



UNIT 3B

Algorithmic Thinking

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

1

Finding the maximum

How do we find the maximum in a sequence of integers shown to us one at a time?

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

2

Finding the maximum

Required: a non-empty *list* of integers.

1. Set *max_so_far* equal to the first number in the *list*.
2. For each number *n* in the *list*:
 - a. If *n* is greater than *max_so_far*, then set *max_so_far* equal to *n*.

Return: *max_so_far* as the maximum of the *list*.

Representing Lists in Ruby

In Ruby, we will use an **array** to represent a list of data values.

```
scores = [78, 93, 80, 68, 100, 94, 85]
```

```
colors = ["red", "green", "blue"]
```

An array is an *ordered* list because the order of the elements matters.

Some Array Operations

```
scores = [78, 93, 80, 68, 100, 94, 85]
```

```
scores.length      => 7  
scores.first       => 78  
scores.last        => 85  
scores.first * 2   => 156  
scores.include?(100) => true  
scores[0]          => 78
```

```
scores << 92  
=> [78, 93, 80, 68, 100, 94, 85, 92]
```

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

5

Finding the max using Ruby

```
def findmax(list)  
  max_so_far = list.first # or list[0]  
  for i in (1..list.length-1) do  
    if list[i] > max_so_far then  
      max_so_far = list[i]  
    end  
  end  
  return max_so_far  
end
```

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

6

Alternate Version

```
def findmax(list)
  max_so_far = list.first
  for item in list do ← “For each item
                        in the list...”
    if item > max_so_far then
      max_so_far = item
    end
  end
  return max_so_far
end
```

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

7

Iterators: Using the **each** method

```
scores = [78, 93, 80, 68, 100, 94, 85]
scores.each { |item| ← “For each item in scores...”
  print item
  print " "
}
=> 78 93 80 68 100 94 85
scores.each { |x| ← “For each x in scores...”
  if x % 2 == 1 then
    print x
    print " "
  end
}
=> 93 85
```

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

8

Alternate Version #2

```
def findmax(list)
  max_so_far = list.first
  list.each { |item|
    if item > max_so_far then
      max_so_far = item
    end
  }
  return max_so_far
end
```

Relational Operators

If we want to compare two integers to determine their relationship, we can use these relational operators:

<	less than	<=	less than or equal to
>	greater than	>=	greater than or equal to
==	equal to	!=	not equal to

```
scores = [78, 93, 80, 68, 100, 94, 85]
scores.length == 7           => true
scores.first > 80            => false
```

Arrays: The `delete_if` method

```
scores = [78, 93, 80, 68, 100, 94, 85]
```

```
scores.delete_if{ |n| n < 80 }
```

“For each element `n` in the array `scores`,
delete `n` if `n` is less than 80.”

```
=> [ 93, 80, 100, 94, 85]
```

```
scores.delete_if{ |n| n % 2 == 0 }
```

Sieve of Eratosthenes

To make a list of every prime number less than `n`:

1. Create an array *numlist* with every integer from 2 to `n`, in order. (Assume `n > 1`.)
2. Create an empty array *primes*.
3. Copy the first number in *numlist* to the end of *primes*.
4. Iterate over *numlist* to remove every number that is a multiple of the most recently discovered prime number.
5. Halt if every number in *numlist* is prime. Otherwise, go back to step 3.

Arrays: Two Special Cases

```
values = []
```

```
=> []
```

This is the empty array (an array with 0 length).

```
values = Array(1..8)
```

```
=> [1, 2, 3, 4, 5, 6, 7, 8]
```

Starting the algorithm in Ruby

```
def sieve(n)  
  numlist = Array(2..n)  
  primes = []  
  primes << numlist.first  
  
  ...  
end
```

Removing multiples of a prime

Where is the most recent prime added to the `primes` list?

```
primes.last
```

How do we determine whether a number `x` is a multiple of the most recent prime?

Use the modulo operator!

```
x % primes.last == 0
```

If `x` is a multiple of the most recent prime, it's not prime!

```
numlist.delete_if { |x| x % primes.last == 0 }
```

Continuing the algorithm in Ruby

```
def sieve(n)
  numlist = Array(2..n)
  primes = []
  primes << numlist.first
  numlist.delete_if { |x|
    x % primes.last == 0
  }
  ...
end
```


This algorithm has a loop

We need to repeat the following two steps:

```
primes << numlist.first  
numlist.delete_if { |x| x % primes.last == 0 }
```

Example: start with `numlist = Array(2..25)`

```
primes = [2]  
numlist = [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
```

```
primes = [2, 3]  
numlist = [5, 7, 11, 13, 17, 19, 23, 25]
```

...

When do we stop?

We need to repeat the following two steps:

```
primes << numlist.first  
numlist.delete_if { |x| x % primes.last == 0 }
```

while what is true?

```
numlist.length > 0  
or  numlist.length >= 1  
or  numlist.length != 0
```

Final Algorithm in Ruby

```
def sieve(n)
  numlist = Array(2..n)
  primes = []
  while numlist.length > 0 do
    primes << numlist.first
    numlist.delete_if { |x|
      x % primes.last == 0
    }
  end
  return primes
end
```