Exercise

write some php code that multiplies two numbers the old-fashioned way... the product of 5 and 8 is:

8 + 8 + 8 + 8 + 8 = 40

your code should define two variables to be multiplied and after computing the product print out the product

www.c9.io

NOTE: run php multiply.php

```php
<?php
    $x = 5;
    $y = 8;
    $prod = 0;
    for( $i = 1; $i <= $x; $i++ ) {
        $prod = $prod + $y;
    }
    print( "$prod" );
?>
```

```php
<?php
    $x = 5;
    $y = 8;
    $prod = 0;
    for( $i = 1; $i <= $x; $i++ ) {
        $prod = $prod + $y;
    }
    print( "$prod" );
?>
```

always think about edge cases

Loops

```
$i = 0;
while( $i <= 10 ) {
    print( "$i\n" );
    $i++;
}
```

same as Javascript

Loops

```php
<?php
    $i = -10;
    while( $i <= 10 ) {
        if( $i != 0 ) {
            $r = 1/$i;
            print( "$r\n" );
            $i++;
        }
    }
?>
```

where is the bug?

Loops

```php
<?php
    $i = -10;
    while( $i <= 10 ) {
        if( $i != 0 ) {
            $r = 1/$i;
            print( "$r\n" );
        }
        $i++;
    }
?>
```

where is the bug?

Loops

```
<?php
    $i = -10;
    while( $i <= 10 ) {
        if( $i == 0 ) {
            continue;
        }
        $r = 1/$i;
        print( "$r\n" );
        $i++;
    }
?>
```

continue: stop executing *this* iteration of the loop

Loops

```php
<?php
    $i = -10;
    while( $i <= 10 ) {
        if( $i == 0 ) {
            continue;
        }
        $r = 1/$i;
        print( "$r\n" );
        $i++;
    }
?>
```

where is the bug?

Loops

```php
<?php
    $i = -10;
    while( $i <= 10 ) {
        if( $i == 0 ) {
            $i++;
            continue;
        }
        $r = 1/$i;
        print( "$r\n" );
        $i++;
    }
?>
```

Loops

```php
<?php
    $i = -10;
    while( $i <= 10 ) {
        if( $i == 0 ) {
            break;
        }
        $r = 1/$i;
        print( "$r\n" );
        $i++;
    }
?>
```

break: stop executing *entire* loop

Functions

```php
<?php

    function myAdd( $a, $b ) {
        $s = $a + $b;
        return( "$s" );
    }

    $x = myAdd( 1, 2 );

?>
```

Functions

```
<?php

    function myAdd( $a, $b ) {
        $s = $a + $b;
        return( "$s" );
    }

    $x = myAdd( 1, 2 );

?>
```

Functions

```php
<?php

    function myAdd( $a, $b ) {
        $s = $a + $b;
        return( "$s" );
    }

    $x = myAdd( 1, 2 );

?>
```

pass parameters to function

Functions

```php
<?php

    function myAdd( $a, $b ) {
        $s = $a + $b;
        return( $s );
    }

    $x = myAdd( 1, 2 );

?>
```

pass parameters to function

Exercise

write some php code to determine if a number is prime. Define a function “dividesEvenly” that takes as input two parameters (x,y) and returns True if y divides x evenly and False otherwise.

Define a second function isPrime that takes as input a single number (x) and returns True if it is prime and False otherwise. This function should loop over all values between 1 and x-1 checking to see if any of these values divides x.

To test your code, write a loop to test if all numbers between 1 and 100 are prime.

www.c9.io

```php
<?php

   function dividesEvenly( $x, $y ) {
       return( ($x % $y) == 0 );
   }

   function isPrime( $x ) {
      for( $i=2 ; $i < $x ; $i++ ) {
          if( dividesEvenly($x,$i) ) {
              return( False );
          }
      }
      return( True );
   }

   for($x=1 ; $x<=100 ; $x++ ) {
    if( isPrime($x) ) {
        print( "$x is prime\n" );
    }
    else {
        print( "$x is not prime\n" );
    }
   }
?>
```

```php
<?php

    function dividesEvenly( $x, $y ) {
        return( ($x % $y) == 0 );
    }

    function isPrime( $x ) {
       for( $i=2 ; $i < $x ; $i++ ) {
           if( dividesEvenly($x,$i) ) {
               return( False );
           }
       }
       return( True );
    }

    for( $x=1 ; $x<=100 ; $x++ ) {
        if( isPrime($x) ) {
            print( "$x is prime\n" );
        }
        else {
            print( "$x is not prime\n" );
        }
    }
?>
```

```php
<?php

    function dividesEvenly( $x, $y ) {
        return( ($x % $y) == 0 );
    }

    function isPrime( $x ) {
        for( $i=2 ; $i < $x ; $i++ ) {
            if( dividesEvenly($x,$i) ) {
                return( False );
            }
        }
        return( True );
    }

    for( $x=1 ; $x<=100 ; $x++ ) {
        if( isPrime($x) ) {
            print( "$x is prime\n" );
        }
        else {
            print( "$x is not prime\n" );
        }
    }
?>
```

```php
<?php

    function dividesEvenly( $x, $y ) {
        return( ($x % $y) == 0 );
    }

    function isPrime( $x ) {
        for( $i=2 ; $i < $x ; $i++ ) {
            if( dividesEvenly($x,$i) ) {
                return( False );
            }
        }
        return( True );
    }

    for( $x=1 ; $x<=100 ; $x++ ) {
        if( isPrime($x) ) {
            print( "$x is prime\n" );
        }
        else {
            print( "$x is not prime\n" );
        }
    }
?>
```

```php
<?php

    function dividesEvenly( $x, $y ) {
        return( ($x % $y) == 0 );
    }

    function isPrime( $x ) {
       for( $i=2 ; $i < $x ; $i++ ) {
           if( dividesEvenly($x,$i) ) {
               return( False );
           }
       }
       return( True );
    }

    for( $x=1 ; $x<=100 ; $x++ ) {
        if( isPrime($x) ) {
            print( "$x is prime\n" );
        }
        else {
            print( "$x is not prime\n" );
        }
    }
?>
```

```php
<?php

    function dividesEvenly( $x, $y ) {
        return( ($x % $y) == 0 );
    }

    function isPrime( $x ) {
       for( $i=2 ; $i < $x ; $i++ ) {
           if( dividesEvenly($x,$i) ) {
               return( False );
           }
       }
       return( True );
    }

    for( $x=1 ; $x<=100 ; $x++ ) {
        if( isPrime($x) ) {
            print( "$x is prime\n" );
        }
        else {
            print( "$x is not prime\n" );
        }
    }
?>
```

Arrays

```
$A = array(); // not necessary, but a good idea
$A[0] = "one";
$A[1] = "two";
$A[2] = "three";
```

Arrays

```php
$A = array(); // not necessary, but a good idea

$A[0] = "one";
$A[1] = "two";
$A[2] = "three";

print( "$A" ); // output: Array
```

Arrays

```
$A = array(); // not necessary, but a good idea

$A[0] = "one";
$A[1] = "two";
$A[2] = "three";

print_r( $A ); // output:Array
                        (
                            [0] => one
                            [1] => two
                            [2] => three
                        )
```

Arrays

```
$A = array(); // not necessary, but a good idea

$A[0] = "one";
$A[1] = "two";
$A[2] = "three";

for( $i=0 ; $i<sizeof($A) ; $i++ ) {
   print( "$A[$i]\n" );
}
```

