THE UNIVERSITY OF VERMONT
COLLEGE OF ENGINEERING &
MATHEMATICAL SCIENCES

Hash Tables: Rehashing

CS 124 / Department of Computer Science

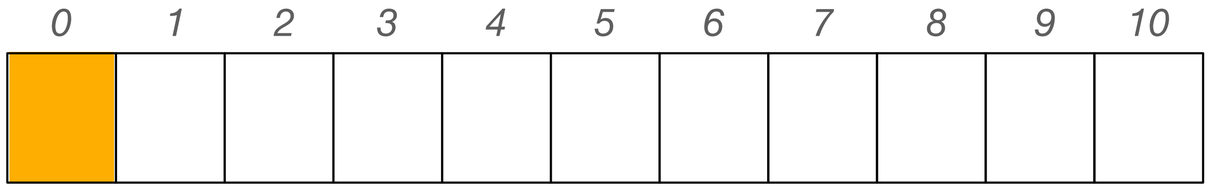
Questions from our previous video

- What can we do when we run out of space in our hash table?
- If we set our stride to some value greater than one, why is it a good idea to have a hash table size that's a prime number?

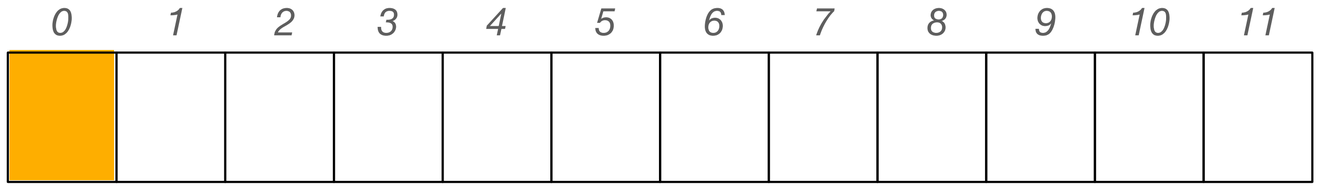
Questions from our previous video

- What can we do when we run out of space in our hash table?
- If we set our stride to some value greater than one, why is it a good idea to have a hash table size that's a prime number?

Why prime numbers are good sizes



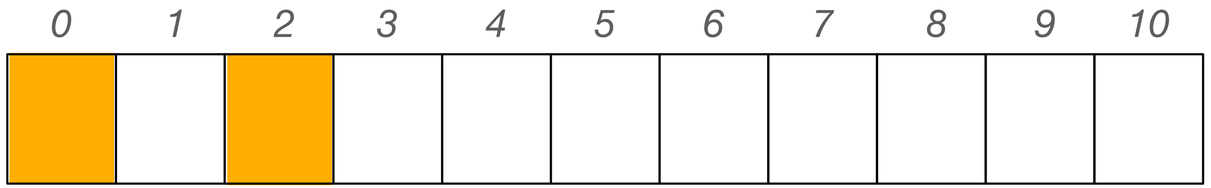
size = 11 (prime)



size = 12 (not prime)

stride = 2

Why prime numbers are good sizes

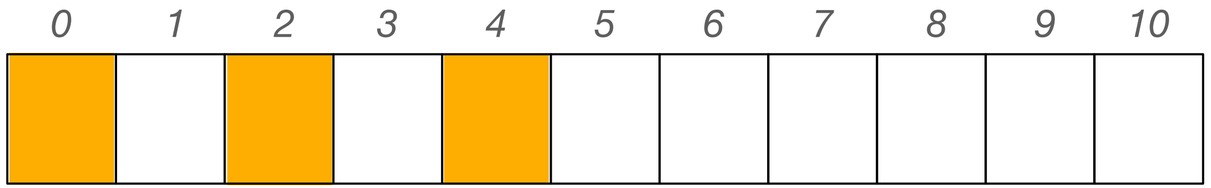


size = 11 (prime)

size = 12 (not prime)

stride = 2

Why prime numbers are good sizes

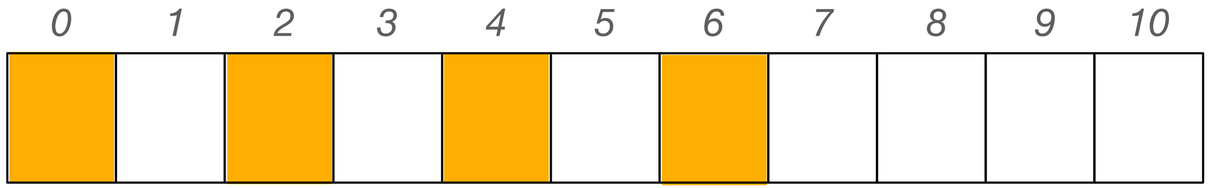


size = 11 (prime)

