## Overview of Asymptotic Notation

<table>
<thead>
<tr>
<th>Asymptotic Notation</th>
<th>( \lim_{n \to \infty} \frac{f(n)}{g(n)} )</th>
<th>Alternate Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(n) = O(g(n)) ) (&quot;big-Oh&quot;)</td>
<td>constant or 0</td>
<td>( f(n) \leq \lim g(n) )</td>
</tr>
<tr>
<td>( f(n) = \Omega(g(n)) ) (&quot;big-Omega&quot;)</td>
<td>constant or ( \infty )</td>
<td>( f(n) \geq \lim g(n) )</td>
</tr>
<tr>
<td>( f(n) = \Theta(g(n)) ) (&quot;big-Theta&quot;)</td>
<td>constant</td>
<td>( f(n) = \lim g(n) )</td>
</tr>
<tr>
<td>( f(n) = o(g(n)) ) (&quot;little-oh&quot;)</td>
<td>0</td>
<td>( f(n) &lt; \lim g(n) )</td>
</tr>
<tr>
<td>( f(n) = \omega(g(n)) ) (&quot;little-omega&quot;)</td>
<td>( \infty )</td>
<td>( f(n) &gt; \lim g(n) )</td>
</tr>
</tbody>
</table>
Analyzing Divide-and-conquer Algs

Remainder

\[ T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n) \]

- \# of subproblems that we recursively solved
- Size of each subproblem (really, \( \left\lfloor \frac{n}{b} \right\rfloor \) or \( \frac{n}{b} \))
- Total time for divide and combine steps together
- \# statements (time) when input is size n (worst-case)
- Base: \( T(c) = \Theta(1) \) problem size where you no longer recurse
Master Method

Solve asymptotically any recurrence of form

\[ T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k (\log n)^c) \]

\[ T(c) = \Theta(1) \text{ for some constant } c \]

Constants \( a \geq 1, \ b > 1, \ l \geq 0, \ k \geq 0 \)

Compare \( l \) and \( \log_b a \)

\( n^c \quad n^{\log_b a} \text{ is \# of times termination condition is reached} \)
Case 1: $l < \log_b a$

$T(n) = \Theta(n^{\log_b a})$

Master Method

$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^l \log(n)^k)$

Case 2: $l = \log_b a$

$T(n) = \Theta(f(n) \cdot \log n) = \Theta\left(n^{l} (\log n)^{k+1}\right)$

Case 3: $l > \log_b a$

$T(n) = \Theta(f(n)) = \Theta\left(n^{l} (\log n)^{k}\right)$
min \((A, P, r)\)  
if \(P = r\) return \(A[p]\)  
\(q = (p + r) / 2\)  
\(m_1 = \min(A, P, q)\)  
\(m_2 = \min(A, q + 1, r)\)  
return \(\min(m_1, m_2)\)

\(T(n) = 2T(n/2) + \Theta(1)\)  
\(n = A.length + 1\)
\[ T(n) = 2 T\left(\frac{n}{2}\right) + \Theta(1) \]

\[ a = b = 2 \quad \log_b a = 1 \]

\[ l = 0, \quad k = 0 \quad n^0 = 1 \quad (\log n)^0 = 1 \]

\[ \log_b a > l \]

\[ T(n) = \Theta(n) \]

Why is this not the best algorithm choice?
\[ T(n) = 3T\left(\frac{n}{3}\right) + \Theta(n) > \log n \]

\[ a = b = 3 \quad \log_b a = 1 \]

\[ \lambda = 1 \quad k = 0 \]

\[ \lambda = \log_b a \]

\[ T(n) = \Theta(n \log \log n) \]
Return to maximum contiguous subsequence problem

\( \leq 1 \)

Split the array in half \( \Theta(1) \)
  remember left & right index of the portion being considered

\( 2T(\frac{n}{2}) \) Recursively solve 2 halves

\( \Theta(n) \) \(/\)
  compute running sums from middle to both left & right, remember max & then add two maxs to see if better than result from both recursive calls
\( \Theta(n) \) think of as growing as \( c \cdot n \)

\( c \) some constant

\[ \Theta\left(\frac{n}{2}\right) = c \cdot \frac{n}{2} = \frac{c}{2} \cdot n \]

why no meaning?
Saw

\[ T(n) = 2T(n/2) + C \cdot n \]

\[ = C \cdot n \log_2 n \]

with the right termination \( T(1) = 1 \)
\( n/2 \text{ elements} \)

- Running +
- Remember max

\( \frac{n}{2} \text{ (looks at each element)} \)

\( \frac{n}{2} \text{ addition if bigger set variable} \)

\( C \cdot \frac{n}{2} = \Theta(n) \text{ (Constant)} \)
$T(n) = 2T(n/2) + \Theta(n)$

$= \Theta(n \log n)$

Better than brute-force $\Theta(n^2)$

Not as good as best alg $\Theta(n)$
Nested Loops (much more on pre-recorded lecture)

\[ \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=j+1}^{n} c \]

\[ \Theta(n^2) \]

\[ \Theta(n) \]
\[
\sum_{j=0}^{n-1} \text{values of } j,
\]

\[
\begin{array}{ccc}
0 & 1, \ldots, n-1 & n-1 \\
1 & 2, \ldots, n-1 & n-2 \\
2 & & n-3 \\
& & \\
\vdots & & \\
\vdots & & \\
n-1 & & 1 \\
\end{array}
\]

\[
\sum_{i=1}^{n-1} (n^2) = \frac{n(n-1)}{2}
\]
Binary Search

input sorted array + value

5 8)9 12 17 20 21 22 27

x < 17

recurse here

x = 10
binary search \((A, p, r, x)\)

\[ q = (p + r) / 2 \]

- If \(x < A[q]\)
  - binary search \((A, p, q-1, x)\)
- Else
  - binary search \((A, q, r, x)\)

**Termination**
- \(\Theta(1)\)

**Split**
- \(\Theta(1)\)

**No combine**
\[ T(1) = \Theta(1) \]
\[ T(n) = T(n/2) + \Theta(1) + \log n \]

\[ a = 1, \quad b = 2 \quad \log_2 1 = 0 \]
\[ k = l = 0 \]
\[ l = \log_2 a \]
\[ 2^l = 1 \]

Case 2
\[ T(n) = \Theta(\log n) \quad \sqrt{\log_2 8} = 3 \]
Strassen's Matrix Multiplication

\[ n \left[ \begin{array}{c} a_1 \\ a_2 \\ \vdots \end{array} \right] \left[ \begin{array}{c} b_1 \\ b_2 \\ \vdots \end{array} \right] = \left[ \begin{array}{c} c_1 \\ c_2 \\ \vdots \end{array} \right] \]

\[ n^2 \text{ entries} \]

\[ \Theta(n) \text{ time each} \]

\[ \Theta(n^3) \]
\[
\begin{bmatrix}
\frac{n}{2} & a & b \\
\sqrt{c} & d \\
\end{bmatrix}
\times
\begin{bmatrix}
e & f \\
g & h \\
\end{bmatrix}
\]

Obvious divide and conquer

\[
a = 8, \quad b = 2, \quad \log_b a = 3 > l \quad \Rightarrow \quad T(n) = \Theta(n^3)
\]

\[
T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2)
\]

\[
l = 2, \quad k = 0
\]
Find a clever to instead
use only 7 matrix multiplys

$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$

$log_b a = log_2 7 > \ell = 2$

$T(n) = \Theta(n^{log_2 7})$