Arrays

```
$A = array(); // not necessary, but a good idea

$A[0] = "one";
$A[1] = "two";
$A[2] = "three";

foreach( $A as $a ) {
   print( "$a\n" );
}
```

```
[0, 3, 2, 3, 4, 3, 2, 2, 1, 7, 6, 4, 5, 8, 3, 2, \
1, 2, 1, 1, 1, 0, 0, 0, 2, 2, 3, 4, 5, 6, 2, 9, 8]
```

```
$digits = Array(0, 3, 2, 3, 4, 3, 2, 2, 1, 7, 6, 4, 5, 8, 3, 2, \
                1, 2, 1, 1, 1, 0, 0, 0, 2, 2, 3, 4, 5, 6, 2, 9, 8);

$count = Array(0,0,0,0,0,0,0,0,0,0);
```

```
$digits = Array(0, 3, 2, 3, 4, 3, 2, 2, 1, 7, 6, 4, 5, 8, 3, 2, \
              1, 2, 1, 1, 1, 0, 0, 0, 2, 2, 3, 4, 5, 6, 2, 9, 8);


$count = Array(0,0,0,0,0,0,0,0,0,0);
foreach( $digits as $d ) {
    $count[$d]++;
}
print_r( $count );
```

```
$digits = Array(0, 3, 2, 3, 4, 3, 2, 2, 1, 7, 6, 4, 5, 8, 3, 2, \
              1, 2, 1, 1, 1, 0, 0, 0, 2, 2, 3, 4, 5, 6, 2, 9, 8);


$count = Array(0,0,0,0,0,0,0,0,0,0);
foreach( $digits as $d ) {
    $count[$d]++;
}
print_r( $count );
```

```
[4, 5, 8, 5, 3, 2, 2, 1, 2, 1]
```

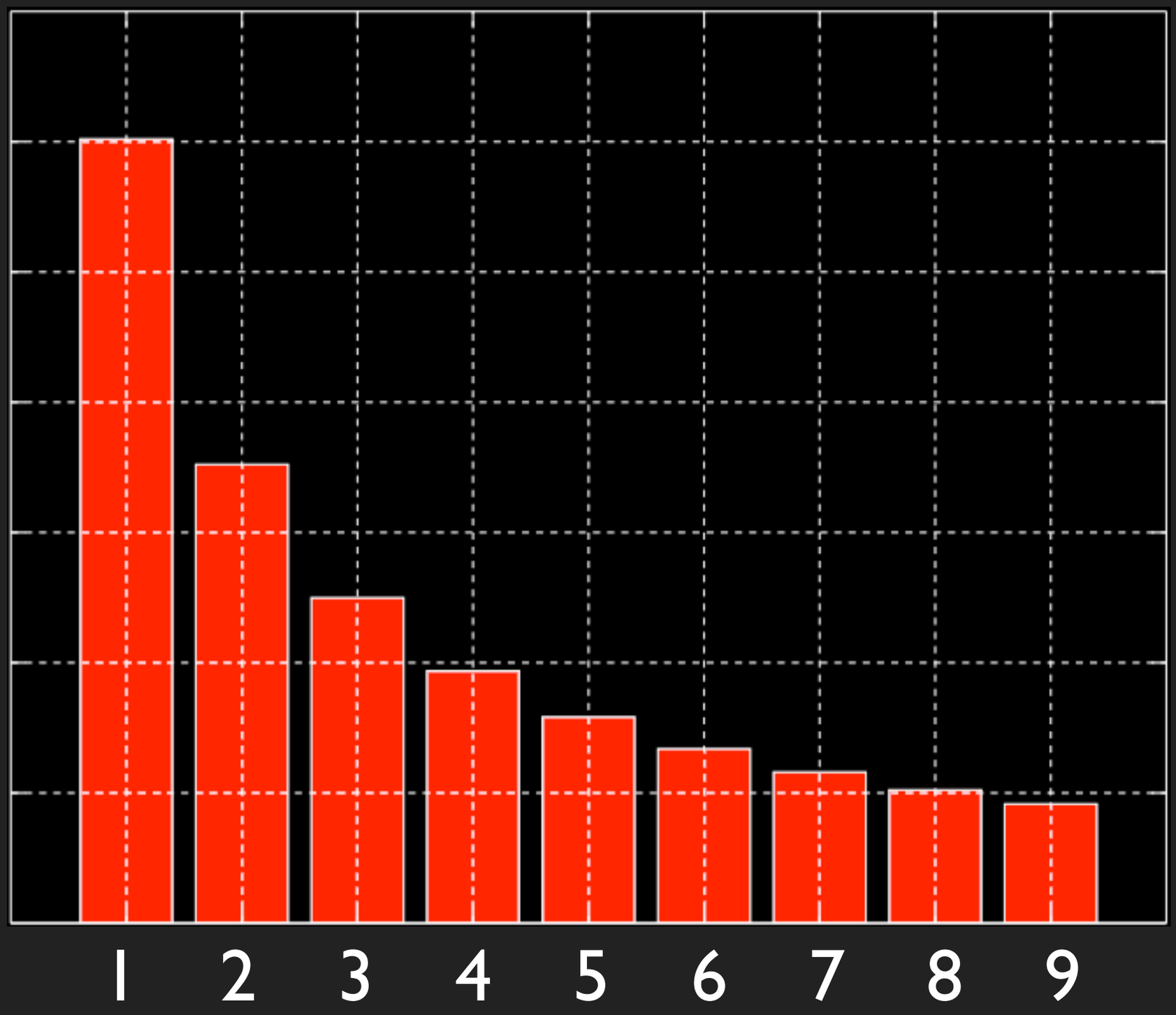
Benford's Law

	population (millions)
China	1,458
India	1,398
United States	352
Indonesia	273
Brazil	223
Pakistan	226
Bangladesh	198
Nigeria	208
Russia	137
Japan	126

Benford's Law

	population (millions)
China	1,458
India	1,398
United States	352
Indonesia	273
Brazil	223
Pakistan	226
Bangladesh	198
Nigeria	208
Russia	137
Japan	126

Benford's Law



Benford's Law

http://testingbenfordslaw.com

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

```
counter["was"] ?
```

Associative Arrays (Dictionaries)

```
$B = Array();
$B["one"] = 1;
$B["two"] = 2;
$B["three"] = 3;
```

Dictionaries

A dictionary is a data structure that stores relationships between key-value pairs.

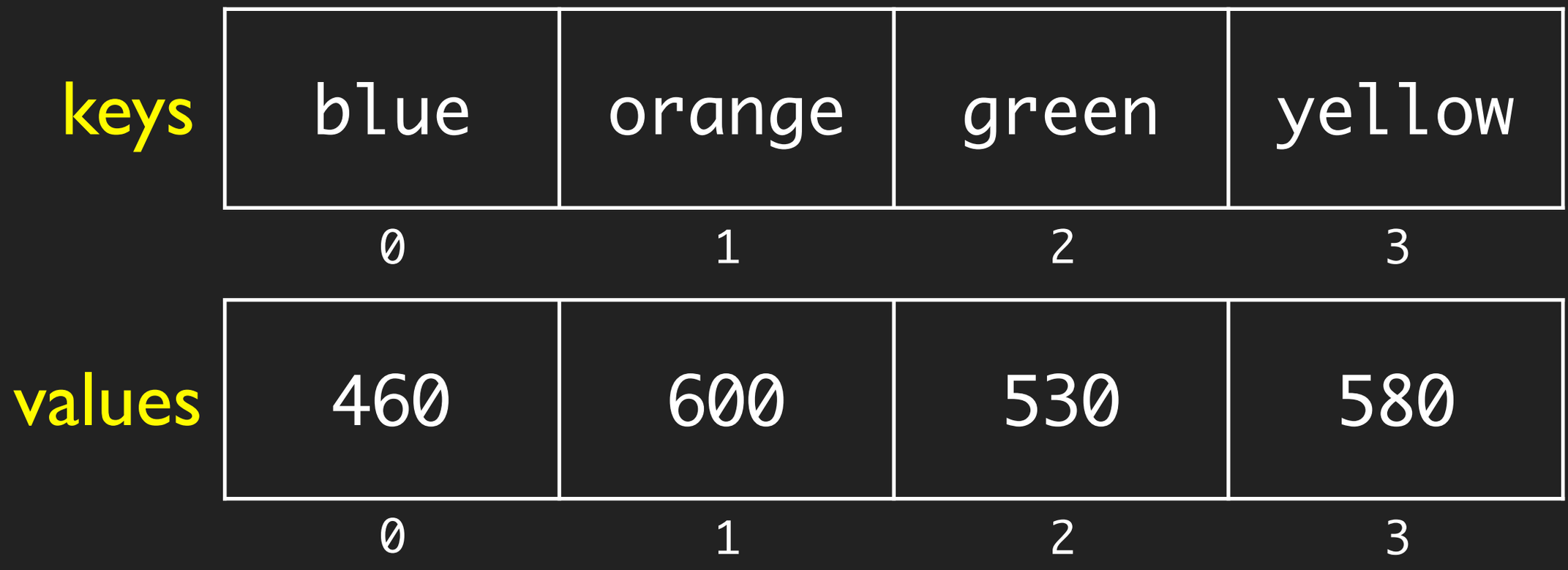
The key is the item used to index into the dictionary.

