Winter 2016
COMP-250: Introduction to Computer Science
Lecture 12, February 18, 2016
Master Theorem
(CLRS 4.3)
Master Theorem

Used for many divide-and-conquer recurrences

\[ T(n) = aT\left(\frac{n}{b}\right) + f(n), \]

where \( a \geq 1, b > 1, \) and \( f(n) > 0. \)

\[ a = \text{(constant) number of sub-instances}, \]
\[ b = \text{(constant) size ratio of sub-instances}, \]
\[ f(n) = \text{time used for dividing and recombining}. \]

Based on the *master theorem* (Theorem 4.1).

Compare \( n^{\log_b a} \) vs. \( f(n) \):
Proof by recursion tree

\[ T(n) = aT(n/b) + f(n) \]

\[ T(n) = \sum a^k f(n/b^k) \]
Master Theorem

\[ T(n) = aT(n/b) + f(n) \]

**Case 1:** \( f(n) \) is \( O(n^L) \) for some constant \( L < \log_b a \).

**Solution:** \( T(n) \) is \( \Theta(n^{\log_b a}) \)

**Case 2:** \( f(n) \) is \( \Theta(n^{\log_b a \log^k n}) \), for some \( k \geq 0 \).

**Solution:** \( T(n) \) is \( \Theta(n^{\log_b a \log^{k+1} n}) \)

**Case 3:** \( f(n) \) is \( \Omega(n^L) \) for some constant \( L > \log_b a \) and \( f(n) \) satisfies the regularity condition \( af(n/b) \leq cf(n) \) for some \( c < 1 \) and all large \( n \).

**Solution:** \( T(n) \) is \( \Theta(f(n)) \)
Master Theorem

\[ T(n) = aT\left(\frac{n}{b}\right) + f(n), \]
where \( a \geq 1, b > 1, \) and \( f(n) > 0. \)

**Case 1:** \( f(n) \) is \( O(n^L) \) for some constant \( L < \log_b a. \)
\( (f(n) \) is polynomially smaller than \( n^{\log_b a}. \))

**Solution:** \( T(n) \) is \( \Theta(n^{\log_b a}) \)
\( (\text{Intuitively: cost is dominated by leaves.}) \)
Master Theorem

**Case 1:** \( f(n) \) is \( O(n^L) \) for some constant \( L < \log_b a \).

**Solution:** \( T(n) \) is \( \Theta(n^{\log_b a}) \)

\[
T(n) = 5T(n/2) + \Theta(n^2)
\]

Compare \( n^{\log_2 5} \) vs. \( n^2 \).

Since \( 2 < \log_2 5 \) use **Case 1**

**Solution:** \( T(n) \) is \( \Theta(n^{\log_2 5}) \)
Master Theorem

\[ T(n) = aT(n/b) + f(n) , \]
where \( a \geq 1, b > 1, \) and \( f(n) > 0. \)

**Simple Case 2:** \( f(n) \) is \( \Theta(n^{\log_b a}). \)

**Solution:** \( T(n) \) is \( \Theta(n^{\log_b a \log n}) \)
Master Theorem

\[ T(n) = aT\left(\frac{n}{b}\right) + f(n) , \]
where \( a \geq 1, b > 1, \) and \( f(n) > 0. \)

**Case 2:** \( f(n) \) is \( \Theta(n^{\log_b a} \log^k n) \), for some \( k \geq 0. \)

**Solution:** \( T(n) \) is \( \Theta(n^{\log_b a} \log^{k+1} n) \)

(Intuitively: cost is \( n^{\log_b a} \lg^k n \) at each level, and there are \( \Theta(\lg n) \) levels.)
Master Theorem

Case 2: $f(n)$ is $\Theta(n^{\log_{b}a \log^{k}n})$, for some $k \geq 0$.

Solution: $T(n)$ is $\Theta(n^{\log_{b}a \log^{k+1}n})$

$$T(n) = 27T(n/3) + \Theta(n^3 \log n)$$

Compare $n^{\log_{3}27}$ vs. $n^3$.

Since $3 = \log_{3}27$ use Case 2

Solution: $T(n)$ is $\Theta(n^3 \log^2 n)$
Master Theorem

\[ T(n) = aT(n/b) + f(n), \]
where \( a \geq 1, b > 1, \) and \( f(n) > 0. \)

**Case 3:** \( f(n) \) is \( \Omega(n^L) \) for some constant \( L > \log_b a \)
and \( f(n) \) satisfies the regularity condition \( af(n/b) \leq cf(n) \) for some \( c < 1 \) and all large \( n. \)
\( (f(n) \) is polynomially greater than \( n^{\log_b a}. \))

**Solution:** \( T(n) \) is \( \Theta(f(n)) \)
(Intuitively: cost is dominated by root.)
**Master Theorem**

**Case 3:** $f(n)$ is $\Omega(n^L)$ for some constant $L > \log_b a$
and $f(n)$ satisfies the regularity condition $af(n/b) \leq cf(n)$ for some $c < 1$ and all large $n$.

