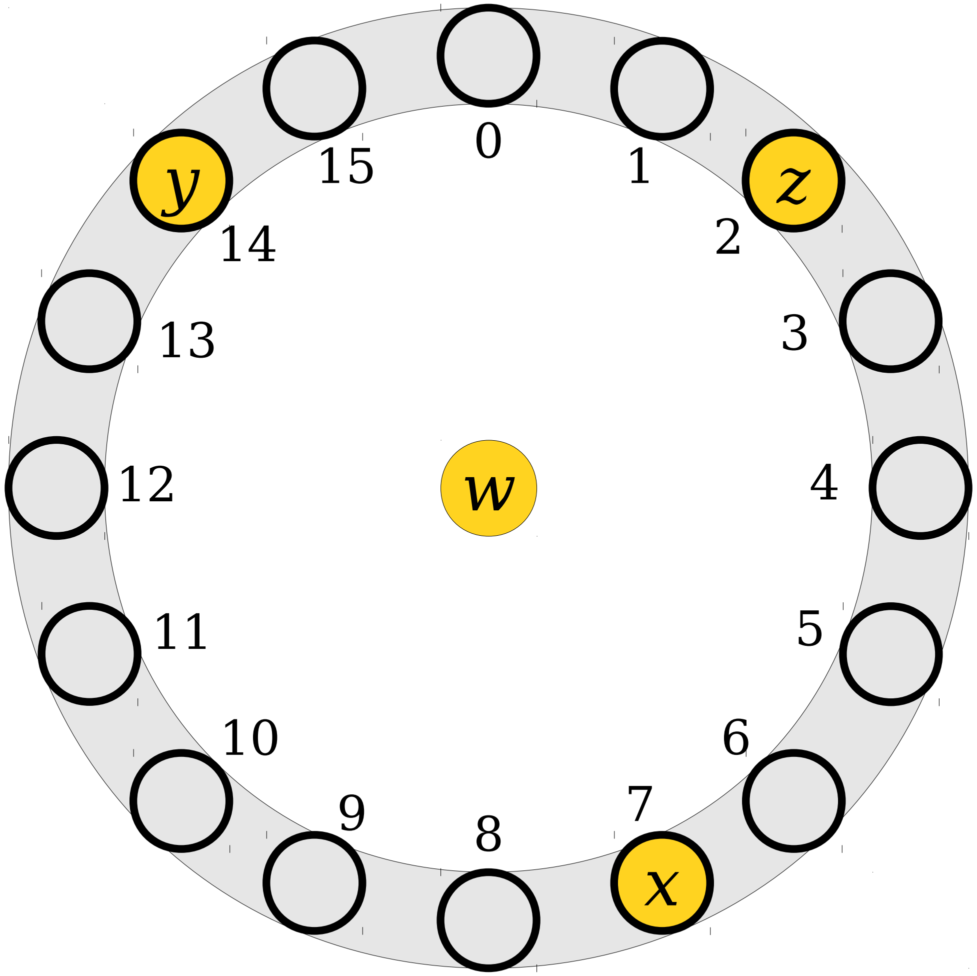
## Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



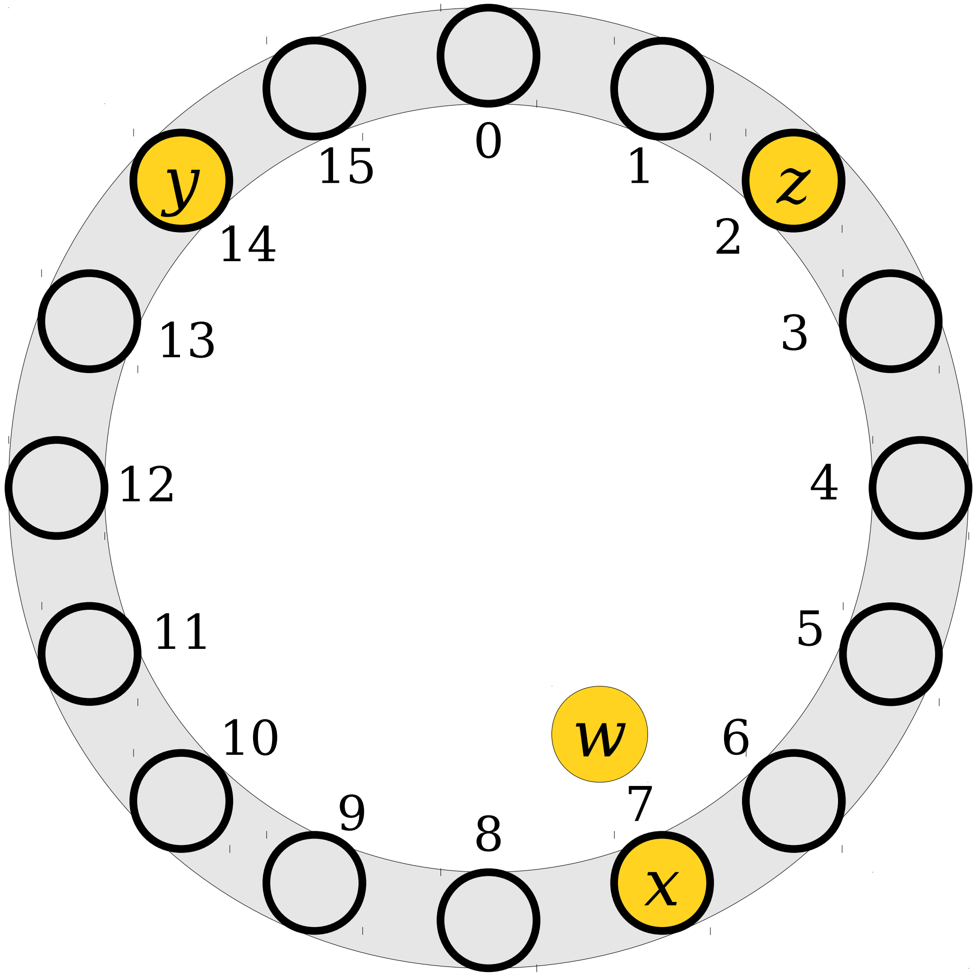
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



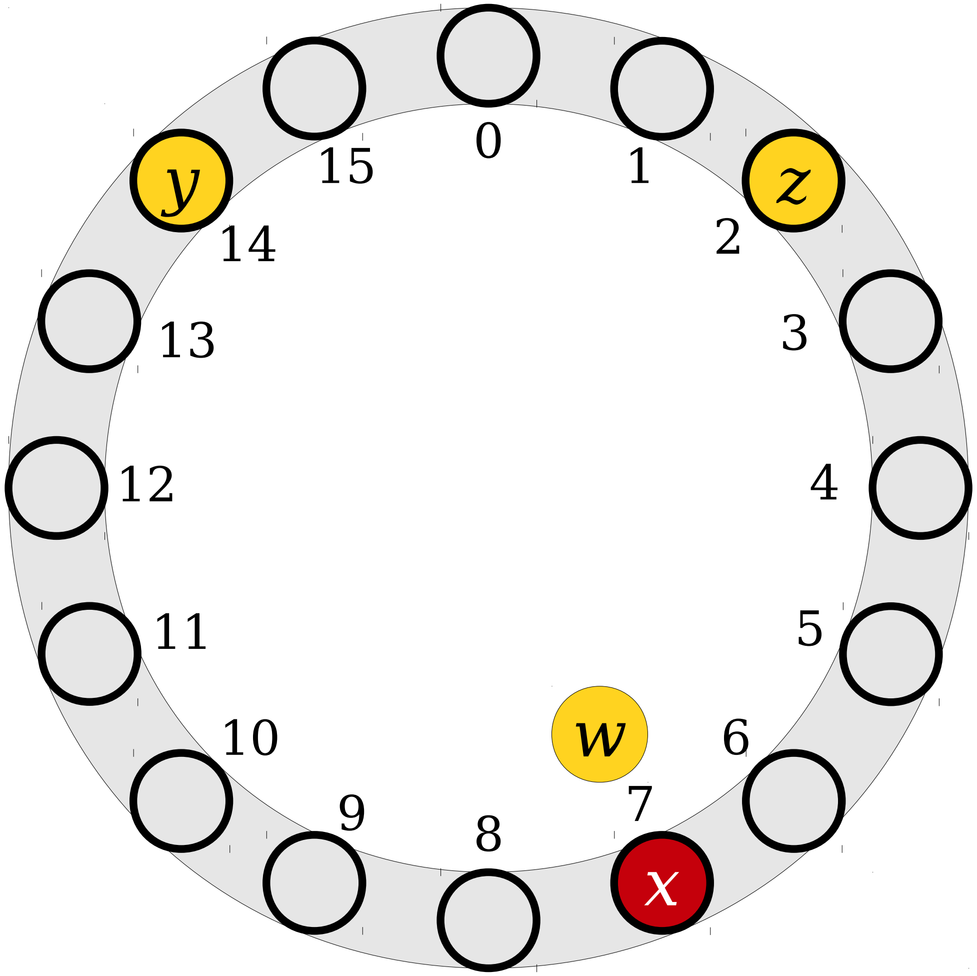
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



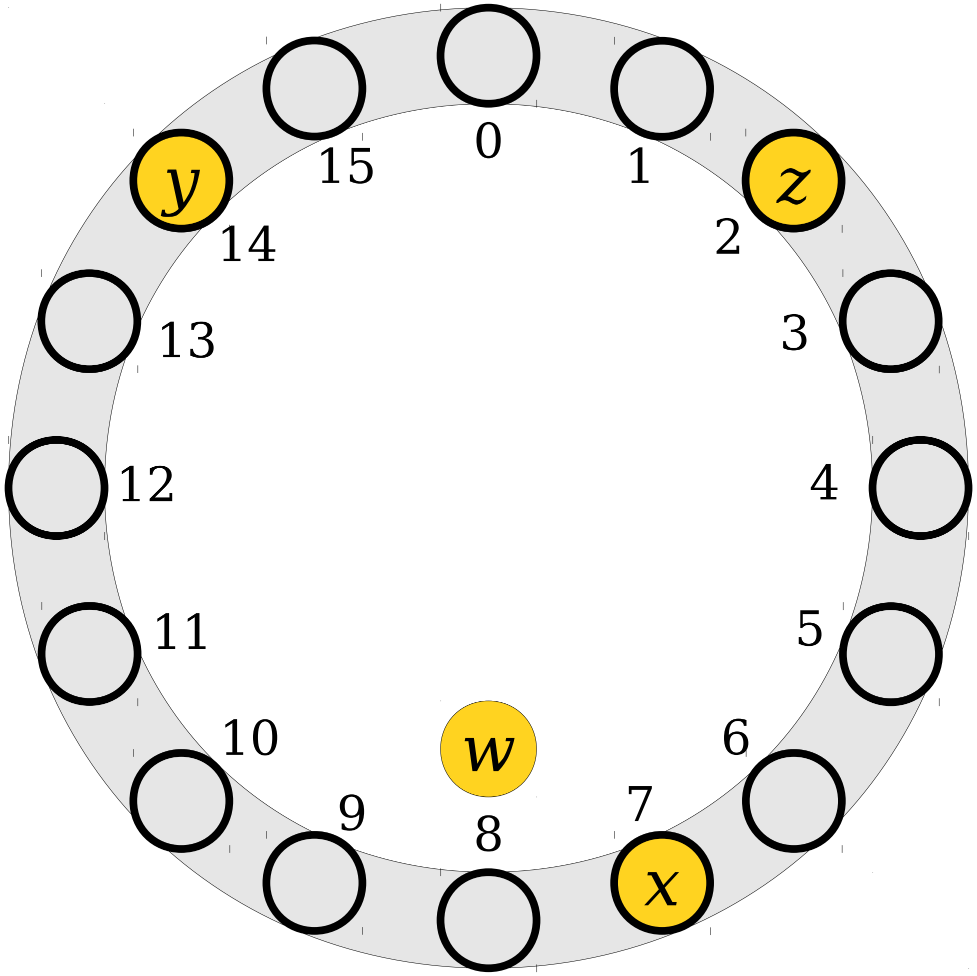
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



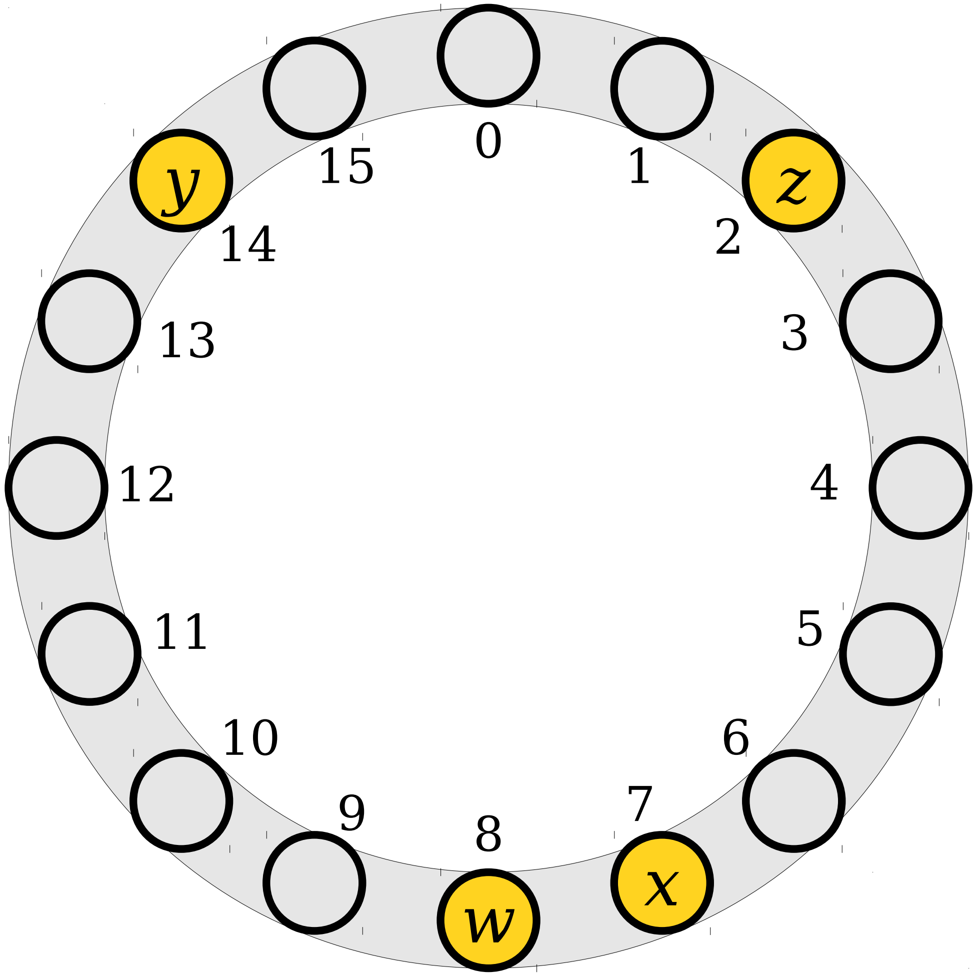
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



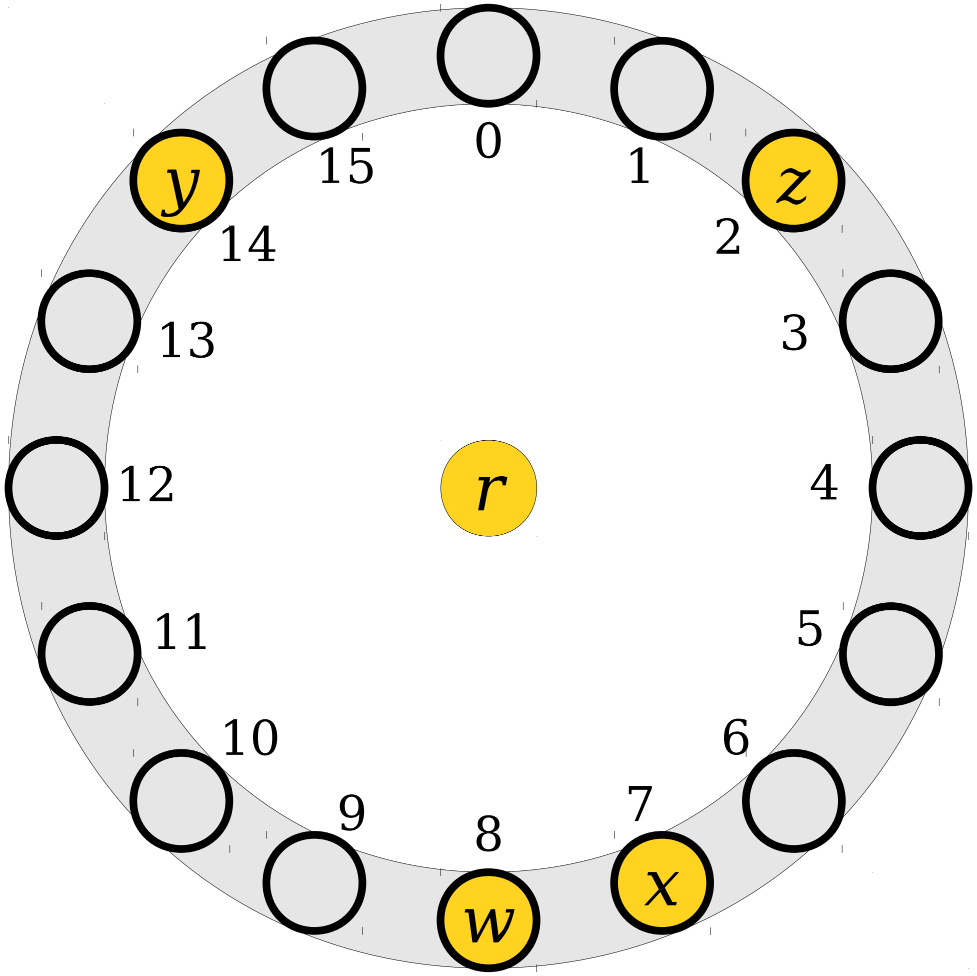
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



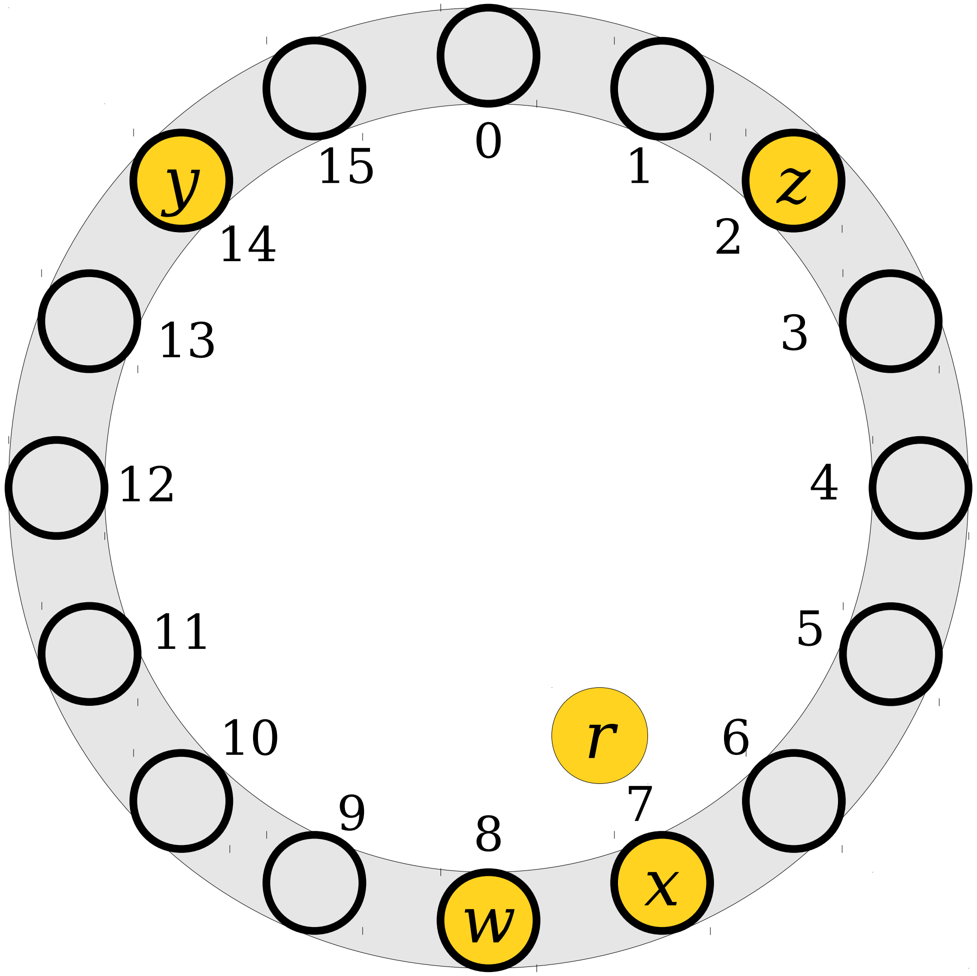
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



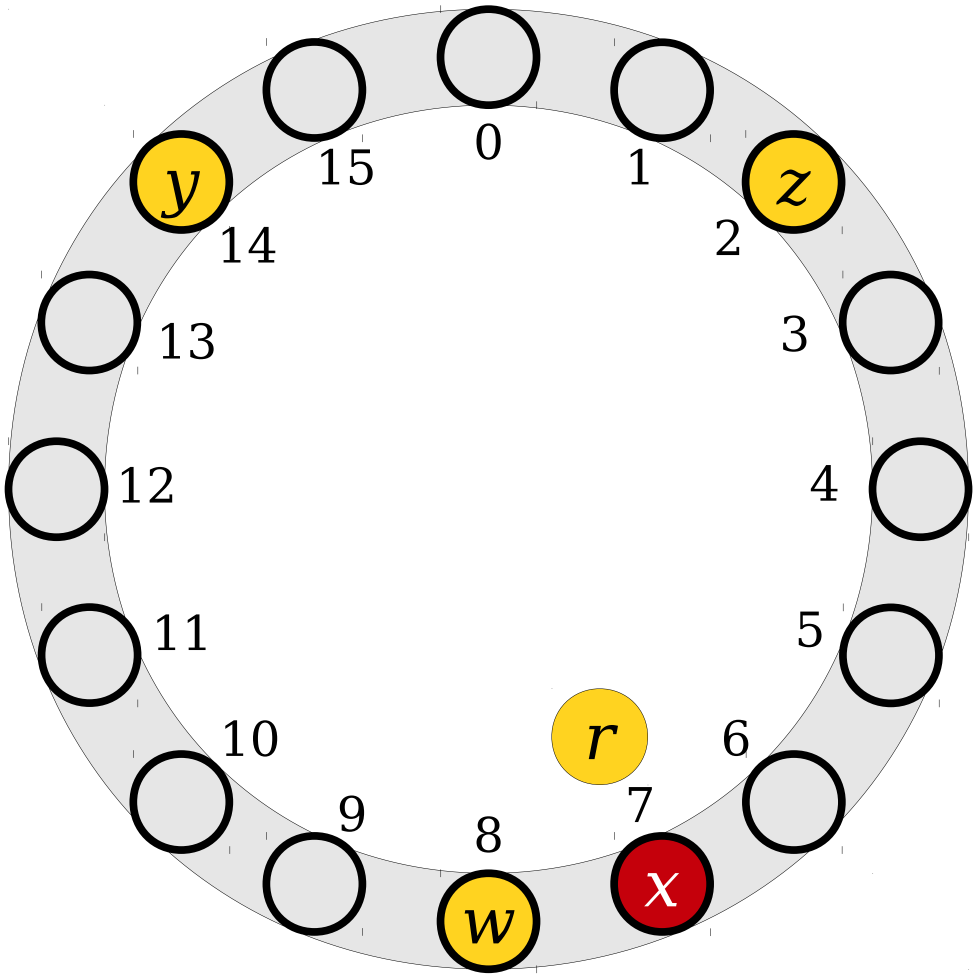
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



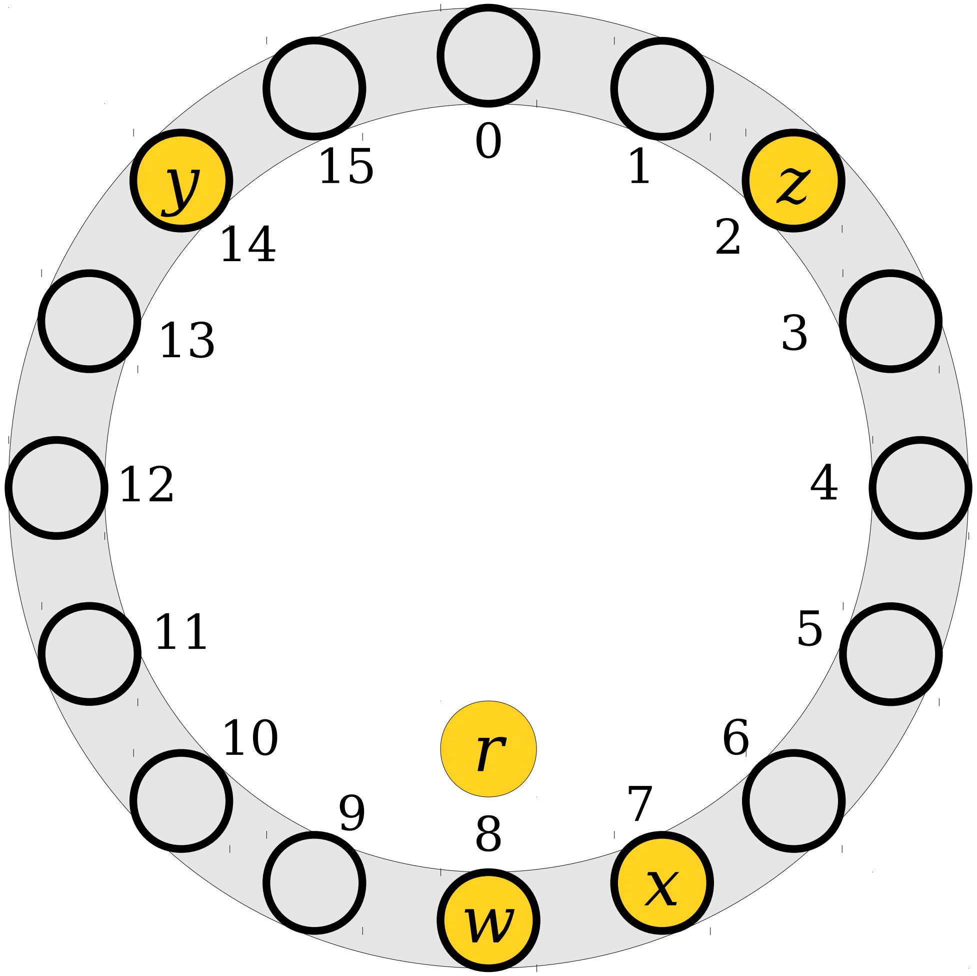
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



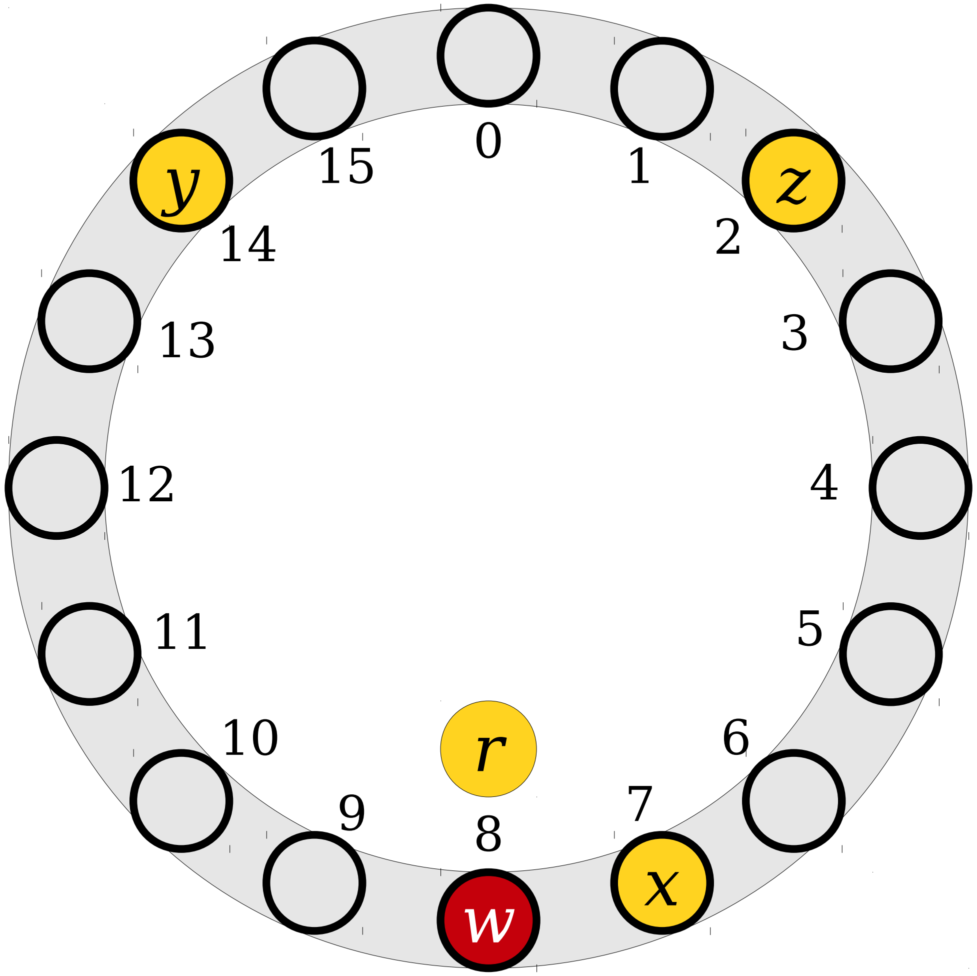
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



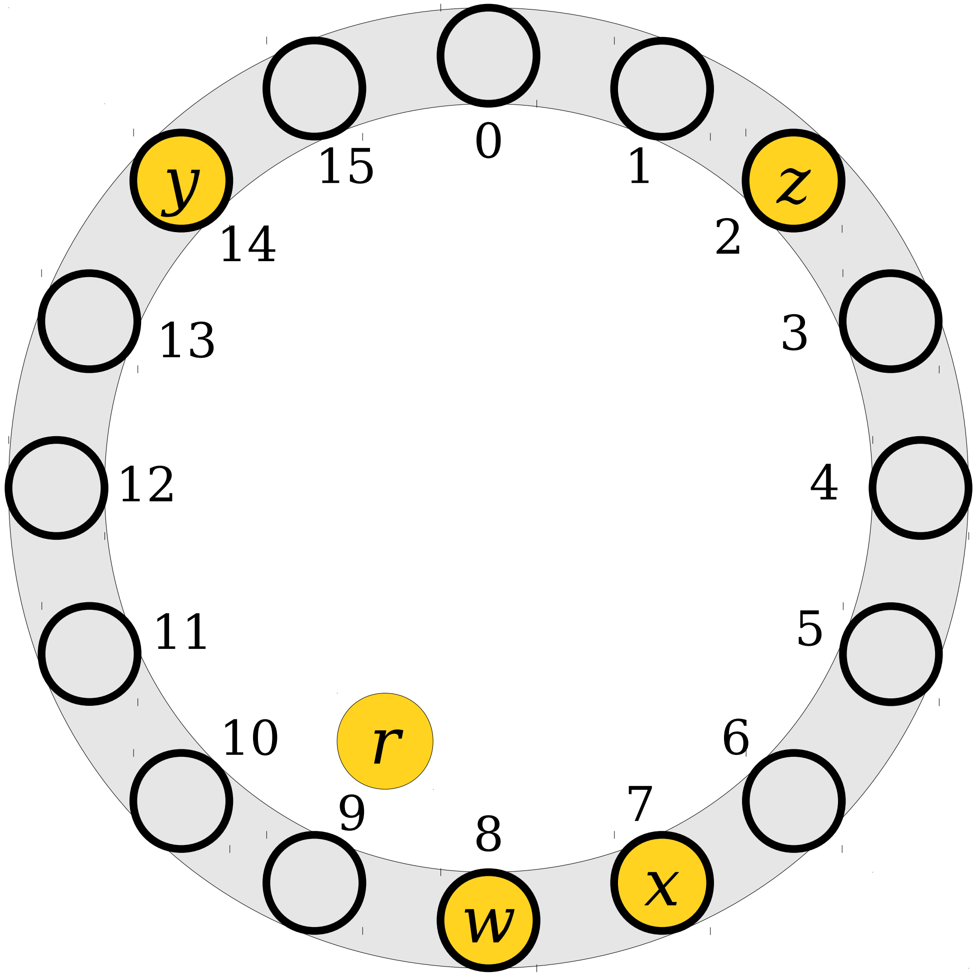
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



