Warm Up!

What’s the theta bound for the runtime function for this piece of code?

```java
public void method1(int n) {
    if (n <= 100) {
        System.out.println(":3");
    } else {
        System.out.println(":D");
        for (int i = 0; i<16; i++) {
            method1(n / 4);
        }
    }
}
```

Master Theorem

\[
T(n) = \begin{cases} 
  d & \text{if } n \text{ is at most some constant} \\
  aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise}
\end{cases}
\]

Where \( f(n) \) is \( \Theta(n^c) \)

- If \( \log_b a < c \) then \( T(n) \in \Theta(n^c) \)
- If \( \log_b a = c \) then \( T(n) \in \Theta(n^c \log n) \)
- If \( \log_b a > c \) then \( T(n) \in \Theta(n^{\log_b a}) \)

Take 2 Minutes

1. [www.pollev.com/cse373activity](http://www.pollev.com/cse373activity) for participating in our active learning questions.

2. [https://www.pollev.com/cse373studentqs](https://www.pollev.com/cse373studentqs) to ask your own questions
Warm Up!

What’s the theta bound for the runtime function for this piece of code?

```java
public void method1(int n) {
    if (n <= 100) {
        System.out.println(":3");
    } else {
        System.out.println(":D");
        for (int i = 0; i<16; i++) {
            method1(n / 4);
        }
    }
}
```

Equation:

\[
T(n) = \begin{cases} 
\text{constant work} & \text{if } n \leq 100 \\
16T\left(\frac{n}{4}\right) + \text{constant work} & \text{otherwise}
\end{cases}
\]

Master Theorem:

\[
T(n) = \begin{cases} 
d & \text{if } n \text{ is at most some constant} \\
aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise}
\end{cases}
\]

Where \( f(n) \) is \( \Theta(n^c) \)

1. If \( \log_b a < c \) \( \Rightarrow \) \( T(n) \in \Theta(n^c) \)
2. If \( \log_b a = c \) \( \Rightarrow \) \( T(n) \in \Theta(n^c \log n) \)
3. If \( \log_b a > c \) \( \Rightarrow \) \( T(n) \in \Theta(n^{\log_b a}) \)

Take 2 Minutes

1. [www.pollev.com/cse373activity](http://www.pollev.com/cse373activity) for participating in our active learning questions.
2. [https://www.pollev.com/cse373studentqs](https://www.pollev.com/cse373studentqs) to ask your own questions.
Warm Up!

What’s the theta bound for the runtime function for this piece of code?

```java
public void method1(int n) {
    if (n <= 100) {
        System.out.println(":3");
    } else {
        System.out.println(":D");
        for (int i = 0; i<16; i++) {
            method1(n / 4);
        }
    }
}
```

$$T(n) = \begin{cases} 
\text{constant work} & \text{if } n \leq 100 \\
16T\left(\frac{n}{4}\right) + \text{constant work} & \text{otherwise}
\end{cases}$$

$$a = 16, b = 4, c = 0$$

---

**Master Theorem**

$$T(n) = \begin{cases} 
d & \text{if } n \text{ is at most some constant} \\
aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise}
\end{cases}$$

Where $$f(n)$$ is $$\Theta(n^c)$$

1. If $$\log_b a < c$$ then $$T(n) \in \Theta(n^c)$$
2. If $$\log_b a = c$$ then $$T(n) \in \Theta(n^c \log n)$$
3. If $$\log_b a > c$$ then $$T(n) \in \Theta(n^{\log_b a})$$

---

Take 2 Minutes

1. [www.pollev.com/cse373activity](http://www.pollev.com/cse373activity) for participating in our active learning questions.
2. [https://www.pollev.com/cse373studentqs](https://www.pollev.com/cse373studentqs) to ask your own questions.
Warm Up!