The value is the item stored in the dictionary.

Dictionaries

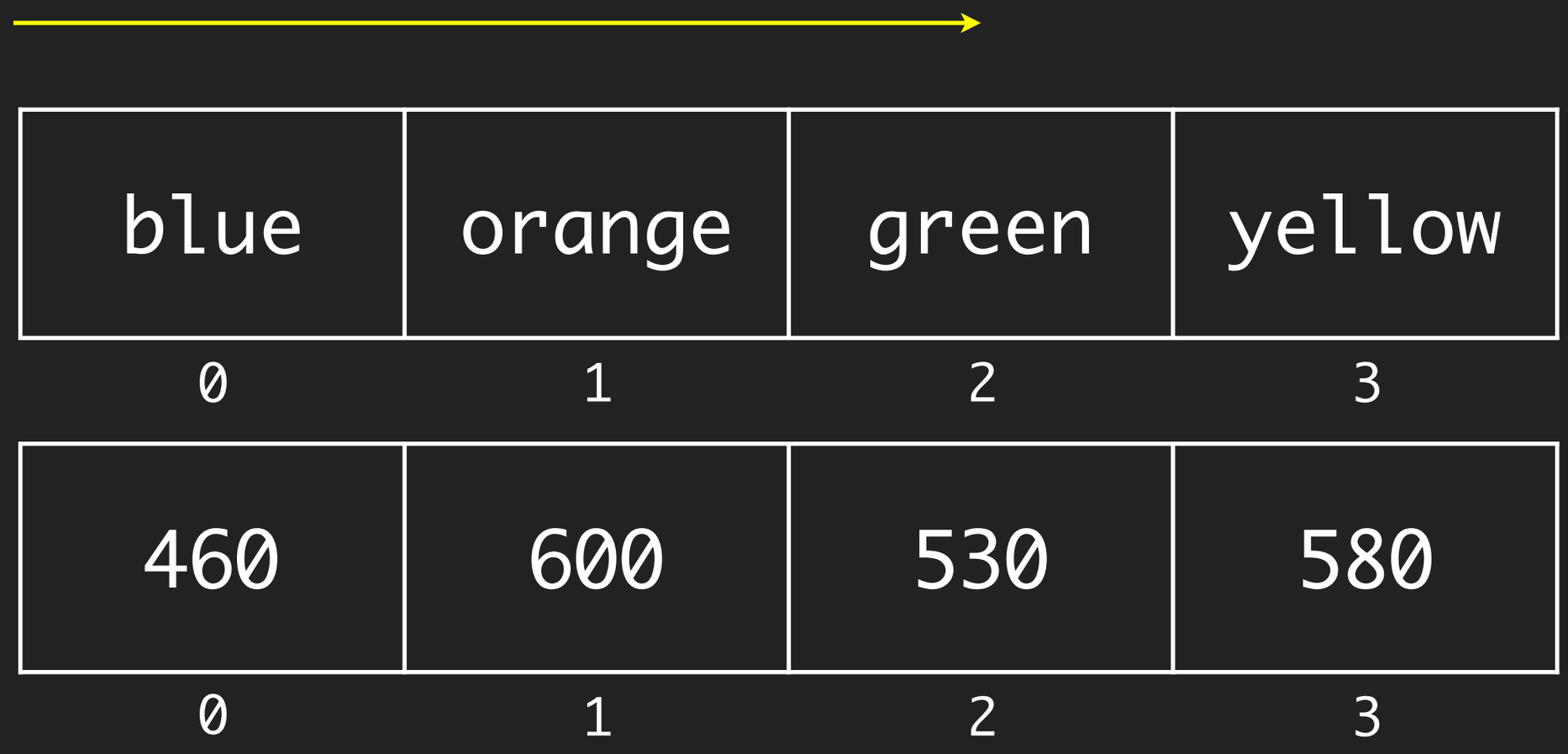
```
my_dictionary = Array();

my_dictionary["blue"] = 460;
my_dictionary["orange"] = 600;
my_dictionary["green"] = 530;
my_dictionary["yellow"] = 580;
```