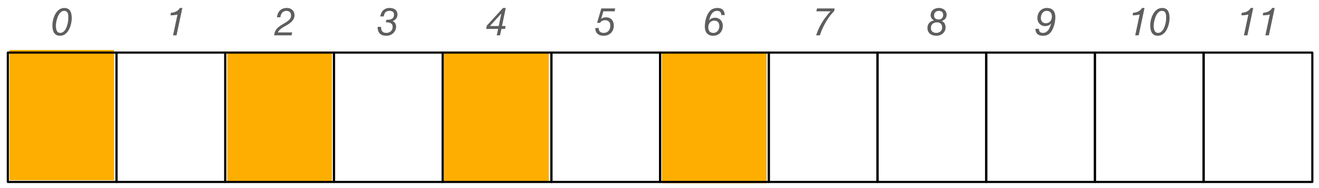
size = 12 (not prime)

stride = 2

Why prime numbers are good sizes



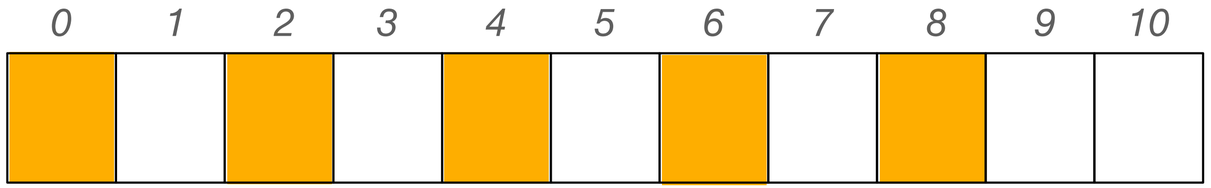
size = 11 (prime)



size = 12 (not prime)

stride = 2

Why prime numbers are good sizes

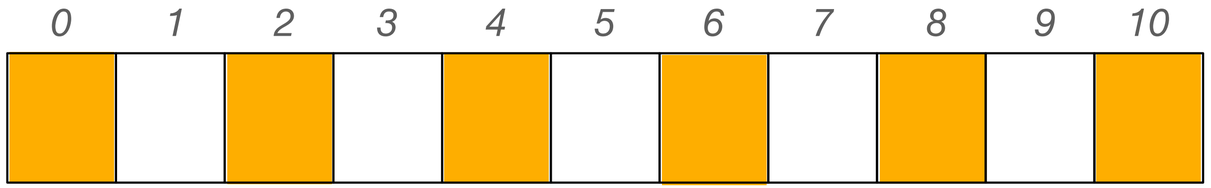


size = 11 (prime)

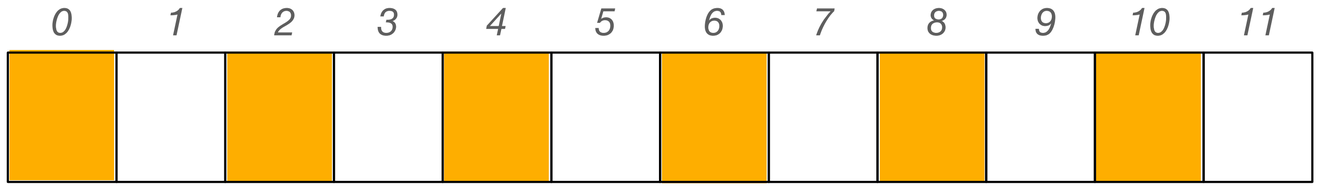
size = 12 (not prime)

stride = 2

Why prime numbers are good sizes



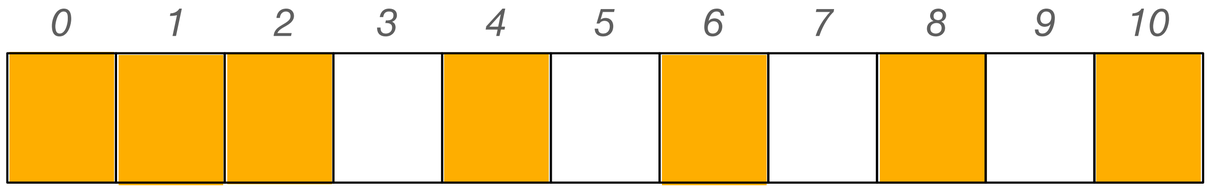
size = 11 (prime)



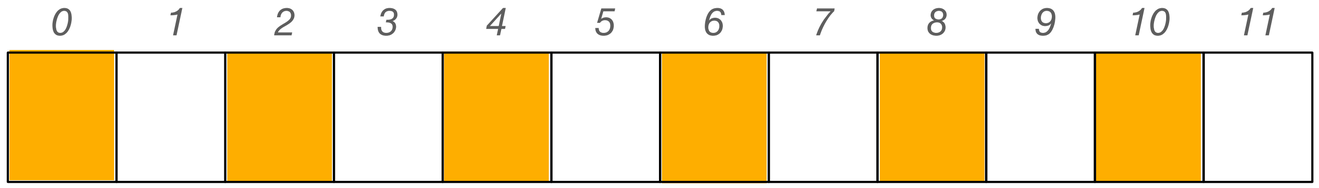
size = 12 (not prime)