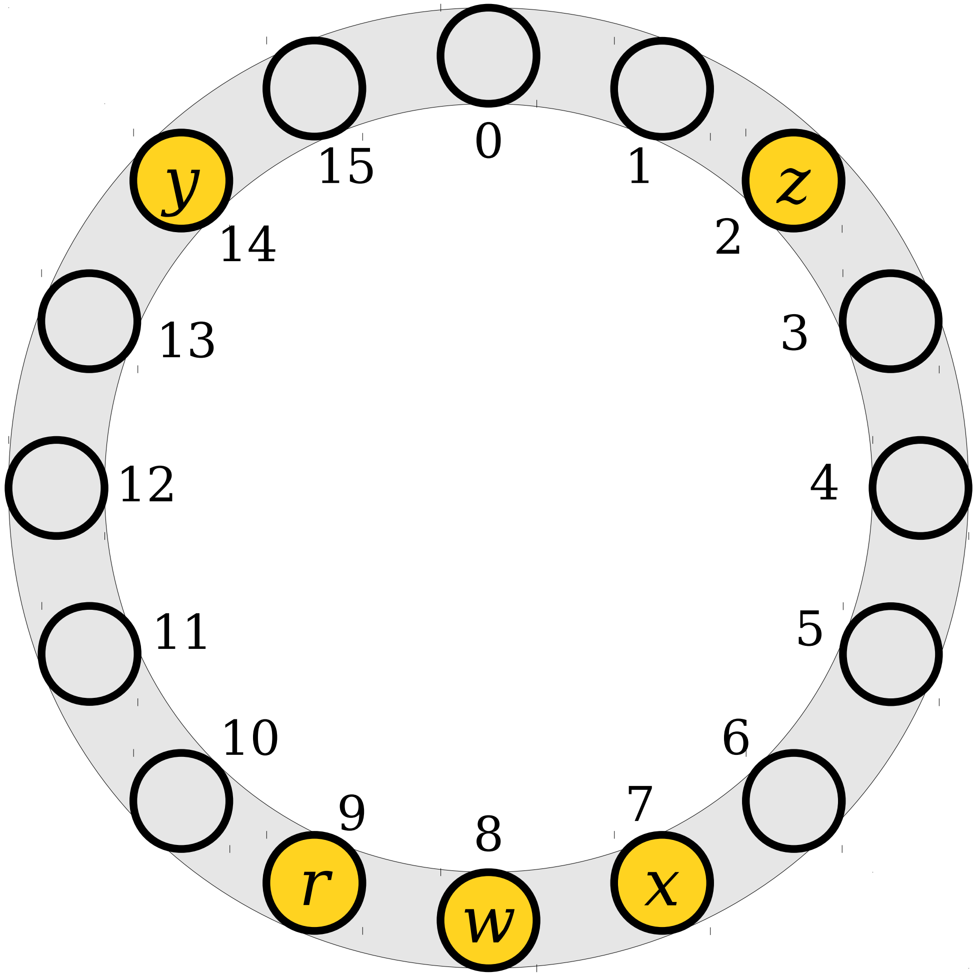
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



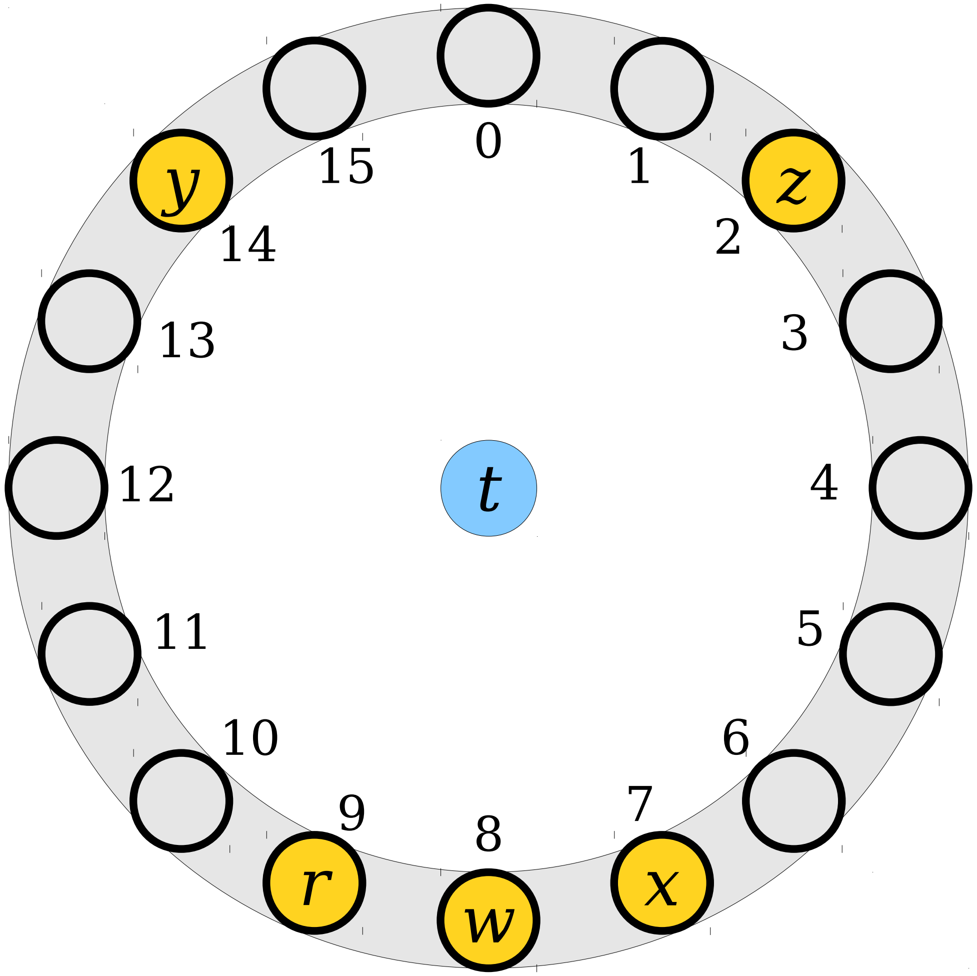
# Linear Probing

- ***Linear probing*** is a simple open-addressing hashing strategy.
- To insert an element $x$, compute $h(x)$ and try to place $x$ there.
- If that spot is occupied, keep moving through the array, wrapping around at the end, until a free spot is found.



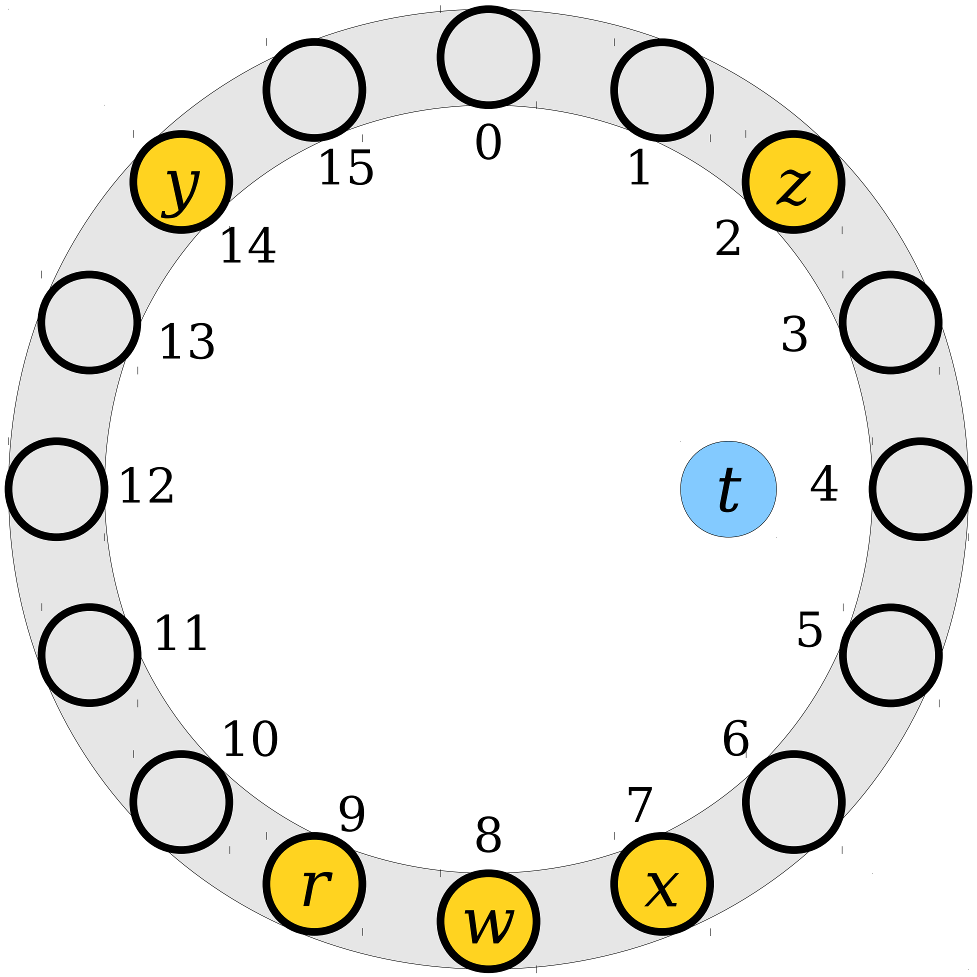
# Linear Probing

- To look up an element *x*, compute *h*(*x*) and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



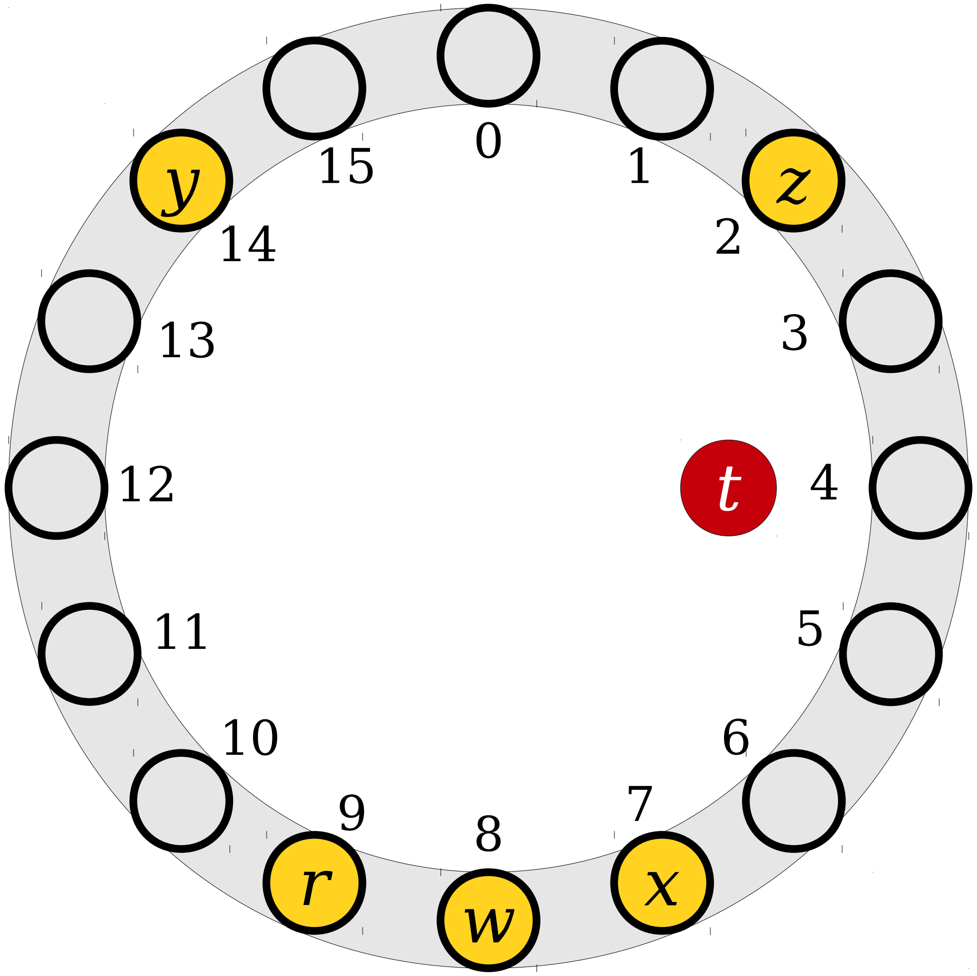
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



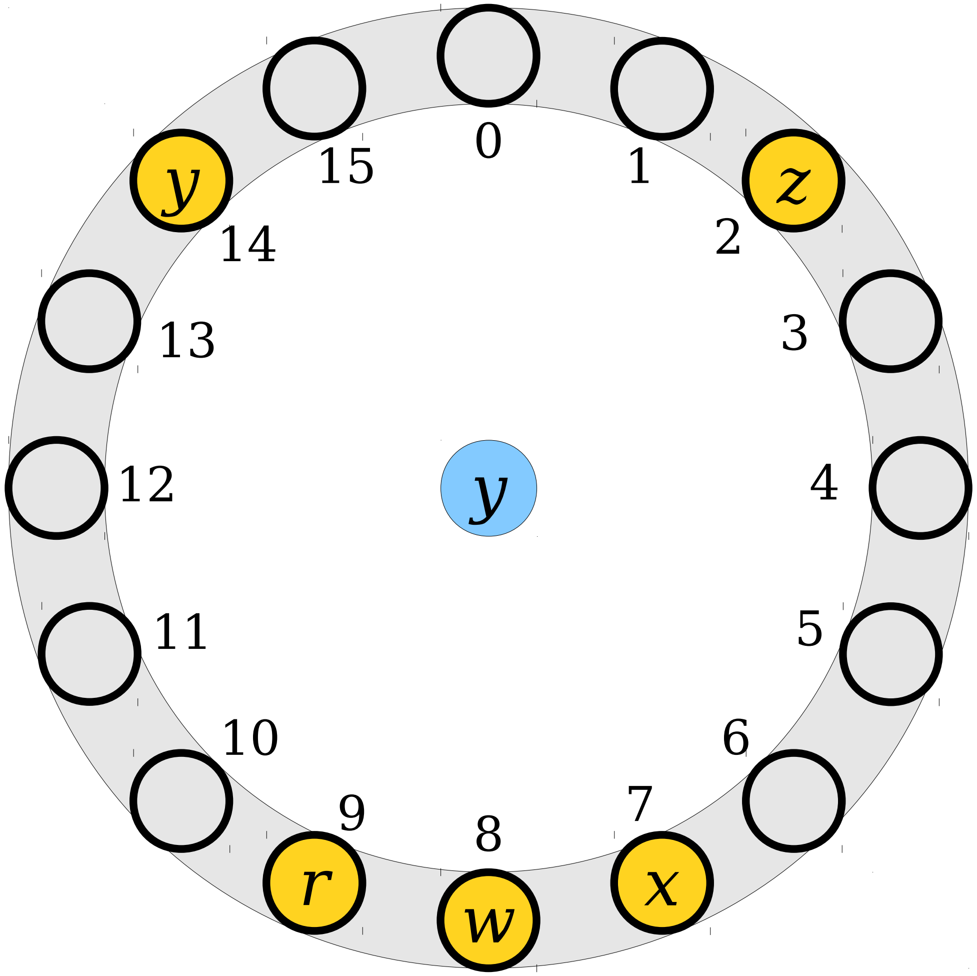
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



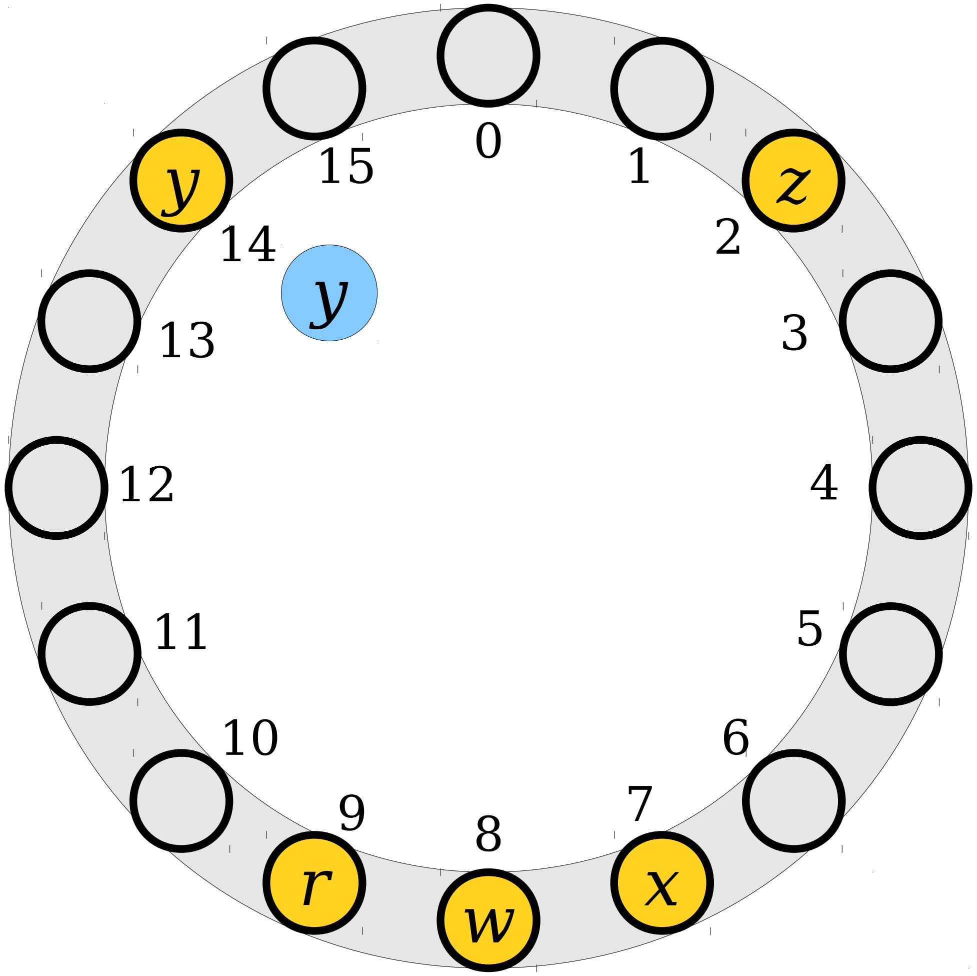
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



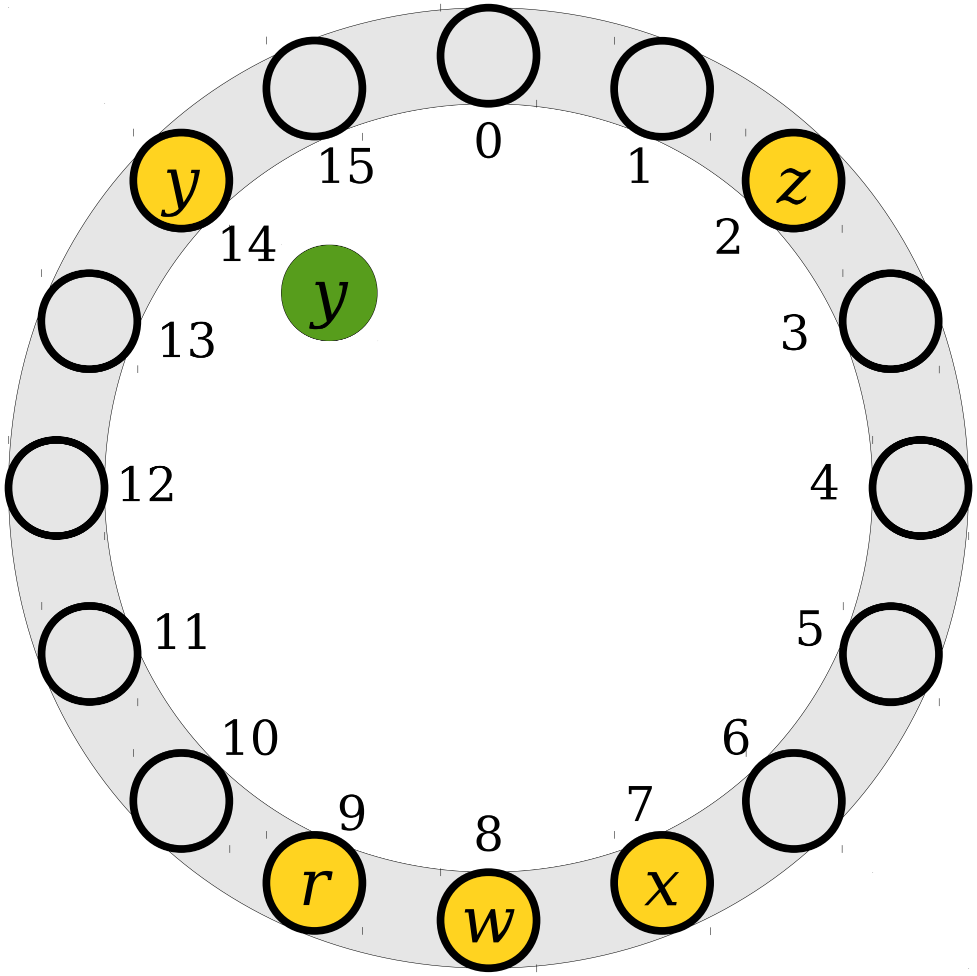
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



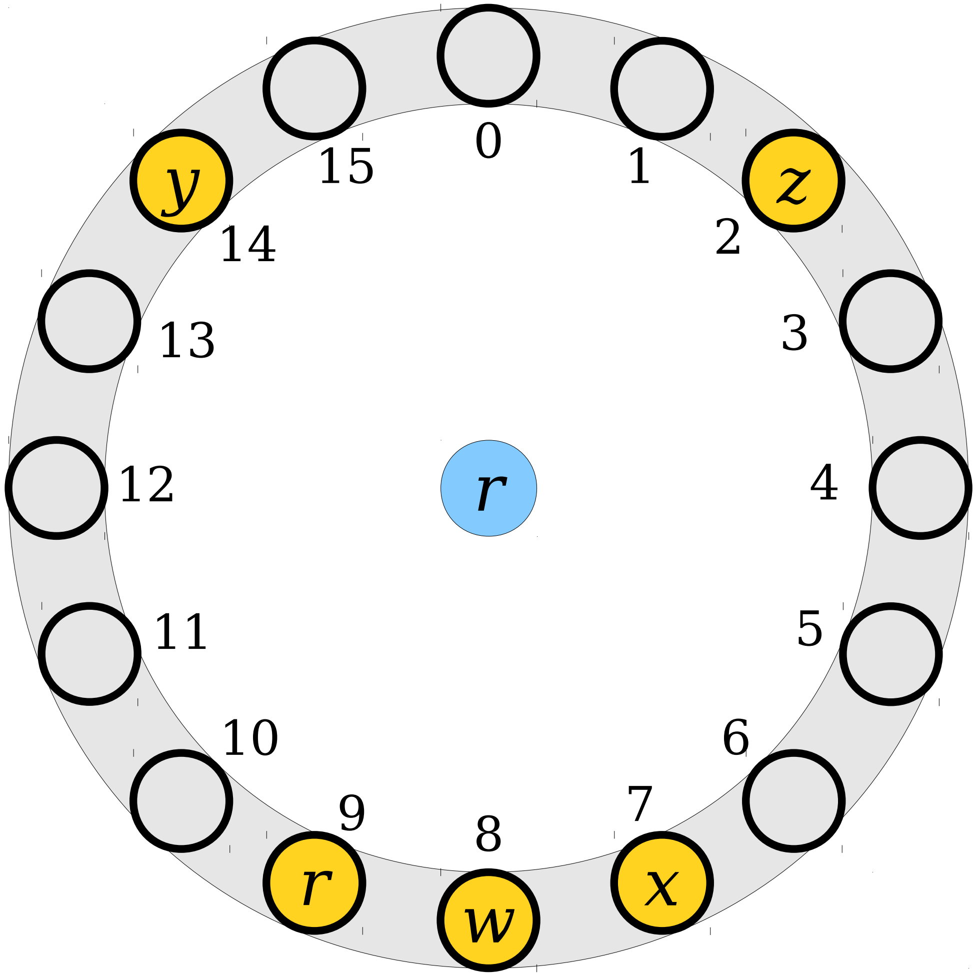
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



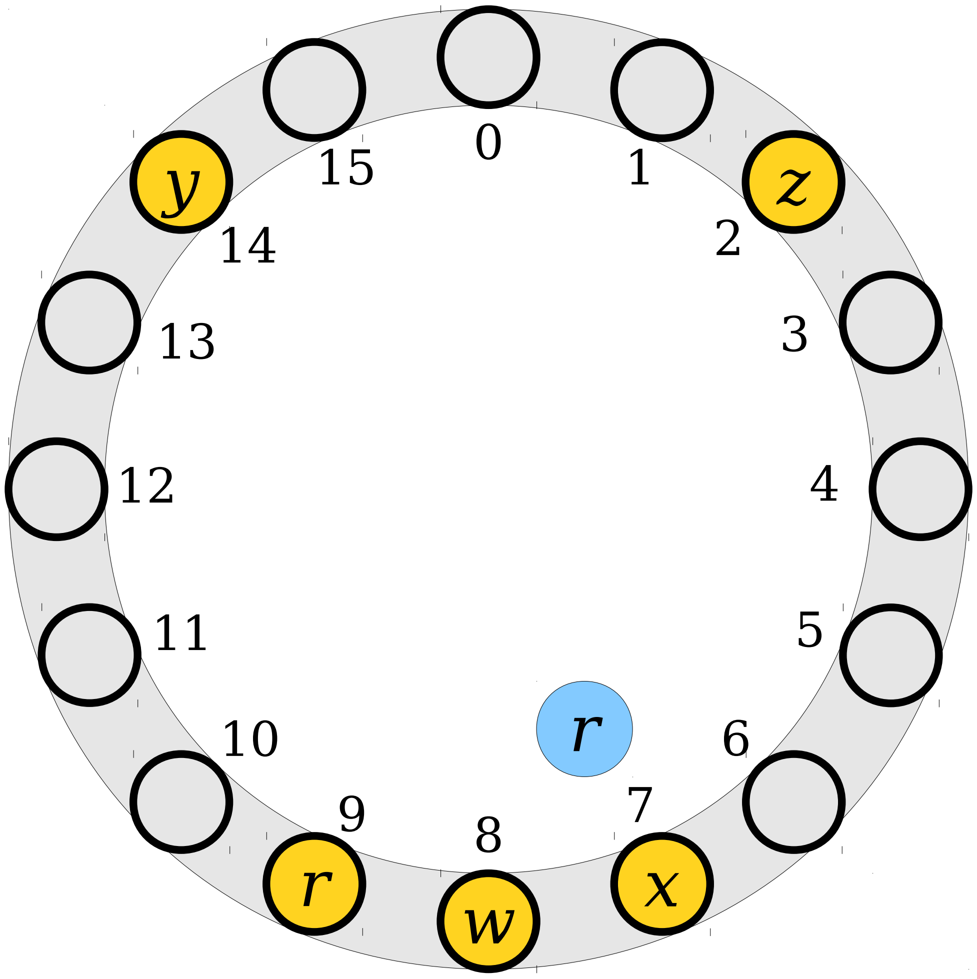
# Linear Probing

- To look up an element *x*, compute *h*(*x*) and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



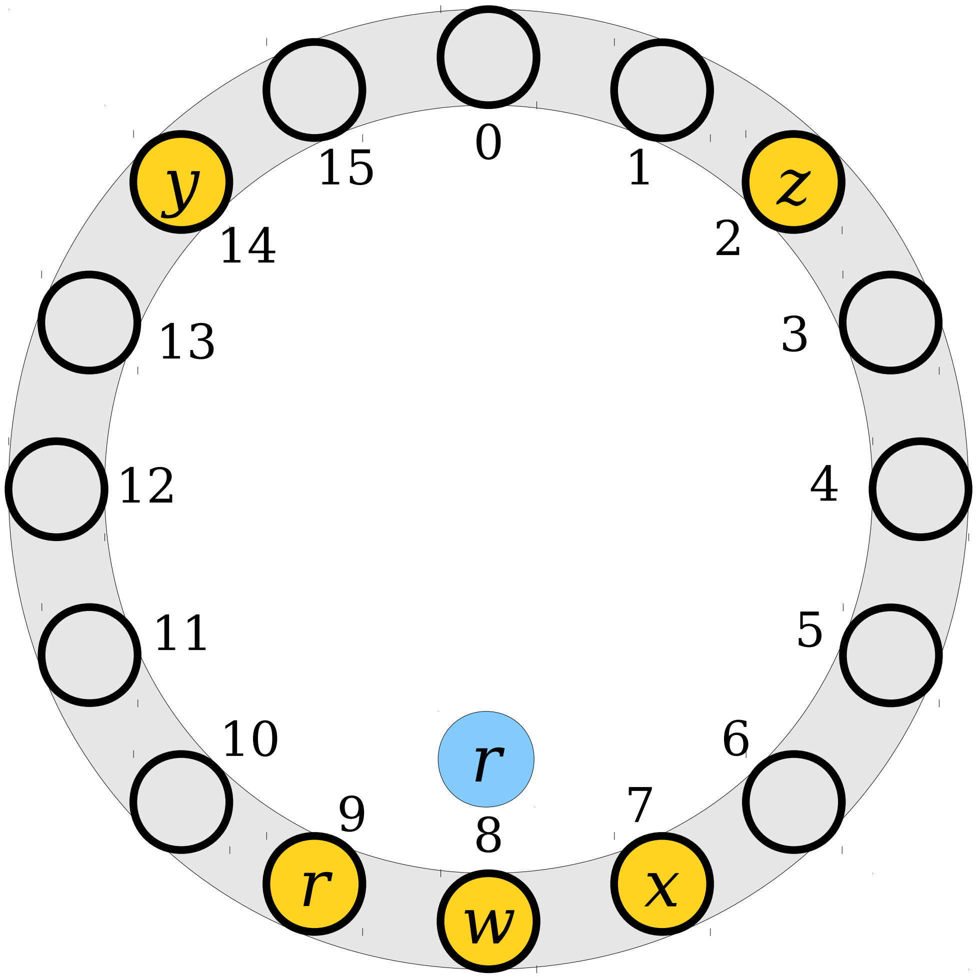
# Linear Probing

- To look up an element *x*, compute *h*(*x*) and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