What’s the theta bound for the runtime function for this piece of code?

```java
public void method1(int n) {
    if (n <= 100) {
        System.out.println(":3");
    } else {
        System.out.println(":D");
        for (int i = 0; i<16; i++) {
            method1(n / 4);
        }
    }
}
```

\[
T(n) = \begin{cases} 
\text{constant work} & \text{if } n \leq 100 \\
16T\left(\frac{n}{4}\right) + \text{constant work} & \text{otherwise}
\end{cases}
\]

\[
a = 16, \ b = 4, \ c = 0
\]

Master Theorem

\[
T(n) = \begin{cases} 
d & \text{if } n \text{ is at most some constant} \\
 aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise}
\end{cases}
\]

Where \( f(n) \) is \( \Theta(n^c) \)

- If \( \log_b a < c \) then \( T(n) \in \Theta(n^c) \)
- If \( \log_b a = c \) then \( T(n) \in \Theta(n^c \log n) \)
- If \( \log_b a > c \) then \( T(n) \in \Theta(n^{\log_b a}) \)

---

Take 2 Minutes

1. [www.pollev.com/cse373activity](http://www.pollev.com/cse373activity) for participating in our active learning questions.

2. [https://www.pollev.com/cse373studentqs](https://www.pollev.com/cse373studentqs) to ask your own questions.
What’s the theta bound for the runtime function for this piece of code?

```java
public void method1(int n) {
    if (n <= 100) {
        System.out.println(":3");
    } else {
        System.out.println(":D");
        for (int i = 0; i<16; i++) {
            method1(n / 4);
        }
    }
}
```

`T(n) = \begin{cases} 
\text{constant work} & \text{if } n \leq 100 \\
16T\left(\frac{n}{4}\right) + \text{constant work} & \text{otherwise}
\end{cases}
```

```latex
\log_4 16 \ ? 0 \\
\log_4 16 > 0 \quad T(n) \in \Theta(n^{\log_b a})
```

\( a = 16, b = 4, c = 0 \)

---

**Master Theorem**

\[ T(n) = \begin{cases} 
d & \text{if } n \text{ is at most some constant} \\
aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise}
\end{cases} \]

Where \( f(n) \) is \( \Theta(n^c) \)

- If \( \log_b a < c \) then \( T(n) \in \Theta(n^c) \)
- If \( \log_b a = c \) then \( T(n) \in \Theta(n^c \log n) \)
- If \( \log_b a > c \) then \( T(n) \in \Theta(n^{\log_b a}) \)

---

**Take 2 Minutes**

1. [www.pollev.com/cse373activity](http://www.pollev.com/cse373activity) for participating in our active learning questions.

2. [https://www.pollev.com/cse373studentqs](https://www.pollev.com/cse373studentqs) to ask your own questions
What’s the theta bound for the runtime function for this piece of code?

```java
public void method1(int n) {
    if (n <= 100) {
        System.out.println(":3");
    } else {
        System.out.println(":D");
        for (int i = 0; i<16; i++) {
            method1(n / 4);
        }
    }
}
```

Master Theorem

\[
T(n) = \begin{cases} 
\text{constant work} & \text{if } n \leq 100 \\
16T\left(\frac{n}{4}\right) + \text{constant work} & \text{otherwise}
\end{cases}
\]

Where \( f(n) = \Theta(n^c) \)

- If \( \log_b a < c \) then \( T(n) \in \Theta(n^c) \)
- If \( \log_b a = c \) then \( T(n) \in \Theta(n^c \log n) \)
- If \( \log_b a > c \) then \( T(n) \in \Theta(n^{\log_b a}) \)

For the given code, we have:

- \( a = 16 \), \( b = 4 \), \( c = 0 \)
- \( \log_4 16 = 0 \)
- \( \log_4 16 > 0 \) \( \Rightarrow T(n) \in \Theta(n^{\log_4 16}) = \Theta(n^2) \)
Administrivia

project1 due Wednesday 11:59pm
exercise1 due Friday 11:59pm

Piazza
- try to use the search bar before you post / use descriptive summary titles when possible - thank you

Office Hours
- please come to them! Piazza is super busy and office hours are less busy. We promise we won’t bite 😊

anonymous feedback
- lectures + projects are so different?
- it’ll be better soon - we didn’t have time to cram in Deques (it’s probably the least relevant ADT) or make a new assignment this quarter. We’re aware, should be better for the future since the next few projects are all implementing new data structures and algorithms (HashMap, Binary Heaps, Graphs, ArrayDisjointSets, Kruskal’s and Dijkstra’s algorithms) we talk about directly in lecture.
- exercises + experiments should help connect lecture topics (while also not being _too_ much time)
- is there time to give an intro to the projects in lecture?
- maybe barely – we can try to see if it fits on a week-to-week
Questions
Code Analysis Process

- **Best case**
  - Model of best-case runtime $f(n)$
  - Asymptotic analysis
  - Best-case upper bound $O(n)$
  - Best-case lower bound $\Omega(n)$
  - Best-case tight fit $\Theta(n)$

- **Worst case**
  - Model of worst-case runtime $f(n)$
  - Asymptotic analysis
  - Worst-case upper bound $O(n)$
  - Worst-case lower bound $\Omega(n)$
  - Worst-case tight fit $\Theta(n)$

If code is recursive:
- **Recurrence**
- **Closed Form**
- **Master Theorem**
Recursive Patterns

Pattern #1: Halving the Input
   Binary Search $\Theta(\log n)$

Pattern #2: Constant size input and doing work
   Merge Sort $\Theta(n \log n)$

Pattern #3: Doubling the Input
   Calculating Fibonacci
Calculating Fibonacci

public int fib(int n) {
    if (n <= 1) {
        return 1;
    }
    return fib(n-1) + fib(n-1);
}

- Each call creates 2 more calls
- Each new call has a copy of the input, almost
- Almost doubling the input at each call

Pattern #3 – Doubling the Input
Calculating Fibonacci Recurrence to Big-Θ

