Searching, Sorting &
Time Complexity

CS 102
Algorithms and Programming II

suggested reading:
Chapter 6 & 14
Big Java: Late Objects
Linear Search

• A linear search begins at one end of a list and examines each element in turn

• Eventually, either the item is found or the end of the list is encountered
Linear Search

- See LinearSearch.java
Binary Search Algorithm

- A binary search first examines the middle element of the list -- if it matches the target, the search is over.
- If it doesn't, only one half of the remaining elements need be searched.
- Since they are sorted, the target can only be in one half of the other.
Binary Search

• The process continues by comparing the middle element of the remaining *viable candidates*

• Each comparison eliminates approximately half of the remaining data

• Eventually, the target is found or the data is exhausted
Binary Search

• Given a key and sorted array $a[]$, find index $i$ such that $a[i] = key$, or report that no such index exists.
Binary Search for 33
Binary Search for 33

```
6  13  14  25  33  43  51  53  64  72  84  93  95  96  97
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
</table>
<pre><code>|    |    |    |    |    |    |   |   |   |    |    |    |    |    |
</code></pre>
    ↑    ↑    ↑
lo     mid    hi
```
Binary Search for 33

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>13</td>
<td>14</td>
<td>25</td>
<td>33</td>
<td>43</td>
<td>51</td>
<td>53</td>
<td>64</td>
<td>72</td>
<td>84</td>
<td>93</td>
<td>95</td>
<td>96</td>
<td>97</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

↑  
lo  

↑  
hi
Binary Search for 33
Binary Search for 33
Binary Search for 33

lo  mid  hi
Binary Search for 33

\[ \text{lo} \quad \uparrow \quad \text{hi} \]
Binary Search for 33

lo
hi
Exercise

• See the binary search method in Arrays class
Time complexity

• A measure of the amount of time required to execute an algorithm.

• Time complexity expresses the relationship between the size of the problem and the run time for the algorithm.

• Time complexity analysis is independent of the programming language chosen to implement the algorithm and the speed of the computer on which the algorithm is executed.
Problem Size

• In many applications, it is easy to come up with a numeric value that specifies the problem size, which is generally denoted by the letter $N$.

• For most array applications, the problem size is simply the size of the array.
Time complexity

• Analysis is usually based on:
  – Number of arithmetic operations performed
  – Number of comparisons made
  – Number of times through a critical loop
  – Number of array elements accessed
  – … etc
Efficiency of Linear Search

- In the worst case—which occurs when the value you’re searching for comes at the end of the array or does not appear at all—linear search requires $N$ steps. On average, it takes approximately half that.
Efficiency of Binary Search

• On each step in the process, the binary search algorithm rules out half of the remaining possibilities.

• In the worst case, the number of steps required is equal to the number of times you can divide the original size of the array in half until there is only one element remaining.
  \[ 1 = \frac{N}{2} / \frac{2}{2} / \frac{2}{2} \ldots / 2 \]

  \( k \) times

• Basic algebra:
  \[ 1 = N / 2^k \]
  \[ 2^k = N \]
  \[ k = \log_2 N \]

Copyright © 2012 Pearson Education, Inc.
Sorting

• One of the most important computing applications

• The process of arranging a list of items in a particular order

• Original List: 9, 2, 4, 5, 8, 1, 3

• List Sorted in Ascending Order: 1, 2, 3, 4, 5, 8, 9

Copyright © 2012 Pearson Education, Inc.
Sorting Algorithms

• There are many sorting algorithms, which vary in efficiency

• We will examine one specific algorithm, which is quadratic in time – (note that this is not the best method available):
  – Selection Sort
Selection Sort

• The strategy of Selection Sort:
  – select a value and put it in its final place in the list
  – repeat for all other values

• In more detail (sorting in ascending order):
  – find the smallest value in the list
  – switch it with the value in the first position
  – find the next smallest value in the list
  – switch it with the value in the second position
  – repeat until all values are in their proper places
Selection Sort

Scan right starting with 3.
1 is the smallest. Exchange 1 and 3.

Scan right starting with 9.
2 is the smallest. Exchange 9 and 2.

Scan right starting with 6.
3 is the smallest. Exchange 6 and 3.

Scan right starting with 6.
6 is the smallest. Exchange 6 and 6.
Selection Sort
Examples

• See SelectionSort.java
• See SelectionSortString.java
Comparing Sorts

• The Selection Sort algorithm has an outer loop that scan all elements, and an inner loop that compare the value of the outer loop with almost all values in the list

• Approximately $N^2$ number of comparisons are made to sort a list of size $n$

• We therefore say that selection sort algorithm is of order $N^2$

• There are other sorting algorithms that are more efficient: $\text{order } N \log_2 N$
Advantages / Disadvantages of Selection Sort

• Advantage: it is easy to program.

• Disadvantage: it is very slow.

• Much work in computer science has been geared towards creating more efficient sorting algorithms.

• For example:
  – Merge Sort
  – Heapsort
  – Quick Sort
Comparison of sorting algorithms

Graph from Alan Jepson University of Toronto.