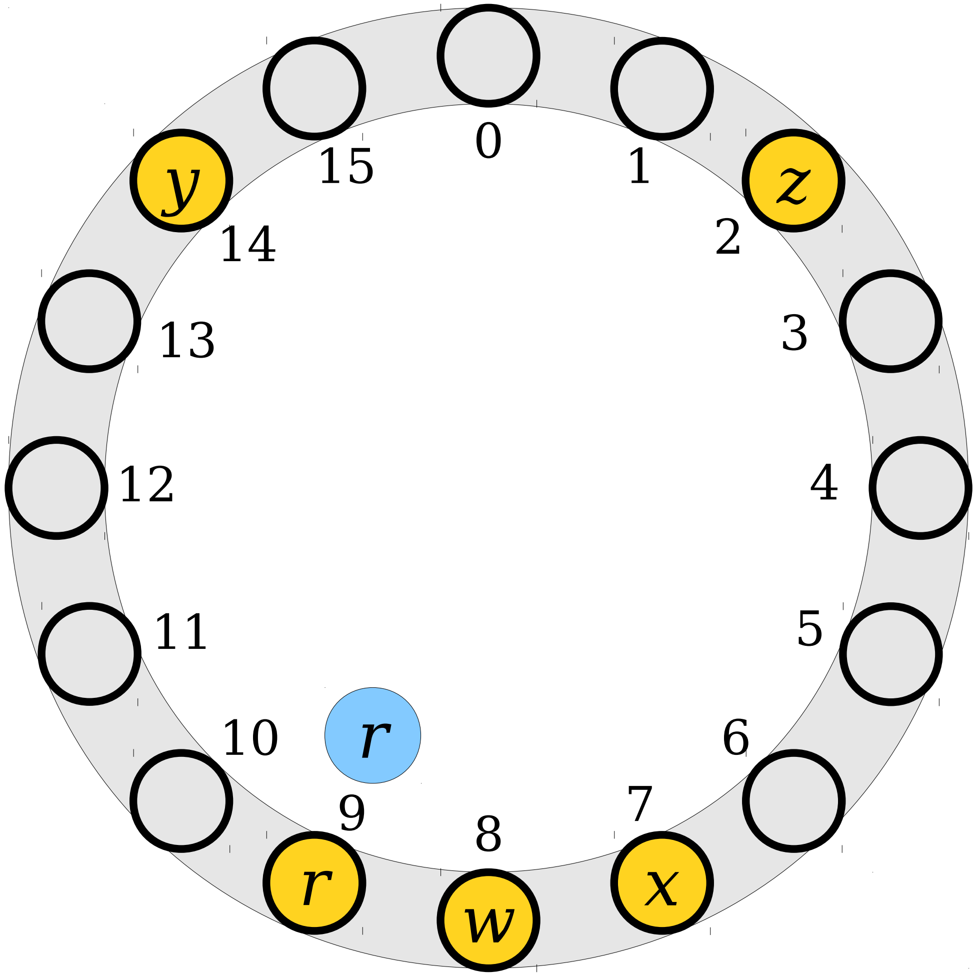
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



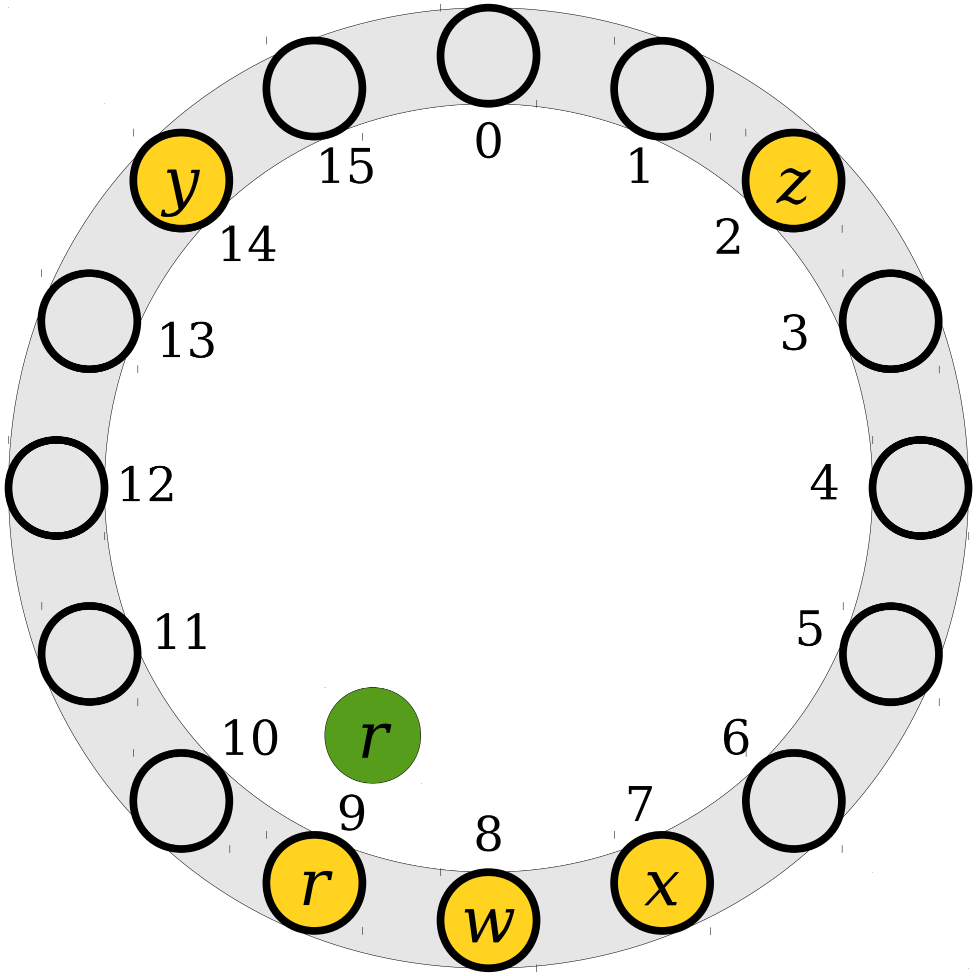
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



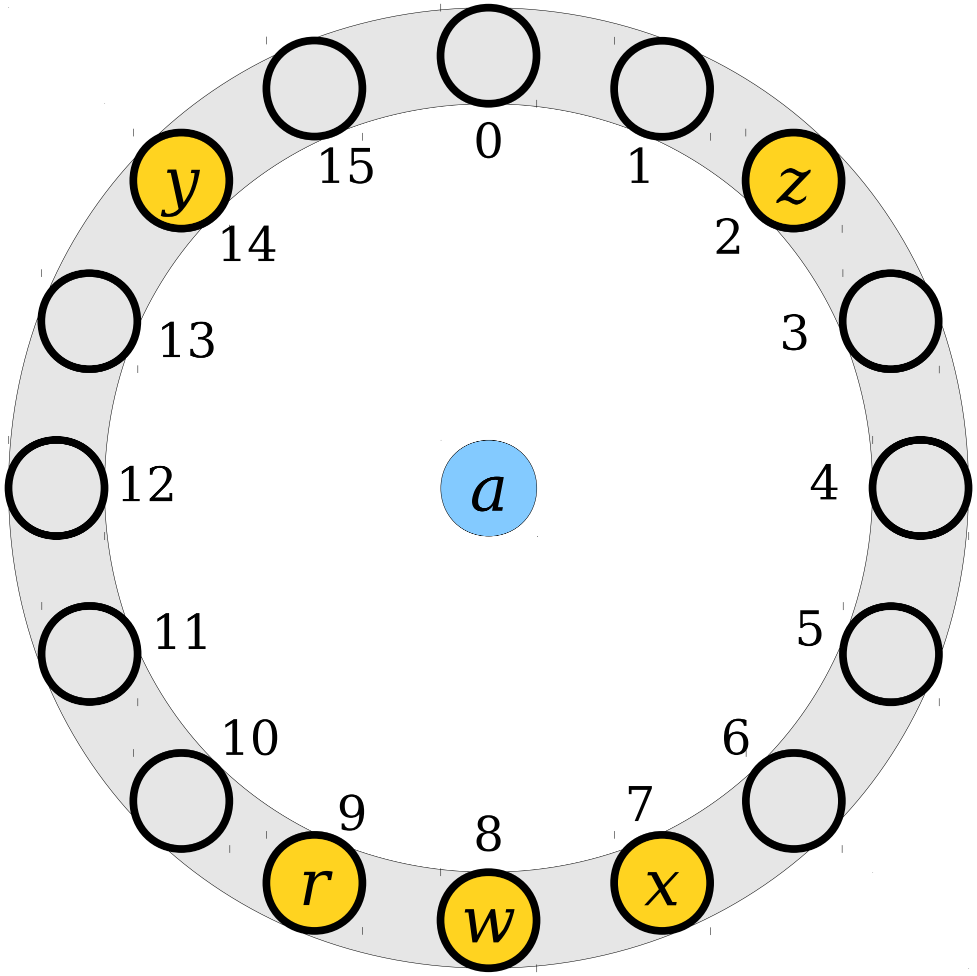
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



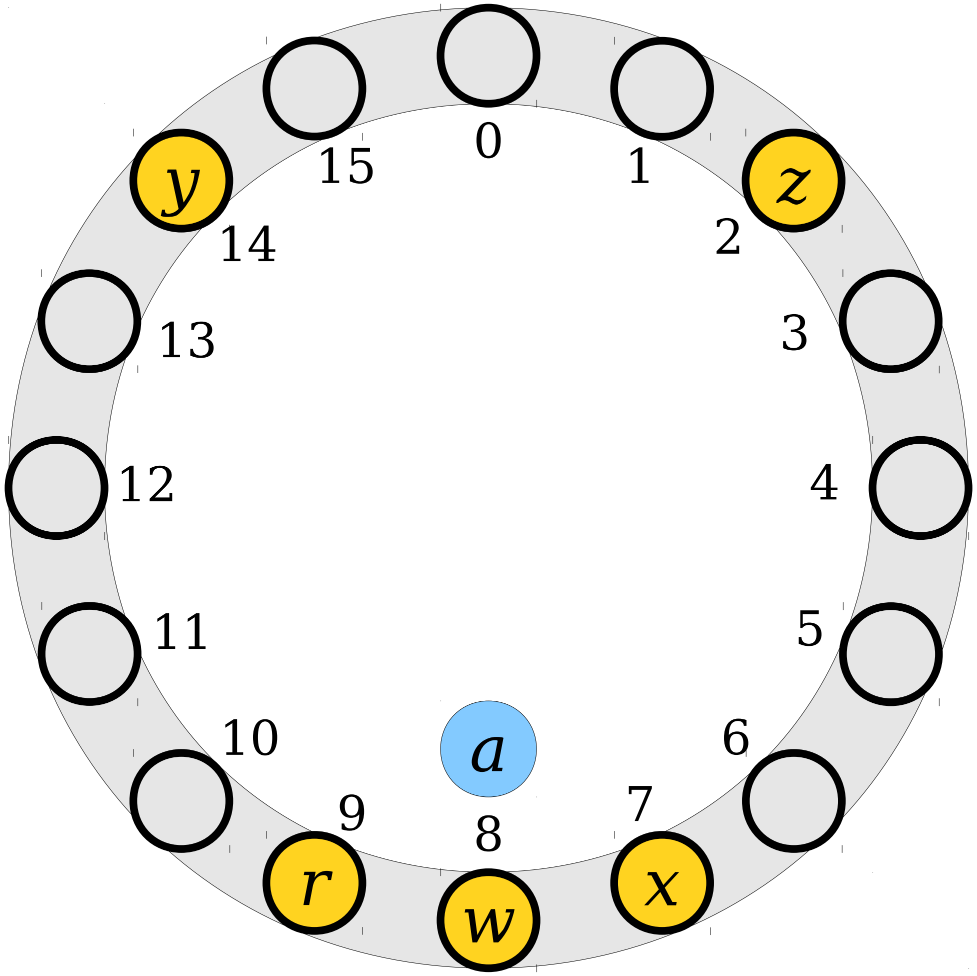
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



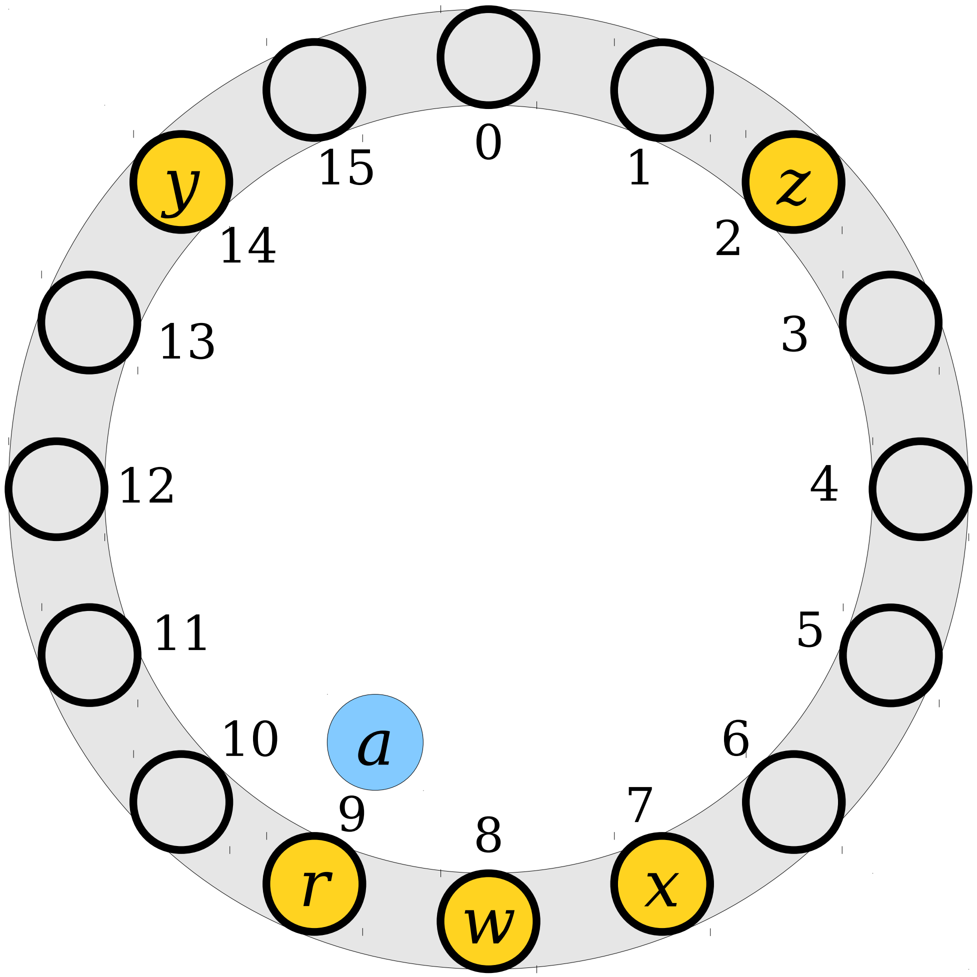
# Linear Probing

- To look up an element *x*, compute *h*(*x*) and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



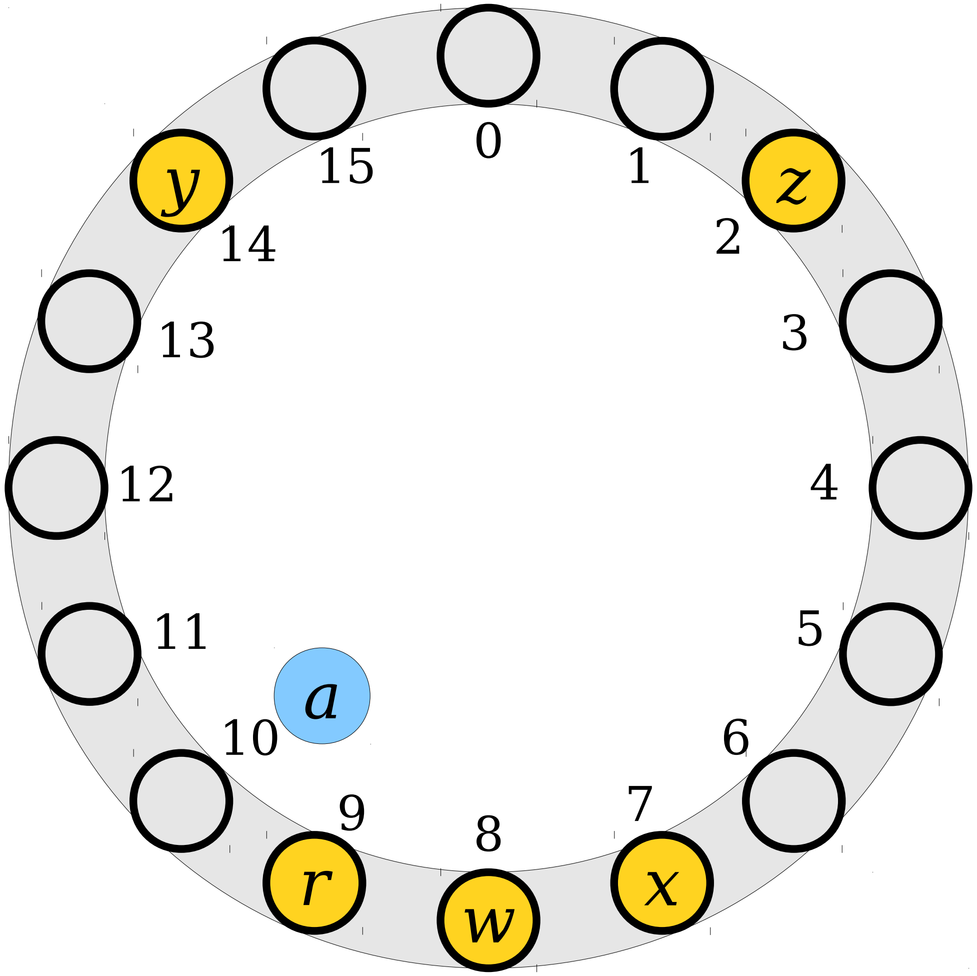
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



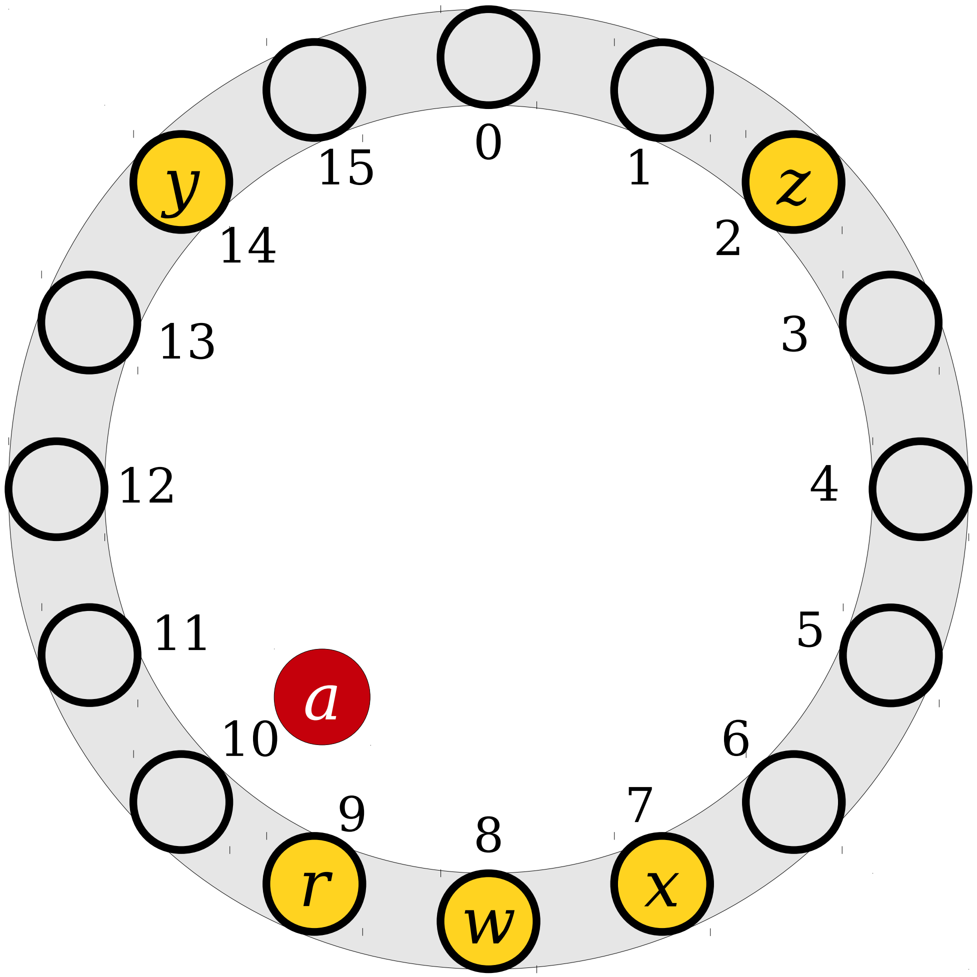
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



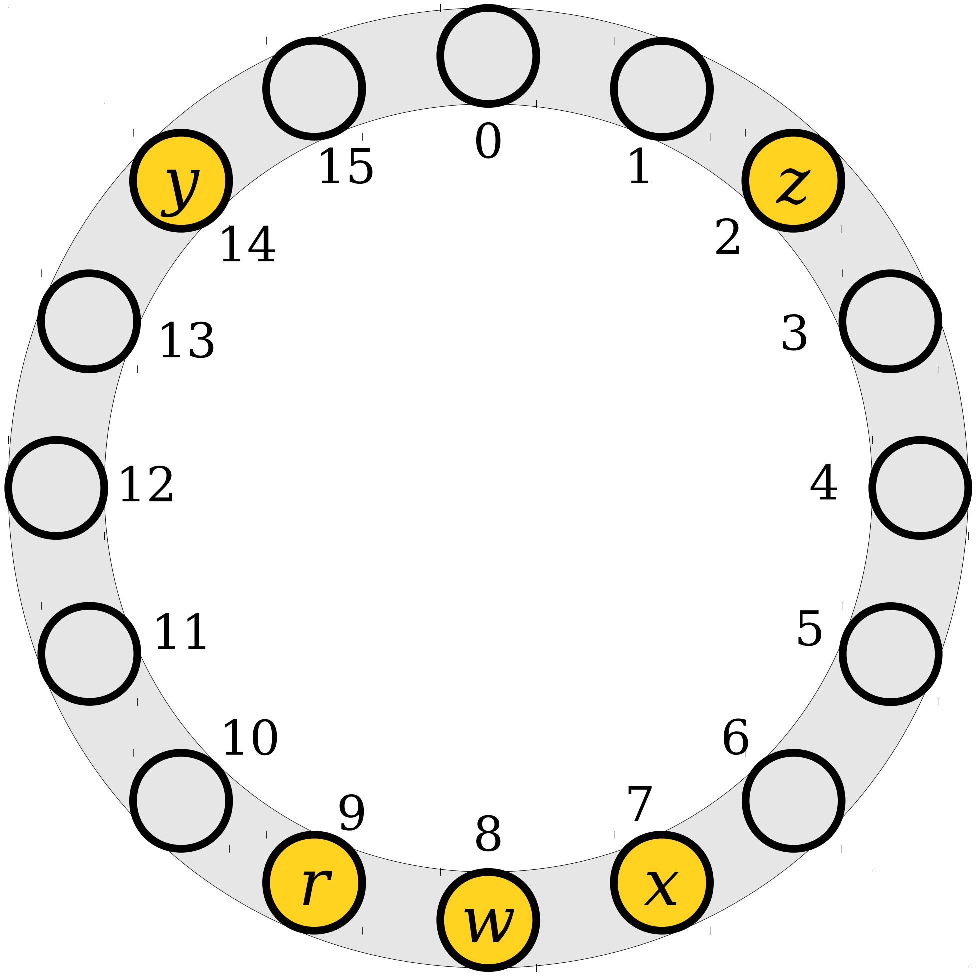
# Linear Probing

- To look up an element $x$, compute $h(x)$ and start looking there.
- Move around the ring until either the element is found or a blank spot is detected.
- (We'll assume the load factor prohibits us from inserting so many elements that there are no free spaces.)



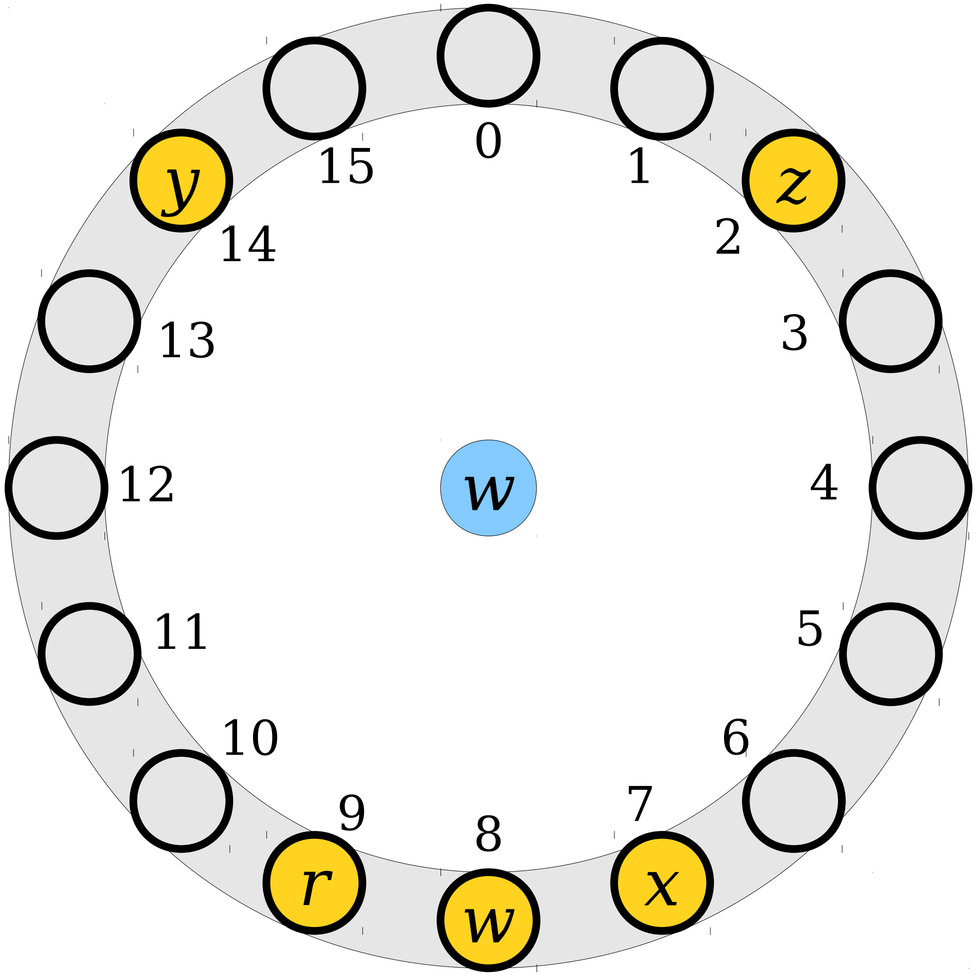
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



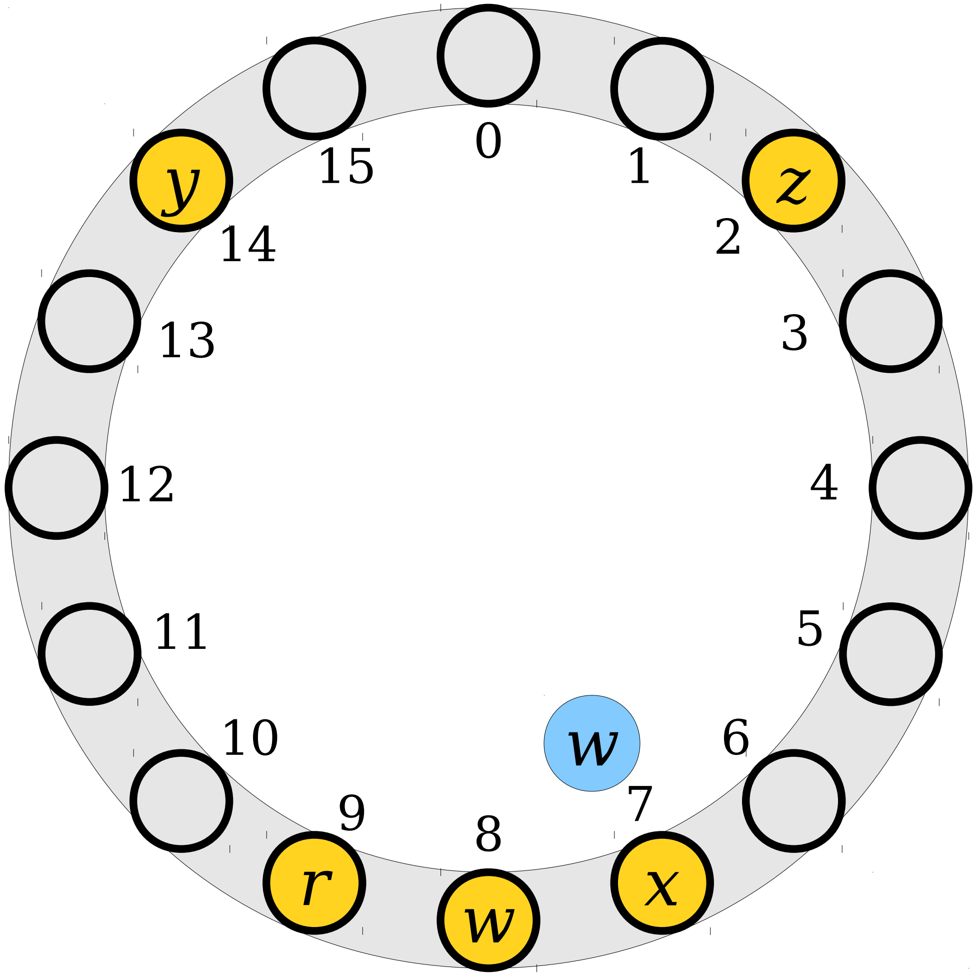
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