```java
public int f(int n) {
    if (n <= 1) {
        return 1;
    }
    return f(n - 1) + f(n - 1);
}
```

Can we use master theorem?

```latex
\begin{align*}
T(n) &= \begin{cases} 
d & \text{when } n \leq 1 \\
2T(n - 1) + c & \text{otherwise}
\end{cases}
\end{align*}
```

Can we use master theorem?

Uh oh, our model doesn’t match that format...

Can we intuit a pattern?

\begin{align*}
T(1) &= d \\
T(2) &= 2T(2-1) + c = 2(d) + c \\
T(3) &= 2T(3-1) + c = 2(2(d) + c) + c = 4d + 3c \\
T(4) &= 2T(4-1) + c = 2(4d + 3c) + c = 8d + 7c \\
T(5) &= 2T(5-1) + c = 2(8d + 7c) + c = 16d + 25c
\end{align*}

Looks like something’s happening but it’s tough

Maybe geometry can help!

Take 1 min to respond to activity

Finish the recurrence, what is the model for the recursive case?
Calculating Fibonacci Recurrence to Big-$\Theta$

How many layers in the function call tree?

How many layers will it take to transform “$n$” to the base case of “1” by subtracting 1

For our example, $4 \rightarrow$ Height = $n$

How many function calls per layer?

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

How many function calls on layer $k$?

$$2^{k-1}$$

How many function calls TOTAL for a tree of $k$ layers?

$$1 + 2 + 3 + 4 + \ldots + 2^{k-1}$$

$T(n) = \begin{cases} 
\text{d when } n \leq 1 \\
2T(n-1) + c \text{ otherwise}
\end{cases}$
Calculating Fibonacci Recurrence to Big-Θ

Patterns found:

How many layers in the function call tree? \( n \)
How many function calls on layer \( k \)? \( 2^{k-1} \)
How many function calls TOTAL for a tree of \( k \) layers?

\[ 1 + 2 + 4 + 8 + \ldots + 2^{k-1} \]

Total runtime = (total function calls) x (runtime of each function call)

Total runtime = \((1 + 2 + 4 + 8 + \ldots + 2^{k-1})\) x (constant work)

\[ 1 + 2 + 4 + 8 + \ldots + 2^{k-1} = \sum_{i=1}^{k-1} 2^i = \frac{2^k - 1}{2 - 1} = 2^k - 1 \]

**Summation Identity**

Finite Geometric Series

\[ \sum_{i=1}^{k-1} x^i = \frac{x^k - 1}{x - 1} \]

\( T(n) = 2^n - 1 \in \Theta(2^n) \)
Recursive Patterns

Pattern #1: Halving the Input
 Binary Search $\Theta(\log n)$

Pattern #2: Constant size input and doing work
 Merge Sort $\Theta(n \log n)$

Pattern #3: Doubling the Input
 Calculating Fibonacci $\Theta(2^n)$
Questions
Tree Method
Code Analysis Process

If code is recursive:
- Recurrence
- Closed Form
- Master Theorem
- Tree Method

Best-case upper bound \(O(n)\)
Best-case lower bound \(\Omega(n)\)
Best-case tight fit \(\Theta(n)\)
Worst-case upper bound \(O(n)\)
Worst-case lower bound \(\Omega(n)\)
Worst-case tight fit \(\Theta(n)\)
Recurrence to Big Θ Techniques

A recurrence is a mathematical function that includes itself in its definition. This makes it very difficult to find the dominating term that will dictate the asymptotic growth.

Solving the recurrence or “finding the closed form” is the process of eliminating the recursive definition. So far, we’ve seen three methods to do so:

1. **Apply Master Theorem**
   - Pro: Plug and chug convenience
   - Con: only works for recurrences of a certain format

2. **Unrolling**
   - Pro: Least complicated setup
   - Con: requires intuitive pattern matching

3. **Tree Method**
   - Pro: Plug and chug
   - Con: Complex setup

\[
T(n) = \begin{cases} 
  d & \text{when } n \leq 1 \\
  2T(n-1) + c & \text{otherwise}
