CMPSC 250
Analysis of algorithms

Dr. Aravind Mohan
Allegheny College

February 7, 2018
Lecture 07 - 1 Finishing up on Asymptotic Notation
Asymptotic Analysis of Running Time

- Use $O$-notation to express number of primitive operations executed as functions of input size.
- Comparing asymptotic running times
  - an algorithm that runs in $O(n)$ time is better than one that runs in $O(n^2)$
  - similarly $O(\log n)$ is better than $O(n)$
  - hierarchy of functions: $\log n < n < n^2 < n^3 < 2^n$
- **Caution!** Beware of very large constant factors. An algorithm running in time $1,000,000 \ n$ is still $O(n)$ but might be less efficient than one running in time $2n^2$, which is $O(n^2)$
Example of Asymptotic Analysis

Algorithm: prefixAverages1(X)
Input: An n-element array X of numbers
Output: An n-element array A of numbers such that A[i] is the average of elements X[0], X[1], ..., X[n]

for i <- 0 to n-1 do{
    a <- 0
    for j <- 0 to i do{
        a <- a + X[j]
    }
    A[i] = a/i+1
}
return array A

• Running time is $O(n^2)$
Can we do better?

Algorithm: prefixAverages2(X)
Input: An n-element array X of numbers
Output: An n-element array A of numbers such that A[i] is the average of elements X[0], X[1], ..., X[n]

\[
\begin{align*}
s & \leftarrow 0 \\
\text{for } i & \leftarrow 0 \text{ to } n-1 \text{ do} \\
& \quad s \leftarrow s + X[i] \\
& \quad A[i] \leftarrow s/i+1 \\
\end{align*}
\]

return array A

- Running time is $O(n)$
Some Terminology

- Logarithmic: $O(\log n)$
- Linear: $O(n)$
- Quadratic: $O(n^2)$
- Polynomial: $O(n^k), k \geq 1$
- Exponential: $O(a^n), a > 1$
## Comparison of Running Times

<table>
<thead>
<tr>
<th>Running Time</th>
<th>Maximum problem size (n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 second</td>
</tr>
<tr>
<td>400n</td>
<td>2500</td>
</tr>
<tr>
<td>2n log n</td>
<td>4096</td>
</tr>
<tr>
<td>2n^2</td>
<td>707</td>
</tr>
<tr>
<td>n^4</td>
<td>31</td>
</tr>
<tr>
<td>2^x</td>
<td>19</td>
</tr>
</tbody>
</table>
Asymptotic Notation

- The "big-Omega" or $\Omega$ - notation
  - asymptotic lower bound
  - $f(n) = \Omega(g(n))$ if there exists constant $c$ and $n_0$ such that $cg(n) \leq f(n)$ for $n \geq n_0$
  - Used to describe best case running times or lower bounds of algorithmic problems
    - E.g., lower-bound of searching in an unsorted array is $\Omega(n)$
The big-Omega notation

\[ f(n) = \Omega(g(n)) \]

\[ n_0 \]
Asymptotic Notation

- The ”big-Theta” or $\Theta$ - notation
  - asymptotic tight bound
  - $f(n) = \Theta(g(n))$ if there exists constant $c_1$, $c_2$, and $n_0$ such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for $n \geq n_0$
  - $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$
  - $O(f(n))$ is often misused instead of $\Theta(f(n))$
The big-Theta notation

\[ f(n) = \Theta(g(n)) \]
Lecture 07 - 2 Sorting
What is Sorting?

- Organizing a collection of data into either ascending or descending order
  - Internal sorting - main memory
  - External sorting - secondary storage
Some Applications of Sorting?

- Searching in databases: we can do binary search on sorted data
- Facebook friend list
- Amazon online ordering system
How to Measure the Performance of Sorting Algorithms?

- Asymptotic Analysis
- We should count the number of key comparisons and the number of moves.
Types of Sorting Algorithms?

- Exchange Sorting
  - Selection Sort
  - Bubble Sort
  - Quick Sort
- Tree Sorting
  - Heap Sort
- Insertion Sort
- Merge Sort
Sorting an Array of Integers

The picture shows an array of six integers that we want to sort from smallest to largest.
Selection Sort Technique

- Start by finding the smallest entry
Selection Sort Technique

- Start by finding the smallest entry
- Swap the smallest entry with the first entry
Selection Sort Technique

- Start by finding the smallest entry
- Swap the smallest entry with the first entry
Selection Sort Technique

- Part of the array is now sorted
Selection Sort Technique

- Find the smallest element in the unsorted side
Selection Sort Technique

- Find the smallest element in the unsorted side
- Swap with the front of the unsorted side
Selection Sort Technique

- We have increased the size of the sorted side by one element
Selection Sort Technique

- The process continues...

![Selection Sort Diagram]
Selection Sort Technique

- The process continues...
Selection Sort Technique

- The process continues...
Selection Sort Technique

- The process keeps adding one more number to the sorted side.
- The sorted side has the smallest numbers, arranged from small to large
Selection Sort Technique

- We can stop when the unsorted side has just one number, since that number must be the largest number.
Selection Sort Technique

- The array is now sorted
- We repeatedly selected the smallest element and moved this element to the front of the unsorted side.
Selection Sort Algorithm

Selection Sort – Pseudocode

Output: $A[1..n]$ sorted in descending order

1. for $i \leftarrow 1$ to $n - 1$
2.  $\text{min} \leftarrow i$
3.  for $j \leftarrow i + 1$ to $n$  \{Find the $i$th smallest element.\}
4.  if $A[j] < A[\text{min}]$ then
5.     $\text{min} \leftarrow j$
6.  end for
7. if $\text{min} \neq i$ then interchange $A[i]$ and $A[\text{min}]$
8. end for
Selection Sort Running Time

- Best = Worst = Average Case Running Time: $O(n^2)$
Reading Assignments

- Chapter 01 - Pg 186 - 191
- Chapter 02 - Pg 248, 249
Questions