stride = 2

Why prime numbers are good sizes



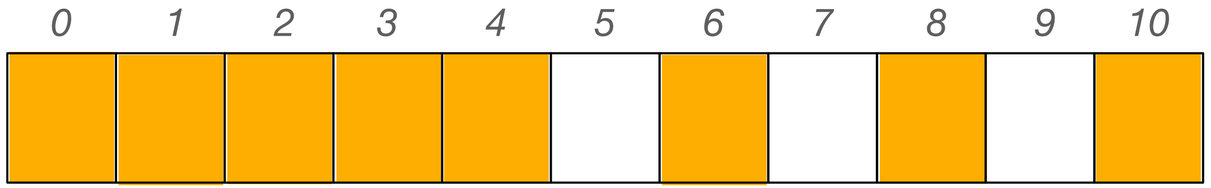
size = 11 (prime)



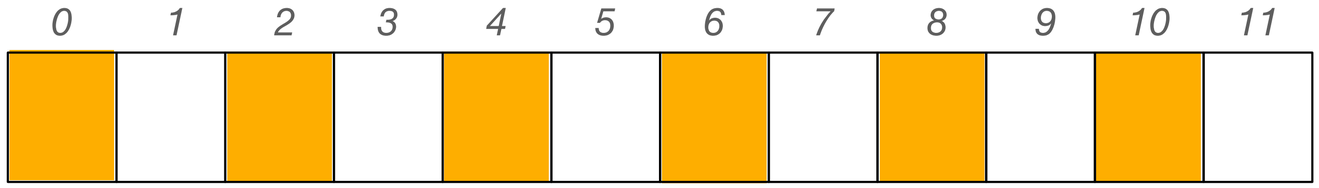
size = 12 (not prime)

stride = 2

Why prime numbers are good sizes



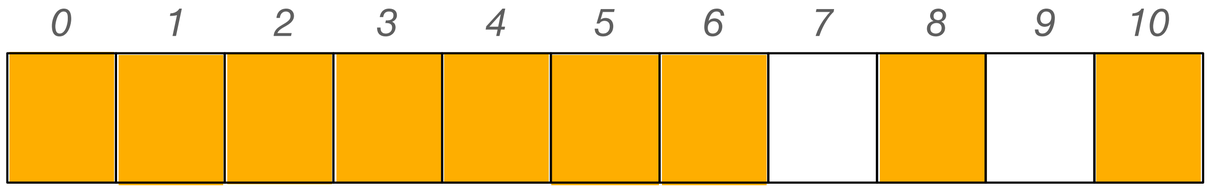
size = 11 (prime)



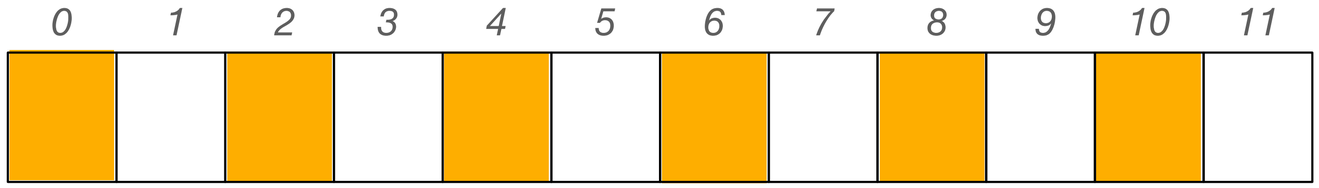
size = 12 (not prime)

stride = 2

Why prime numbers are good sizes



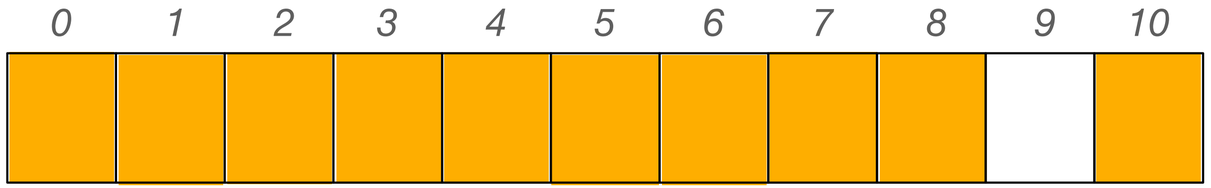
size = 11 (prime)



