CS 171: Introduction to Computer Science II

Algorithm Analysis + Simple Sorting

Li Xiong
Today

• Algorithm Analysis (cont)
• Simple sorting algorithms
Tilde Notation

• Tilde notation: ignore insignificant terms
• Definition: we write $f(n) \sim g(n)$ if $f(n)/g(n)$ approaches 1 as $n$ grows

• $2n + 10 \sim 2n$
• $3n^3 + 20n^2 + 5 \sim 3n^3$
Big-Oh Notation

• Given functions $f(n)$ and $g(n)$, we say that $f(n)$ is $O(g(n))$ if there are positive constants $c$ and $n_0$ such that $f(n) \leq cg(n)$ for $n \geq n_0$

• Example: $2n + 10$ is $O(n)$
  – pick $c = 3$ and $n_0 = 10$
Important Functions in Big-Oh Analysis

- Constant: \( 1 \)
- Logarithmic: \( \log n \)
- Linear: \( n \)
- N-Log-N: \( n \log n \)
- Quadratic: \( n^2 \)
- Cubic: \( n^3 \)
- Polynomial: \( n^d \)
- Exponential: \( 2^n \)
- Factorial: \( n! \)
log-log plot

Typical orders of growth

- exponential
- cubic
- quadratic
- linearithmic
- linear
- logarithmic
- constant
Practical method for Big-Oh Analysis

- Write down cost function $f(n)$
  1. Look for highest-order term (tilde notation)
  2. Drop constant factors

- Examples
  - $3n^3 + 20n^2 + 5$
  - $n \log n + 10$
## Common notations for algorithm analysis

<table>
<thead>
<tr>
<th>notation</th>
<th>provides</th>
<th>example</th>
<th>shorthand for</th>
<th>used to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tilde</td>
<td>leading term</td>
<td>(\sim 10N^2)</td>
<td>(10N^2)</td>
<td>provide approximate model</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(10N^2 + 22N\log N)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(10N^2 + 2N + 37)</td>
<td></td>
</tr>
<tr>
<td>Big Theta</td>
<td>asymptotic growth rate</td>
<td>(\Theta(N^2))</td>
<td>(\frac{1}{2}N^2)</td>
<td>classify algorithms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(10N^2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(5N^2 + 22N\log N + 3N)</td>
<td></td>
</tr>
<tr>
<td>Big Oh</td>
<td>(\Theta(N^2)) and smaller</td>
<td>(O(N^2))</td>
<td>(10N^2)</td>
<td>develop upper bounds</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(100N)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(22N\log N + 3N)</td>
<td></td>
</tr>
<tr>
<td>Big Omega</td>
<td>(\Theta(N^2)) and larger</td>
<td>(\Omega(N^2))</td>
<td>(\frac{1}{2}N^2)</td>
<td>develop lower bounds</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(N^5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(N^3 + 22N\log N + 3N)</td>
<td></td>
</tr>
</tbody>
</table>
Useful Approximations

• Harmonic sum
  
  \[ 1 + 1/2 + 1/3 + \ldots + 1/N \approx \ln N \]

• Triangular sum
  
  \[ 1 + 2 + 3 + \ldots + N = N(N+1)/2 \approx N^2/2 \]

• Geometric sum
  
  \[ 1 + 2 + 4 + \ldots + N = 2N - 1 \approx 2N \text{ when } N = 2^n \]

• Stirling’s approximation
  
  \[ \lg N! = \lg 1 + \lg 2 + \lg 3 + \ldots + \lg N \approx N \lg N \]
### Common order-of-growth classifications

<table>
<thead>
<tr>
<th>growth rate</th>
<th>name</th>
<th>typical code framework</th>
<th>description</th>
<th>example</th>
<th>$T(2N) / T(N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>constant</td>
<td>$a = b + c$;</td>
<td>statement</td>
<td>add two numbers</td>
<td>$1$</td>
</tr>
</tbody>
</table>
| $O(\log N)$| logarithmic   | while $(N > 1)$
{ N = N / 2; ... } | divide in half    | binary search    | $\sim 1$       |
| $O(N)$      | linear        | for (int $i = 0$; $i < N$; $i++$)
{ ... } | loop              | find the maximum | $2$           |
| $O(N \log N)$| linearithmic | [see mergesort lecture] | divide and conquer| mergesort       | $\sim 2$       |
| $O(N^2)$    | quadratic     | for (int $i = 0$; $i < N$; $i++$)
  for (int $j = 0$; $j < N$; $j++$)
  { ... } | double loop       | check all pairs  | $4$           |
| $O(N^3)$    | cubic         | for (int $i = 0$; $i < N$; $i++$)
  for (int $j = 0$; $j < N$; $j++$)
  for (int $k = 0$; $k < N$; $k++$)
  { ... } | triple loop       | check all triples| $8$           |
| $O(2^N)$    | exponential   | [see combinatorial search lecture] | exhaustive search | check all subsets | $T(N)$       |
## Practical implications of Order-or-growth

