



UNIT 4A

Iteration: Searching

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

1

Goals of this Unit

- Study an iterative algorithm called linear search that finds the first occurrence of a target in a collection of data.
- Study an iterative algorithm called insertion sort that sorts a collection of data into non-decreasing order.
- Learn how these algorithm scale as the size of the collection grows.
- Express the amount of work each algorithm performs as a function of the amount of data being processed.

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

2

Searching



Built-in Search in Ruby

```
movies = ["up", "wall-e", "toy story",  
          "monsters inc", "cars", "bugs life",  
          "finding nemo", "the incredibles",  
          "ratatouille"]
```

```
movies.index("cars")      => 4
```

```
movies.index("shrek")     => nil
```

```
movies.index("Up")       => nil
```

```
movies.include?("wall-e") => true
```

```
movies.include?("toy")   => false
```

A Little More about Strings

You can use relational operators to compare strings.

Comparisons are done character by character using ASCII codes.

```
"smithers" > "burns"           => true
"homer" < "marge"             => true
"homer" < "Marge"             => false
"clancy" > "cletus"           => false
"bart" < "bartholomew"       => true
```

Extended ASCII table

1	30 !	65 A	97 a	129 0	161 i	195 A	229 á
2 -	34 "	66 B	98 b	130 ,	162 ¢	196 Á	230 â
3 -	35 #	67 C	99 c	131 /	163 £	197 À	231 ã
4 -	36 \$	68 D	100 d	132 "	164 ¤	198 Ã	232 ä
5 -	37 %	69 E	101 e	133 .	165 ¥	199 Ä	233 å
6 -	38 &	70 F	102 f	134 +	166 ¦	200 Å	234 æ
7 -	39 '	71 G	103 g	135 =	167 §	201 Æ	235 ç
8 -	40 (72 H	104 h	136 ^	168 ¨	202 Ç	236 è
9 -	41)	73 I	105 i	137 %	169 ©	203 È	237 é
10 -	42 *	74 J	106 j	138 \$	170 ª	204 É	238 ê
11 -	43 +	75 K	107 k	139 <	171 «	205 Ê	239 ë
12 0	44 ,	76 L	108 l	140 >	172 ¬	206 Ë	240 ì
13 -	45 -	77 M	109 m	141 ñ	173 ¯	207 Ì	241 í
14 #	46 .	78 N	110 n	142 Ž	174 ®	208 Í	242 î
15 x	47 /	79 O	111 o	143 ò	175 ¯	209 Î	243 ï
16 -	48 0	80 P	112 p	144 ó	176 ´	210 Ï	244 ð
17 ◀	49 1	81 Q	113 q	145 ´	177 ±	211 Ò	245 é
18 0	50 2	82 R	114 r	146 ¨	178 ¨	212 Ó	246 ê
19 !!	51 3	83 S	115 s	147 ¨	179 ¨	213 Ô	247 ë
20 ¶	52 4	84 T	116 t	148 ¨	180 ¨	214 Õ	248 ì
21 -	53 5	85 U	117 u	149 -	181 µ	215 Ö	249 í
22 -	54 6	86 V	118 v	150	182 ¶	216 Ø	250 ð
23 -	55 7	87 W	119 w	151 —	183 ·	217 Ù	251 ó
24 ^	56 8	88 X	120 x	152 ~	184 ¸	218 Ú	252 ô
25 -	57 9	89 Y	121 y	153 ™	185 ¸	219 Û	253 õ
26 →	58 :	90 Z	122 z	154 §	186 ¸	220 Ü	254 ö
27 .	59 ;	91 [123 {	155 ¸	187 ¸	221 Ý	255 ý
28 -	60 <	92 \	124	156 œ	188 ¼	222 Þ	
29 -	61 =	93]	125 }	157 ð	189 ½	223 ß	
30 -	62 >	94 ^	126 ~	158 ð	190 ¾	224 à	
31 -	63 ?	95 _	127 ¯	159 ÿ	191 ù		
32 -	64 @	96 `	128 ¯	160	192 À		

Containment

Design an algorithm that returns **true** if a list contains a desired “key”, or **false** otherwise.

A contains? method

```
def contains?(list, key)
  index = 0
  while index < list.length do
    if list[index] == key then
      return true
    end
    index = index + 1
  end
  return false
end
```

← What happens if we execute **return** before we reach the end of the method?

A contains? method – version 2

```
def contains?(list, key)
  for item in list do
    if item == key then
      return true
    end
  end
  return false
end
```

A contains? method – version 3

```
def contains?(list, key)
  list.each { |item|
    if item == key then
      return true
    end
  }
  return false
end
```

A contains? method – version 4

```
def contains?(list, key)
  list.each { |x| return true if x == key }
  return false
end
```

Important note: You can use this method on keys of any type, as long as the key's type matches the type of the elements in the array.

Search

Design an algorithm that returns the index of the first occurrence of a key in a list if the key is present, or **nil** otherwise.

A search method

```
def search(list, key)
  index = 0
  while index < list.length do
    if list[index] == key then
      return index
    end
    index = index + 1
  end
  return nil
end
```

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

13

Sorry...

```
def search(list, key)
  for item in list do
    if item == key then
      return index
    end
  end
  return nil
end
```

Why can't we do this?

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

14

Ok, but...

```
def search(list, key)
  for item in list do
    if item == key then
      return list.index(key)
    end
  end
  return nil
end
```

← What's undesirable about this?