Dictionaries

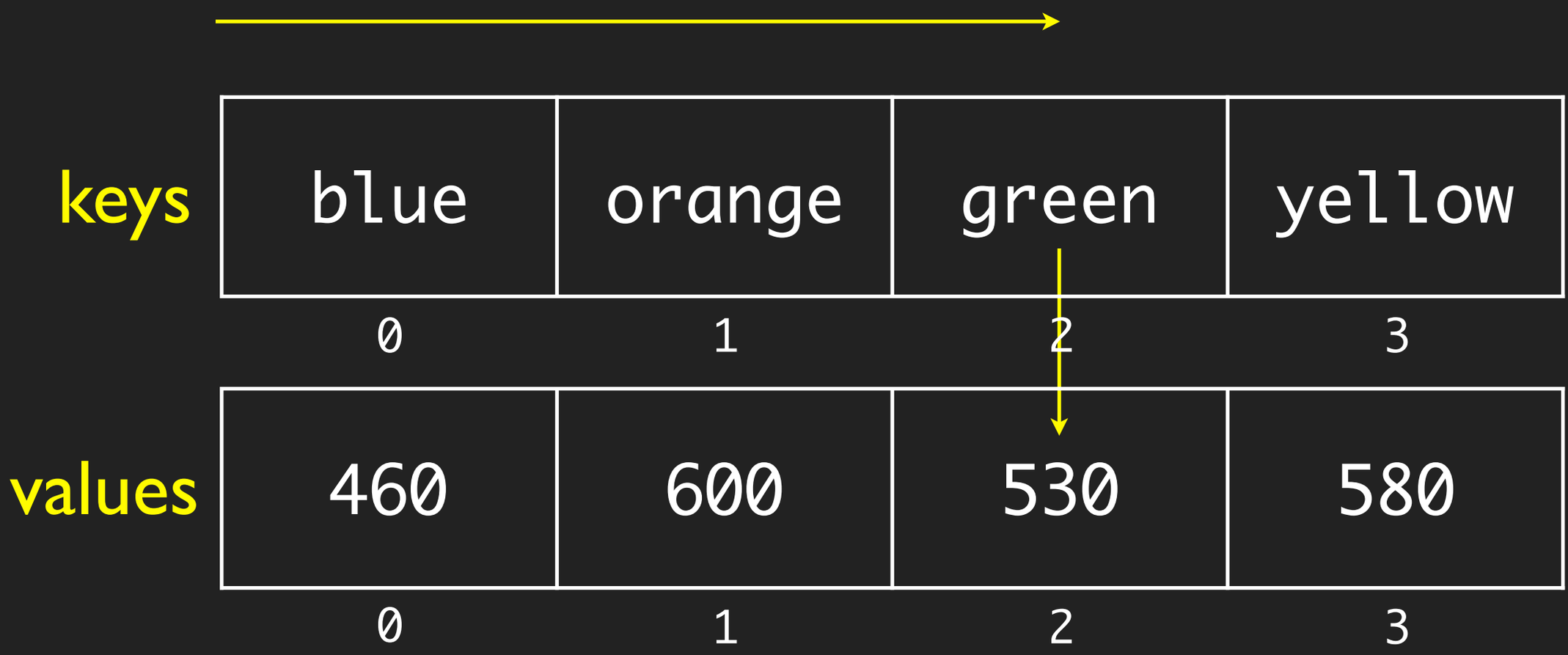



```
print my_dictionary["green"]
```

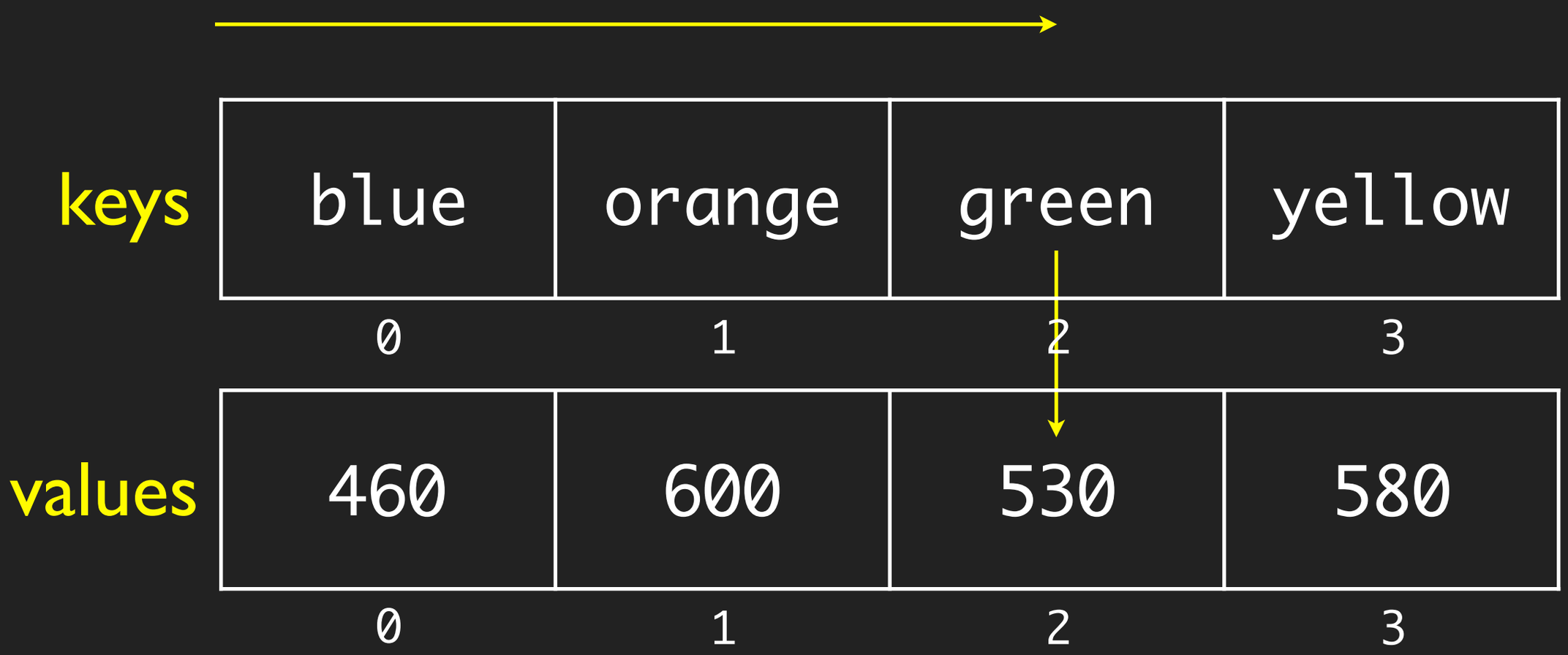
Dictionaries



Dictionaries



Dictionaries



O(n)

Hashing

blue	orange	green	yellow
460	600	530	580
0	1	2	3

Hashing

blue	orange	green	yellow	*key*
460	600	530	580	
0	1	2	3	*index*

f(key) = index

f() is *efficient* to compute

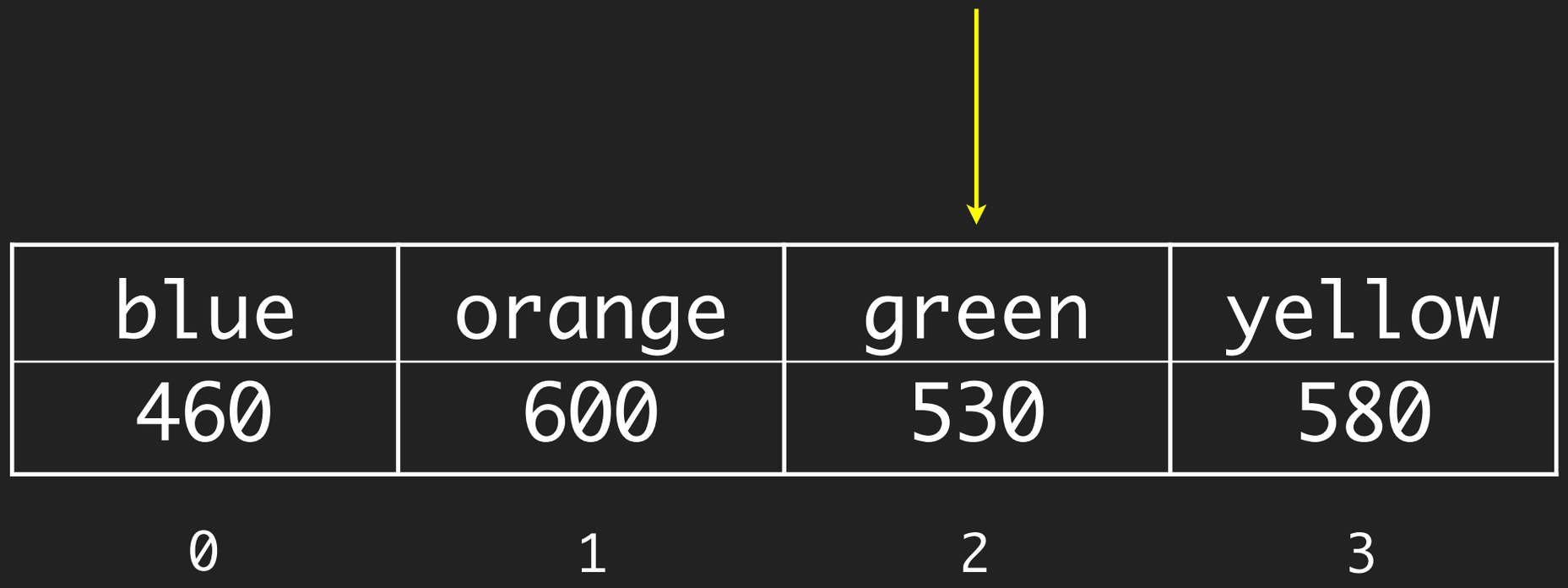
f() produces a *unique* index for each key