\end{cases}
\]

**Master Theorem**

\[
T(n) = \begin{cases} 
  d & \text{if } n \text{ is at most some constant} \\
  aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise}
\end{cases}
\]

- \(T(1) = d\)
- \(T(2) = 2T(2-1) + c = 2(d) + c\)
- \(T(3) = 2T(3-1) + c = 2(2(d) + c) + c = 4d + 3c\)
- \(T(4) = 2T(4-1) + c = 2(4d + 3c) + c = 8d + 7c\)
- \(T(5) = 2T(5-1) + c = 2(8d + 7c) + c = 16d + 25c\)
**Tree Method**

Draw out call stack, what is the input to each call? How much work is done by each call?

**How much work is done at each layer?**

64 for this example -> n work at each layer

Work is variable per layer, but across the entire layer work is constant - always n

**How many layers are in our function call tree?**

Hint: how many levels of recursive calls does it take *binary search* to get to the base case?

Height = \( \log_2 n \)

It takes \( \log_2 n \) divisions by 2 for n to be reduced to the base case 1

\( \log_2 64 = 6 \) -> 6 levels of this tree

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

... and so on...
Tree Method

\[ T(n) = \begin{cases} 
1 & \text{when } n \leq 1 \\
2T\left(\frac{n}{2}\right) + n & \text{otherwise}
\end{cases} \]

<table>
<thead>
<tr>
<th>Recursive level</th>
<th>How many nodes at each level?</th>
<th>How much work done by each node?</th>
<th>How much work across each level?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>( n )</td>
<td>( n )</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>( \frac{n}{2} )</td>
<td>( n )</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>( \frac{n}{4} )</td>
<td>( n )</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>( \frac{n}{8} )</td>
<td>( n )</td>
</tr>
<tr>
<td>log ( n )</td>
<td>( n )</td>
<td>1</td>
<td>( n )</td>
</tr>
</tbody>
</table>
Tree Method Practice

1. What is the size of the input on level $i$?
   \[ \frac{n}{2^i} \]

2. What is the work done by each node on the $i^{th}$ recursive level?
   \[ \frac{n}{2^i} \]

3. What is the number of nodes at level $i$?
   \[ 2^i \]

4. What is the total work done at the $i^{th}$ recursive level?
   
   \[ \text{numNodes} \times \text{workPerNode} = 2^i \left( \frac{n}{2^i} \right) = n \]

5. What value of $i$ does the last level occur?
   \[ \frac{n}{2^i} = 1 \rightarrow n = 2^i \rightarrow i = \log_2 n \]

6. What is the total work across the base case level?
   \[ \text{numNodes} \times \text{workPerNode} = 2^{\log_2 n} (1) = n \]

Combining it all together...

\[ T(n) = \begin{cases} 1 & \text{when } n \leq 1 \\ 2T \left( \frac{n}{2} \right) + n & \text{otherwise} \end{cases} \]

<table>
<thead>
<tr>
<th>Level ($i$)</th>
<th>Number of Nodes</th>
<th>Work per Node</th>
<th>Work per Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$n$</td>
<td>$n$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>$\frac{n}{2}$</td>
<td>$n$</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>$\frac{n}{4}$</td>
<td>$n$</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>$\frac{n}{8}$</td>
<td>$n$</td>
</tr>
<tr>
<td>$\log_2 n$</td>
<td>$n$</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

\[ T(n) = \sum_{i=0}^{\log_2 n-1} n + n = n \log_2 n + n = \Theta(n \log n) \]
Questions
Summations
Modeling Complex Loops

Write a mathematical model of the following code

