Basic Idea

• Convert key to an integer
• Use that integer to index into a list
• ‘abc’
  
  = 011000010110001001100011
  
  = 6,382,179

Good way to convert names to indices?
Names to Indices

• Advantage: Unique names mapped to unique indices
• Disadvantage: VERY space inefficient
• Consider a table containing MIT’s ~4,000 undergraduates
  – Assume longest name is 20 characters
  – Each character 8 bits, so 160 bits per name
  – How many entries will table have?
    • $2^{160}$
How Big is $2^{160}$?
# Hash Tables

<table>
<thead>
<tr>
<th>Key</th>
<th>Hash</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>4</td>
<td>101</td>
</tr>
<tr>
<td>Mark</td>
<td>8</td>
<td>351</td>
</tr>
<tr>
<td>Andrea</td>
<td>2</td>
<td>220</td>
</tr>
<tr>
<td>Mike</td>
<td>4</td>
<td>521</td>
</tr>
</tbody>
</table>

Key | Hash | Value |
---|------|-------|
John | 4 | 101 |
Mark | 8 | 351 |
Andrea | 2 | 220 |
Mike | 4 | 521 |

- John: [(‘Andrea’, 220)]
- Mark: [(‘John’, 101), (‘Mike’, 521)]
- Andrea: [(‘Mark’, 351)]
Four Major Topics

• Learning a language for expressing computations – Python
  – Almost all of it
• Learning about the process of writing and debugging a program – Be systematic
• Learning to estimate computational complexity
• Learning a few recipes – algorithms
Why Python?

• Relatively easy to learn and use
  – Simple syntax
  – Interpretive, which makes debugging easier
  – Don’t have to worry about managing memory

• Modern
  – Supports currently stylish mode of programming, object-oriented

• Increasingly popular
  – Used in an increasing number of subjects at MIT and elsewhere
  – Increasing use in industry
  – Large and ever growing set of libraries
Writing, Testing, and Debugging Programs

• Take it a step at time
  – Understand problem
  – Think about overall structure and algorithms independently of expression in programming language
  – Break into small parts
  – Identify useful abstractions (data and functional)
  – Code and unit test a part at a time
  – First functionality, then efficiency

• Be systematic
  – Ask yourself why program did what it did, not why it didn’t do what you wanted it to do.
Algorithms

- Big O notation
  - Orders of growth
  - Exponential, Polynomial, Linear, Log
- Kinds of Algorithms
  - Exhaustive enumeration, Guess and check, Successive approximation
  - Recursion, iteration
- Specific algorithms
  - E.g., Binary search, Merge sort
What Next

• Many of you have worked very hard
  – TA’s and I appreciate it
• Only you know your return on investment
  – Take a look at early problem sets
  – Think about what you’d be willing tackle now
• Remember that you can write programs to get answers
• There are other CS courses you are prepared to take
  – 6.0002, 6.01, 6.004, 6.034, 6.005, 6.006
• You could even major in Course VI
6.0002

- Time to digest and really learn how to use Python
- Statistical thinking
- Using computation to build useful models
  - Understanding the world
  - Turning data into useful information