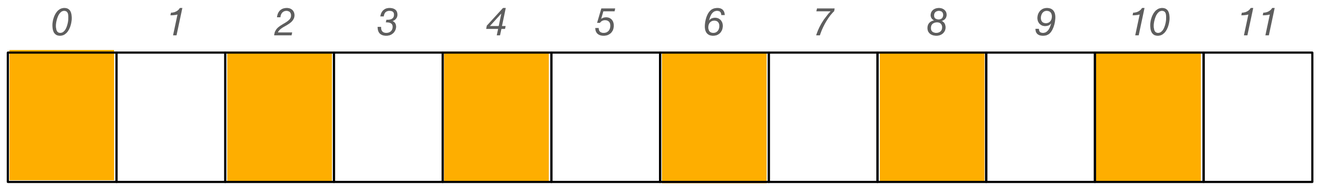
size = 12 (not prime)

stride = 2

Why prime numbers are good sizes



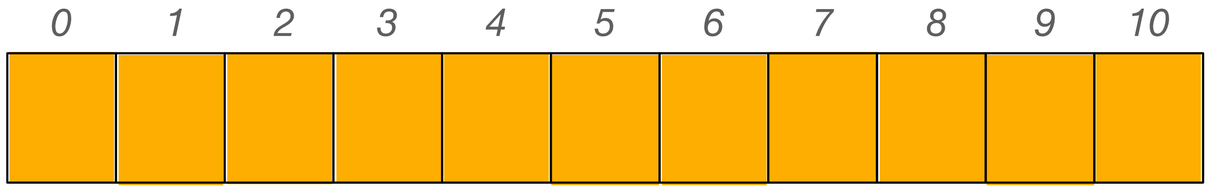
size = 11 (prime)



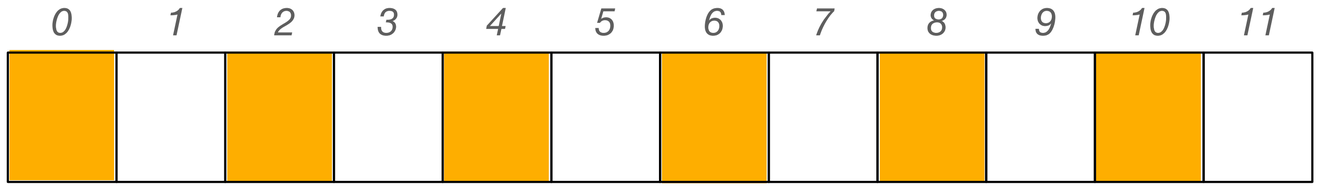
size = 12 (not prime)

stride = 2

Why prime numbers are good sizes



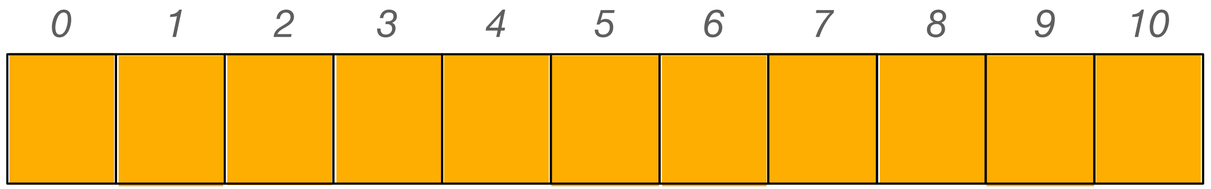
size = 11 (prime)



size = 12 (not prime)

stride = 2

Why prime numbers are good sizes



size = 11 (prime)

size = 12 (not prime)

If the stride and the table size are not coprime, then you can't probe the entire table from any given starting point!

Now on to reashing....

Rehashing

The more entries we have in our hash table, the more likely it is that we have a collision on our next insert.

The more entries we have in our table, the longer the typical probe sequence becomes for inserting, finding, and removing entries.

Rehashing

The more entries we have in our hash table, the more likely it is that we have a collision on our next insert.

The more entries we have in our table, the longer the typical probe sequence becomes for inserting, finding, and removing entries.

Performance degrades!

Rehashing

We set a limit -- a maximum value -- for our load factor (a.k.a. fill percentage)

On inserts, we check our load factor. If it exceeds this limit, we create a new bigger table, and we insert all the objects from the old table into the new table.

Since the table size changes, the index calculated from our hash function will change for each item, hence the term "rehashing." All objects will get a new hash value when inserted into the new table.

Rehashing

We set a limit -- a maximum value -- for our load factor (a.k.a. fill percentage)

On inserts, we check our load factor. If it exceeds this limit, we create a new bigger table, and we insert all the objects from the old table into the new table.

Since the table size changes, the index calculated from our hash function will change for each item, hence the term "rehashing." All objects will get a new hash value when inserted into the new table.

But how big should we make our new table?

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11		

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17		

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17	34	

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17	34	37

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17	34	37
21		

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17	34	37
21	42	

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17	34	37
21	42	43

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17	34	37
21	42	43
103		

Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17	34	37
21	42	43
103	206	

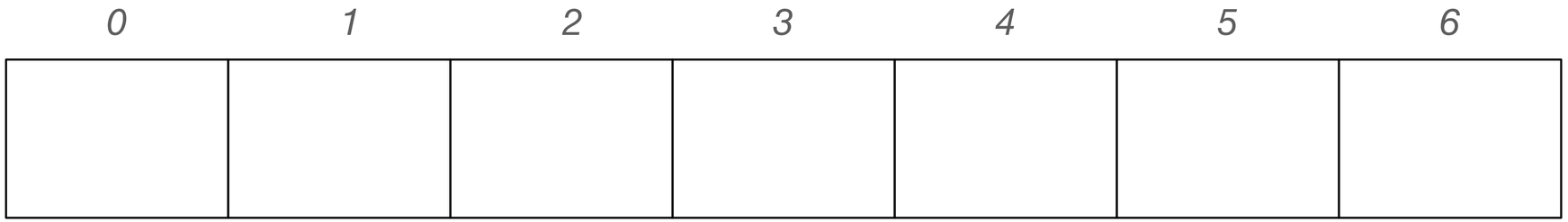
Rehashing

As a rule of thumb, we take the old table size, double it and then find the next prime number and use this as a size for our new table.