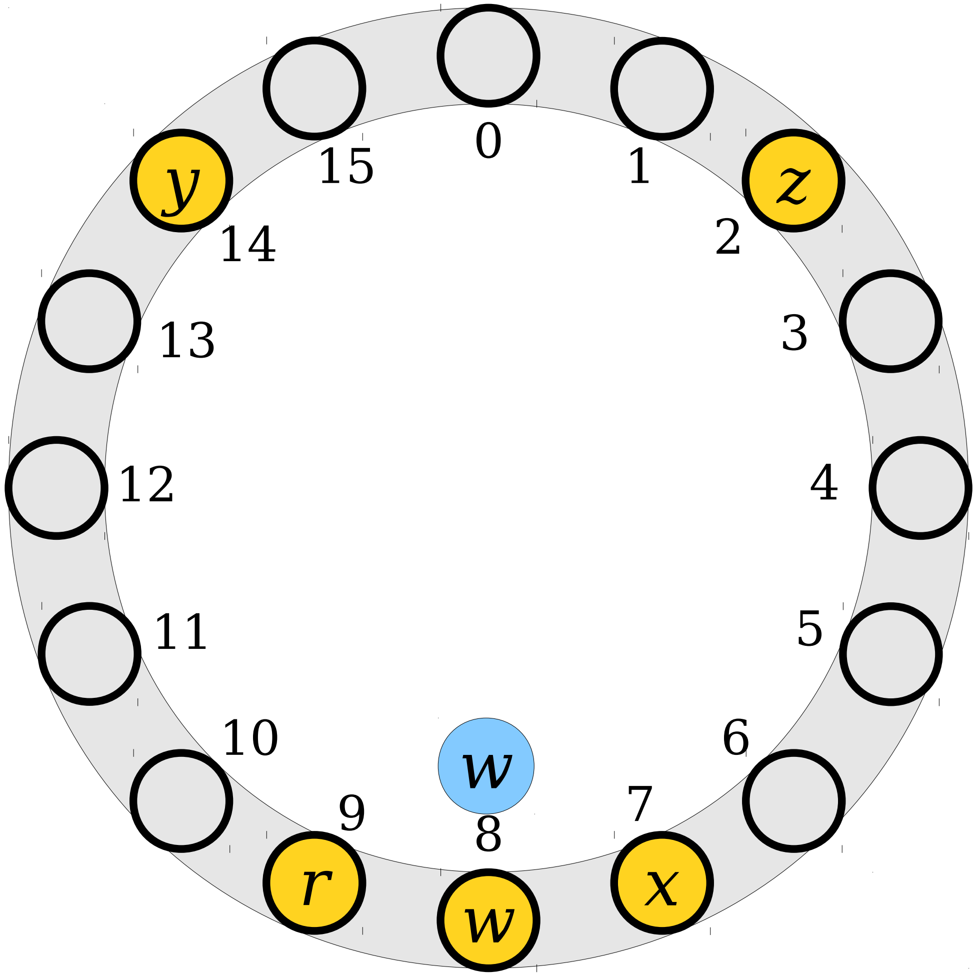
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



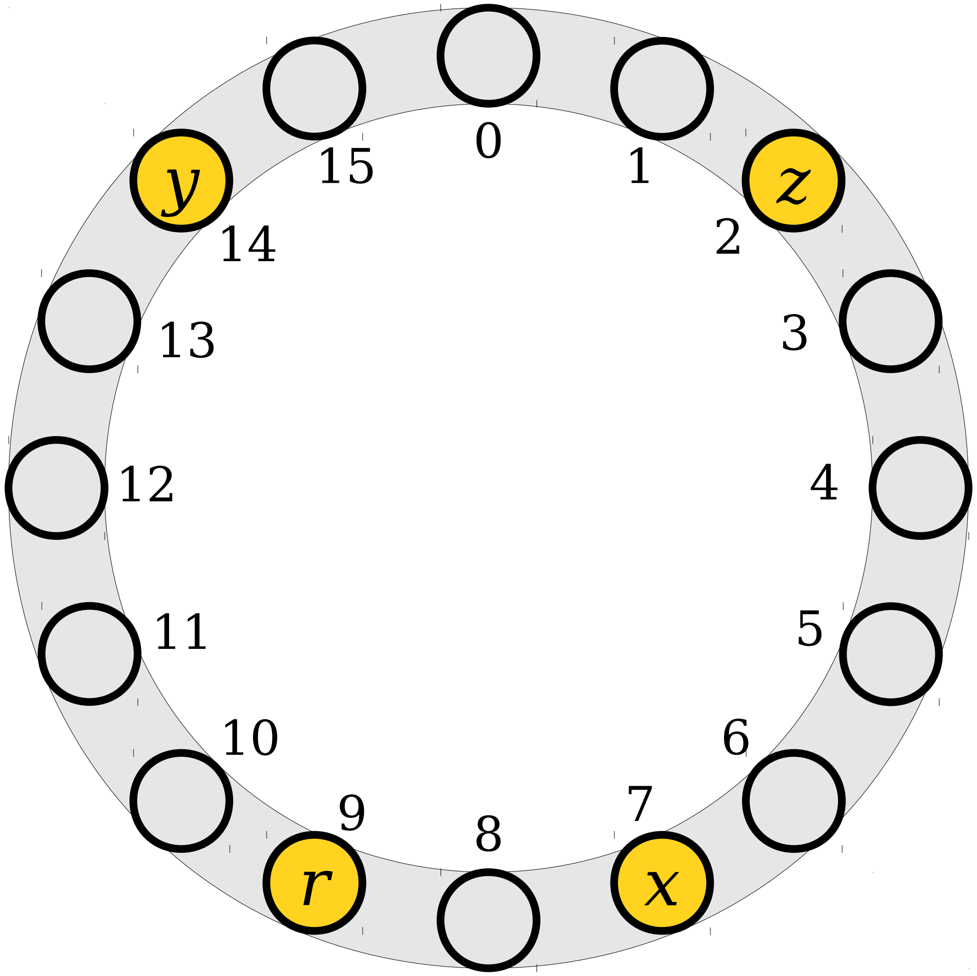
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



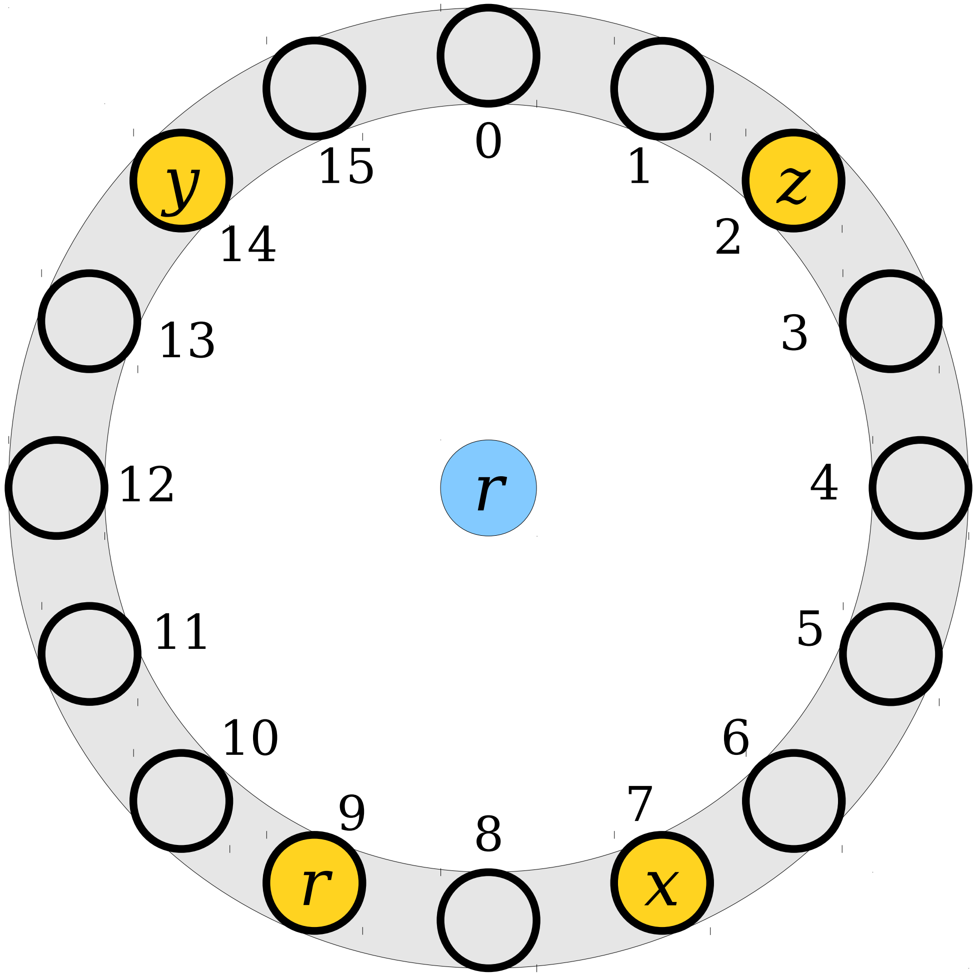
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



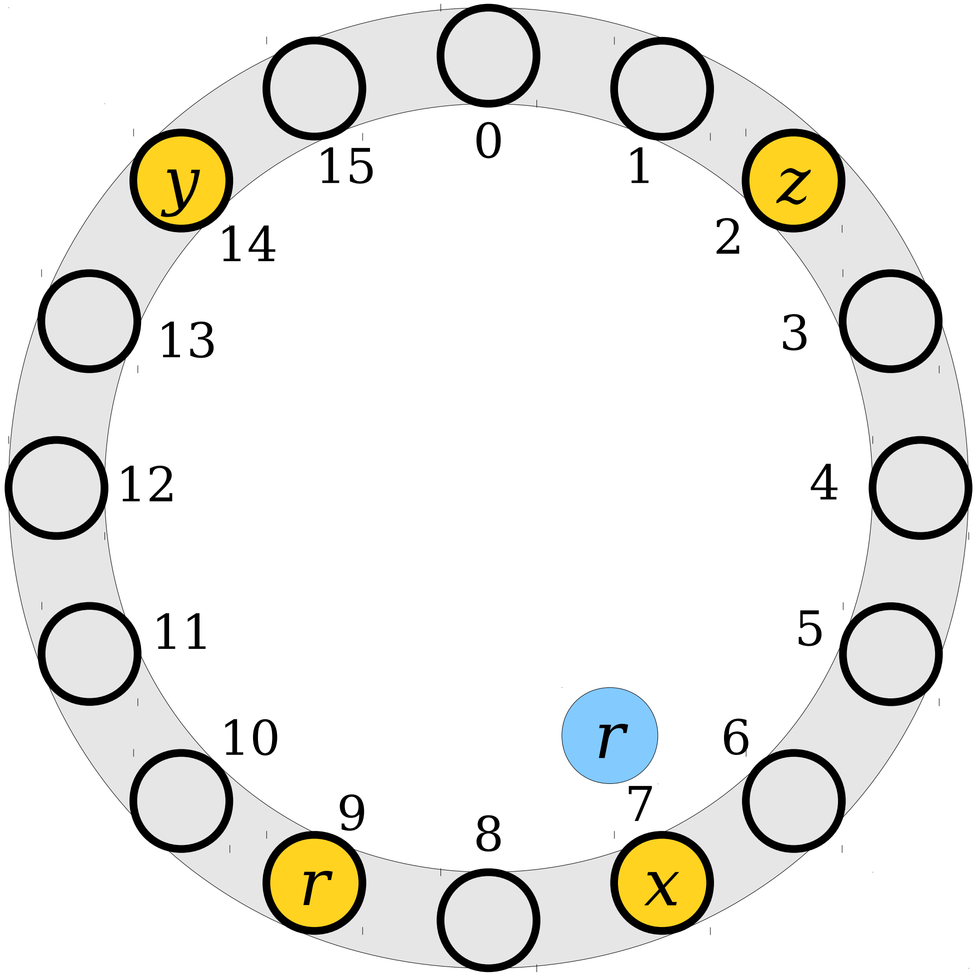
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



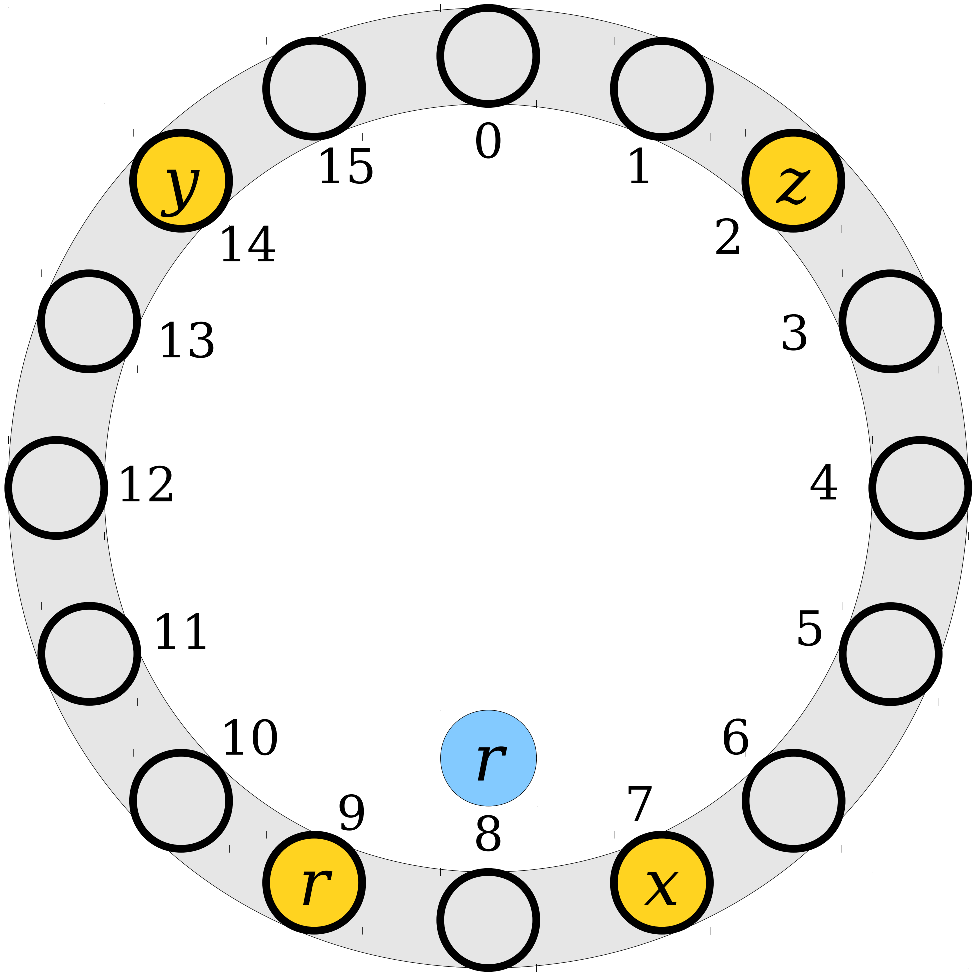
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



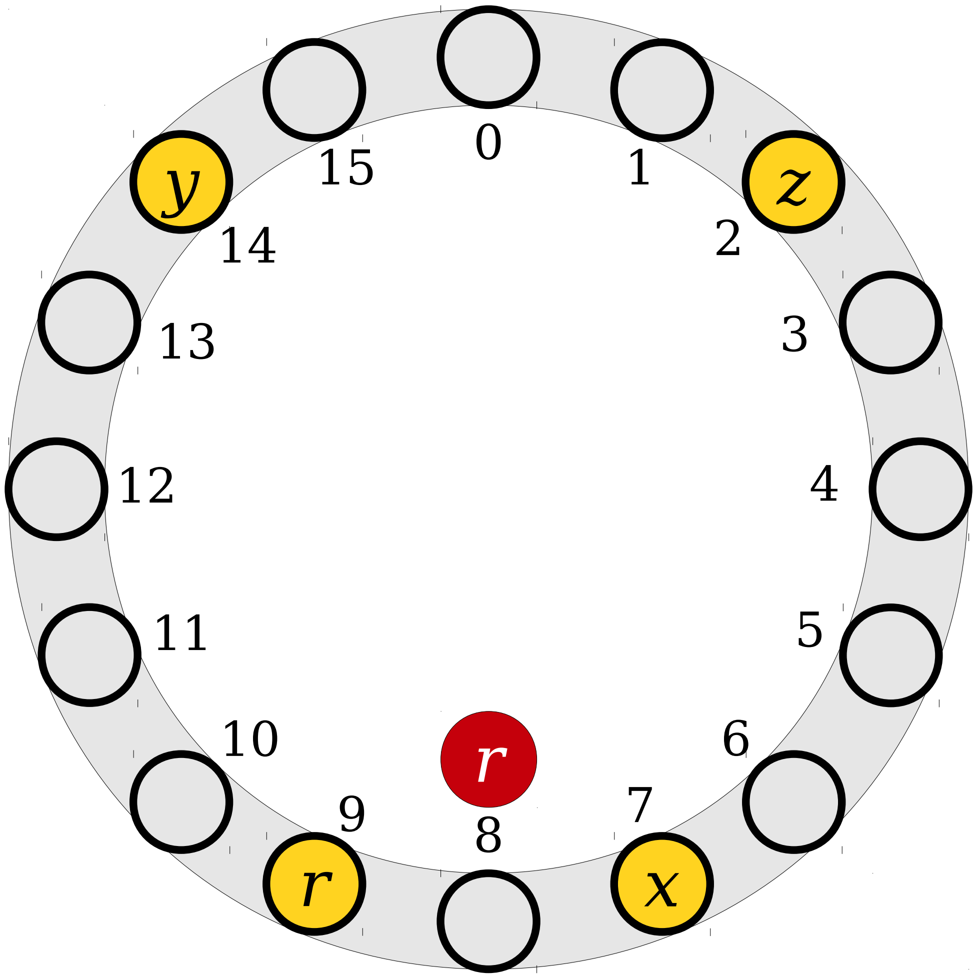
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



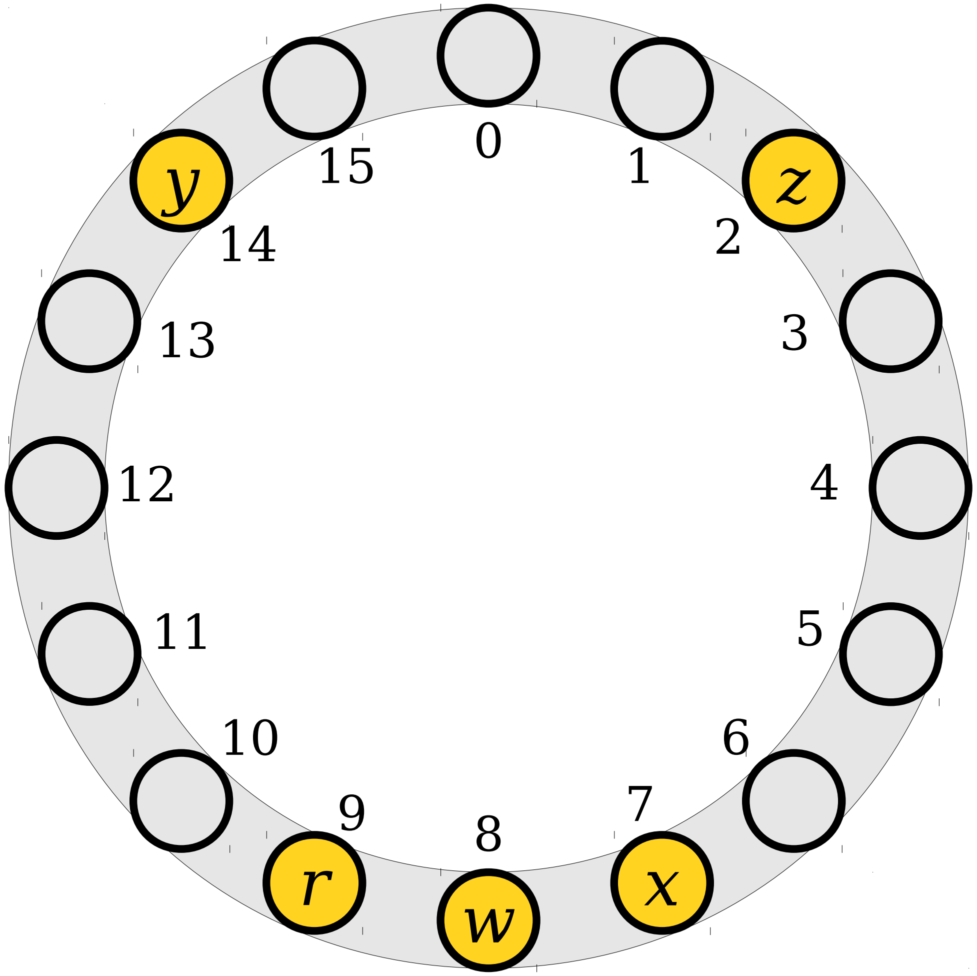
# Linear Probing

- Deletions are a bit trickier than in chained hashing.
- We cannot just do a search and remove the element where we find it.
- Why?



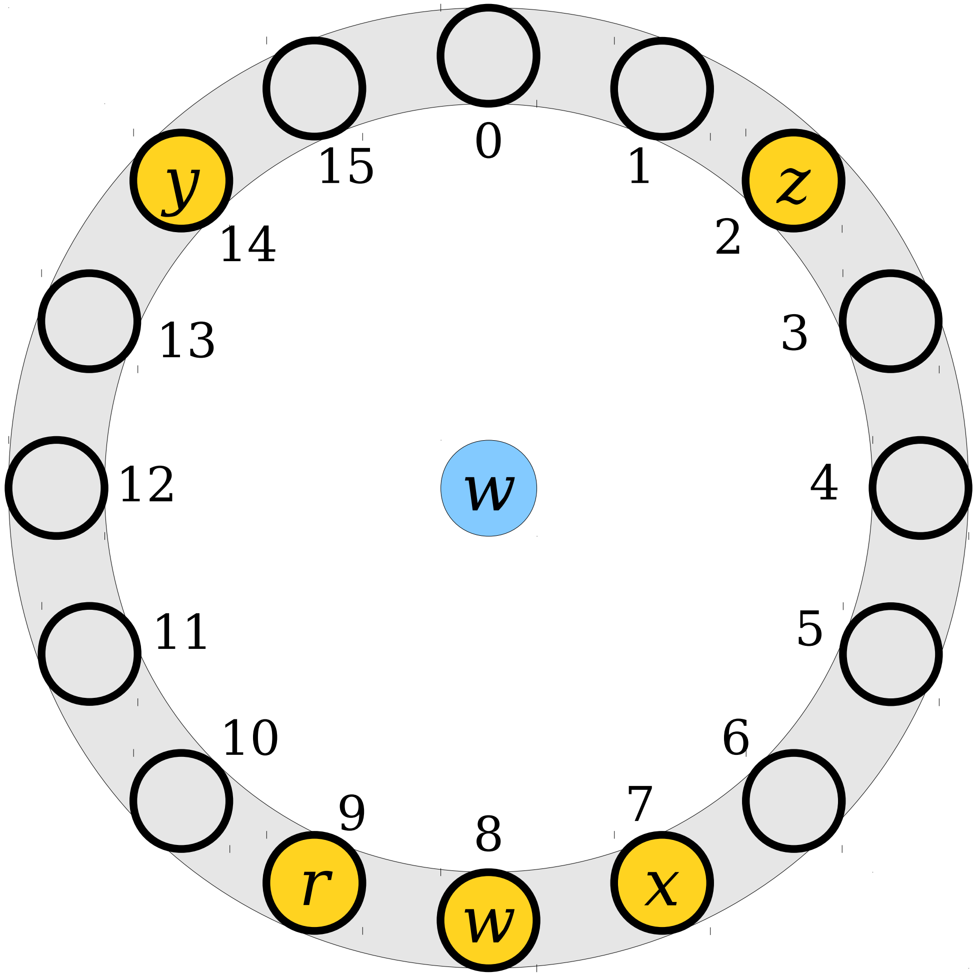
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
    - You need to watch out for wraparounds.
    - When inserting, feel free to replace any tombstone you encounter.



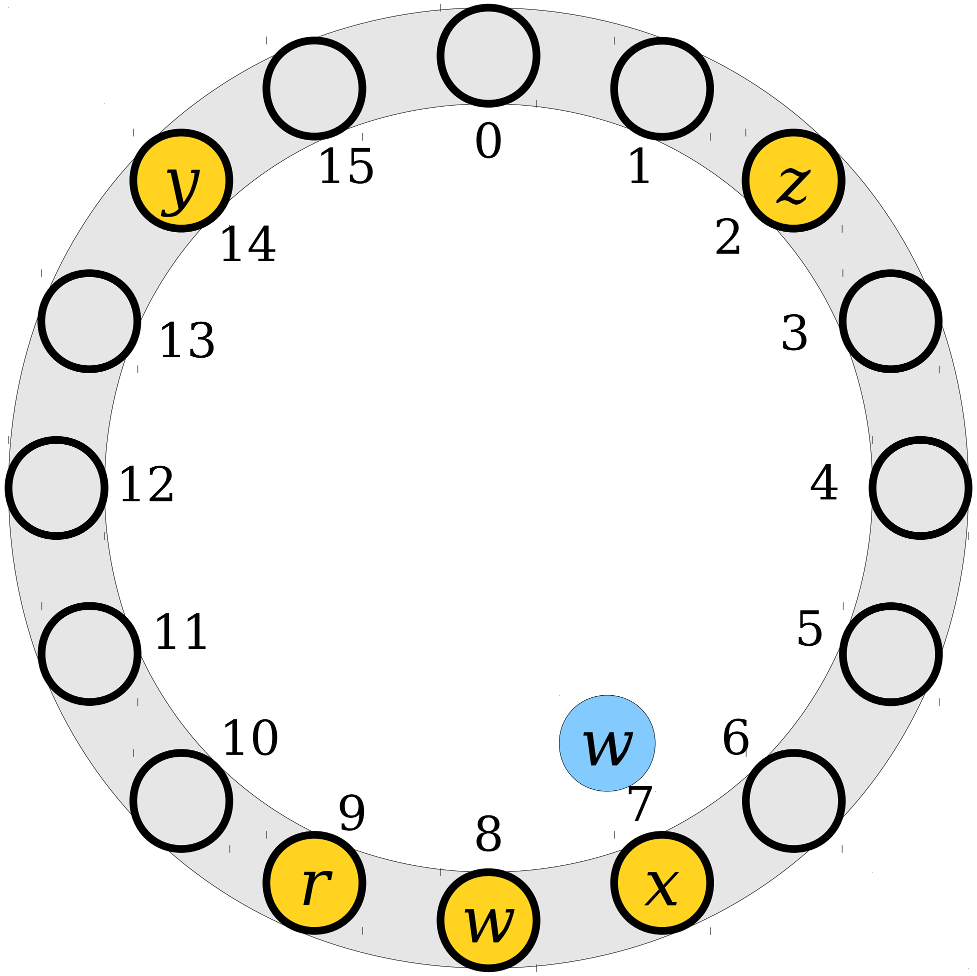
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
    - You need to watch out for wraparounds.
    - When inserting, feel free to replace any tombstone you encounter.



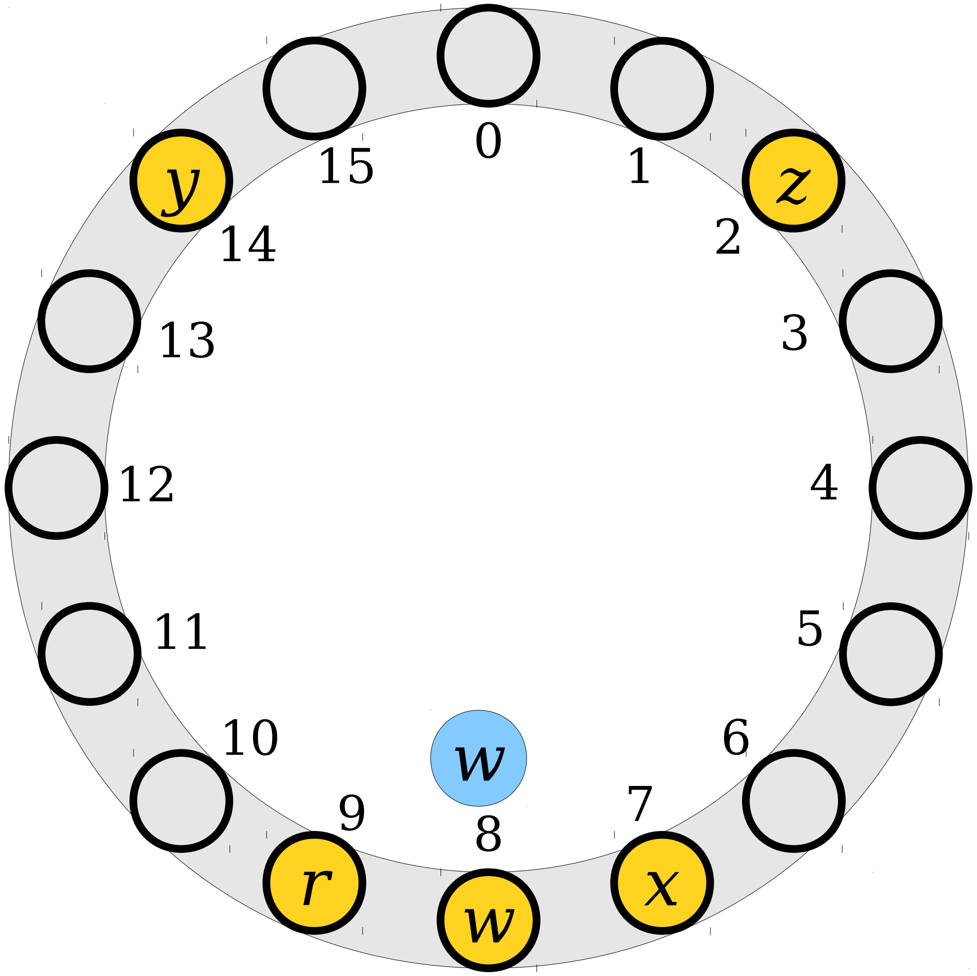
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
  - You need to watch out for wraparounds.
  - When inserting, feel free to replace any tombstone you encounter.



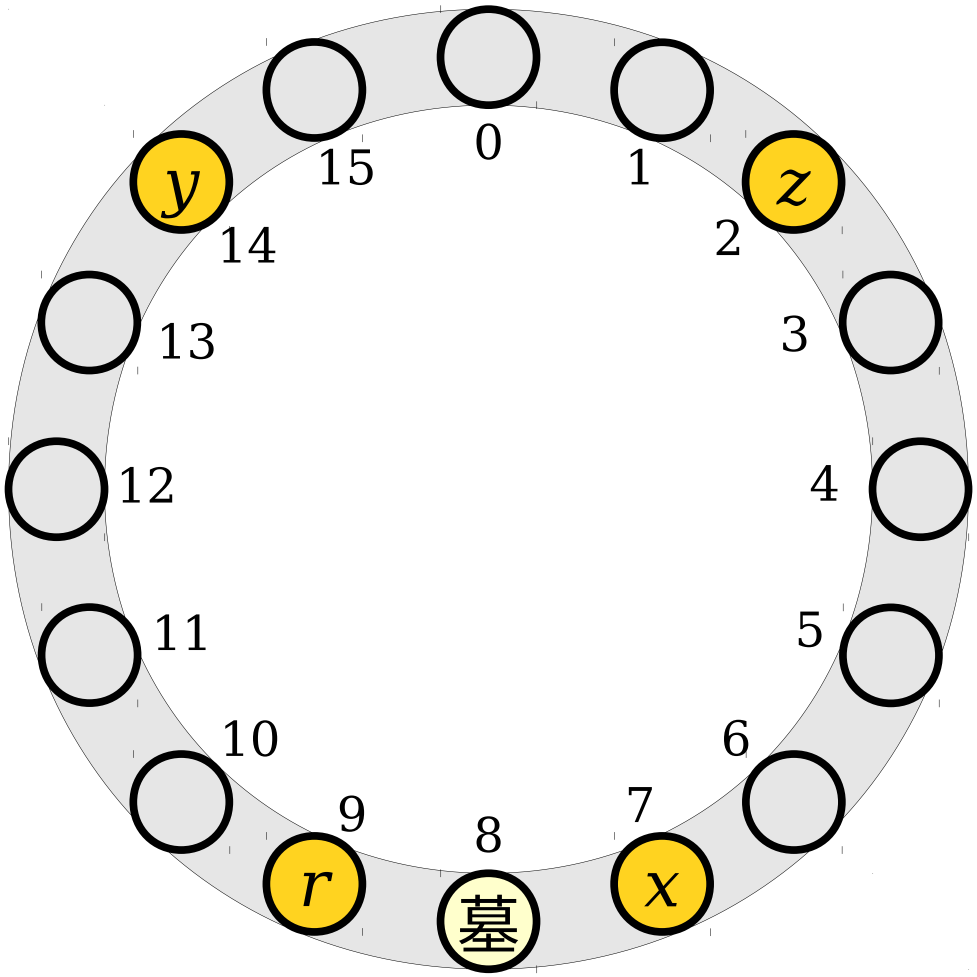
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
    - You need to watch out for wraparounds.
    - When inserting, feel free to replace any tombstone you encounter.