Hashing

blue	orange	green	yellow
460	600	530	580
0	1	2	3

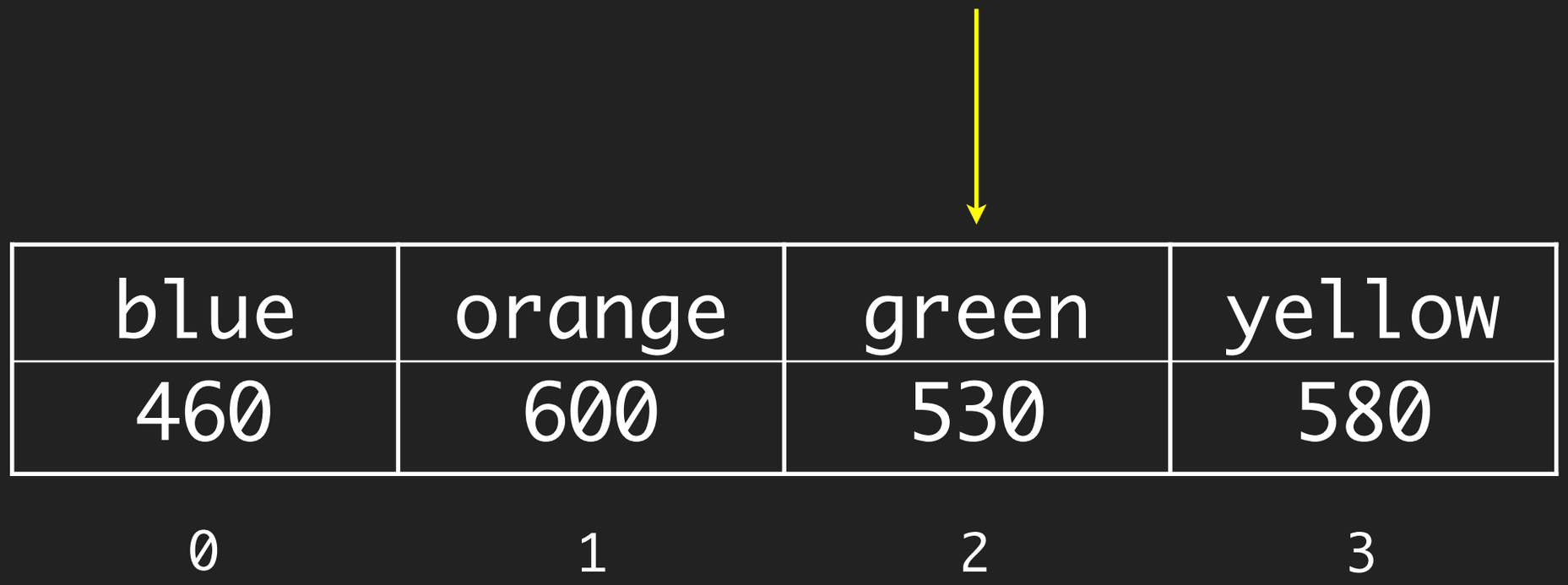
f("blue") = 0
f("orange") = 1
f("green") = 2
f("yellow") = 3

Hashing




```
print my_dictionary["green"]
```

Hashing




```
print my_dictionary["green"]
```

O(1)

Hashing

```php
<?php

function myHash( $key, $n ) {
	$hash_code = 0;
	for( $i=0 ; $i<strlen($key) ; $i++ ) {
		$hash_code += hash_code + ord($key[$i]);
	}
	return( $hash_code % n );
}

print( myHash("test",3) );

?>
```

Hashing

```php
<?php

function myHash( $key, $n ) {
	$hash_code = 0;
	for( $i=0 ; $i<strlen($key) ; $i++ ) {
		$hash_code += hash_code + ord($key[$i]);
	}
	return( $hash_code % n );
}

print( myHash("test",4) );

?>
```

Hashing

```php
<?php

function myHash( $key, $n ) {
	$hash_code = 0;
	for( $i=0 ; $i<strlen($key) ; $i++ ) {
		$hash_code += hash_code + ord($key[$i]);
	}
	return( $hash_code % n );
}

print( myHash("test",4) );

?>
```

Hashing

```php
<?php

function myHash( $key, $n ) {
	$hash_code = 0;
	for( $i=0 ; $i<strlen($key) ; $i++ ) {
		$hash_code += hash_code + ord($key[$i]);
	}
	return( $hash_code % n );
}

print( myHash("test",4) );

?>
```

```
ord("a") = 97
ord("b") = 98
...
```

Hashing

```php
<?php

function myHash( $key, $n ) {
      $hash_code = 0;
      for( $i=0 ; $i<strlen($key) ; $i++ ) {
            $hash_code += hash_code + ord($key[$i]);
      }
      return( $hash_code % n );
}

print( myHash("test",4) );

?>
```

Hashing

```
<?php

function myHash( $key, $n ) {
      $hash_code = 0;
      for( $i=0 ; $i<strlen($key) ; $i++ ) {
            $hash_code += hash_code + ord($key[$i]);
      }
      return( $hash_code % n );
}

print( myHash("test",4) );

?>
```

efficient?

Hashing

```
<?php

function myHash( $key, $n ) {
      $hash_code = 0;
      for( $i=0 ; $i<strlen($key) ; $i++ ) {
            $hash_code += hash_code + ord($key[$i]);
      }
      return( $hash_code % n );
}

print( myHash("test",4) );

?>
```

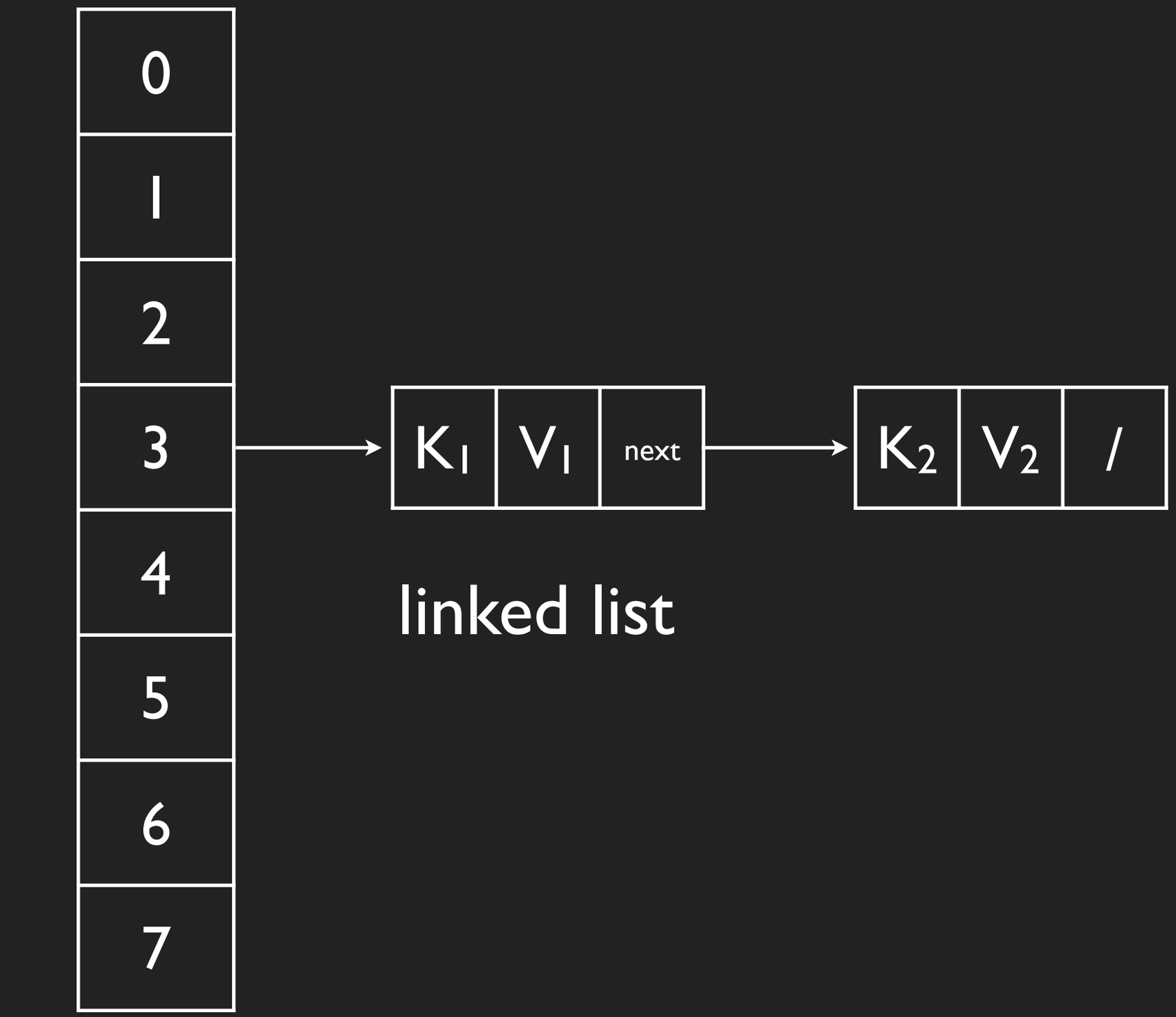
unique?

