UNIT 4A
Iteration: Searching
Searching
Searching

Twitter – 1.6 billion queries per day !!!!

- **Google:** 34,000 searches per second (2 million per minute; 121 million per hour; 3 billion per day; 88 billion per month, figures rounded)
- **Yahoo:** 3,200 searches per second (194,000 per minute; 12 million per hour; 280 million per day; 8.4 billion per month, figures rounded)
- **Bing:** 927 searches per second (56,000 per minute; 3 million per hour; 80 million per day; 2.4 billion per month, figures rounded)

http://searchengineland.com
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.
Review
Nested Loops revisited..

- Nested for example
Ruby `times` method

- Syntax: `some_integer.times { statements }
- `Some_integer.upto(i)`
- `Some_integer.downto(i)`
Containers
Type of Containers

• A set is an _________ container

• A list is an _________ container

• An array is a _________

• The difference between arrays and lists?
Ruby Arrays

Empty Array  \( A = [] \)

Examples:

Create:  (from a range)  (as a list)

Access:  (first)  (last)  (any index)
Why Study Containers?

• Organizing data for processing

• Example
  – Music player
    • How to store music?
      – MyPlayList
    • How to design the program?
  • How to write the code?
Searching
What can we search?
Designing a simple search algorithm

• The problem: Given a list of items
  – Find if a specific item exists or not
  – Find if more than one of the specific items exists
  – Find the first item in the list
  – Find the last item in the list
Class Demo
From demo to program

• How to store data?

• How to write a search function?

• Write code
  – Using basic loops
  – Using ruby methods
Ruby Array methods

- Suppose A is an array
  - A.include?(arg)
  - A.index?(arg)
  - A.length
Ruby Style search

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

movies.include?("wall-e") =>
movies.include?("toy") =>
movies.index("cars") =>
movies.index("shrek") =>
movies.index("Up") =>
How strings are represented
A Little More about Strings

• You can use relational operators to compare strings: <, <=, >, >=, ==, !=

• Comparisons are done character by character using ASCII codes.
**Extended ASCII table**

| 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| r  | t  | l  | j  | p  | %  | &  | '  | (  | )  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100| 101| 102| 103| 104| 105| 106| 107| 108| 109| 110| 111| 112| 113| 114| 115| 116| 117| 118| 119| 120| 121| 122| 123| 124| 125| 126| 127| 128| 129| 130| 131| 132| 133| 134| 135| 136| 137| 138| 139| 140| 141| 142| 143| 144| 145| 146| 147| 148| 149| 150| 151| 152| 153| 154| 155| 156| 157| 158| 159| 160| 161| 162| 163| 164| 165| 166| 167| 168| 169| 170| 171| 172| 173| 174| 175| 176| 177| 178| 179| 180| 181| 182| 183| 184| 185| 186| 187| 188| 189| 190| 191| 192| 193| 194| 195| 196| 197| 198| 199| 200| 201| 202| 203| 204| 205| 206| 207| 208| 209| 210| 211| 212| 213| 214| 215| 216| 217| 218| 219| 220| 221| 222| 223| 224| 225| 226| 227| 228| 229| 230| 231| 232| 233| 234| 235| 236| 237| 238| 239| 240| 241| 242| 243| 244| 245| 246| 247| 248| 249| 250| 251| 252| 253| 254| 255|
Exercise on String Comparison

"Steelers" > "Jets" =>
"steelers" > "Jets" =>
"Steelers" > "jets" =>
"Steelers Nation" > "Steelers" =>
"Steelers Nation" > "Steelers" =>
A contains? method
Find Last

Design an algorithm that returns the index of the last occurrence of a key in a list if the key is present, or \texttt{nil} otherwise.