Old size	2 x old size	Next prime
11	22	23
17	34	37
21	42	43
103	206	211

Rehashing

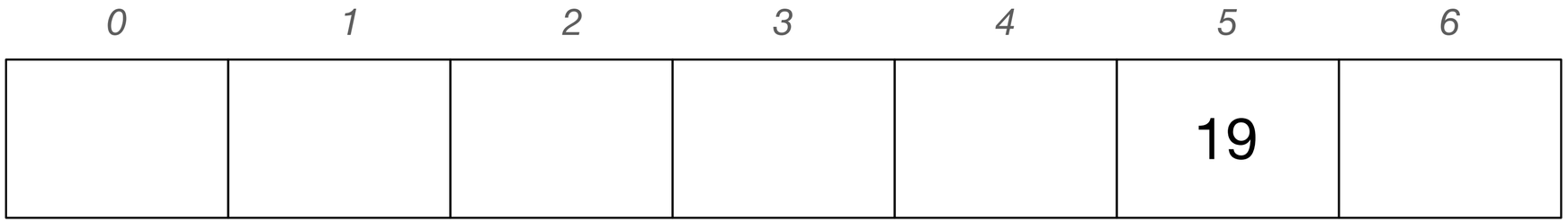
Max. load factor (fill percentage): 50%



Hash function:
$f(x) = x \bmod 7$

Rehashing

Max. load factor (fill percentage): 50%

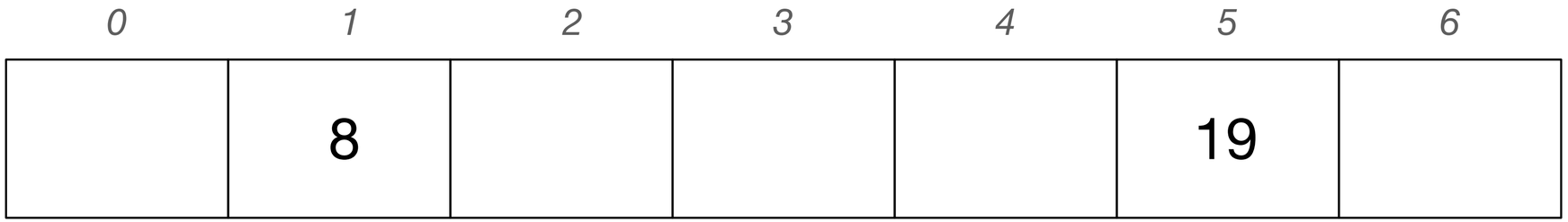


Hash function:
$f(x) = x \bmod 7$

Load: 14.2%

Rehashing

Max. load factor (fill percentage): 50%

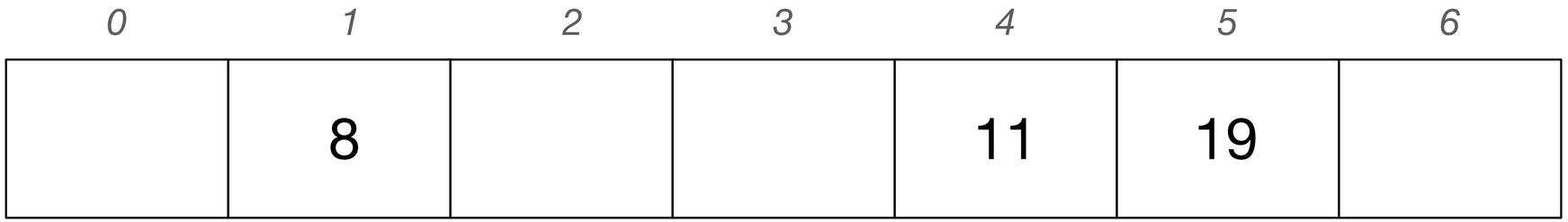


Hash function:
$f(x) = x \bmod 7$