Hashing

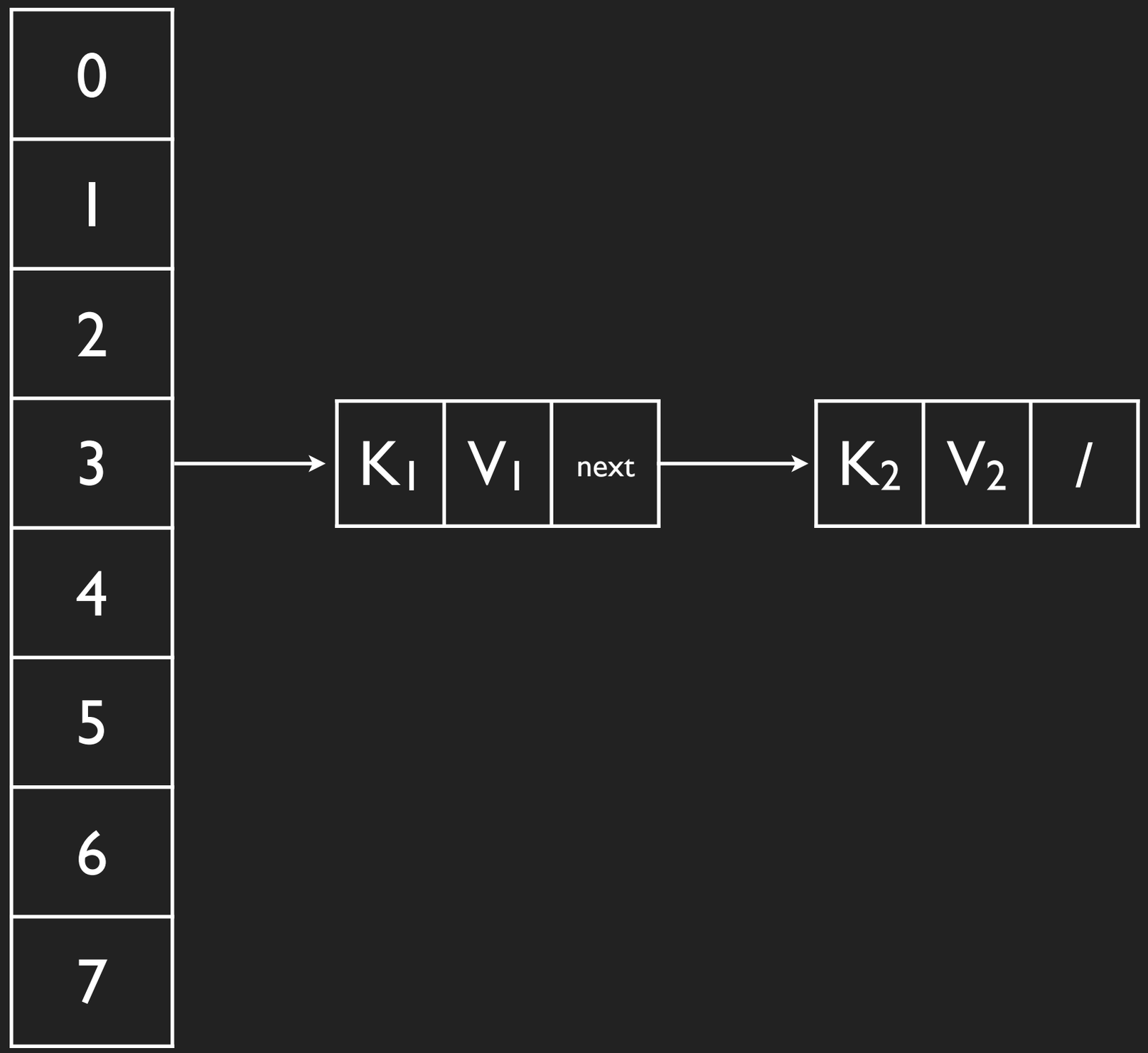
```
<?php

function myHash( $key, $n ) {
      $hash_code = 0;
      for( $i=0 ; $i<strlen($key) ; $i++ ) {
            $hash_code += hash_code + ord($key[$i]);
      }
      return( $hash_code % n );
}

print( myHash("test",4) );

?>
```

myHash("rat") == myHash("tar")

Hashing



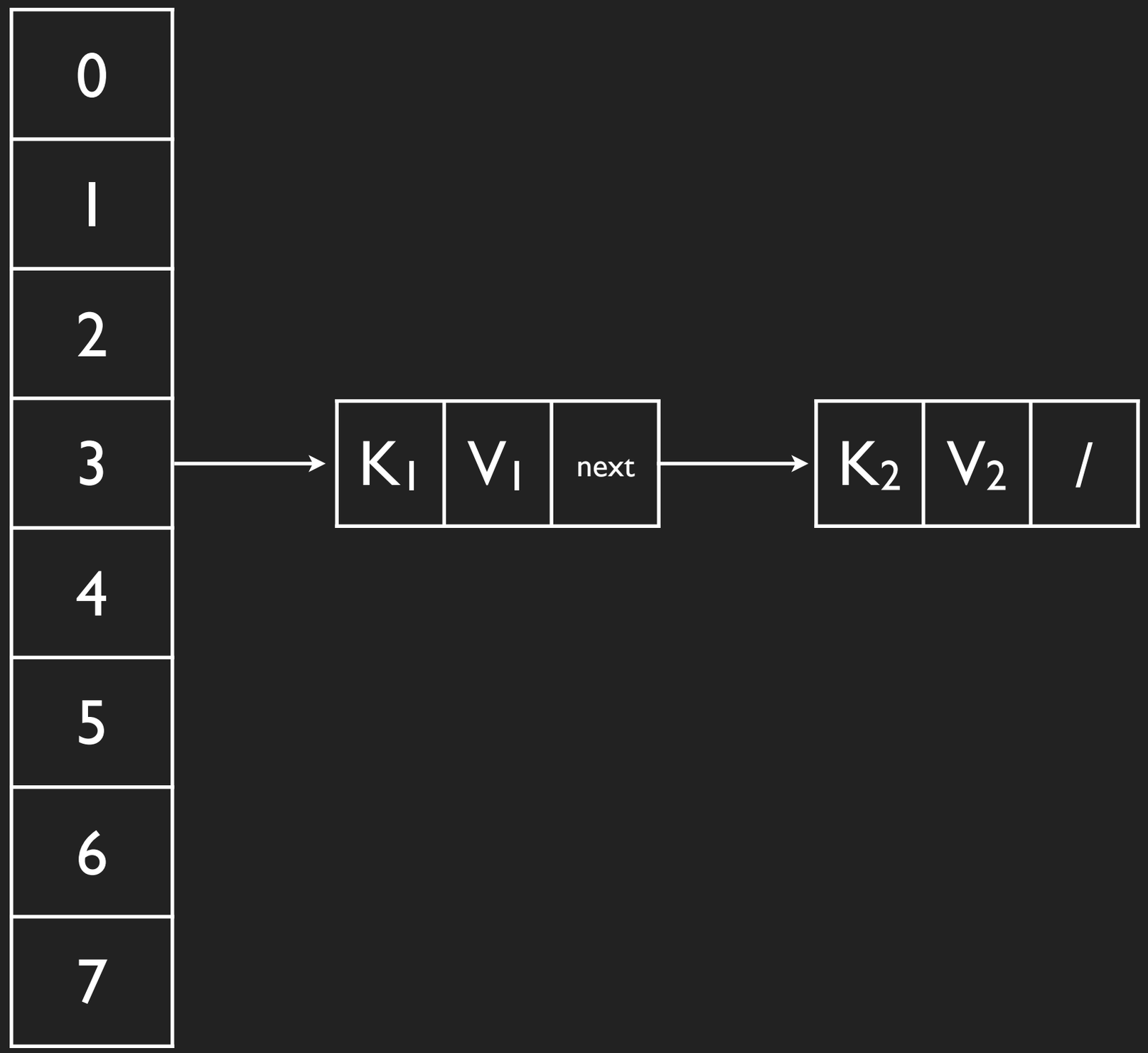
Hashing



hash table

best-case: ?