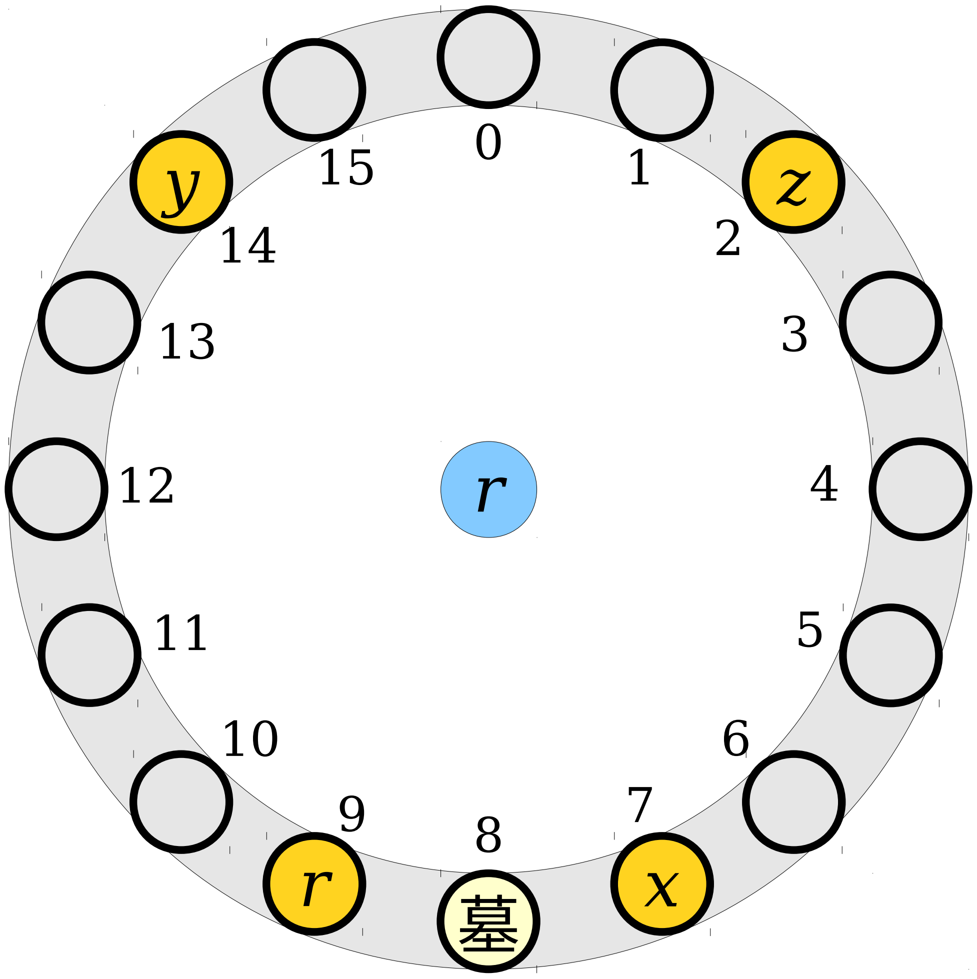
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
  - You need to watch out for wraparounds.
  - When inserting, feel free to replace any tombstone you encounter.



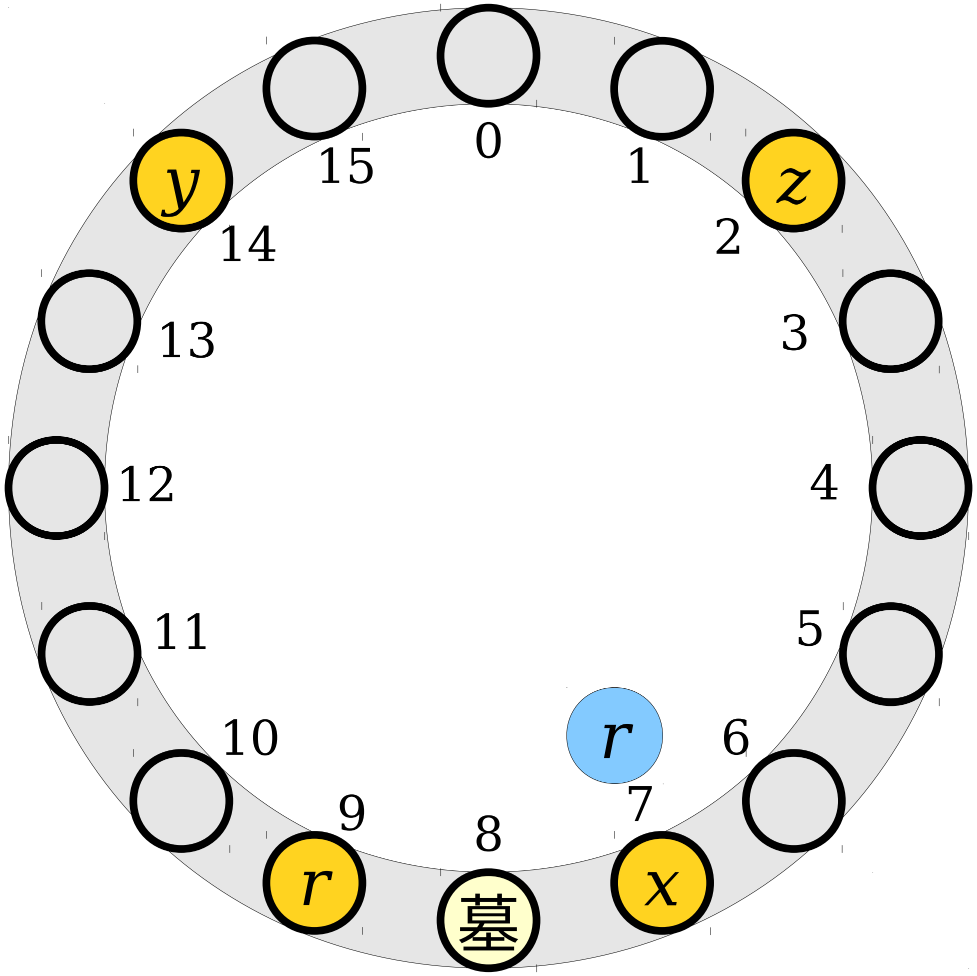
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
  - You need to watch out for wraparounds.
  - When inserting, feel free to replace any tombstone you encounter.



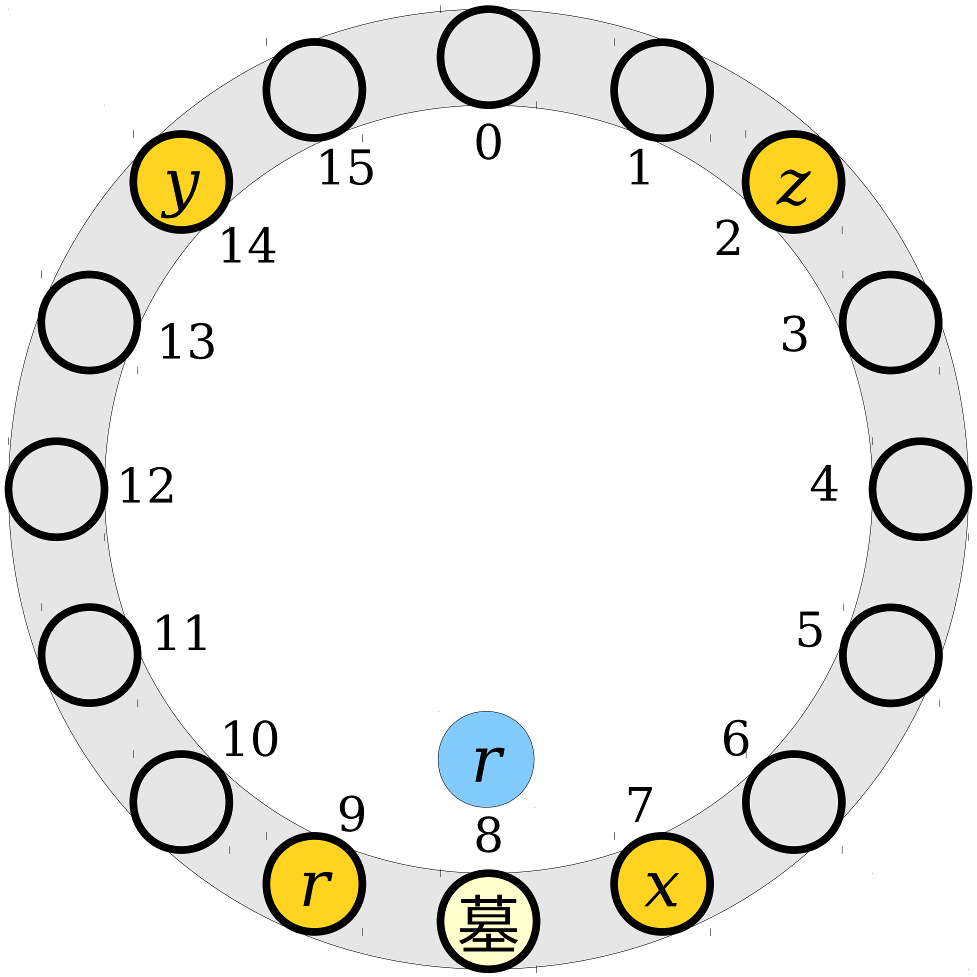
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
  - You need to watch out for wraparounds.
  - When inserting, feel free to replace any tombstone you encounter.



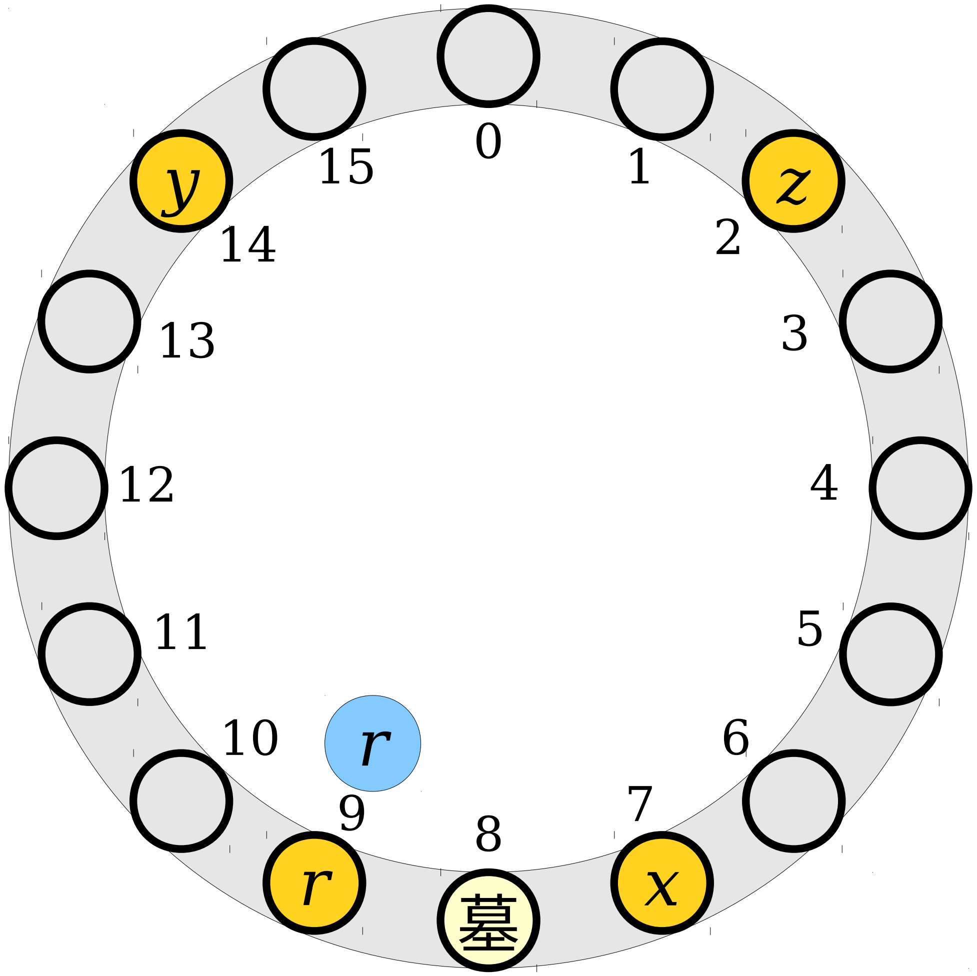
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
    - You need to watch out for wraparounds.
    - When inserting, feel free to replace any tombstone you encounter.



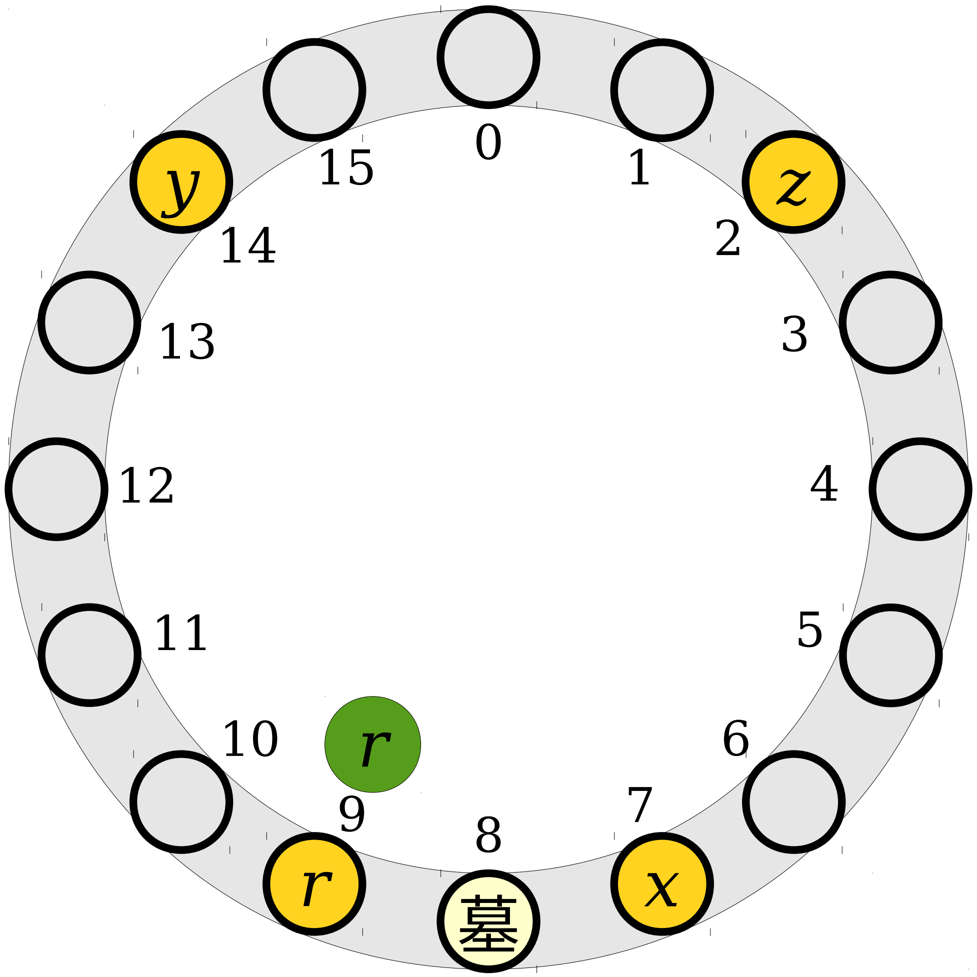
# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
  - You need to watch out for wraparounds.
  - When inserting, feel free to replace any tombstone you encounter.



# Linear Probing

- Deletions are often implemented using ***tombstones***.
- When removing an element, mark that the cell is empty and was previously occupied.
- When doing a lookup, don't stop at a tombstone. Instead, keep the search going.
    - You need to watch out for wraparounds.
    - When inserting, feel free to replace any tombstone you encounter.

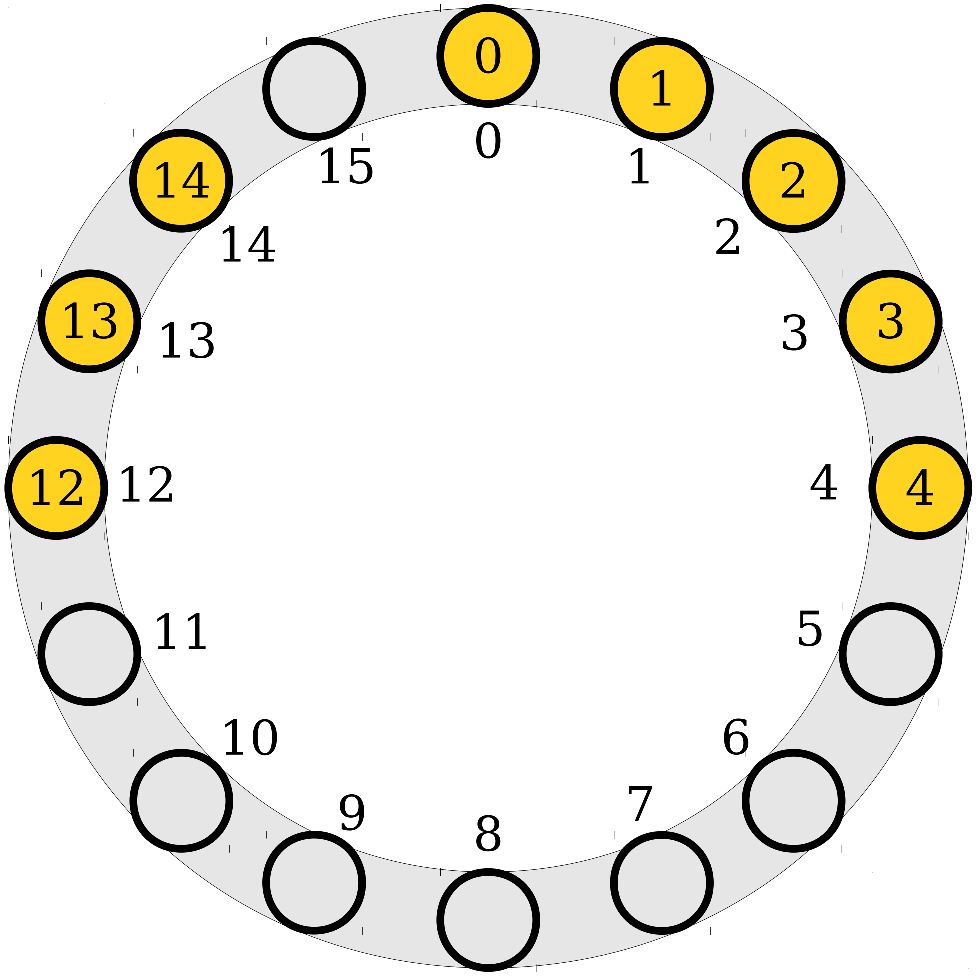


# Linear Probing in Practice

- In practice, linear probing is one of the fastest general-purpose hashing strategies available.
- This is surprising – it was originally invented in 1954! It's pretty amazing that it still holds up so well.
- Why is this?
  - ***Low memory overhead:*** just need an array and a hash function.
  - ***Excellent locality:*** when collisions occur, we only search in adjacent locations in the array.
  - ***Great cache performance:*** a combination of the above two factors.

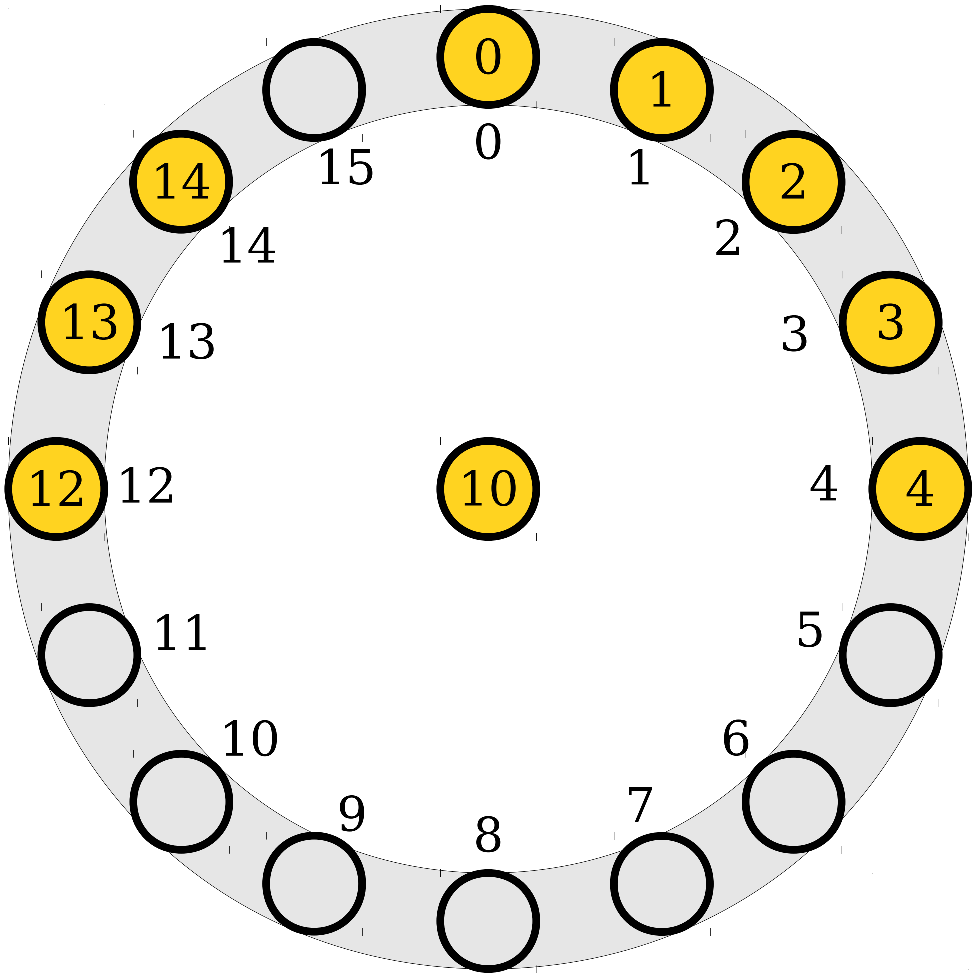
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



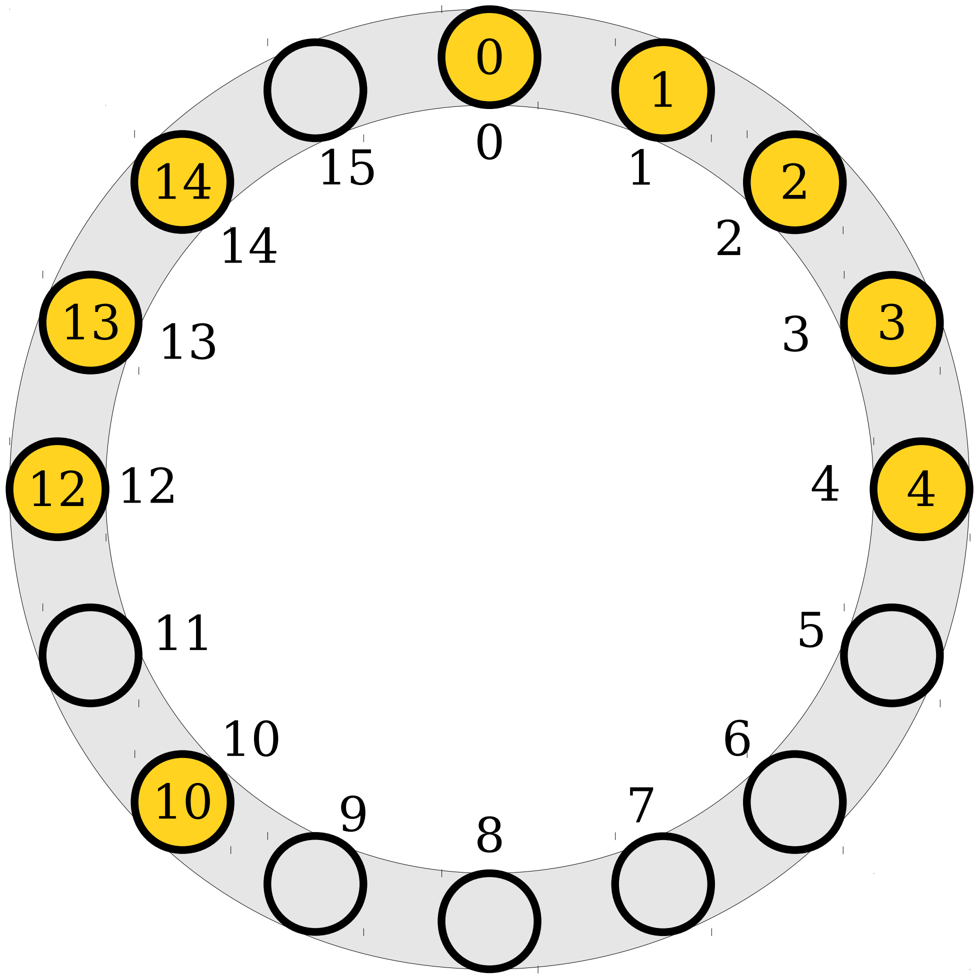
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



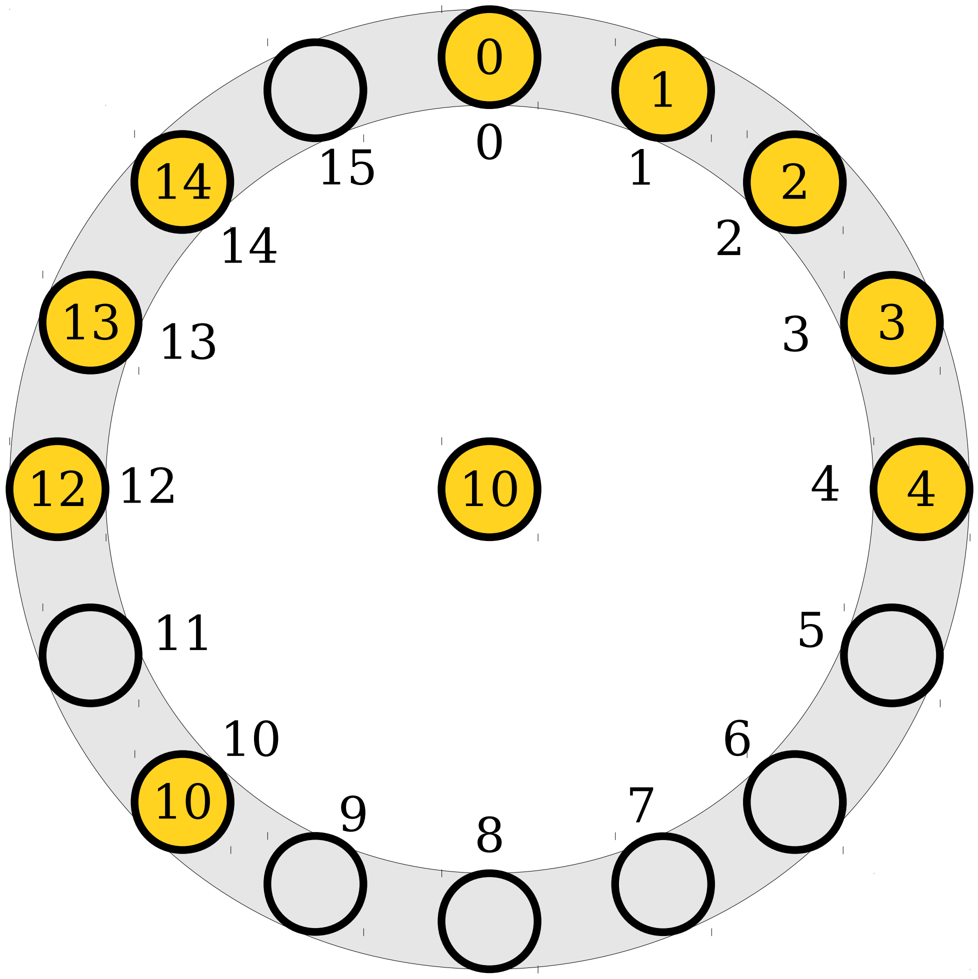
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



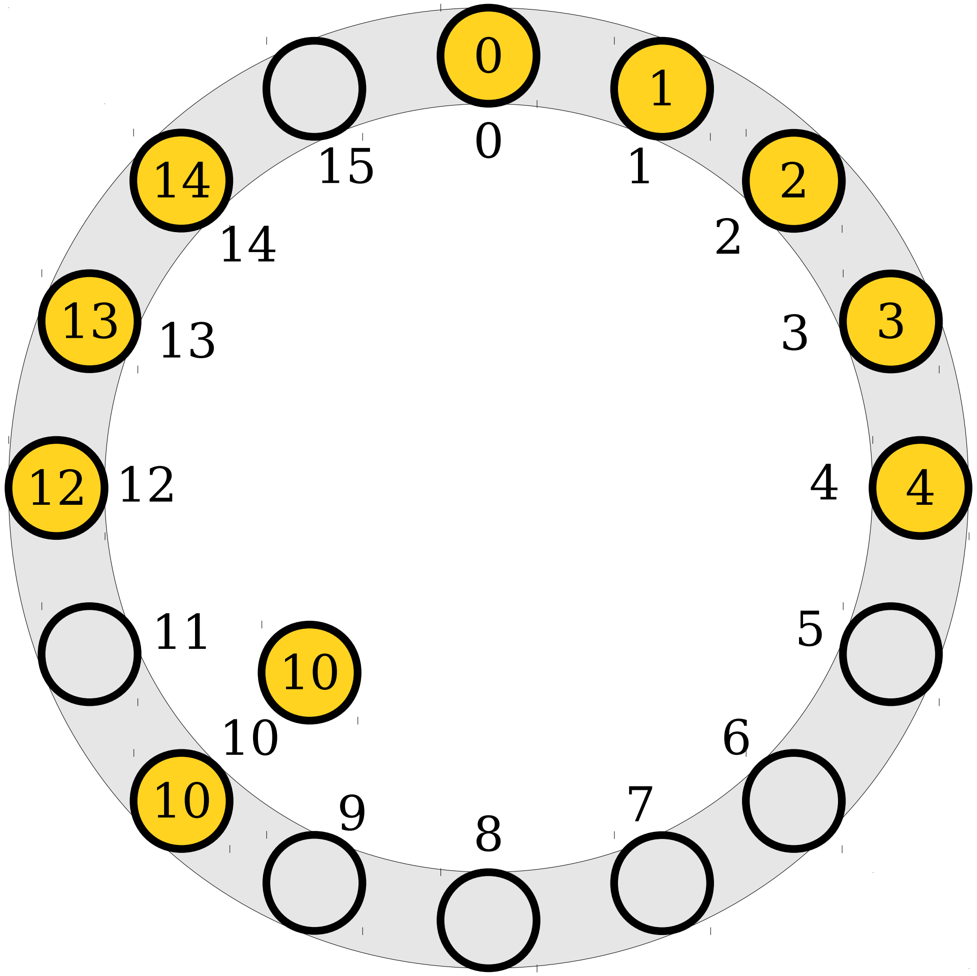
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