Load: 28.6%

Rehashing

Max. load factor (fill percentage): 50%

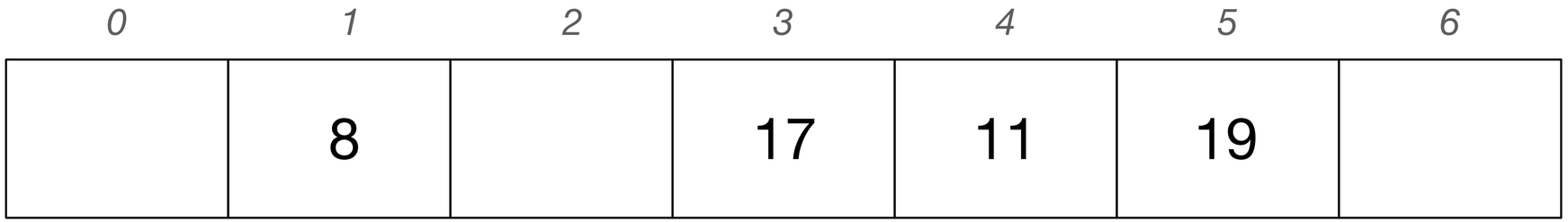


Hash function:
$f(x) = x \bmod 7$

Load: 42.9%

Rehashing

Max. load factor (fill percentage): 50%



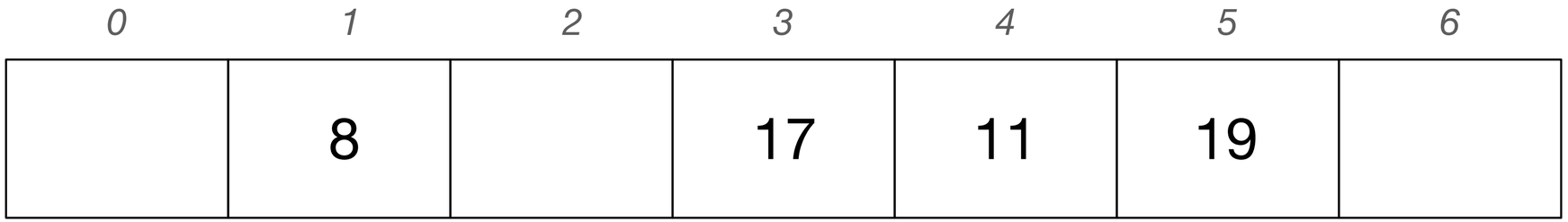
Hash function:
$f(x) = x \bmod 7$

Load: 57.1%

Time to rehash!

Rehashing

Max. load factor (fill percentage): 50%



Hash function:
$f(x) = x \bmod 7$

Load: 57.1%

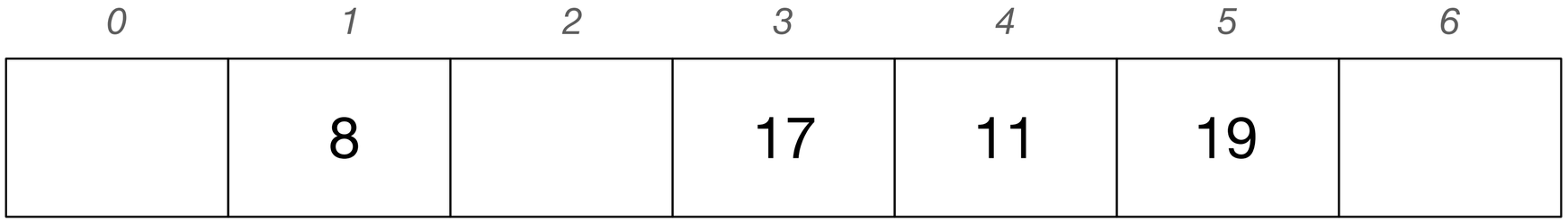
Time to rehash!