```java
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.println("Hello!");
    }
}
```

Keep an eye on loop bounds!
Modeling Complex Loops

```java
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.print("Hello! ");
    }
    System.out.println();
}
```

Definition: Summation

$$\sum_{i=a}^{b} f(i) = f(a) + f(a + 1) + f(a + 2) + \ldots + f(b-2) + f(b-1) + f(b)$$

$$T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} 1$$

How do we model this part? Summations!

$$1 + 2 + 3 + 4 + \ldots + n = \sum_{i=1}^{n} i$$

What is the Big O?
Simplifying Summations

for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.println("Hello!");
    }
}

\[
T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n-1} 1 \cdot i = 1 \sum_{i=0}^{n-1} i = \frac{n(n - 1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n = O(n^2)
\]

- Summation of a constant: \[\sum_{i=0}^{n-1} c = cn\]
- Factoring out a constant: \[\sum_{i=a}^{b} cf(i) = c \sum_{i=a}^{b} f(i)\]
- Gauss’s Identity: \[\sum_{i=0}^{n-1} i = \frac{n(n - 1)}{2}\]

Find closed form using summation identities (given on exams)
Questions
Appendix
Tree Method Practice

\[ T(n) = \begin{cases} 
4 & \text{when } n \leq 1 \\
3T\left(\frac{n}{4}\right) + n^2 & \text{otherwise}
\end{cases} \]

Answer the following questions:
1. What is the size of the input on level \( i \)?
2. What is the work done by each node on the \( i^{th} \) recursive level?
3. What is the number of nodes at level \( i \)?
4. What is the total work done at the \( i^{th} \) recursive level?
5. What value of \( i \) does the last level occur?
6. What is the total work across the base case level?
Tree Method Practice

1. What is the size of the input on level $i$?
   \[
   \frac{n}{4^i}
   \]

2. What is the work done by each node on the $i^{th}$ recursive level?
   \[
   \left(\frac{n}{4^i}\right)^2
   \]

3. What is the number of nodes at level $i$?
   \[
   3^i
   \]

4. What is the total work done at the $i^{th}$ recursive level?
   \[
   3^i \left[\left(\frac{n}{4^i}\right)^2\right] = \left(\frac{3}{16}\right)^i n^2
   \]

5. What value of $i$ does the last level occur?
   \[
   \frac{n}{4^i} = 1 \rightarrow n = 4^i \rightarrow i = \log_4 n
   \]

6. What is the total work across the base case level?
   \[
   3^{\log_4 n} \cdot 4
   \]

Combining it all together...

\[
T(n) = \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i n^2 + 4n^{\log_3 4}
\]

**Table:**

<table>
<thead>
<tr>
<th>Level (i)</th>
<th>Number of Nodes</th>
<th>Work per Node</th>
<th>Work per Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$n^2$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>$\left(\frac{n}{4}\right)^2$</td>
<td>$\frac{3}{4} n^2$</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>$\left(\frac{n}{4^2}\right)^2$</td>
<td>$\frac{3^2}{4^4} n^2$</td>
</tr>
<tr>
<td>base</td>
<td>$3^{\log_4 n}$</td>
<td>4</td>
<td>$4 \times 3^{\log_4 n}$</td>
</tr>
</tbody>
</table>
Tree Method Practice

\[ T(n) = \sum_{i=0}^{\log_4 n - 1} \left( \frac{3}{16} \right)^i n^2 + 4n^{\log_4 3} \]

Identities are on the [webpage](#). You don’t need to memorize them.

So what’s the big-\( \Theta \)...

\[ T(n) = n^2 \left( -\frac{16}{13} \right) \left( \frac{3}{16} \right)^{\log_4 n} + \left( \frac{16}{13} \right) n^2 + 4n^{\log_4 3} \]