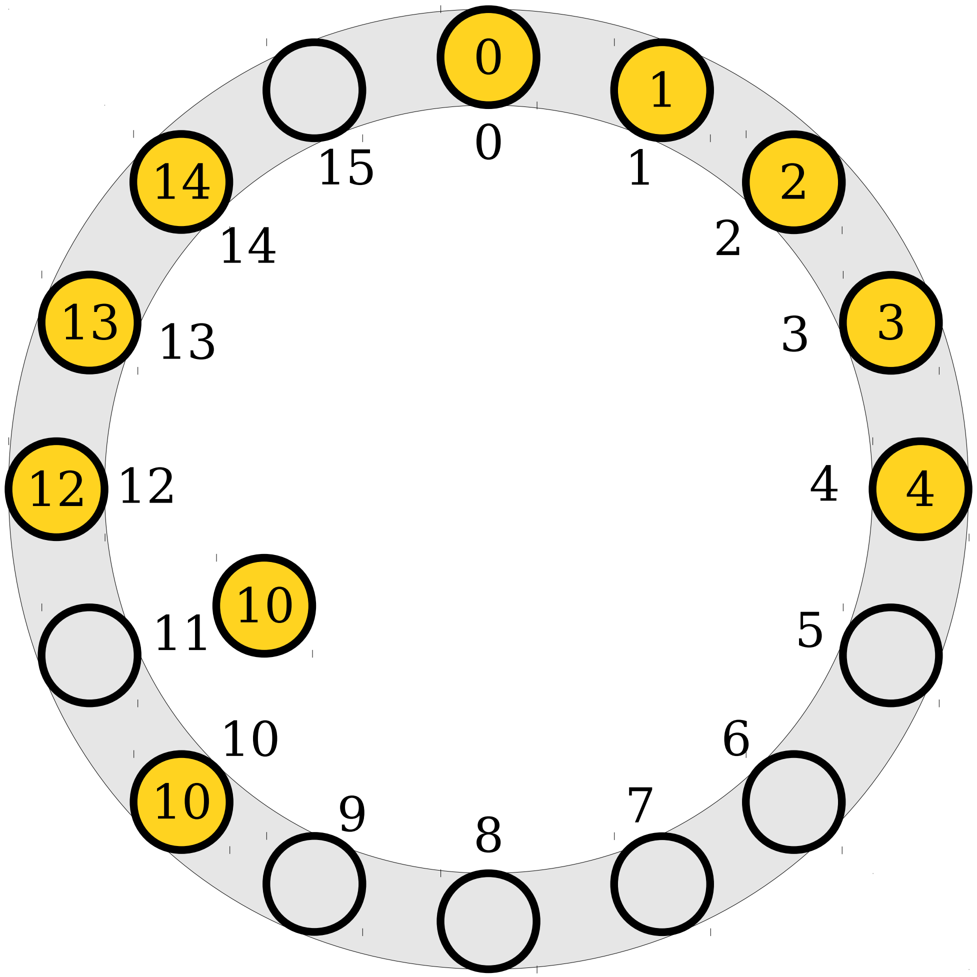
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



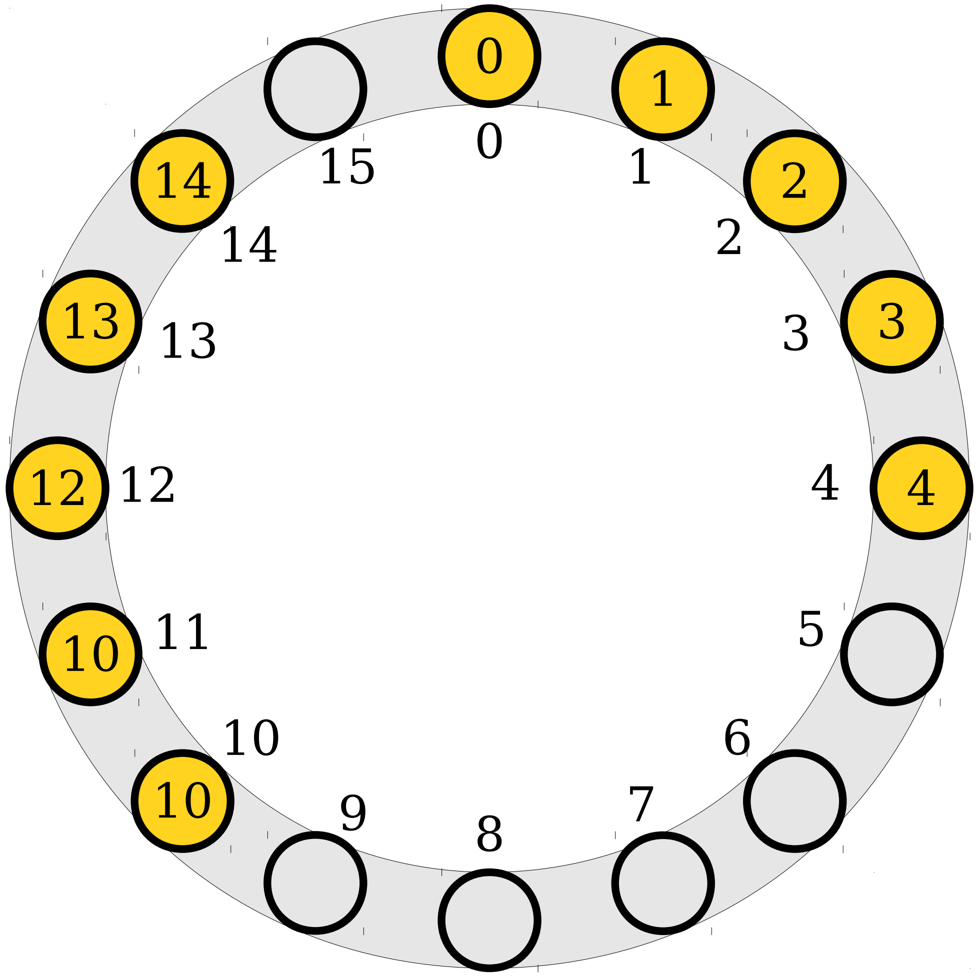
## The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



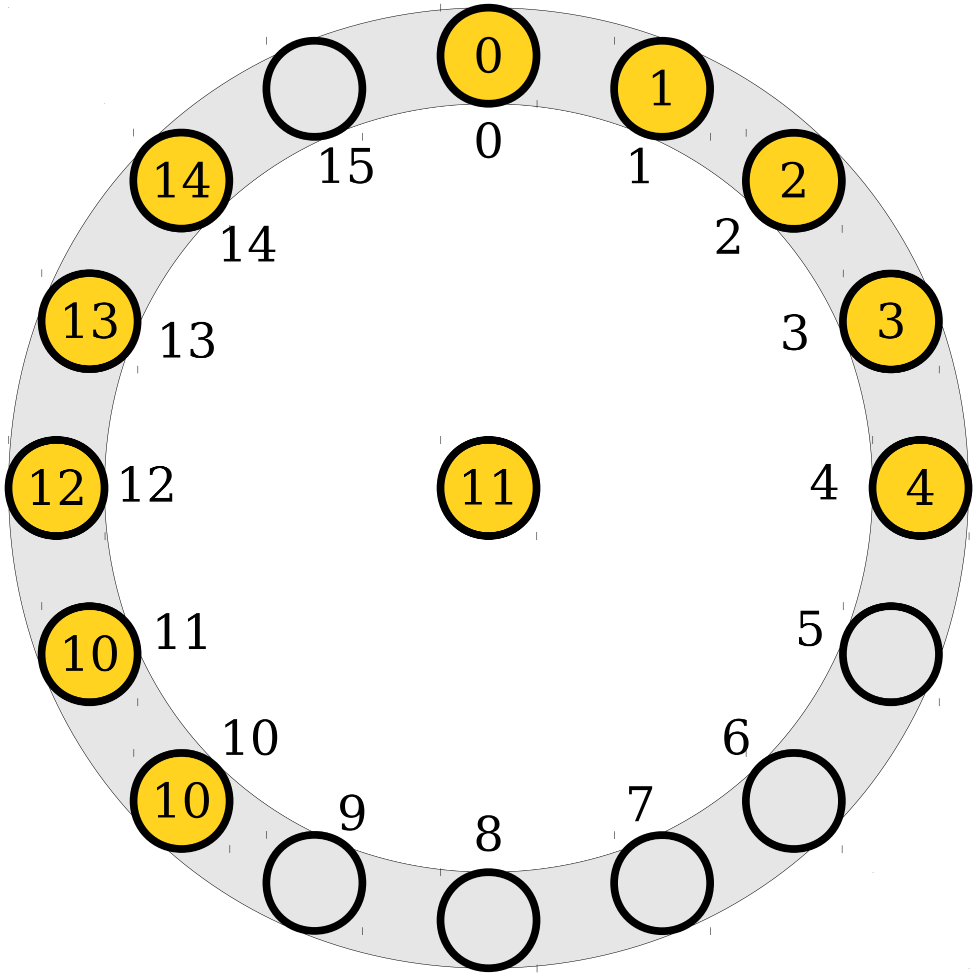
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



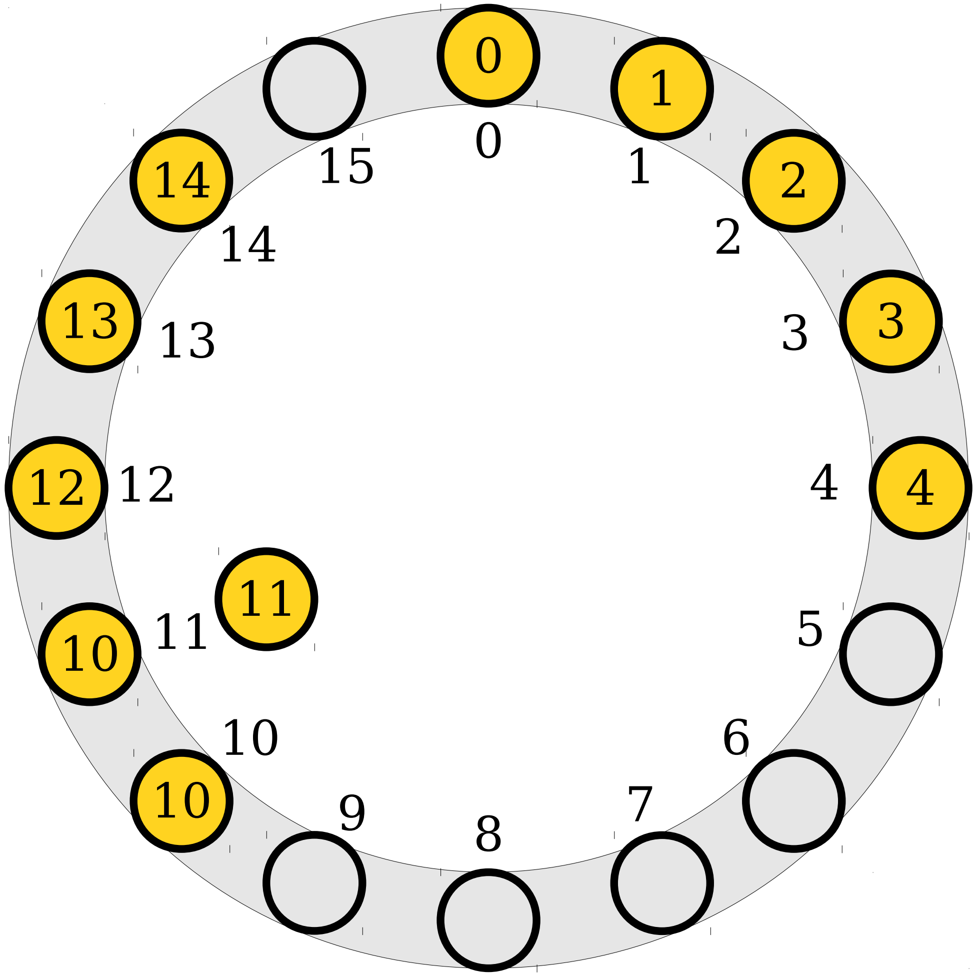
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



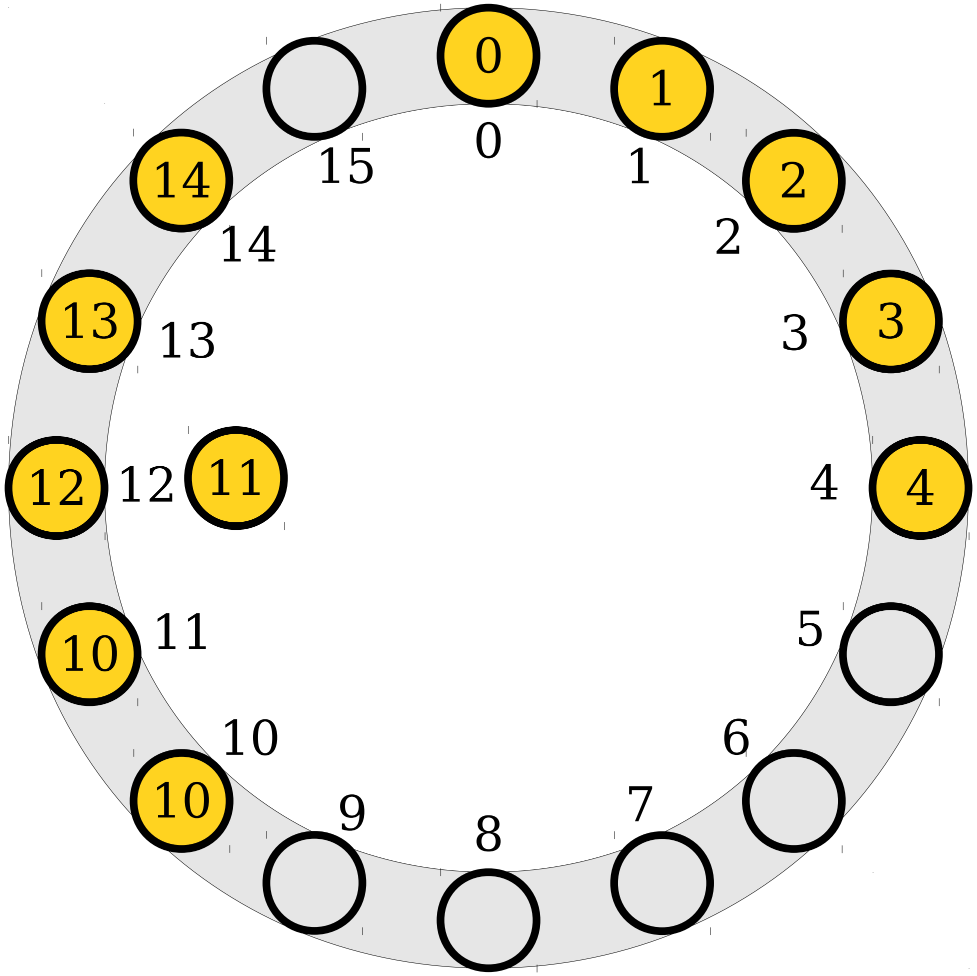
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



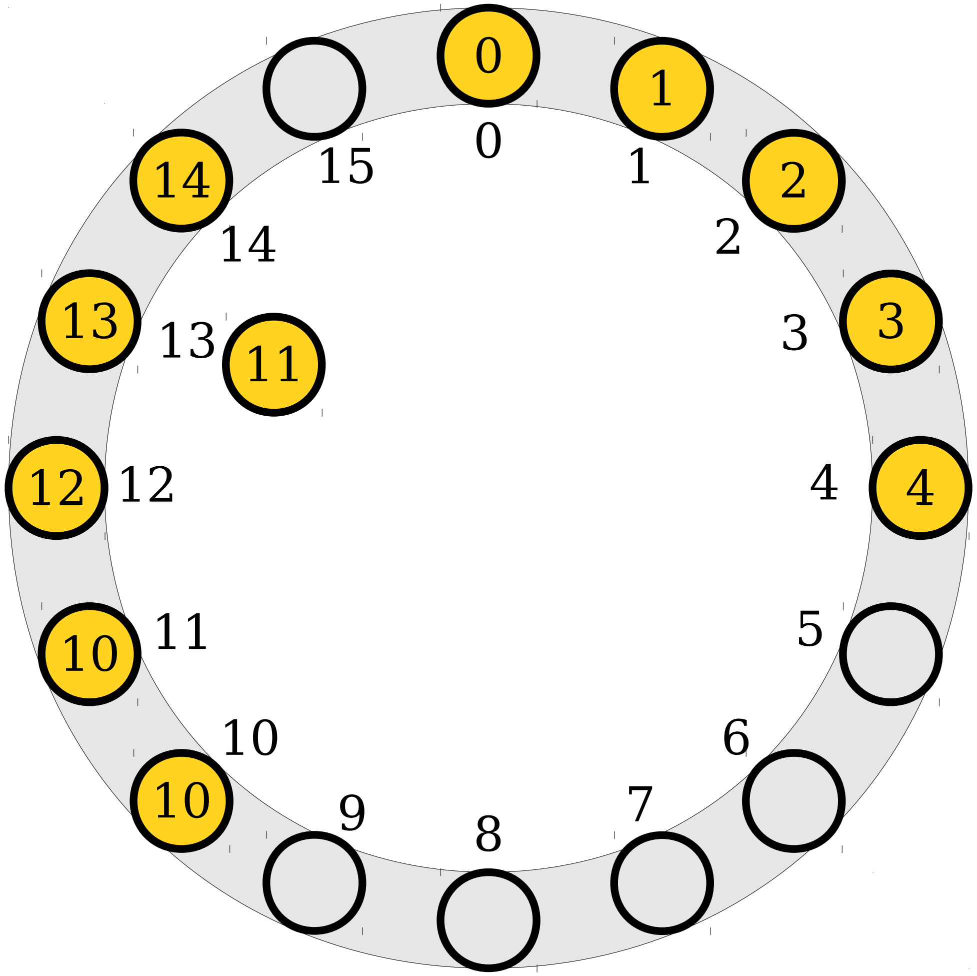
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



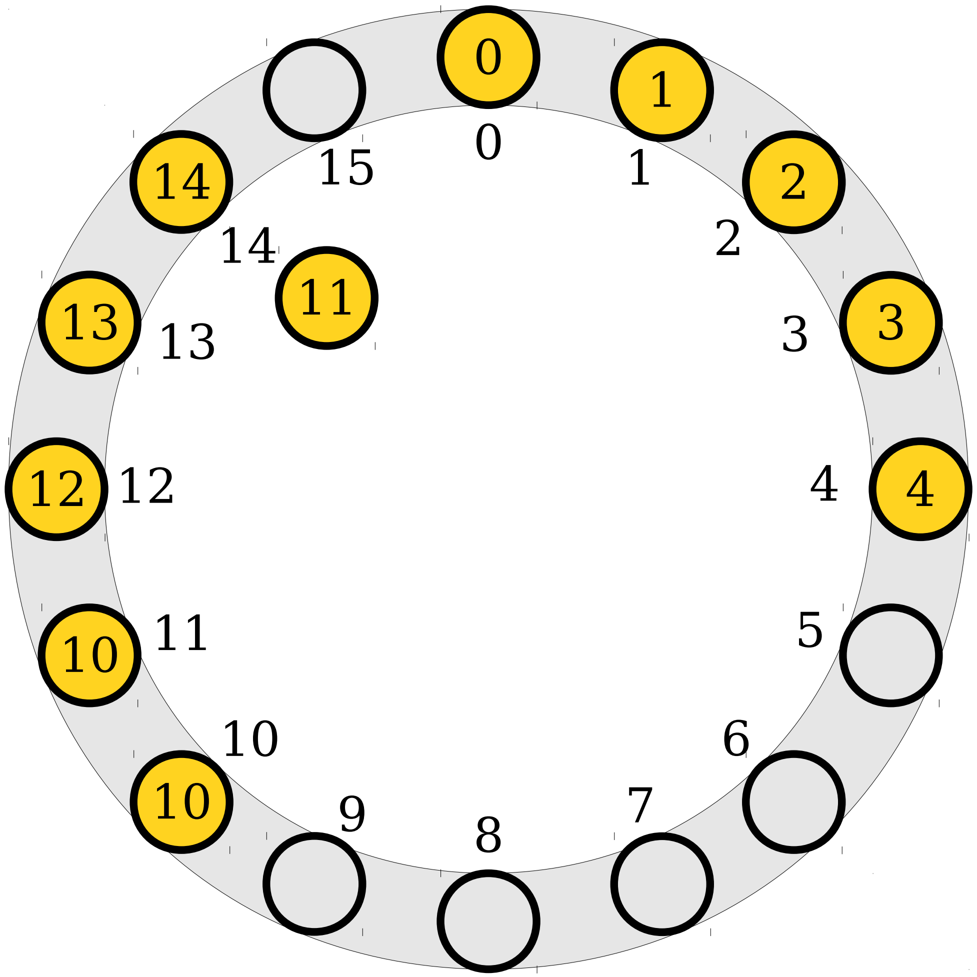
## The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



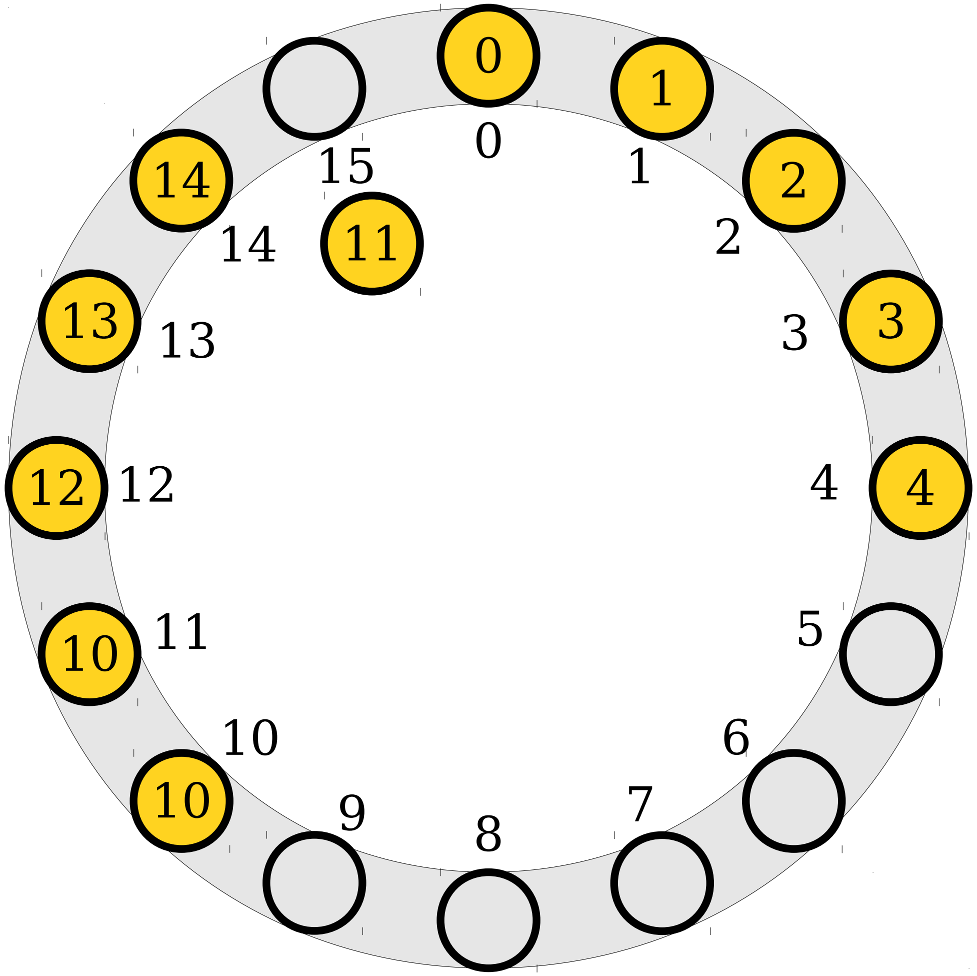
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



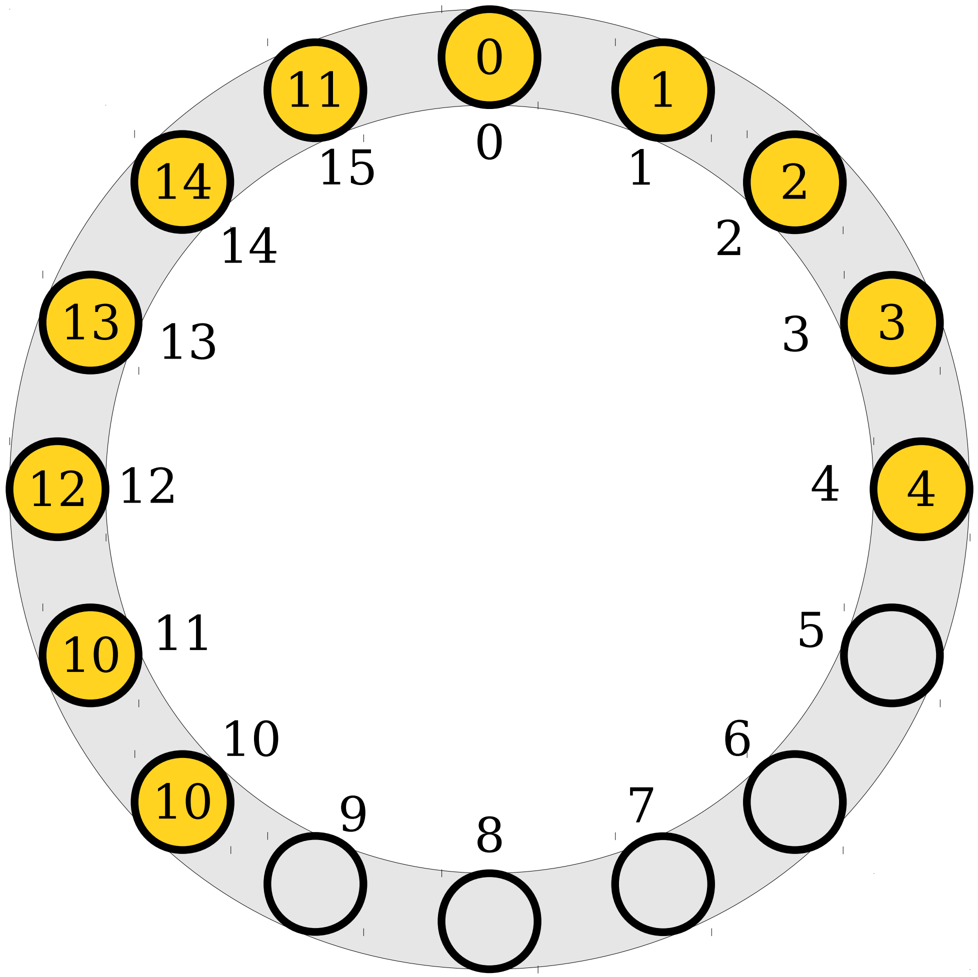
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



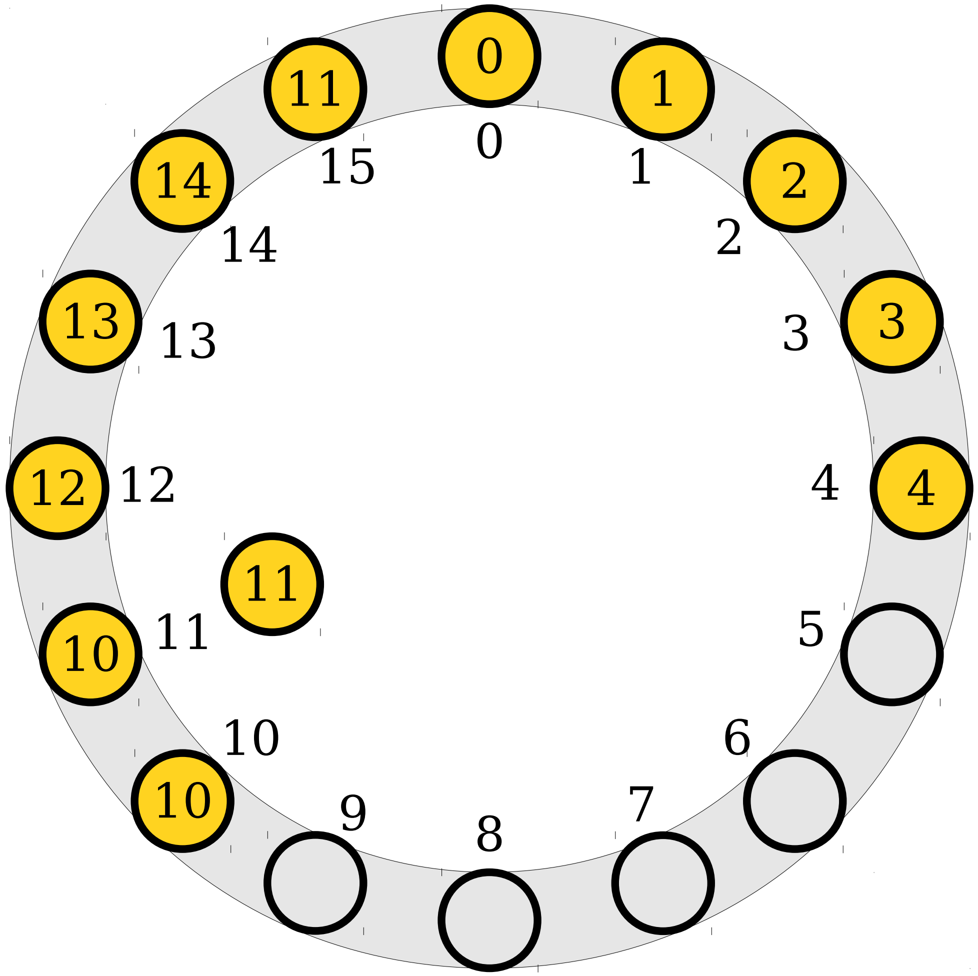
# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



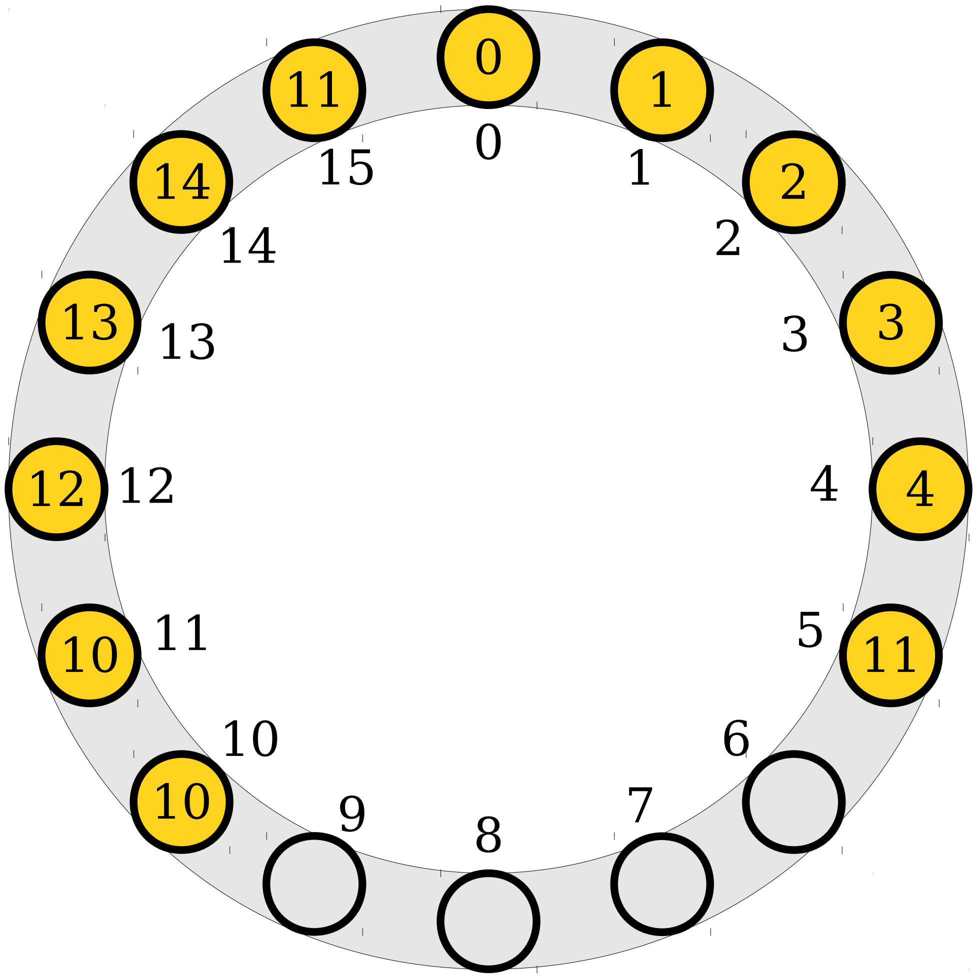
## The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



# The Weakness

- Linear probing exhibits severe performance degradations when the load factor gets high.
- The number of collisions tends to grow as a function of the number of existing collisions.
- This is called ***primary clustering***.