7 x 2 = 14

next prime is 17

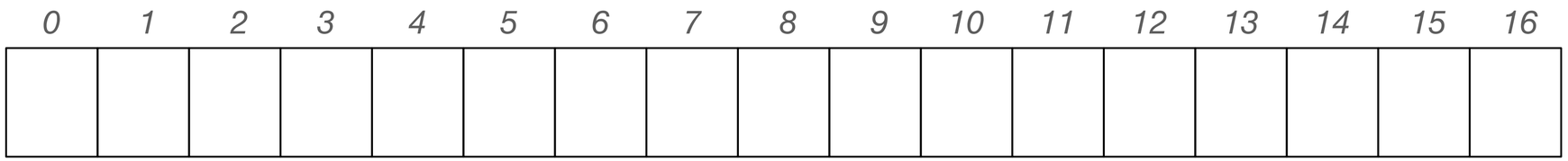
Rehashing

Max. load factor (fill percentage): 50%



Hash function:
$f(x) = x \bmod 7$

Load: 57.1%

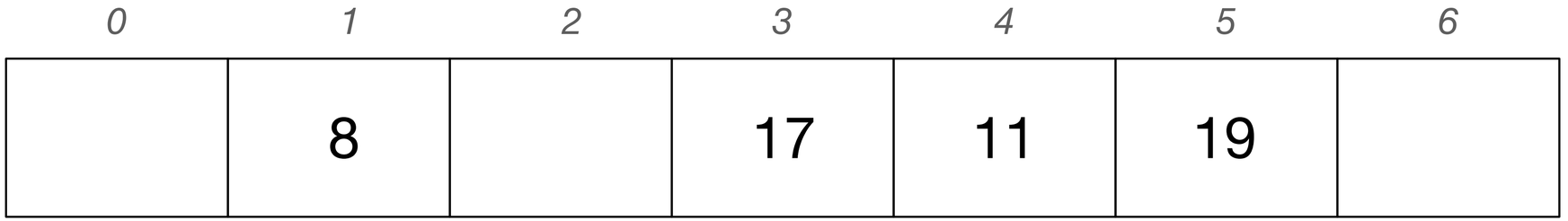


Hash function:
$f(x) = x \bmod 17$

Load: 0.0%

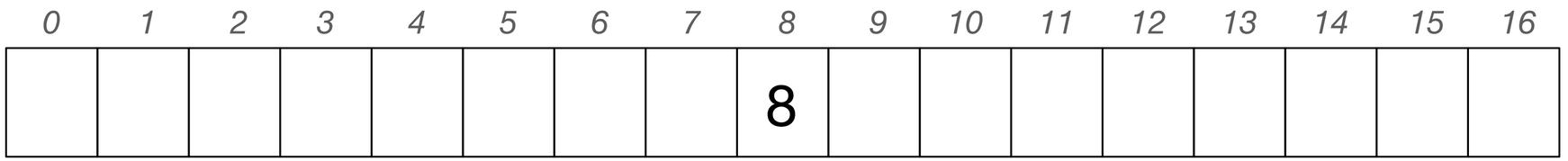
Rehashing

Max. load factor (fill percentage): 50%



Hash function:
$f(x) = x \bmod 7$

Load: 57.1%

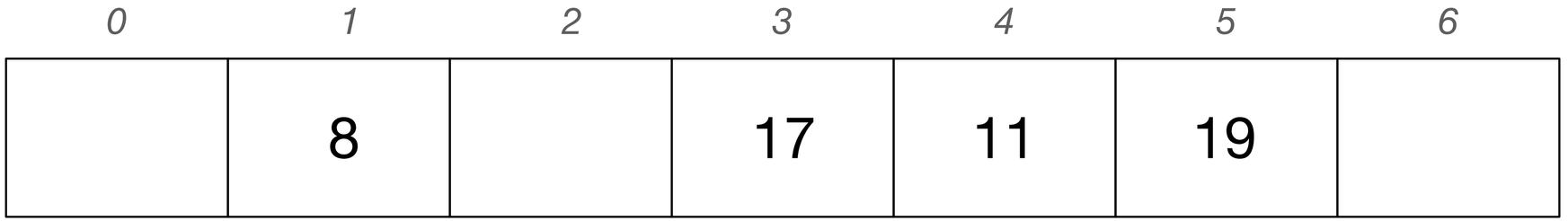


Hash function:
$f(x) = x \bmod 17$

Load: 5.8%

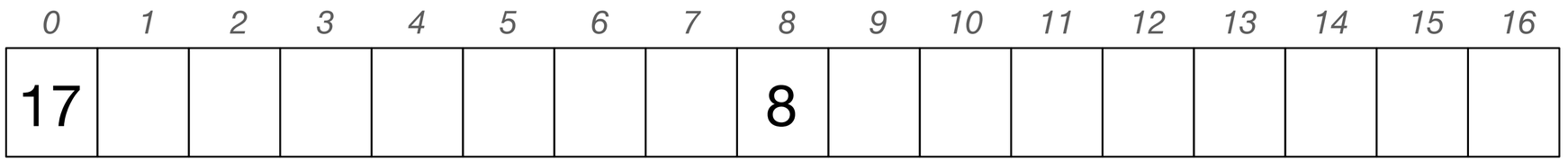
Rehashing

Max. load factor (fill percentage): 50%



Hash function:
$f(x) = x \bmod 7$

Load: 57.1%

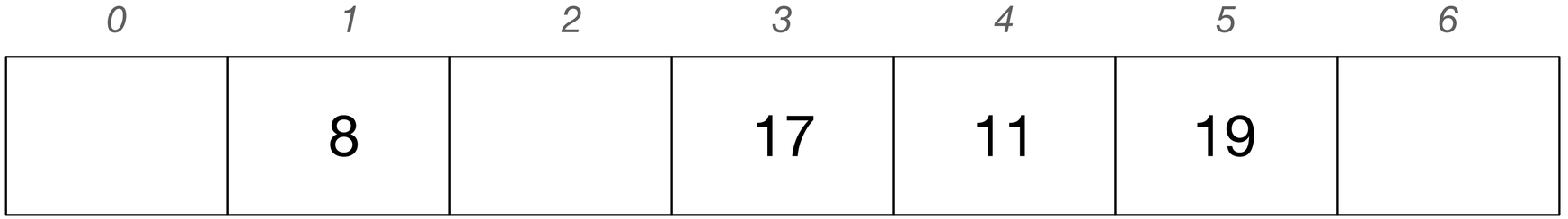


Hash function:
$f(x) = x \bmod 17$

Load: 11.7%

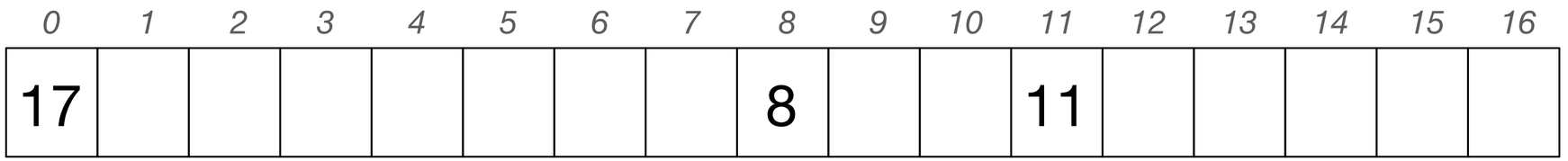
Rehashing

Max. load factor (fill percentage): 50%



Hash function:
$f(x) = x \bmod 7$

Load: 57.1%

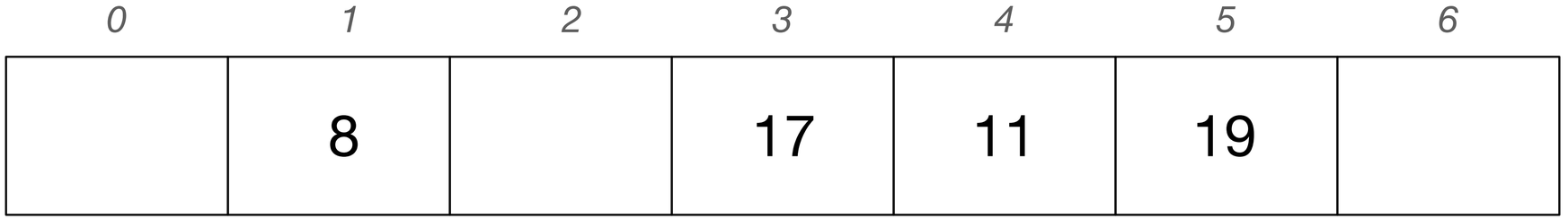


Hash function:
$f(x) = x \bmod 17$

Load: 17.6%

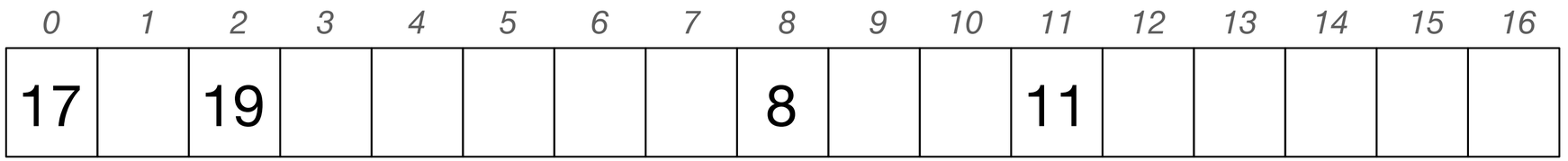
Rehashing

Max. load factor (fill percentage): 50%