**Solution:** $T(n)$ is $\Theta(f(n))$

**What’s with the Case 3 regularity condition?**

- Generally not a problem.
- It always holds whenever $f(n) = n^k$ and $f(n)$ is $\Omega(n^{\log_b a + \epsilon})$ for constant $\epsilon > 0$. 
**Master Theorem**

**Case 3:** $f(n)$ is $\Omega(n^L)$ for some constant $L > \log_b a$ and $f(n)$ satisfies the regularity condition $af(n/b) \leq cf(n)$ for some $c<1$ and all large $n$.

**Solution:** $T(n)$ is $\Theta(f(n))$

\[
T(n) = 5T(n/2) + \Theta(n^3)
\]

Compare $n^{\log_2 5}$ vs. $n^3$.

Since $3 > \log_2 5$ use **Case 3**

\[
af(n/b) = 5(n/2)^3 = 5/8 \ n^3 \leq cn^3, \text{ for } c = 5/8
\]

**Solution:** $T(n)$ is $\Theta(n^3)$
Case 2: \( f(n) \) is \( \Theta(n^{\log_b a} \log^k n) \), for some \( k \geq 0 \).

Solution: \( T(n) \) is \( \Theta(n^{\log_b a} \log^{k+1} n) \)

\[
T(n) = 27T(n/3) + \Theta(n^3/\log n)
\]

Compare \( n^{\log_3 27} \) vs. \( n^3 \).

Since \( 3 = \log_3 27 \) use **Case 2**

**but** \( n^3/\log n \) is **not** \( \Theta(n^3 \log^k n) \) for \( k \geq 0 \)

Cannot use Master Method.
Divide-and-Conquer Paradigm
Divide-and-Conquer

Divide-and-conquer.
- Break up problem into several parts.
- Solve each part recursively.
- Combine solutions to sub-problems into overall solution.

Most common usage.
- Break up problem of size $n$ into two equal parts of size $n/2$.
- Solve two parts recursively.
- Combine two solutions into overall solution in linear time.

Consequence.
- Straightforward: $n^2$.
- Divide-and-conquer: $n \log n$.

Divide et impera.
Veni, vidi, vici.
- Julius Caesar
Divide-and-Conquer: Binary Search
Binary Search

Find a value $v$ in a sorted array of elements.

$$[a_0 \leq a_1 \leq, \ldots, \leq a_{\text{Size}-1}]$$

Size = number of elements.
Algorithm: `binarySearch(a, v, low, high)`

Input: array $a$, value $v$, lower and upper bound indices $low, high$ ($low = 0, high = n - 1$ initially)

Output: the index $i$ of element $v$ (if it is present), -1 (if $v$ is not present)

```
if low == high then
  if $a[low] == v$ then
    return low
  else
    return -1
  end if
else
  mid $\leftarrow (low + high)/2$
  if $v \leq a[mid]$ then
    return `binarySearch(a, v, low, mid)`
  else
    return `binarySearch(a, v, mid + 1, high)`
end if
end if
```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>0000</th>
<th></th>
<th>L</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0001</td>
<td></td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0010</td>
<td></td>
<td>H</td>
<td>L=H</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0011</td>
<td></td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>1001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>1010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>1011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>1100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>1101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>1110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>1111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>H</td>
<td>L=H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td></td>
<td></td>
<td>H</td>
<td>L=H</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td></td>
<td>H</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td></td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0101</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0110</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0111</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1001</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1010</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1011</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1100</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1101</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1110</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1111</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td></td>
<td></td>
<td>H</td>
<td></td>
</tr>
</tbody>
</table>
Binary Search