So… how fast is linear probing?

## Time-Out for Announcements!

## Final Project Topics

- Final project topics have been assigned, and we’re really excited to see what you end up making!
- We recommend that you make slow and steady progress on the project over the next couple of weeks.
- We’ll work out a presentation schedule in a week or so.

## Problem Sets

- Problem Set Four is due this Thursday at 2:30PM.
  - Have questions? Stop by office hours or ask on Piazza!
- We're working on grading PS3 right now and will try to get it back to you soon.
- PS5 will go out on Thursday and will be due one week from this Thursday.
- And that's it!

## Later This Week

- Keith will be out of town through the end of the week.
- Rafa and Mitchell will be covering Keith’s office hours at the regular time (2PM – 4PM) in the Huang Basement.
- Sam will be giving Thursday’s lecture on cuckoo hashing (super interesting stuff!)

# CS + SOCIAL GOOD T-SHIRT DRIVE

**White Plaza**

**May 17th**

**1:30-3:30 PM**

Do you have any extra clothes from career fairs or events that you know you're not going to wear? Want to get a head start on your end-of-year cleaning? Why not donate your T-shirts to a good cause *and* get free boba? We're collecting clothes to donate to children's hospitals and other organizations. Trade in 2 T-shirts or other articles of clothing for 1 Sharetea boba!* Bring as many items as you want!

*limit 1 boba per person

# GTGTC Exec Applications

- Girls Teaching Girls to Code (GTGTC) is looking for people to serve on next year's executive committee.
- This is an excellent program that's been around for years. It's a great way to make an impact.
- Interested? Apply **here** by this Sunday.

Back to CS166!

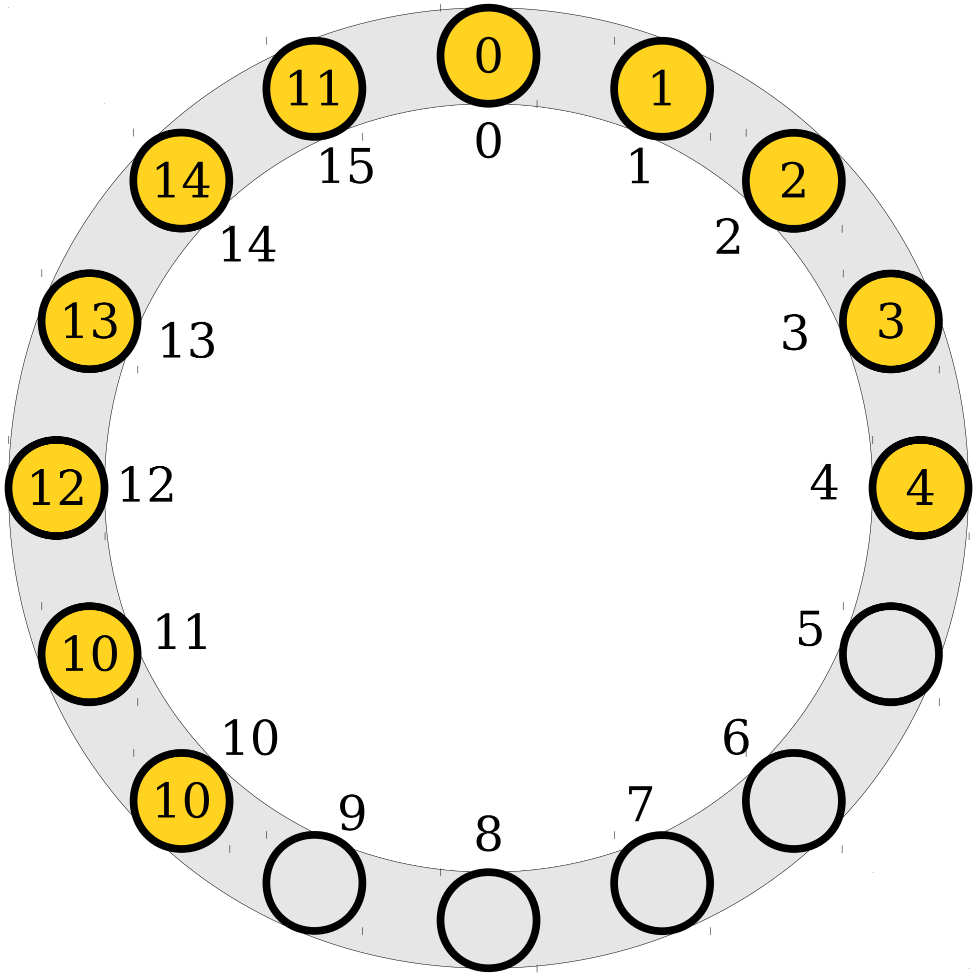
## Analyzing Linear Probing

You probably saw an analysis of chained hash tables in CS161.

What makes linear probing different, interesting, or noteworthy?

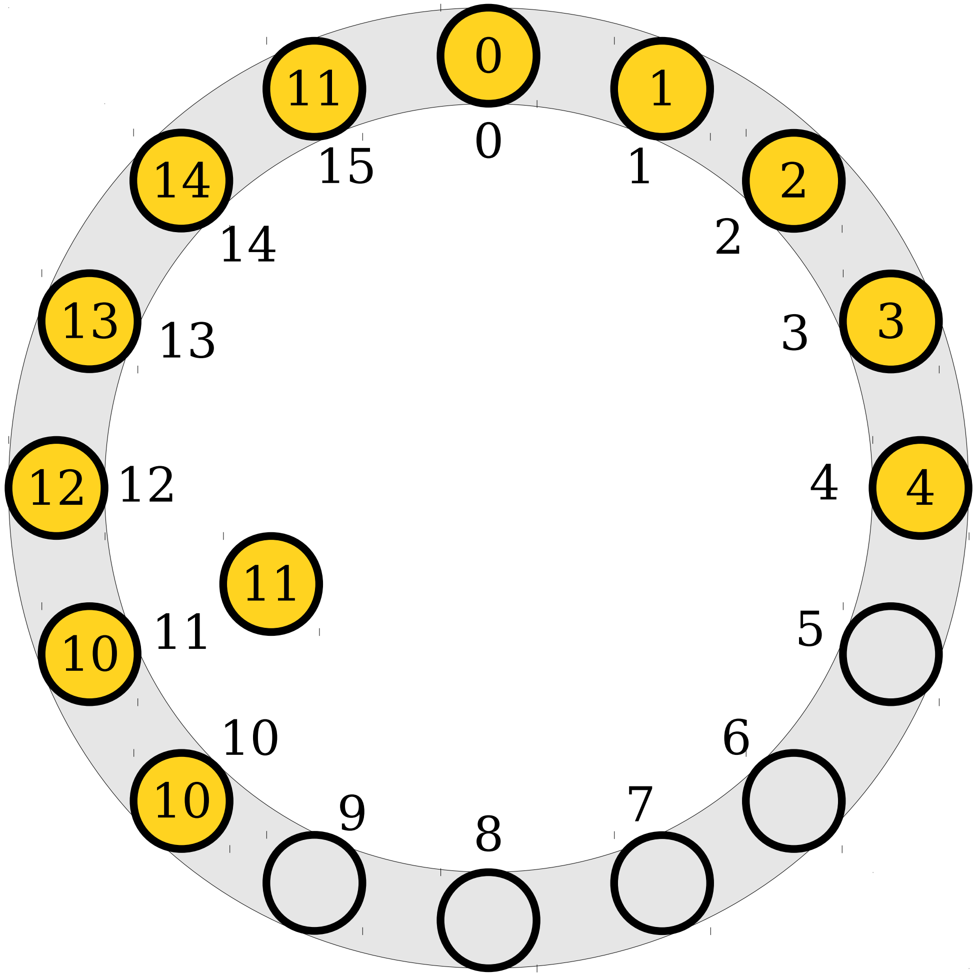
# Why Linear Probing is Different

- In chained hashing, collisions only occur when two values have exactly the same hash code.
- In linear probing, collisions can occur between elements with entirely different hash codes.
- To analyze linear probing, we need to know more than just how many elements collide with us.



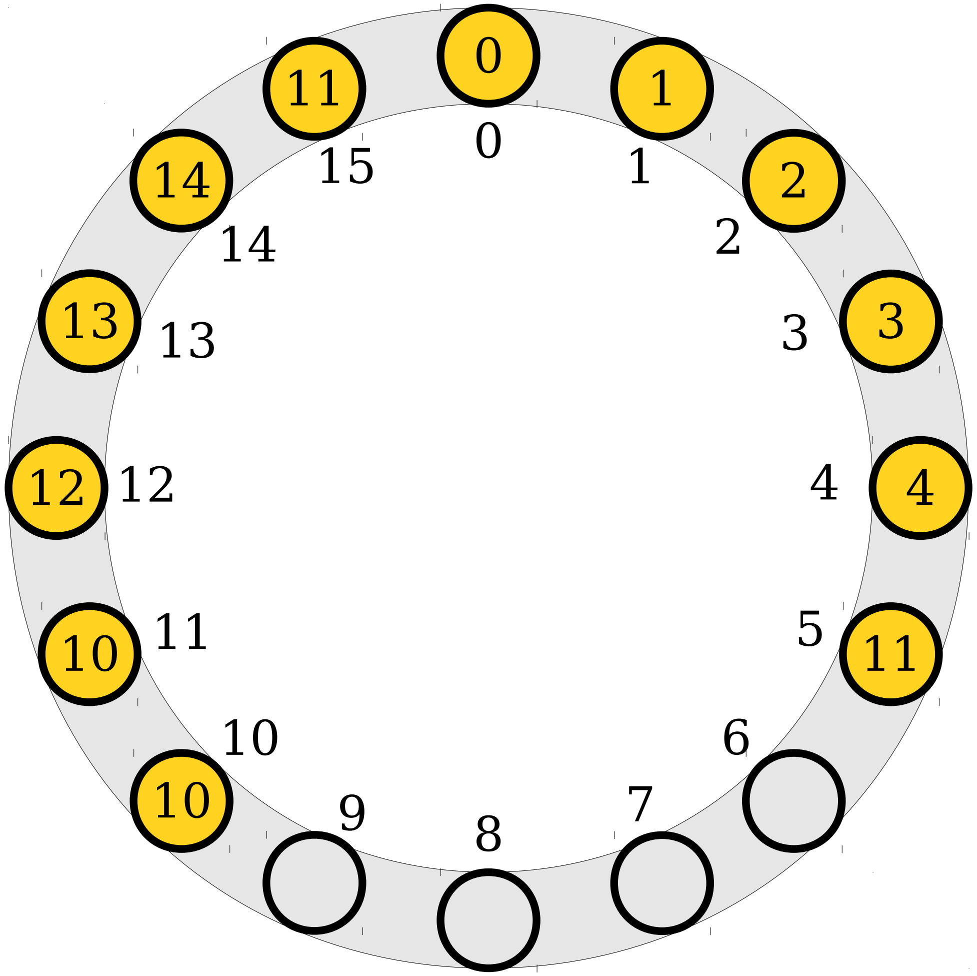
# Why Linear Probing is Different

- In chained hashing, collisions only occur when two values have exactly the same hash code.
- In linear probing, collisions can occur between elements with entirely different hash codes.
- To analyze linear probing, we need to know more than just how many elements collide with us.



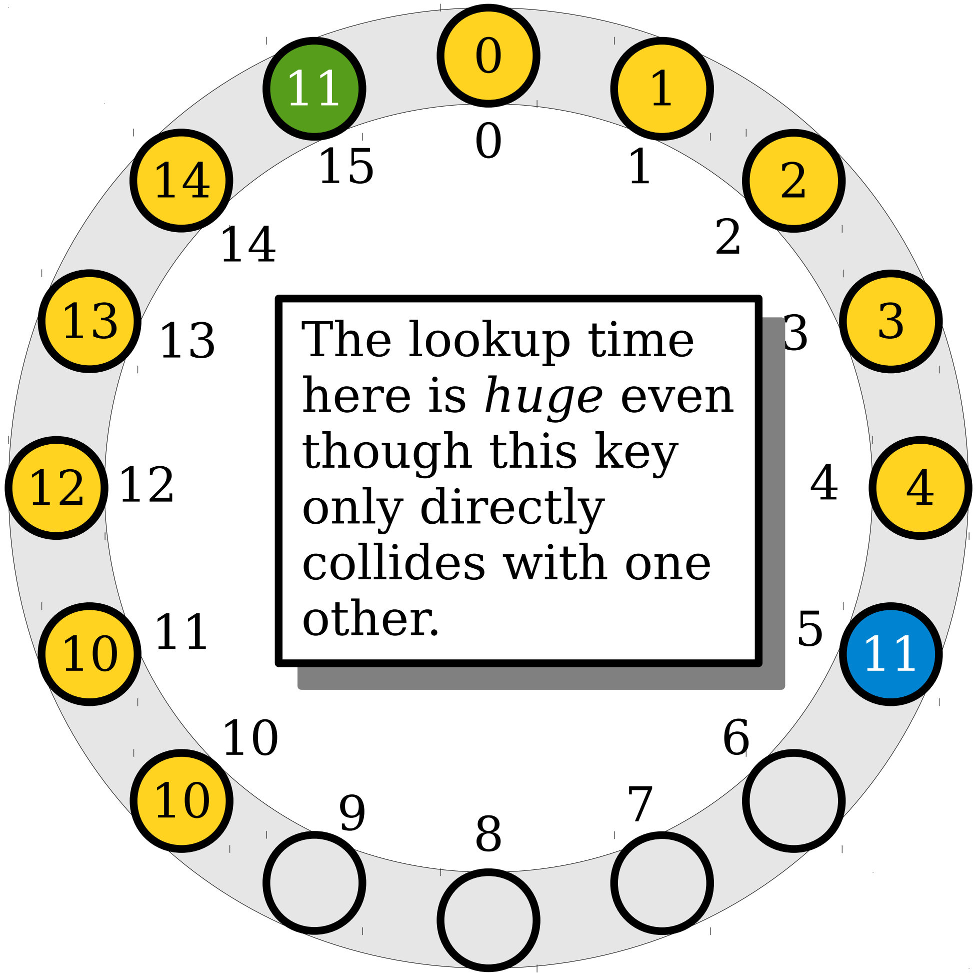
# Why Linear Probing is Different

- In chained hashing, collisions only occur when two values have exactly the same hash code.
- In linear probing, collisions can occur between elements with entirely different hash codes.
- To analyze linear probing, we need to know more than just how many elements collide with us.



# Why Linear Probing is Different

- In chained hashing, collisions only occur when two values have exactly the same hash code.
- In linear probing, collisions can occur between elements with entirely different hash codes.
- To analyze linear probing, we need to know more than just how many elements collide with us.



The lookup time here is *huge* even though this key only directly collides with one other.

# Some Brief History

- In 1954, Gene Amdahl, Elaine McGraw, and Arthur Samuel invent linear probing as a subroutine for an assembler.
- In 1962, Don Knuth, in his first ever analysis of an algorithm, proves that linear probing takes expected time O(1) for lookups if the hash function is truly random (***n***-wise independence).
- In 1995, Schmidt and Siegel proved **O(log *n*)**-independent hash functions guarantee fast performance for linear probing, but note that such hash functions either take a long time to evaluate or require a lot of space.
- In 2006, Anna Pagh et al. proved that **5**-independent hash functions give expected constant-time lookups. (This is the analysis we'll see today.) These hash functions can be stored in O(1) space and evaluated in O(1) time.
- In 2007, Mitzenmacher and Vadhan proved that **2**-independence will give expected O(1)-time lookups, assuming there's some measure of randomness in the keys.
- In 2010, Pătrașcu and Thorup proved that **5**-independence is the minimum independence needed for adversarially-chosen keys.

## The Analysis!

For simplicity, let's assume a load factor of $\alpha = 1/3$.

A ***region of size m*** is a consecutive set of $m$ locations in the hash table.

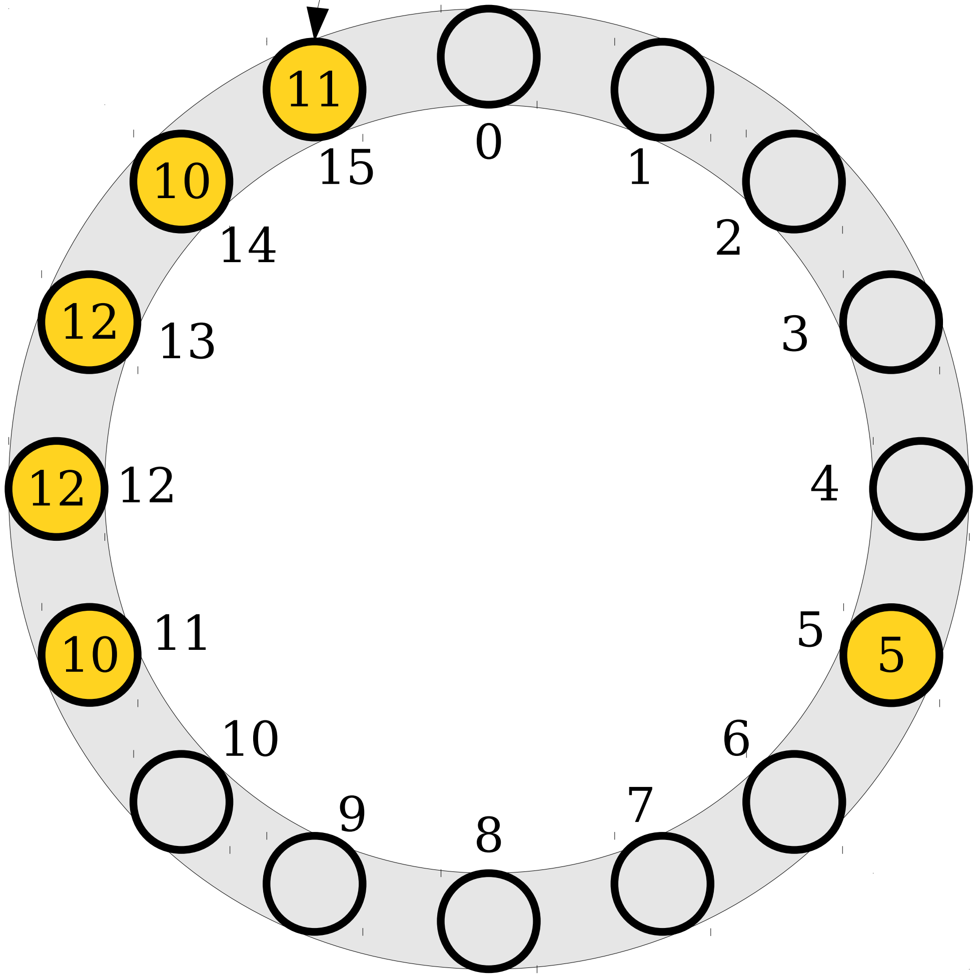
An element $x$ ***hashes to*** region $R$ if $h(x) \in R$, though $x$ may not be placed in $R$.

On expectation, a region of size $2^s$ should have at most $1/3 \cdot 2^s$ elements hash to it.

It would be very unlucky if a region had twice as many elements in it as expected.

A region of size $2^s$ is ***overloaded*** if at least $2/3 \cdot 2^s$ elements hash to it.

This element is far from home. ☹



***Intuition:*** If an element ends up far from its home location, then some large region near its home has to be overloaded.

***Theorem:*** The probability that an element $x_a$ ends up between $2^s$ and $2^{s+1}$ steps from its home location is upper-bounded by

**$c$ · Pr[ the region of size $2^s$ centered on $h(x_a)$ is overloaded ]**

for some fixed constant $c$ independent of $s$.

***Proof:*** Set up some cleverly-chosen ranges over the hash table and use the pigeonhole principle. See Thorup's lecture notes.

## Analyzing the Runtime

- The cost of looking up some key $x_a$ is bounded from above by the length of the run containing $x_a$.
- The expected cost of performing a lookup is therefore at most

$$O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^{s+1} \cdot \Pr[x_q \text{ is between } 2^s \text{ and } 2^{s+1} \text{ spots from home}]$$

- The previous theorem tells us that this cost is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^{s} \cdot \Pr[\text{the region of size } 2^s \text{ on } h(x_a) \text{ is overloaded}]}$$

- If we can determine the probability that a region of size $2^s$ is overloaded, we'll have a bound on the expected lookup cost for $x_a$.

# Overloaded Regions

- ***Recall:*** A region is a contiguous span of table slots, and we've chosen $\alpha = 1/3$.
- An overloaded region has at least $⅔ \cdot 2^s$ elements in it.
- Let the random variable $\boldsymbol{B_s}$ represent the number of keys that hash into the block of size $2^s$ centered on $h(x_a)$. We want to know

$$\Pr[\, B_s \geq ⅔ \cdot 2^s \,].$$

- Assuming our hash functions are at least 2-independent, we have $\mathrm{E}[B_s] = ⅓ \cdot 2^s$. Then the above quantity is equivalent to

$$\mathbf{Pr}[\, \boldsymbol{B_s} \geq \mathbf{2 \cdot E}[\boldsymbol{B_s}] \,],$$

and looking up an element takes, on expectation, time

$$\mathbf{O(1)} \cdot \sum_{s=0}^{\lceil \log n \rceil} \mathbf{2}^{s} \cdot \mathbf{Pr}[\boldsymbol{B_s} \geq \mathbf{2 \cdot E}[\boldsymbol{B_s}]]$$

# Concentration Inequalities

- The expression

$$\Pr[\ B_s \geq 2 \cdot E[B_s]\ ]$$

  seems like a perfect case to try to use a concentration bound, like we did last Thursday.
- Knowing nothing about $B_s$ other than the fact that it's nonnegative, we could start off by trying to use Markov's inequality:

$$\Pr[\ X \geq c\ ] \quad \leq \quad E[X] / c$$

- Using what we have:

$$\Pr[\ B_s \geq 2 \cdot E[B_s]\ ] \quad \leq \quad E[B_s] / 2 \cdot E[B_s] \quad = \quad \mathbf{½}.$$

- That's a pretty weak bound. What does that do to our analysis?

## A Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1)} \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot \mathbf{E}[B_s]]$$

# A Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot Pr[B_s \ge 2 \cdot E[B_s]]}$$

- Assuming 2-independent hashing, this is

$$O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \ge 2 \cdot E[B_k]]$$

# A Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 2-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_k]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \frac{1}{2} \end{aligned}$$

# A Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 2-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_k]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \frac{1}{2} \\ = \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \end{aligned}$$

# A Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot Pr[\boldsymbol{B_s} \geq 2 \cdot E[\boldsymbol{B_s}]]}$$

- Assuming 2-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_k]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \frac{1}{2} \\ = \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \\ = \quad & \mathbf{O(\boldsymbol{n})} \end{aligned}$$

# A Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 2-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_k]] \\ \leq\ & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \frac{1}{2} \\ =\ & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \\ =\ & \mathbf{O(n)} \end{aligned}$$

- This bound is not at all useful. We're going to need to do better than this!

# Concentration Inequalities

- This analysis used Markov's inequality without any additional knowledge about $B_s$.
- $B_s$ is the number of elements that hash into the block of size $2^s$ near $h(x_a)$. What does that tell us?
- Let $X_{is}$ be an indicator variable that's 1 if $x_i$ hashes into the region of size $2^s$ centered on $h(x_a)$ and 0 otherwise. Then we can write

$$B_s = \sum_{i=1}^{n} X_{is}.$$

- Notice that

$$\mathbf{E}[B_s] = \mathbf{E}[\sum_{i=1}^{n} X_{is}] = \sum_{i=1}^{n} E[X_{is}].$$

# Chernoff Bounds

- Last time, we saw the ***Chernoff bound***, which says that if $X \sim \text{Binom}(n, p)$ and $p < 1/2$, then

$$\Pr[X > n/2] < e^{\frac{-n(1/2-p)^2}{2p}}$$

- We just saw that our variable $B_s$ is the sum of a number of Bernoulli variables $X_{is}$, so it seems like we might be able to apply Chernoff bounds here.
- ***Problem:*** These $X_{is}$ variables are ***not*** independent of one another!
    - We know $h$ is $k$-independent and we know what $h(x_a)$ is.
    - So any other group of $k$-1 hashes are independent, but not all $n$ of them.
- Therefore, $B_s$ is not binomially distributed, so we can't use a Chernoff bound.

# Chebyshev's Inequality

- The last remaining bound that we used last time was ***Chebyshev's inequality***, which states that

$$\Pr[\ |X - E[X]| \geq c\ ] \leq \mathrm{Var}[X] / c^2.$$

- If we can determine Var[$B_s$], then we can try using Chebyshev's inequality to bound the probability that $B_s$ is too large.

# The Variance

$$\text{Var}[B_s] \quad = \quad \text{Var}[\sum_{i=1}^{n} X_{is}]$$

Assume, going forward, that the $X_{is}$'s are pairwise independent.

We're already conditioning on knowing $h(x_a)$.

This means that we need our hash function to be at least ***3-independent*** from this point onward.

## The Variance

$$
\begin{aligned}
\mathrm{Var}[B_s] &= \mathrm{Var}[\sum_{i=1}^{n} X_{is}] \\
&= \sum_{i=1}^{n} \mathrm{Var}[X_{is}]
\end{aligned}
$$

## The Variance

$$\begin{aligned}\operatorname{Var}[B_s] &= \operatorname{Var}[\sum_{i=1}^{n} X_{is}] \\ &= \sum_{i=1}^{n} \operatorname{Var}[X_{is}] \\ &\leq \sum_{i=1}^{n} \operatorname{E}[X_{is}^2]\end{aligned}$$

Standard technique we saw last time: use the fact that Var[$Z$] ≤ E[$Z^2$].

## The Variance

$$\begin{aligned} \operatorname{Var}[B_s] &= \operatorname{Var}[\sum_{i=1}^{n} X_{is}] \\ &= \sum_{i=1}^{n} \operatorname{Var}[X_{is}] \\ &\leq \sum_{i=1}^{n} \operatorname{E}[X_{is}^2] \\ &= \sum_{i=1}^{n} \operatorname{E}[X_{is}] \end{aligned}$$

Standard technique we saw last time: if $Z$ is an indicator variable, then $Z^2 = Z$.

## The Variance

$$\begin{aligned}
\mathrm{Var}[B_s] &= \mathrm{Var}[\sum_{i=1}^{n} X_{is}] \\
&= \sum_{i=1}^{n} \mathrm{Var}[X_{is}] \\
&\leq \sum_{i=1}^{n} \mathrm{E}[X_{is}^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_{is}] \\
&= \mathrm{E}[\sum_{i=1}^{n} X_{is}]
\end{aligned}$$

## The Variance

$$
\begin{aligned}
\mathrm{Var}[B_s] &= \mathrm{Var}[\sum_{i=1}^{n} X_{is}] \\
&= \sum_{i=1}^{n} \mathrm{Var}[X_{is}] \\
&\leq \sum_{i=1}^{n} \mathrm{E}[X_{is}^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_{is}] \\
&= \mathrm{E}[\sum_{i=1}^{n} X_{is}] \\
&= \mathrm{E}[B_s]
\end{aligned}
$$

More generally: if $X$ is a sum of pairwise independent indicator variables, then $\mathrm{Var}[X] \leq \mathrm{E}[X]$.

# Using Chebyshev

- We want to know

$$\Pr[\ B_s \geq 2 \cdot \mathrm{E}[B_s]\ ] \quad = \quad \Pr[\ B_s - \mathrm{E}[B_s] \geq \mathrm{E}[B_s]\ ]$$

- Using Chebyshev's inequality:

$$\begin{aligned} \Pr[\ B_s - \mathrm{E}[B_s] \geq \mathrm{E}[B_s]\ ] \quad &\leq \quad \Pr[\ |B_s - \mathrm{E}[B_s]| \geq \mathrm{E}[B_s]\ ] \\ &\leq \quad \mathrm{Var}[B_s] \ / \ \mathrm{E}[B_s]^2 \\ &\leq \quad \mathrm{E}[B_s] \ / \ \mathrm{E}[B_s]^2 \\ &= \quad 1 \ / \ \mathrm{E}[B_s] \\ &= \quad \mathbf{3 \cdot 2^{-s}}. \end{aligned}$$

# A Better Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]$$

# A Better Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1)} \cdot \sum_{s=0}^{\lceil \log n \rceil} \mathbf{2^s \cdot Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 3-independent hashing, this is

$$O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]$$

## A Better Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 3-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot 3 \cdot 2^{-s} \end{aligned}$$

# A Better Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 3-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot 3 \cdot 2^{-s} \\ = \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 3 \end{aligned}$$

# A Better Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 3-independent hashing, this is

$$\begin{aligned}
& O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]] \\
\leq\ & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot 3 \cdot 2^{-s} \\
=\ & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 3 \\
=\ & \mathbf{O(\log n)}
\end{aligned}$$

# A Better Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \ge 2 \cdot E[B_s]]}$$

- Assuming 3-independent hashing, this is

$$\begin{aligned}
& O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \ge 2 \cdot E[B_s]] \\
\le \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot 3 \cdot 2^{-s} \\
= \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 3 \\
= \quad & \mathbf{O(\log n)}
\end{aligned}$$

- ***Theorem:*** This runtime bound is tight (there's an adversarial choice of a 3-independent hash function that degrades the runtime to this level.)

## Why This Works

- ***Key idea:*** Increasing the degree of independence lets us control the variance of the distribution.
- With 2-independent hashing, we use one degree of independence to condition on knowing where some specific key lands. At that point, we only have one more degree of independence – not enough to control the variance!
- With 3-independent hashing, we use one degree of independence to condition on knowing where the key lands. We can then use the two remaining degrees of independence to control the variance and use Chebyshev's inequality.
- ***Small increases to the independence of a hash function can dramatically tighten concentration bounds.***

***Question:*** If we increase the degree of independence further, can we constrain the spread of the elements in a way that improves our runtime?

