Big-O notation

Lecture 10

Running time of selection sort

- We showed that running selection sort on an array of \( n \) elements takes in the worst case
  \[ T(n) = 1 + 15n + 5n^2 \]

- When \( n \) is large, \( T(n) \approx 5n^2 \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( T(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>661</td>
</tr>
<tr>
<td>20</td>
<td>2301</td>
</tr>
<tr>
<td>30</td>
<td>4951</td>
</tr>
<tr>
<td>40</td>
<td>8601</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1000</td>
<td>5015001</td>
</tr>
<tr>
<td>2000</td>
<td>20030001</td>
</tr>
</tbody>
</table>

Doubling \( n \) quadruples \( T(n) \)

N.B. That is true for any coefficient of \( n^2 \) (not just 5)

Big-O notation

- Goals:
  - Simplify the discussion of algorithm running times
  - Describe how the running time of an algorithm increases as a function of \( n \) (the size of the problem), when \( n \) is LARGE
  - Get rid of terms that become insignificant when \( n \) is large
- We will say things like:
  - The worst-case running time of selectionSort on an array of \( n \) elements is \( O(n^2) \)
  - The worst-case running time of mergeSort on an array of \( n \) elements is \( O(n \log(n)) \)

Big-O definition

- Let \( f(n) \) and \( g(n) \) be two non-negative functions defined on the natural numbers \( N \).
- We say that \( f(n) \) is \( O(g(n)) \) if and only if:
  - There exists an integer \( n_0 \) and a real number \( c \) such that: for all \( n \geq n_0 \), \( f(n) \leq c \cdot g(n) \)

More mathematically, we would write:

\[ \exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R} : \forall n \geq n_0, f(n) \leq c \cdot g(n) \]

- N.B. The constant \( c \) must not depend on \( n \)

Intuition and visualization

- “\( f(n) \) is \( O(g(n)) \)” iff there exists a point \( n_0 \) beyond which \( f(n) \) is less than some fixed constant times \( g(n) \)

\[ f(n) \leq c \cdot g(n) \quad (\text{for } c = 1) \]
Proving big-O relations

- To prove that \( f(n) \) is \( O(g(n)) \), we must find \( n_0 \) and \( c \) such that \( f(n) \leq c \cdot g(n) \).
- Example: Prove that \( 5 + 3 \, n^2 \) is \( O(1 + n^2) \).
  
  We need to pick \( c \) greater than 3. Let’s pick \( c = 5 \).
  
  If we choose \( n_0 = 1 \), we get that if \( n \geq n_0 \), then
  
  \[
  5 + 3 \, n^2 \leq 5 + 5 \, n^2 \quad \text{(since \( n \geq n_0 \))}
  
  = 5 \cdot (1 + n^2)
  
  = c \cdot (1 + n^2)
  
\]

Examples

- Prove that \( 2n + 3 \) is \( O(n) \)

Examples

- Prove that \( f(n) = 10^{100} \) is \( O(1) \)

Examples

- Prove that \( n \, (\sin(n) + 1) \) is \( O(n) \)

Proving that \( f(n) \) is not \( O(g(n)) \)

- To prove that \( f(n) \) is not \( O(g(n)) \), one must show that for any \( n_0 \) and \( c \), there exists an \( n \geq n_0 \) such that \( f(n) > c \cdot g(n) \).

- Procedure: Assume \( n_0 \) and \( c \) are given, and find a value of \( n \) such that \( f(n) > c \cdot g(n) \). The value of \( n \) will usually depend on \( n_0 \) and \( c \).
Examples
• Prove that \( n \, (\sin(n) + 1) \) is \( O(n) \)

Examples
• Prove that \( 2^n \) is not \( O(n^3) \)