Hash function:
$f(x) = x \bmod 7$

Load: 57.1%

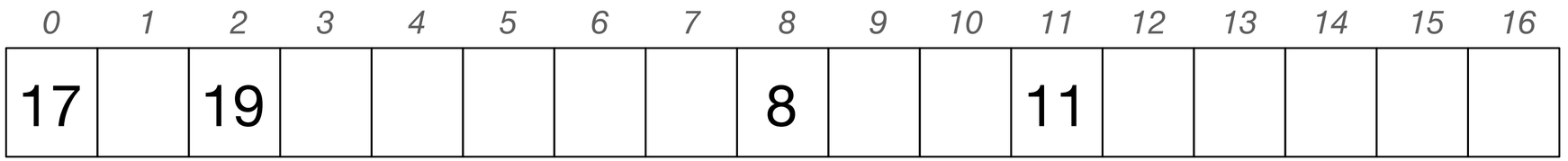


Hash function:
$f(x) = x \bmod 17$

Load: 23.5%

Rehashing

Max. load factor (fill percentage): 50%



Hash function:
$f(x) = x \bmod 17$

Load: 23.5%

Rehashing

Rehash as needed on inserts to keep load factor below threshold

Typically we use load factors between 50% and 75% as thresholds

If we use 50% as our threshold, this means that at least half of our hash table will be empty at any given time. Why is this good?

Rehashing

Rehash as needed on inserts to keep load factor below threshold

Typically we use load factors between 50% and 75% as thresholds

If we use 50% as our threshold, this means that at least half of our hash table will be empty at any given time. Why is this good?

- Reduced frequency of collisions
- Shorter probing sequences

What do we need to implement rehashing?

- A function which when given an integer calculates the next prime number
- A method to perform rehashing
- Modify the insert function to check the load factor