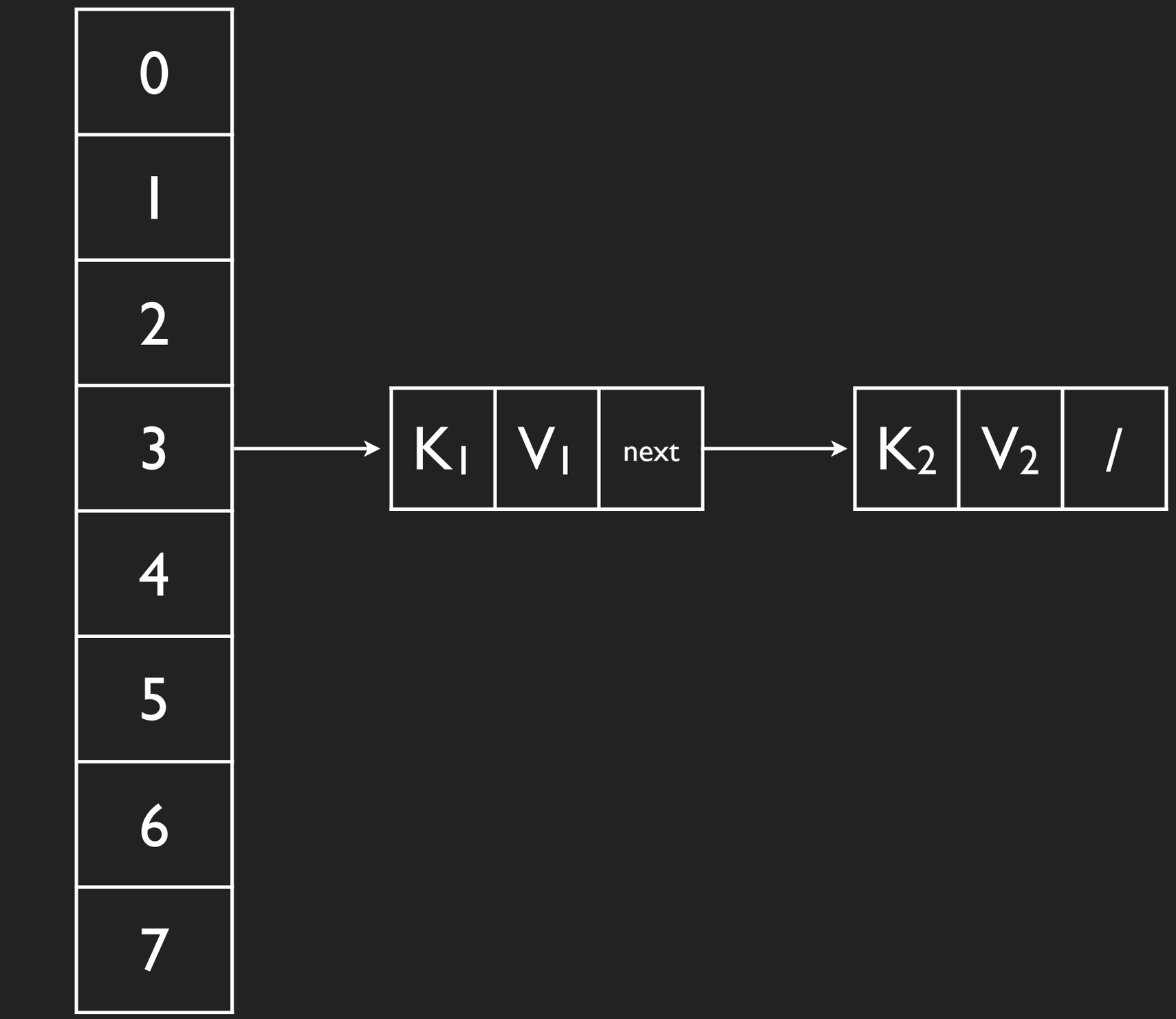
Hashing



hash table

best-case: O(1)

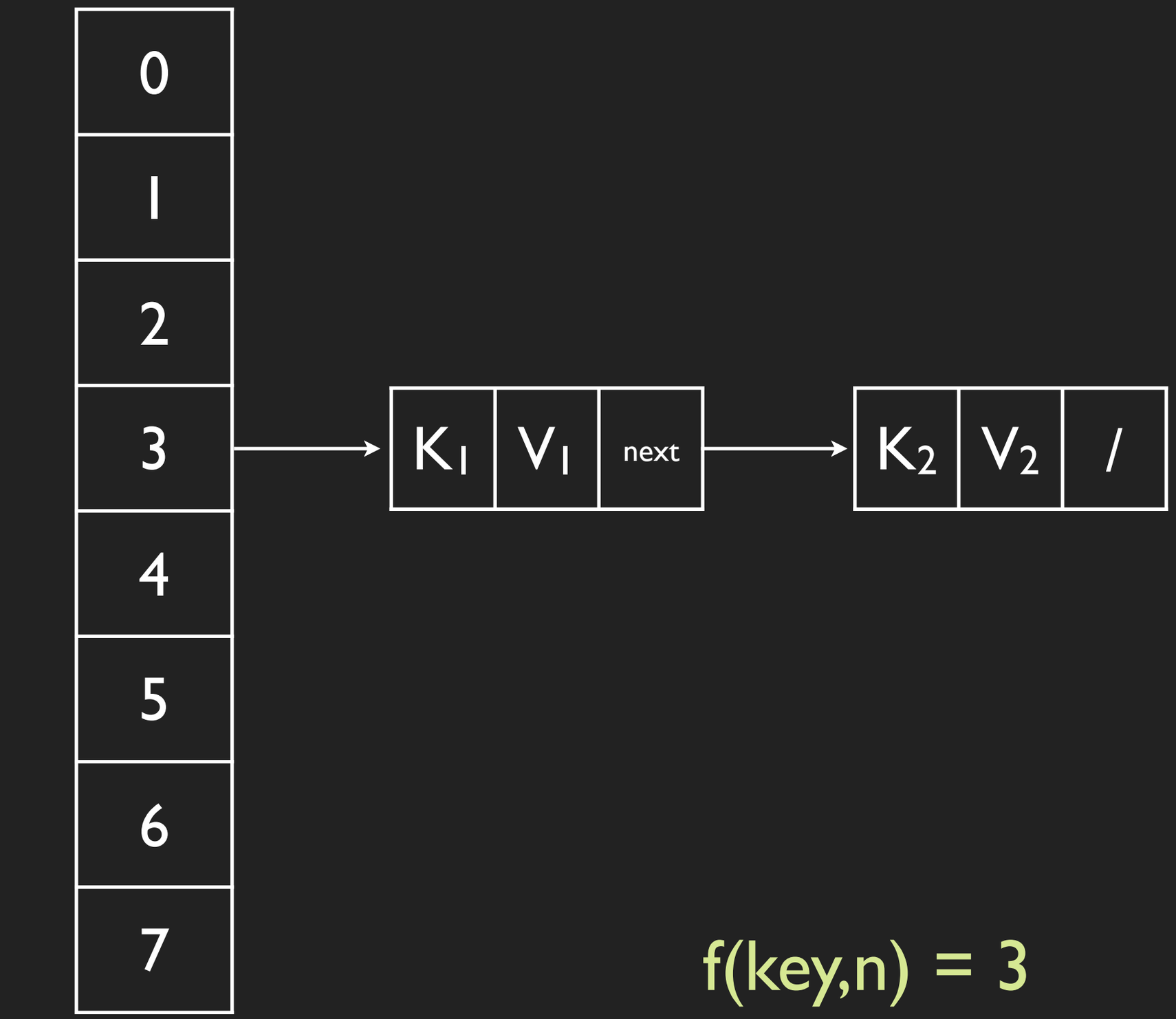
Hashing



hash table

worst-case: ?

