Introduction

Definitions

Data Structure: collections of variables, possibly of different data types, connected in various ways

Algorithms: methods for solving problems that are suited for computer applications

Algorithm-centric View

“Data structures exist as the byproduct of algorithms”
- Algorithms that use time and space as efficiently as possible
- Programs can be made millions of times faster by a well-designed algorithm
Running Times for Search

<table>
<thead>
<tr>
<th>Population</th>
<th>Linear</th>
<th>Logarithmic</th>
</tr>
</thead>
<tbody>
<tr>
<td>EECS 281</td>
<td>12 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>U M</td>
<td>3 sec</td>
<td>1.4 ms</td>
</tr>
<tr>
<td>County</td>
<td>35 sec</td>
<td>18 ms</td>
</tr>
<tr>
<td>Michigan</td>
<td>17 min</td>
<td>23 ms</td>
</tr>
<tr>
<td>USA</td>
<td>8 hours</td>
<td>28 ms</td>
</tr>
<tr>
<td>World</td>
<td>7 days</td>
<td>32 ms</td>
</tr>
<tr>
<td>EECS 281: 120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USA: 276 million</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which Means

- There are many ways to solve an algorithmic problem
- Goal of 281 is to solve problems using most efficient method
  - where ‘efficient’ is defined in terms of time and space
- Key to 281 is ability to successfully analyze different methods/algorithms

Typical Approach to Algorithm Selection

- Define problem to be solved
- Manage (understand) its complexity
- Decompose into smaller subtasks
  - Often, choice of one such subtask is critical to overall efficiency
- Refine solution based on expected usage

*Note that effort to make algorithm efficient may not be worthwhile unless domain is large or algorithm is to be reused often*
Analysis ‘Lite’

Key to 281 is ability to successfully analyze different algorithms

- Let’s analyze an informal problem
  - knowing that a more rigorous analysis could potentially lead to better solutions
- Such analyses will be shown later in our course

Example: Dinner at My House

- After dinner, the table must be cleared
  - there are $m$ people sitting at the table
  - there are $n$ unique items in each place setting
    - plate, glass, fork, knife,
  - Analyze the following three methods of clearing the table in terms of $m$ and $n$

Method 1: Dinner at My House

- Who: an adult
- How: takes half of all of the original items on the table on each trip from the table to the sink
Method 2: Dinner at My House

- Who: the oldest child
- How: moves exactly half of one place setting on each trip from the table to the sink

Method 3: Dinner at My House

- Who: the youngest daughter and oldest son
- How:
  - daughter takes an individual trip from the table to the sink for each item on the table in front of each person at the table
  - However, she is not allowed to move any knives, because she is too young
  - brother moves all knives from the table to the sink in one trip

Running Times for Search

<table>
<thead>
<tr>
<th>Population</th>
<th>Linear</th>
<th>Logarithmic</th>
</tr>
</thead>
<tbody>
<tr>
<td>EECS 281</td>
<td>12 ms</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>U M</td>
<td>3 seconds</td>
<td>1.4 ms</td>
</tr>
<tr>
<td>County</td>
<td>35 seconds</td>
<td>18 ms</td>
</tr>
<tr>
<td>Michigan</td>
<td>17 minutes</td>
<td>23 ms</td>
</tr>
<tr>
<td>USA</td>
<td>8 hours</td>
<td>28 ms</td>
</tr>
<tr>
<td>World</td>
<td>7 days</td>
<td>32 ms</td>
</tr>
</tbody>
</table>

EECS 281: 120 USA: 276 million
Search Revisited

- Cost of algorithm is defined by cost of operations in algorithm
- In search, the following are important:
  - search for name
  - insert new name
  - delete old name

Search Revisited

Logarithmic (aka binary) Search

In search, the following are important:
- search for name:
  - about \( \log n \) operations
- insert new name:
  - to find correct location: about \( \log n \) operations
  - to insert: about \( n/2 \) operations (moves of names)
- delete old name
  - to find correct location: about \( \log n \) operations
  - to delete: about \( n/2 \) operations (moves of names)

Search Revisited

Linear Search

In search, the following are important:
- search for name:
  - about \( n/2 \) operations
- insert new name:
  - 1 operation (always put at beginning/end)
- delete old name
  - to find correct location: about \( n/2 \) operations
  - to delete: about \( n/2 \) operations (moves of names) (about 1 operation if clever!!)
Summary

- Define problem to be solved
- Manage (understand) its complexity
- Decompose into smaller subtasks
  - Often, choice of one such subtask is critical to overall efficiency
- Refine solution based on expected usage

Note that effort to make algorithm efficient may not be worthwhile unless domain is large or algorithm is to be reused often