Closed form:

\[ T(n) = n^2 \left( \frac{3^{\log_4 n}}{16} - 1 \right) + 4n^{\log_4 3} \]

\[ T(n) \in \Theta(n^2) \]
More Tree Method

\[ T(n) = \begin{cases} 
6T\left(\frac{n}{2}\right) + 2n & \text{if } n > 8 \\
3 & \text{otherwise}
\end{cases} \]
Tree Method Practice

\[ T(n) = \begin{cases} 
6T\left(\frac{n}{2}\right) + 2n & \text{if } n > 8 \\
3 & \text{otherwise}
\end{cases} \]

Answer the following questions:
1. What is the size of the input on level \( i \)?
2. What is the work done by each node on the \( i^{th} \) recursive level?
3. What is the number of nodes at level \( i \)?
4. What is the total work done at the \( i^{th} \) recursive level?
5. What value of \( i \) does the last level occur?
6. What is the total work across the base case level?
Tree Method Practice

1. What is the size of the input on level $i$? $\frac{n}{2^i}$

2. What is the work done by each node on the $i^{th}$ recursive level? $2 \cdot \frac{n}{2^i}$

3. What is the number of nodes at level $i$? $6^i$

4. What is the total work done at the $i^{th}$ recursive level? $6^i \left[ 2 \cdot \frac{n}{2^i} \right] = 2 \cdot 3^i \cdot n$

5. What value of $i$ does the last level occur? $\frac{n}{2^i} = 2 \Rightarrow n = 2^{i+1} \Rightarrow i = \log_2(n) - 1$

6. What is the total work across the base case level? $6^{\log_2(n) - 1} \cdot 3$

Combining it all together...

$$T(n) = \begin{cases} 6T \left( \frac{n}{2} \right) + 2n & \text{if } n > 2 \\ 3 & \text{otherwise} \end{cases}$$

<table>
<thead>
<tr>
<th>Level ($i$)</th>
<th>Number of Nodes</th>
<th>Work per Node</th>
<th>Work per Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$2n$</td>
<td>$2n$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>$\frac{2n}{8}$</td>
<td>$\frac{n}{2}$</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>$\frac{2(n)}{8^2}$</td>
<td>$\frac{n}{8}$</td>
</tr>
<tr>
<td>base</td>
<td>$2^{\log_2n-1}$</td>
<td>3</td>
<td>$\frac{3}{2}n^{1/3}$</td>
</tr>
</tbody>
</table>

$$T(n) = \sum_{i=0}^{\log_2(n)-2} 2 \cdot 3^i n + \frac{1}{2} n^{\log_2 6}$$

$$\frac{3 \cdot 6^{\log_2 n}}{6} = \frac{1}{2} \cdot n^{\log_2 6} = \frac{1}{2} \cdot n^{\log_2 6}$$
\[
T(n) = \sum_{i=0}^{\log_2(n)-2} 2 \cdot 3^i + \frac{1}{2} n^{\log_2 6} \\
= 2n \sum_{i=0}^{\log_2(n)-2} 3^i + \frac{1}{2} n^{\log_2 6} \\
= 2n \frac{3^{\log_2(n)-1}}{3-1} + \frac{1}{2} n^{\log_2 6} \\
= n \cdot \frac{n^{\log_2(3)}}{3} + \frac{1}{2} n^{\log_2 6} \\
= \frac{n^{\log_2(3)+1}}{3} + \frac{1}{2} n^{\log_2 6} \\
= \frac{n^{\log_2(6)}}{3} + \frac{1}{2} n^{\log_2 6} = \frac{5}{6} n^{\log_2 6}
\]

finite geometric series
\[
\sum_{i=0}^{n-1} x^i = \frac{x^n - 1}{x - 1}
\]

power of a log
\[
x^{\log_b y} = y^{\log_b x}
\]

\[1 = \log_2 2\]

\[\log_a b + \log_a c = \log_a (bc)\]
public static void primesUpToN(int n) {
    System.out.print("1 2 ");
    for (int i = 3; i <= n; i++) {
        for (int j = 2; j < i; j++){
            if (j != i && j % i == 0) {
                System.out.print(i + " ");
                break;
            }
        }
    }
    System.out.println();
}

\[
T(n) = 1 + \sum_{i=3}^{n} \sum_{j=2}^{i-1} 5 = 1 + \sum_{i=0}^{n-3} \sum_{j=0}^{i-3} 5 = 1 + \sum_{i=0}^{n-3} 5(i - 2) = 1 + 5(\sum_{i=0}^{n-3} i - \sum_{i=0}^{n-3} 2) = = 1 + 5\left(\frac{(n-2)(n-3)}{2}\right) - (n-2)(2)
\]