(This is the theory version of “can we do better?”)

# Generalizing Variance

- The variance of a random variable $X$ is defined as

$$\text{Var}[X] = \text{E}[(X - \text{E}[X])^2].$$

- We can generalize this to higher exponents.
- The ***fourth central moment*** of $X$, denoted **4th[$X$]**, is defined as

$$\mathbf{4th}[X] = \mathbf{E}[(X - \mathbf{E}[X])^4].$$

- Like the variance, 4th[$X$] measures how likely we are to get far away from E[$X$].
- Because of the fourth-power term, 4th[$X$] is much more sensitive to outliers.

# Generalizing Chebyshev

- The ***fourth moment inequality*** states that

$$\Pr[\ |X - E[X]| \geq c\ ] \ \leq\ 4\text{th}[X] / c^4.$$

- ***Proof:*** Let $X$ be a random variable. Then

$$\Pr[\ |X - E[X]| \geq c\ ] \ =\ \Pr[\ (X - E[X])^4 \geq c^4\ ].$$

Let $Y = (X - E[X])^4$. Notice that

$$E[Y] = E[(X - E[X])^4] = 4\text{th}[X],$$

so via Markov's inequality, we have

$$\begin{aligned} \Pr[\ |X - E[X]| \geq c\ ] \ &=\ \Pr[\ Y \geq c^4\ ] \\ &\leq\ E[Y] / c^4 \\ &=\ \mathbf{4th}[X] / c^4.\ \blacksquare \end{aligned}$$

***Good question to ponder:***
why doesn't this work for the third central moment, where $3\text{rd}[X] = (X - E[X])^3$?

# Generalizing Indicator Variance

- ***Theorem:*** If $X$ is an indicator variable for the event $\mathcal{E}$, then $4\text{th}[X] \le \text{E}[X]$.
- ***Proof:*** $X$ takes on value 1 with probability $\Pr[\mathcal{E}]$ and 0 with probability $1 - \Pr[\mathcal{E}]$. Therefore, we have

$$\begin{aligned}
4\text{th}[X] &= \text{E}[(X - \text{E}[X])^4] \\
&= (1 - \Pr[\mathcal{E}])^4 \cdot \Pr[\mathcal{E}] + \Pr[\mathcal{E}]^4(1 - \Pr[\mathcal{E}]) \\
&\le (1 - \Pr[\mathcal{E}])^3 \cdot \Pr[\mathcal{E}] + \Pr[\mathcal{E}]^4 \\
&= \Pr[\mathcal{E}] - \Pr[\mathcal{E}]^4 + \Pr[\mathcal{E}]^4 \\
&= \Pr[\mathcal{E}] \\
&= \text{E}[X]. \blacksquare
\end{aligned}$$

Read this on your own time – it's cute!

## Updating our Analysis

- For linear probing, we're ultimately interested in bounding

$$\Pr[\ B_s \geq 2 \cdot \mathrm{E}[B_s]\ ]$$

  in the case where $B_s$ represents the number of elements hitting a particular block.
- Using 2-independent hashing, the best bound we could use was Markov's inequality, which gave an extremely weak bound.
- Using 3-independent hashing, we could use Chebyshev's inequality, which gave an inverse exponential bound.
- ***Question:*** If we use stronger hash functions, can we tighten this bound using the fourth moment inequality?

What is 4th[$B_s$]?

# The Limits of Our Generalization

- There's a lovely little expression for Var[$X$]:

$$\mathbf{Var}[\boldsymbol{X}] = \mathbf{E}[\boldsymbol{X}^2] - \mathbf{E}[\boldsymbol{X}]^2.$$

- That's because

$$\begin{aligned}
\mathrm{Var}[X] &= \mathrm{E}[(X - \mathrm{E}[X])^2] \\
&= \mathrm{E}[X^2 - 2X \cdot \mathrm{E}[X] + \mathrm{E}[X]^2] \\
&= \mathrm{E}[X^2] - 2\mathrm{E}[X] \cdot \mathrm{E}[X] + \mathrm{E}[X]^2 \\
&= \mathrm{E}[X^2] - 2\mathrm{E}[X]^2 + \mathrm{E}[X]^2 \\
&= \mathbf{E}[\boldsymbol{X}^2] - \mathbf{E}[\boldsymbol{X}]^2.
\end{aligned}$$

- We can try this for fourth moments, but, well, um…

$$\begin{aligned}
\mathrm{4th}[X] &= \mathrm{E}[(X - \mathrm{E}[X])^4] \\
&= \mathrm{E}[X^4 - 4X^3 \cdot \mathrm{E}[X] + 6X^2 \cdot \mathrm{E}[X]^2 - 4X \cdot \mathrm{E}[X]^3 + \mathrm{E}[X]^4] \\
&= \mathrm{E}[X^4] - 4\mathrm{E}[X] \cdot \mathrm{E}[X^3] + 6\mathrm{E}[X]^2\mathrm{E}[X^2] - 4\mathrm{E}[X] \cdot \mathrm{E}[X]^3 + \mathrm{E}[X^4] \\
&= \mathrm{E}[X^4] - 4\mathrm{E}[X] \cdot \mathrm{E}[X^3] + 6\mathrm{E}[X]^2\mathrm{E}[X^2] - 3\mathrm{E}[X]^4
\end{aligned}$$

= **¯_(ツ)_/¯**

## The Fourth Moment

- Let’s see if we can bound 4th[$B_s$].

## The Fourth Moment

- Let's see if we can bound 4th$[B_s]$.

$$\text{4th}[B_s] \quad = \quad \mathrm{E}[(B_s - \mathrm{E}[B_s])^4]$$

## The Fourth Moment

- Let’s see if we can bound 4th[$B_s$].

$$\begin{aligned}
\text{4th}[B_s] &= \mathrm{E}[(B_s - \mathrm{E}[B_s])^4] \\
&= \mathrm{E}[\left(\sum_{i=1}^{n} X_{is} - \sum_{i=1}^{n} \mathrm{E}[X_{is}]\right)^4]
\end{aligned}$$

## The Fourth Moment

- Let’s see if we can bound 4th[$B_s$].

$$\begin{aligned}
4\text{th}[B_s] &= \mathrm{E}[(B_s - \mathrm{E}[B_s])^4] \\
&= \mathrm{E}[\left(\sum_{i=1}^{n} X_{is} - \sum_{i=1}^{n} \mathrm{E}[X_{is}]\right)^4] \\
&= \mathrm{E}[\left(\sum_{i=1}^{n} (X_{is} - \mathrm{E}[X_{is}])\right)^4]
\end{aligned}$$

# The Fourth Moment

- Let’s see if we can bound 4th[$B_s$].

$$\begin{aligned}
4\text{th}[B_s] &= \mathrm{E}[(B_s - \mathrm{E}[B_s])^4] \\
&= \mathrm{E}[\left(\sum_{i=1}^{n} X_{is} - \sum_{i=1}^{n} \mathrm{E}[X_{is}]\right)^4] \\
&= \mathrm{E}[\left(\sum_{i=1}^{n} (X_{is} - \mathrm{E}[X_{is}])\right)^4] \\
&= \mathrm{E}[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} (X_{is} - \mathrm{E}[X_{is}])(X_{js} - \mathrm{E}[X_{js}])(X_{ks} - \mathrm{E}[X_{ks}])(X_{ls} - \mathrm{E}[X_{ls}])]
\end{aligned}$$

# The Fourth Moment

- Let's see if we can bound 4th[$B_s$].

$$
\begin{aligned}
4\text{th}[B_s] &= \mathrm{E}[(B_s - \mathrm{E}[B_s])^4] \\
&= \mathrm{E}[\left(\sum_{i=1}^{n} X_{is} - \sum_{i=1}^{n} \mathrm{E}[X_{is}]\right)^4] \\
&= \mathrm{E}[\left(\sum_{i=1}^{n} (X_{is} - \mathrm{E}[X_{is}])\right)^4] \\
&= \mathrm{E}[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} (X_{is} - \mathrm{E}[X_{is}])(X_{js} - \mathrm{E}[X_{js}])(X_{ks} - \mathrm{E}[X_{ks}])(X_{ls} - \mathrm{E}[X_{ls}])] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is} - \mathrm{E}[X_{is}])(X_{js} - \mathrm{E}[X_{js}])(X_{ks} - \mathrm{E}[X_{ks}])(X_{ls} - \mathrm{E}[X_{ls}])]
\end{aligned}
$$

## The Fourth Moment

- Let’s see if we can bound 4th[$B_s$].

$$\begin{aligned}
\text{4th}[B_s] &= \mathrm{E}[(B_s - \mathrm{E}[B_s])^4] \\
&= \mathrm{E}\left[\left(\sum_{i=1}^{n} X_{is} - \sum_{i=1}^{n} \mathrm{E}[X_{is}]\right)^4\right] \\
&= \mathrm{E}\left[\left(\sum_{i=1}^{n} (X_{is} - \mathrm{E}[X_{is}])\right)^4\right] \\
&= \mathrm{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} (X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])\right] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])]
\end{aligned}$$

- So now we “just” need to simplify this expression. 😃

## Increasing our Independence

- We now have this lovely expression:

$$\text{4th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is} - \mathrm{E}[X_{is}])(X_{js} - \mathrm{E}[X_{js}])(X_{ks} - \mathrm{E}[X_{ks}])(X_{ls} - \mathrm{E}[X_{ls}])]$$

- ***Recall:*** If our hash function is $k$-independent, then we've already used one degree of independence conditioning on knowing where $h(x_a)$ is. That leaves us with $k$-1 degrees of independence.
- Let's suppose we're using a **5-independent** hash function, meaning that any four hash values are independent of one another.
- This allows us to dramatically simplify this expression.

# Exploring this Summation

- The terms of this summation might sometimes range over the same variables at the same time:

$$\text{4th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])]$$

- ***Claim:*** Any term in the above summation where $X_{is}$ is a different random variable than $X_{js}$, $X_{ks}$, and $X_{ls}$ is zero.

- ***Proof:*** Suppose that $X_{is}$ is a different random variable from the others. Then since $X_{is}$, $X_{js}$, $X_{ks}$, and $X_{ls}$ are independent, we have

$$\begin{aligned}
&\mathrm{E}[\ (X_{is} - \mathrm{E}[X_{is}])(X_{js} - \mathrm{E}[X_{js}])(X_{ks} - \mathrm{E}[X_{ks}])(X_{ls} - \mathrm{E}[X_{ls}])\ ] \\
= &\mathrm{E}[X_{is} - \mathrm{E}[X_{is}]] \cdot \mathrm{E}[(X_{js} - \mathrm{E}[X_{js}])(X_{ks} - \mathrm{E}[X_{ks}])(X_{ls} - \mathrm{E}[X_{ls}])\ ] \\
= &0 \cdot \mathrm{E}[(X_{js} - \mathrm{E}[X_{js}])(X_{ks} - \mathrm{E}[X_{ks}])(X_{ls} - \mathrm{E}[X_{ls}])\ ] \\
= &\mathbf{0}
\end{aligned}$$

## Exploring this Summation

- The terms of this summation might sometimes range over the same variables at the same time:

$$\text{4th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])]$$

- ***Claim:*** Every term in this sum is zero *except* for the following:
  - Terms where $i = j = k = l$.
  - Terms where two of $i$, $j$, $k$, and $l$ refer to one value and the other two of $i$, $j$, $k$, and $l$ refer to another.
- ***Proof:*** If a variable appears exactly one time, then by our previous logic the term evaluates to zero. If a variable appears exactly three times, then the other variable appears exactly once and the term evaluates to zero. That leaves behind the two remaining cases here.

## Exploring this Summation

- The terms of this summation might sometimes range over the same variables at the same time:

$$4\text{th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is} - \mathrm{E}[X_{is}])(X_{js} - \mathrm{E}[X_{js}])(X_{ks} - \mathrm{E}[X_{ks}])(X_{ls} - \mathrm{E}[X_{ls}])]$$

- ***Claim:*** Every term in this sum is zero *except* for the following:
  - Terms where $i = j = k = l$.
  - Terms where two of *i*, *j*, *k*, and *l* refer to one value and the other two of *i*, *j*, *k*, and *l* refer to another.

# Exploring this Summation

- The terms of this summation might sometimes range over the same variables at the same time:

$$\text{4th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])]$$

- ***Claim:*** Every term in this sum is zero *except* for the following:
  - Terms where $i = j = k = l$.
  - Terms where two of *i*, *j*, *k*, and *l* refer to one value and the other two of *i*, *j*, *k*, and *l* refer to another.

$$\sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4]$$

# Exploring this Summation

- The terms of this summation might sometimes range over the same variables at the same time:

$$4\text{th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])]$$

- ***Claim:*** Every term in this sum is zero *except* for the following:
  - Terms where $i = j = k = l$.
  - Terms where two of *i, j, k,* and *l* refer to one value and the other two of *i, j, k,* and *l* refer to another.

$$\sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4]$$

# Exploring this Summation

- The terms of this summation might sometimes range over the same variables at the same time:

$$4\text{th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])]$$

- ***Claim:*** Every term in this sum is zero *except* for the following:
  - Terms where $i = j = k = l$.
  - Terms where two of *i, j, k,* and *l* refer to one value and the other two of *i, j, k,* and *l* refer to another.

$$\sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{p=1}^{n}\sum_{q=p+1}^{n} \mathrm{E}[(X_{ps}-\mathrm{E}[X_{ps}])^2(X_{qs}-\mathrm{E}[X_{qs}])^2]$$

Which of *i, j, k,* and *l* refer to the first value?

What's the first value?

What's the second? (It must be different than the first!)

# Exploring this Summation

- The terms of this summation might sometimes range over the same variables at the same time:

$$4\text{th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])]$$

- ***Claim:*** Every term in this sum is zero *except* for the following:
  - Terms where $i = j = k = l$.
  - Terms where two of $i, j, k,$ and $l$ refer to one value and the other two of $i, j, k,$ and $l$ refer to another.

$$\sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2} \sum_{p=1}^{n} \sum_{q=p+1}^{n} \mathrm{E}[(X_{ps}-\mathrm{E}[X_{ps}])^2(X_{qs}-\mathrm{E}[X_{qs}])^2]$$

# Exploring this Summation

- The terms of this summation might sometimes range over the same variables at the same time:

$$\text{4th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-\mathrm{E}[X_{ks}])(X_{ls}-\mathrm{E}[X_{ls}])]$$

- ***Claim:*** Every term in this sum is zero *except* for the following:
  - Terms where $i = j = k = l$.
  - Terms where two of *i*, *j*, *k*, and *l* refer to one value and the other two of *i*, *j*, *k*, and *l* refer to another.

$$\sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2]$$

We'll use *i* and *j* as our summation variables, since that's easier to read.

$$\mathrm{4th}[B_s] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])]$$

$$\begin{aligned}
4\text{th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2]
\end{aligned}$$

Since *h* is 5-independent and we're conditioning on just knowing one hash location ($h(x_a)$), these are independent random variables.

$$\begin{aligned}
\text{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2 (X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2]
\end{aligned}$$

Since $h$ is 5-independent and we're conditioning on just knowing one hash location ($h(x_a)$), these are independent random variables.

$$
\begin{aligned}
\text{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])) \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2} \sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6 \sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2]
\end{aligned}
$$

This is the definition of the fourth central moment.

This is the definition of variance.

So is this.

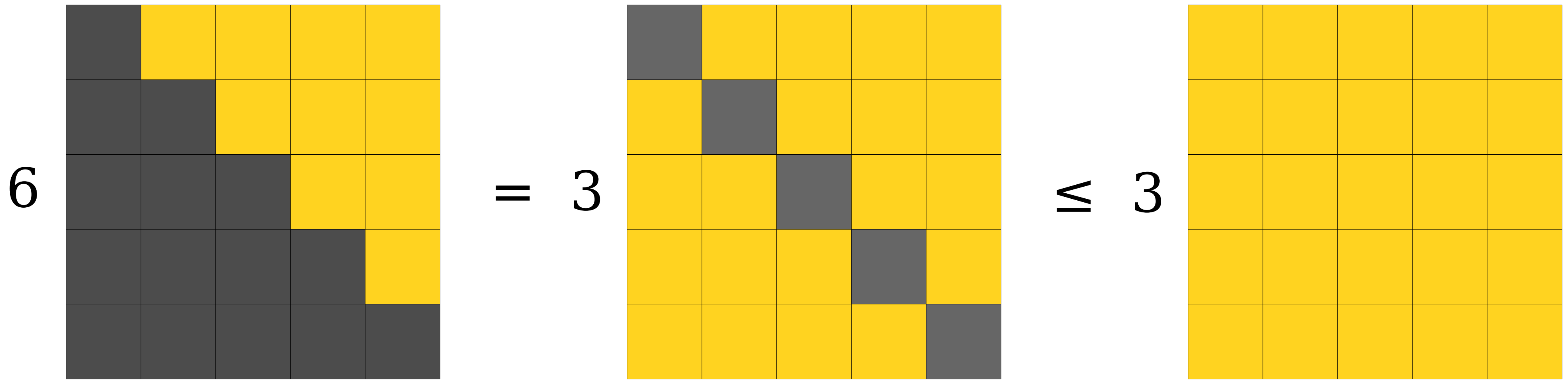
$$
\begin{aligned}
\mathrm{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 6\sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]
\end{aligned}
$$

This is the definition of the fourth central moment.

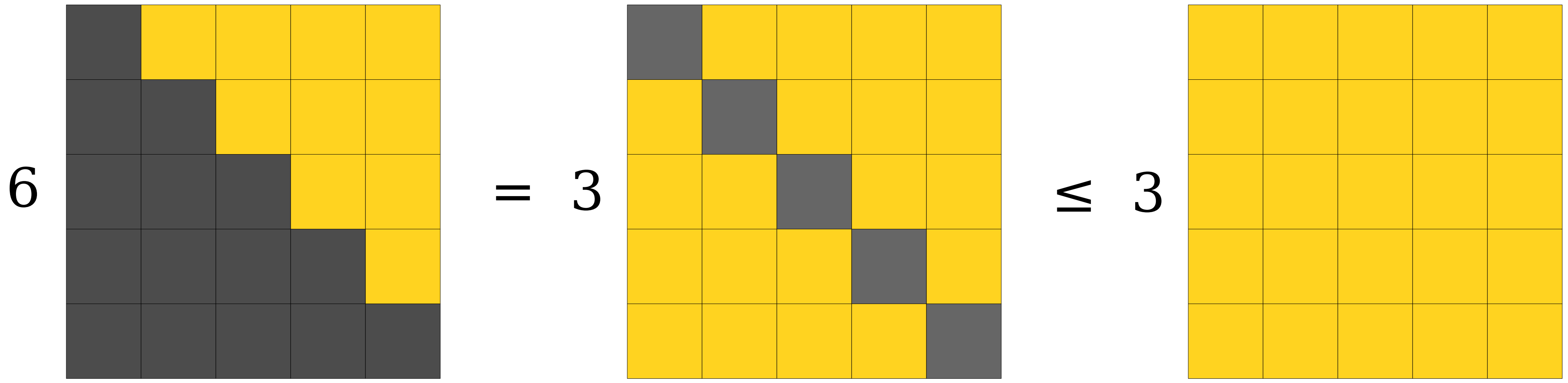
This is the definition of variance.

So is this.

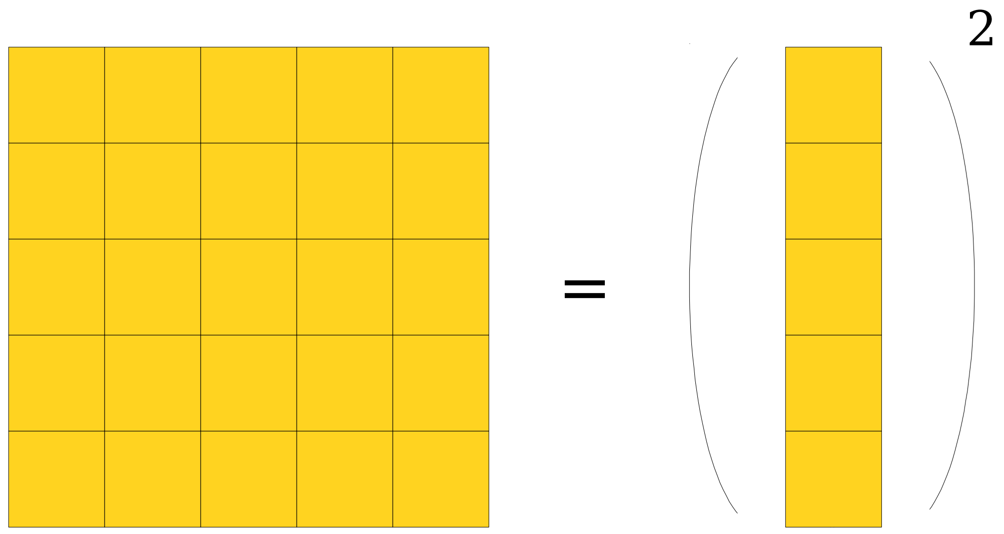
$$\begin{aligned}
\mathrm{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 6\sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]
\end{aligned}$$



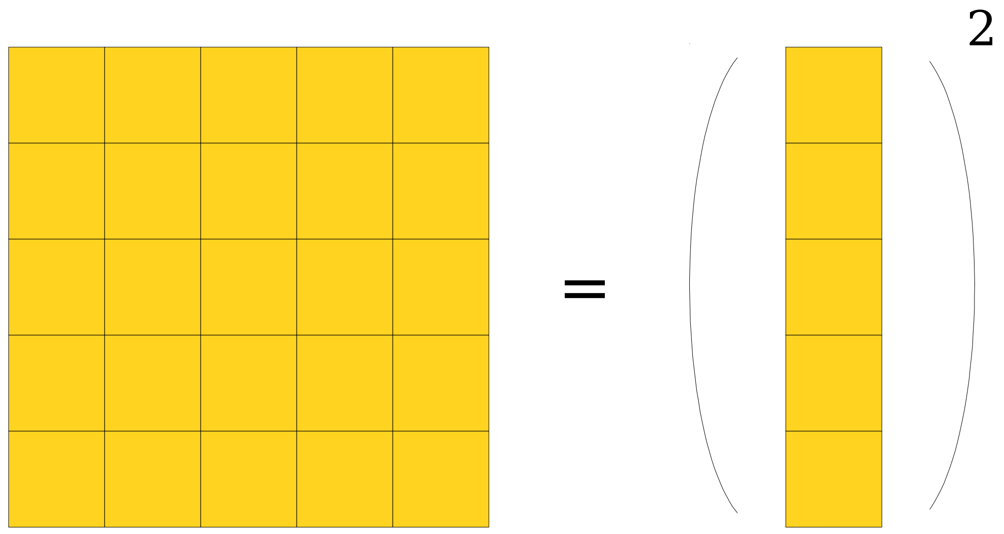
$$
\begin{aligned}
4\text{th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} 4\text{th}[X_{is}] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]\\
&\le \sum_{i=1}^{n} 4\text{th}[X_{is}] + 3\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]
\end{aligned}
$$



$$\begin{aligned}
\text{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \text{4th}[X_{is}] + 6\sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&\leq \sum_{i=1}^{n} \text{4th}[X_{is}] + 3\sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]
\end{aligned}$$



$$
\begin{aligned}
4\text{th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} 4\text{th}[X_{is}] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&\le \sum_{i=1}^{n} 4\text{th}[X_{is}] + 3\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&= \sum_{i=1}^{n} 4\text{th}[X_{is}] + 3\left(\sum_{i=1}^{n} \mathrm{Var}[X_{is}]\right)^2
\end{aligned}
$$



$$
\begin{aligned}
\mathrm{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2 (X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 6 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&\leq \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3 \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3 \left( \sum_{i=1}^{n} \mathrm{Var}[X_{is}] \right)^2
\end{aligned}
$$

$$
\boxed{\sum_{i=1}^{n} \mathrm{Var}[X_{is}] = \mathrm{Var}[\sum_{i=1}^{n} X_{is}] = \mathrm{Var}[B_s]}
$$

$$
\begin{aligned}
\mathrm{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&\le \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\left(\sum_{i=1}^{n} \mathrm{Var}[X_{is}]\right)^2 \\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\mathrm{Var}[B_s]^2
\end{aligned}
$$

$$\boxed{\sum_{i=1}^{n} \mathrm{Var}[X_{is}] = \mathrm{Var}[\sum_{i=1}^{n} X_{is}] = \mathrm{Var}[B_s]}$$

$$
\begin{aligned}
\mathrm{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]\\
&\le \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]\\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\left(\sum_{i=1}^{n} \mathrm{Var}[X_{is}]\right)^2\\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\,\mathrm{Var}[B_s]^2
\end{aligned}
$$

If $X$ is an indicator, then $\mathrm{4th}[X] \le \mathrm{E}[X]$.

We know from our 3-independence analysis that $\mathrm{Var}[B_s] \le \mathrm{E}[B_s]$

$$
\begin{aligned}
4\mathrm{th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} 4\mathrm{th}[X_{is}] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&\le \sum_{i=1}^{n} 4\mathrm{th}[X_{is}] + 3\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&= \sum_{i=1}^{n} 4\mathrm{th}[X_{is}] + 3\left(\sum_{i=1}^{n} \mathrm{Var}[X_{is}]\right)^2 \\
&= \sum_{i=1}^{n} 4\mathrm{th}[X_{is}] + 3\mathrm{Var}[B_s]^2 \\
&\le \sum_{i=1}^{n} \mathrm{E}[X_{is}] + 3\mathrm{E}[B_s]^2
\end{aligned}
$$

If $X$ is an indicator, then $4\mathrm{th}[X] \le \mathrm{E}[X]$.

We know from our 3-independence analysis that $\mathrm{Var}[B_s] \le \mathrm{E}[B_s]$

$$
\begin{aligned}
\mathrm{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]\\
&\leq \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]\\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\left(\sum_{i=1}^{n} \mathrm{Var}[X_{is}]\right)^2\\
&= \sum_{i=1}^{n} \mathrm{4th}[X_{is}] + 3\mathrm{Var}[B_s]^2\\
&\leq \sum_{i=1}^{n} \mathrm{E}[X_{is}] + 3\mathrm{E}[B_s]^2
\end{aligned}
$$

$$\begin{aligned}
\text{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2] \\
&= \sum_{i=1}^{n} \text{4th}[X_{is}] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&\leq \sum_{i=1}^{n} \text{4th}[X_{is}] + 3\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}] \\
&= \sum_{i=1}^{n} \text{4th}[X_{is}] + 3\left(\sum_{i=1}^{n} \mathrm{Var}[X_{is}]\right)^2 \\
&= \sum_{i=1}^{n} \text{4th}[X_{is}] + 3\mathrm{Var}[B_s]^2 \\
&\leq \sum_{i=1}^{n} \mathrm{E}[X_{is}] + 3\mathrm{E}[B_s]^2 \\
&= \mathrm{E}[B_s] + 3\mathrm{E}[B_s]^2
\end{aligned}$$

$$\begin{aligned}
\text{4th}[B_s] &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])(X_{js}-\mathrm{E}[X_{js}])(X_{ks}-E[X_{ks}])(X_{ls}-E[X_{ls}])]\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + \binom{4}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^4] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}[(X_{is}-\mathrm{E}[X_{is}])^2]\mathrm{E}[(X_{js}-\mathrm{E}[X_{js}])^2]\\
&= \sum_{i=1}^{n} \text{4th}[X_{is}] + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]\\
&\le \sum_{i=1}^{n} \text{4th}[X_{is}] + 3\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Var}[X_{is}]\mathrm{Var}[X_{js}]\\
&= \sum_{i=1}^{n} \text{4th}[X_{is}] + 3\left(\sum_{i=1}^{n} \mathrm{Var}[X_{is}]\right)^2\\
&= \sum_{i=1}^{n} \text{4th}[X_{is}] + 3\mathrm{Var}[B_s]^2\\
&\le \sum_{i=1}^{n} \mathrm{E}[X_{is}] + 3\mathrm{E}[B_s]^2\\
&= \mathrm{E}[B_s] + 3\mathrm{E}[B_s]^2\\
&\le \mathbf{4E}[\boldsymbol{B_s}]^{\mathbf{2}}
\end{aligned}$$

(As long as $\mathrm{E}[B_s] \ge 1$, which we can assume if we're talking about sufficiently large regions.)

## The Net Result

- We've just shown that

$$\text{4th}[B] \leq 4 \cdot \text{E}[B]^2$$

- Phew! That was crazy. But at least we now have a bound on the fourth moment, which lets us use the fourth moment inequality!

# Fourth Moments for Victory

- Using the fourth moment inequality:

$$\begin{aligned}
\Pr[\ B_s \geq 2\mathrm{E}[B_s]\ ] &= \Pr[\ B_s - \mathrm{E}[B_s] \geq \mathrm{E}[B_s]\ ] \\
&\leq 4\mathrm{th}[B_s] / \mathrm{E}[B_s]^4 \\
&\leq 4\cdot\mathrm{E}[B_s]^2 / \mathrm{E}[B_s]^4 \\
&= 4 / \mathrm{E}[B_s]^2 \\
&= 4 / (1/3 \cdot 2^s)^2 \\
&= \mathbf{36 \cdot 2^{-2s}}.
\end{aligned}$$

- Notice that this is exponentially better than our previous bound!

# A Strong Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1)} \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot \mathrm{E}[B_s]]$$

## A Strong Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 5-independent hashing, this is

$$O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]$$

# A Strong Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 5-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot 36 \cdot 2^{-2s} \end{aligned}$$

# A Strong Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 5-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot 36 \cdot 2^{-2s} \\ = \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 36 \cdot 2^{-s} \end{aligned}$$

## A Strong Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot Pr[B_s \geq 2 \cdot E[B_s]]}$$

- Assuming 5-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot 36 \cdot 2^{-2s} \\ = \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 36 \cdot 2^{-s} \\ = \quad & \mathbf{O(1)} \end{aligned}$$

# A Strong Runtime Bound

- The expected cost of looking up $x_a$ in a linear probing table is

$$\mathbf{O(1)} \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]]$$

- Assuming 5-independent hashing, this is

$$\begin{aligned} & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot \Pr[B_s \geq 2 \cdot E[B_s]] \\ \leq \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 2^s \cdot 36 \cdot 2^{-2s} \\ = \quad & O(1) \cdot \sum_{s=0}^{\lceil \log n \rceil} 36 \cdot 2^{-s} \\ = \quad & \mathbf{O(1)} \end{aligned}$$

- We've finally obtained an **O(1)** bound on the cost of operations in a chained hash table – provided that we use 5-independent hashing!

## What Just Happened?

- With one degree of independence, we could obtain the *expected value* and use that to bound the probability with ***Markov's inequality***.
- Using two degrees of independence, we could obtain the *variance* and use that to bound the probability with ***Chebyshev's inequality***.
- Using four degrees of independence, we could obtain the *fourth central moment* and use that to bound the probability with the ***fourth moment bound***.
- Increasing the strength of a hash function allows us to obtain more central moments and, therefore, to tighten our bound more than might initially be suspected.

## More to Explore

- Mitzenmacher and Vadhan’s paper “Why Simple Hash Functions Work” provides a fundamentally different strategy for analyzing linear probing.
- Pătrașcu and Thorup’s paper on the lower bound for 5-independence here gives a glimpse of how you’d argue that these bounds can’t be improved.

## Next Time

- ***Cuckoo Hashing***
  - Hashing with worst-case O(1) lookups!
- ***The Cuckoo Graph***
  - Random graphs for Fun and Profit.