0 1 2 3
L L L
* 4 5
H H
H L=L

0 1 2 3
L L L
* 4 5
H H
H L=L

0 1 2 3
L L L
* 4 5
H H
H L=L

0 1 2 3
L L L
* 4 5
H H
H L=L

0 1 2 3
L L L
* 4 5
H H
H L=L
Binary Search

0   L
1
2
3   L   L
   * 4
   L   H
   H   L=H
5   H   H
Binary Search

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>*</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>
```

L = Low
H = High
L = H = Target
Binary Search

0  L
1
2
3  L  L
* 4  H  L=H
5  H  H
Recurrence Relation

**Def.** $T(n) = \text{number of comparisons to find } v \text{ among } n \text{ sorted elements.}$

**Binary Search recurrence.**

$$
\begin{align*}
T(n) &= \begin{cases} 
1 & \text{if } n = 1 \\
T(n/2) + 1 & \text{if } n > 1 
\end{cases}
\end{align*}
$$

**Solution.** $T(n)$ is $O(\log n)$ (Master Theorem Case 2).
Divide-and-Conquer Multiplication
Add. Given two n-digit integers $a$ and $b$, compute $a + b$.
- $\Theta(n)$ bit operations.

Multiply. Given two n-digit integers $a$ and $b$, compute $a \times b$.
- Grade School solution: $\Theta(n^2)$ bit operations.
D&C Multiplication

To multiply two n-digit integers:

- Multiply four \( \frac{n}{2} \)-digit integers.
- Add two \( \frac{n}{2} \)-digit integers, and shift to obtain result.

\[
x = 2^{n/2} \cdot x_1 + x_0
\]
\[
y = 2^{n/2} \cdot y_1 + y_0
\]
\[
xy = \left(2^{n/2} \cdot x_1 + x_0\right)\left(2^{n/2} \cdot y_1 + y_0\right) = 2^n \cdot x_1y_1 + 2^{n/2} \cdot (x_1y_0 + x_0y_1) + x_0y_0
\]

\[
T(n) = 4T(n/2) + \Theta(n) \quad \Rightarrow \quad T(n) \text{ is } \Theta(n^2)
\]

assumes \( n \) is a power of 2
Telescoping Proof

**Claim.**

\[
T(n) = \underbrace{4T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n)}_{\text{add, shift}} \quad \Rightarrow \quad T(n) \text{ is } \Theta(n^2)
\]

assumes \( n \) is a power of 2

**Pf.** For \( n > 1 \):

\[
T(n)/n = 4T(n/2)/n + C
= 2 T(n/2)/(n/2) + C
= 2 \left[ 2 T(n/4)/(n/4) + C \right] + C
= 4 T(n/4)/(n/4) + 2C + C
= 4 \left[ 2 T(n/8)/(n/8) + C \right] + 2C + C
= 8 T(n/8)/(n/8) + 4C + 2C + C
\]

...  

\[
= n T(1)/1 + n/2 C + n/4 C + ... + 4C + 2C + C
= C (n/2+n/4+...+2+1) = C(n-1).
\]
Karatsuba Multiplication

To multiply two n-digit integers:

- Add two \( \frac{n}{2} \) digit integers.
- Multiply three \( \frac{n}{2} \)-digit integers.
- Add, subtract, and shift \( \frac{n}{2} \)-digit integers to obtain result.

\[
\begin{align*}
x &= 2^{n/2} \cdot x_1 + x_0 \\
y &= 2^{n/2} \cdot y_1 + y_0 \\
xy &= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0 \\
&= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot \left( (x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0 \right) + x_0 y_0
\end{align*}
\]

Theorem. [Karatsuba-Ofman, 1962] Can multiply two n-digit integers in \( O(n^{1.585}) \) bit operations.

\[
T(n) \leq T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + T\left(1+\left\lfloor \frac{n}{2} \right\rfloor \right) + \Theta(n)
\]

\( \Rightarrow T(n) \text{ is } O(n^{\log_2 3}) \text{ is } O(n^{1.585}) \)
Karatsuba Recursion Tree

\[ T(n) = \begin{cases} 
0 & \text{if } n = 1 \\
3T(n/2) + n & \text{otherwise}
\end{cases} \]

\[ T(n) = \sum_{k=0}^{\log_2 n} n \left(\frac{3}{2}\right)^k = n \frac{\left(\frac{3}{2}\right)^{1+\log_2 n} - 1}{\frac{3}{2} - 1} = 3n^{\log_2 3} - 2 \]
Karatsuba Multiplication

Generalization: $O(n^{1+\varepsilon})$ for any $\varepsilon > 0$.

Best known: $n \log n \cdot 2^{O(\log^* n)}$

where $\log^*(x) = \begin{cases} 0 & \text{if } x \leq 1 \\ 1 + \log^*(\log x) & \text{if } x > 1 \end{cases}$

Conjecture: $\Omega(n \log n)$ but not proven yet.
Winter 2016
COMP-250: Introduction to Computer Science
Lecture 12, February 18, 2016