Hashing

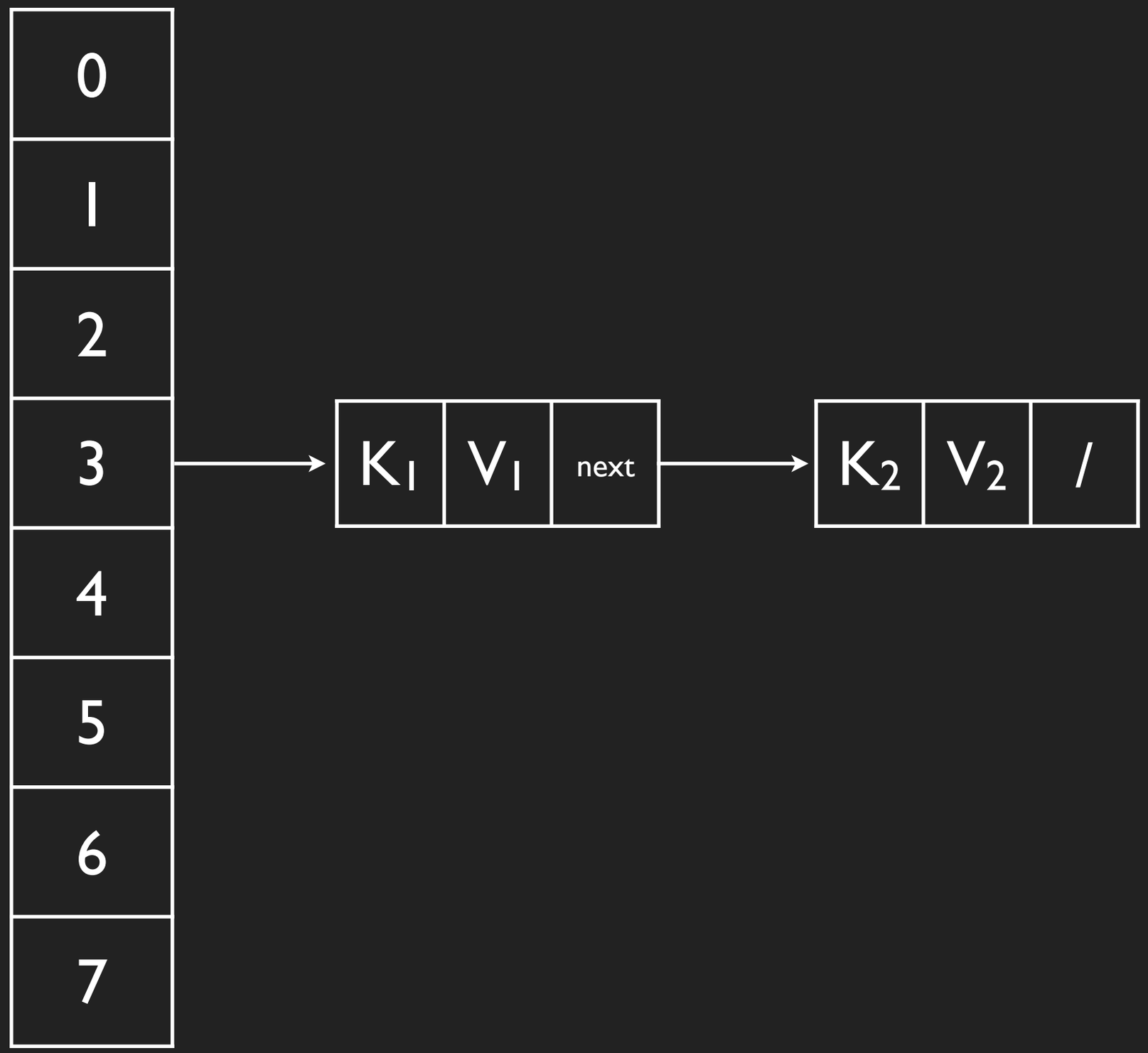


hash table

f(key,n) = 3

worst-case: O(n)

Hashing



hash table

worst-case: O(m), m <<< n

word frequency

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

word frequency

1. parse string into individual words

word frequency

1. parse string into individual words
2. eliminate punctuation

word frequency

1. parse string into individual words
2. eliminate punctuation
3. convert to lower-case

word frequency

1. parse string into individual words
2. eliminate punctuation
3. convert to lower-case
4. build dictionary

word frequency

1. parse string into individual words
2. eliminate punctuation
3. convert to lower-case
4. build dictionary

1. if element is in dictionary, increment count
2. if element is not in dictionary, initialize count to 1

```php
<?php

$s = . . .;

$s = str_replace( ",", "", $s );
$s = str_replace( " -", "", $s );
$s = str_replace( ".", "", $s );
$s = strtolower( $s );
print( $s );

$S = explode( " ", $s );
print_r( $S );

$D = Array();
for( $i=0 ; $i<sizeof($S) ; $i++ ) {
	$key = $S[$i];
	if( array_search( $key, $D ) ) {
		$D[$key]++;
	} else {
		$D[$key] = 1;
	}
}
print_r( $D );

?>
```

```php
<?php

$s = . . .;

$s = str_replace( ",", "", $s );
$s = str_replace( " -", "", $s );
$s = str_replace( ".", "", $s );
$s = strtolower( $s );
print( $s );

$S = explode( " ", $s );
print_r( $S );

$D = Array();
for( $i=0 ; $i<sizeof($S) ; $i++ ) {
     $key = $S[$i];
     if( array_search( $key, $D ) ) {
          $D[$key]++;
     } else {
          $D[$key] = 1;
     }
}
print_r( $D );

?>
```

```php
<?php

$s = . . .;

$s = str_replace( ",", "", $s );
$s = str_replace( " -", "", $s );
$s = str_replace( ".", "", $s );
$s = strtolower( $s );
print( $s );

$S = explode( " ", $s );
print_r( $S );

$D = Array();
for( $i=0 ; $i<sizeof($S) ; $i++ ) {
     $key = $S[$i];
     if( array_search( $key, $D ) ) {
          $D[$key]++;
     } else {
          $D[$key] = 1;
     }
}
print_r( $D );

?>
```

```php
<?php

$s = . . .;

$s = str_replace( ",", "", $s );
$s = str_replace( " -", "", $s );
$s = str_replace( ".", "", $s );
$s = strtolower( $s );
print( $s );

$S = explode( " ", $s );
print_r( $S );

$D = Array();
for( $i=0 ; $i<sizeof($S) ; $i++ ) {
	$key = $S[$i];
	if( array_search( $key, $D ) ) {
		$D[$key]++;
	} else {
		$D[$key] = 1;
	}
}
print_r( $D );

?>
```

```php
<?php

$s = . . .;

$s = str_replace( ",", "", $s );
$s = str_replace( " -", "", $s );
$s = str_replace( ".", "", $s );
$s = strtolower( $s );
print( $s );

$S = explode( " ", $s );
print_r( $S );

$D = Array();
for( $i=0 ; $i<sizeof($S) ; $i++ ) {
     $key = $S[$i];
     if( array_search( $key, $D ) ) {
          $D[$key]++;
     } else {
          $D[$key] = 1;
     }
}
print_r( $D );

?>
```