<table>
<thead>
<tr>
<th>growth rate</th>
<th>problem size solvable in minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1970s</td>
</tr>
<tr>
<td>1</td>
<td>any</td>
</tr>
<tr>
<td>log N</td>
<td>any</td>
</tr>
<tr>
<td>N</td>
<td>millions</td>
</tr>
<tr>
<td>N log N</td>
<td>hundreds of thousands</td>
</tr>
<tr>
<td>N^2</td>
<td>hundreds</td>
</tr>
<tr>
<td>N^3</td>
<td>hundred</td>
</tr>
<tr>
<td>2^N</td>
<td>20</td>
</tr>
</tbody>
</table>
for (int i = 0; i < n; i++) {
    for (int j = i; j < n; j++) {
        sum += i*j;
    }
}

Example 4

```c
for (int i = 0; i < n; i++) {
    for (int j = i; j < n; j++) {
        sum += i*j;
    }
}
```

$$n + (n-1) + (n-2) + \ldots + 1 + 0 = \frac{n(n+1)}{2}$$

is

$$0.5 \ (n^2 + n) \Rightarrow O(n^2)$$
Example 5

double product = 1.0;
for (int i = 1; i <= n; i *= 2) {
    product *= i;
}


Example 5: Solution

double product = 1.0;
for (int i = 1; i <= n; i *= 2) {
    product *= i;
}

• This has a logarithmic cost:

  \[ O(\log_2 n) \]

or \[ O(\log n) \] as the change of base is merely a matter of a constant factor.
double product = 1.0;
for (int i = 1; i <= n; i *= 2) {
    for (int j = 1; j <= i; j++) {
        product *= j;
    }
}
Example 6

• What about this:

```java
double product = 1.0;
for (int i = 1; i <= n; i *= 2) {
    for (int j = 1; j <= i; j++) {
        product *= j;
    }
}
```

\[1+2+4+8+...+n\] is \(O(n)\)
Review Question

• What is the Order of growth (big-oh) of the following code?

```cpp
for (int i=1; i<=N; ++i) {
    for (int j=1; j<=N; j*=2) {
        count++;
        count++;
    }
}
```
Search in Ordered vs. Unordered Array

- What’s the big O function for linear search?
- Binary search?
Search in Ordered vs. Unordered Array

• What’s the big O function for linear search? O(N)
• Binary search? O(lgN)
• Binary search has much better running time, particularly for large-scale problems
Today

• Algorithm Analysis (cont)
• Simple sorting algorithms
### Sorting problem

**Ex.** Student records in a university.

<table>
<thead>
<tr>
<th>Item</th>
<th>Key</th>
<th>Code</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen</td>
<td>3</td>
<td>991-878-4944</td>
<td>308 Blair</td>
</tr>
<tr>
<td>Rohde</td>
<td>2</td>
<td>232-343-5555</td>
<td>343 Forbes</td>
</tr>
<tr>
<td>Gazsi</td>
<td>4</td>
<td>766-093-9873</td>
<td>101 Brown</td>
</tr>
<tr>
<td>Furia</td>
<td>1</td>
<td>766-093-9873</td>
<td>101 Brown</td>
</tr>
<tr>
<td>Kanaga</td>
<td>3</td>
<td>898-122-9643</td>
<td>22 Brown</td>
</tr>
<tr>
<td>Andrews</td>
<td>3</td>
<td>664-480-0023</td>
<td>097 Little</td>
</tr>
<tr>
<td>Battle</td>
<td>4</td>
<td>874-088-1212</td>
<td>121 Whitman</td>
</tr>
</tbody>
</table>

**Sort.** Rearrange array of $N$ items into ascending order.

<table>
<thead>
<tr>
<th>Item</th>
<th>Key</th>
<th>Code</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrews</td>
<td>3</td>
<td>664-480-0023</td>
<td>097 Little</td>
</tr>
<tr>
<td>Battle</td>
<td>4</td>
<td>874-088-1212</td>
<td>121 Whitman</td>
</tr>
<tr>
<td>Chen</td>
<td>3</td>
<td>991-878-4944</td>
<td>308 Blair</td>
</tr>
<tr>
<td>Furia</td>
<td>1</td>
<td>766-093-9873</td>
<td>101 Brown</td>
</tr>
<tr>
<td>Gazsi</td>
<td>4</td>
<td>766-093-9873</td>
<td>101 Brown</td>
</tr>
<tr>
<td>Kanaga</td>
<td>3</td>
<td>898-122-9643</td>
<td>22 Brown</td>
</tr>
<tr>
<td>Rohde</td>
<td>2</td>
<td>232-343-5555</td>
<td>343 Forbes</td>
</tr>
</tbody>
</table>
Sorting Problem

• How do you sort a hand of poker cards?
Simple sort

• Bubble sort
• Selection sort
• Insertion sort
Two useful sorting abstractions

**Helper functions.** Refer to data through compares and exchanges.

**Less.** Is item v less than w?

```java
private static boolean less(Comparable v, Comparable w) {
    return v.compareTo(w) < 0;
}
```

**Exchange.** Swap item in array a[] at index i with the one at index j.

```java
private static void exch(Comparable[] a, int i, int j) {
    Comparable swap = a[i];
    a[i] = a[j];
    a[j] = swap;
}
```
Bubble Sort

• Intuition:
  – Find the biggest number.
  – Find the second biggest number.
  – Find the third biggest number.
  – ...

• This gives you an ordering of the numbers.

• Bubble sort achieves this by repeatedly swapping two adjacent numbers.
Bubble Sort

• After one pass, we find the biggest number.

• It’s like the biggest ‘bubble’ floats to the top of the surface, hence the name ‘bubble sort’.
Bubble Sort

- In the second pass, we repeat the same process, but now we only have N-1 numbers to work on.
- The third pass is the same, with only N-2 numbers.
- ...
- Repeat until all players are in order.
Analysis of Bubble Sort

• Number of comparisons?

• Number of swaps?
Analysis of Bubble Sort

• Number of comparisons?

\[
\frac{N(N-1)}{2} = O(N^2)
\]

• Number of swaps?

**best case:** \( O(1) \)

**worst cast:** \[
\frac{N(N-1)}{2} = O(N^2)
\]

**average:** \[
\frac{N(N-1)}{4} = O(N^2)
\]
Selection Sort

1. Keep track of the index of the smallest number in each round.
2. Swap the smallest number towards the beginning of the array.
3. Repeat the above two steps.
Selection Sort

**Algorithm.** ↑ scans from left to right.

**Invariants.**
- Entries the left of ↑ (including ↑) fixed and in ascending order.
- No entry to right of ↑ is smaller than any entry to the left of ↑.
Selection Sort

- Move the pointer to the right.
  
  ```
  i++;
  ```

- Identify index of minimum entry on right.
  
  ```
  int min = i;
  for (int j = i+1; j < N; j++)
    if (less(a[j], a[min]))
      min = j;
  ```

- Exchange into position.
  
  ```
  exch(a, i, min);
  ```
Selection Sort Implementation

```java
public class Selection {
    public static void sort(Comparable[] a) {
        int N = a.length;
        for (int i = 0; i < N; i++) {
            int min = i;
            for (int j = i+1; j < N; j++)
                if (less(a[j], a[min]))
                    min = j;
            exch(a, i, min);
        }
    }

    private static boolean less(Comparable v, Comparable w) {
        /* as before */
    }

    private static void exch(Comparable[] a, int i, int j) {
        /* as before */
    }
}
```
Selection Sort

• Online demo

• Gypsy dance demo
  – http://www.youtube.com/watch?v=Ns4TPTC8whw
Selection Sort

- Number of comparisons?

- Number of swaps?
Selection Sort

• Number of comparisons?
  \( O(N^2) \)

• Number of swaps?
  \( O(N) \)