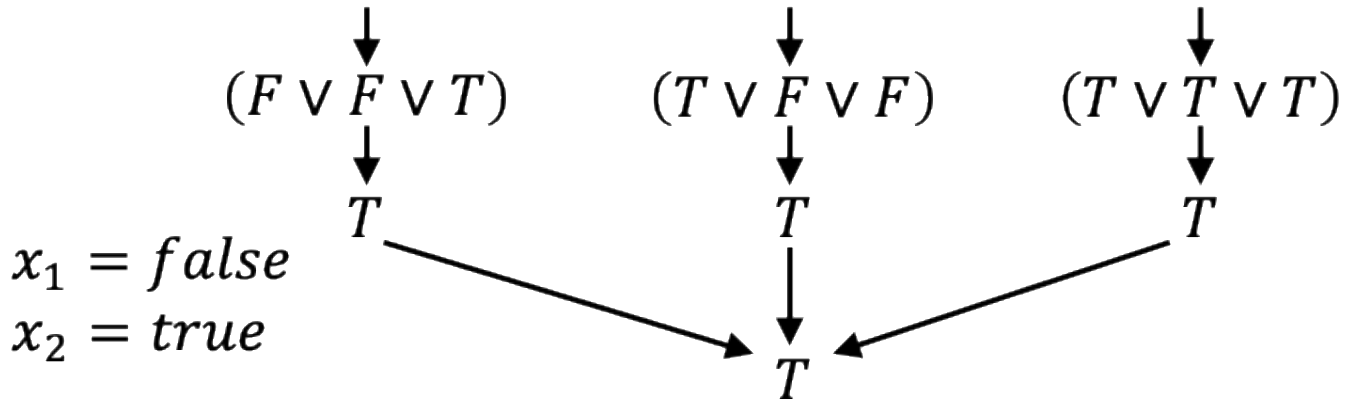
Test 2 Review
CSCI 432

Theoretical CS News

Theoretical CS News

3−SAT:

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



Theoretical CS News

Exponential Time Hypothesis (ETH):
3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

3−SAT:

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

$$(F \vee F \vee T) \quad (T \vee F \vee F) \quad (T \vee T \vee T)$$

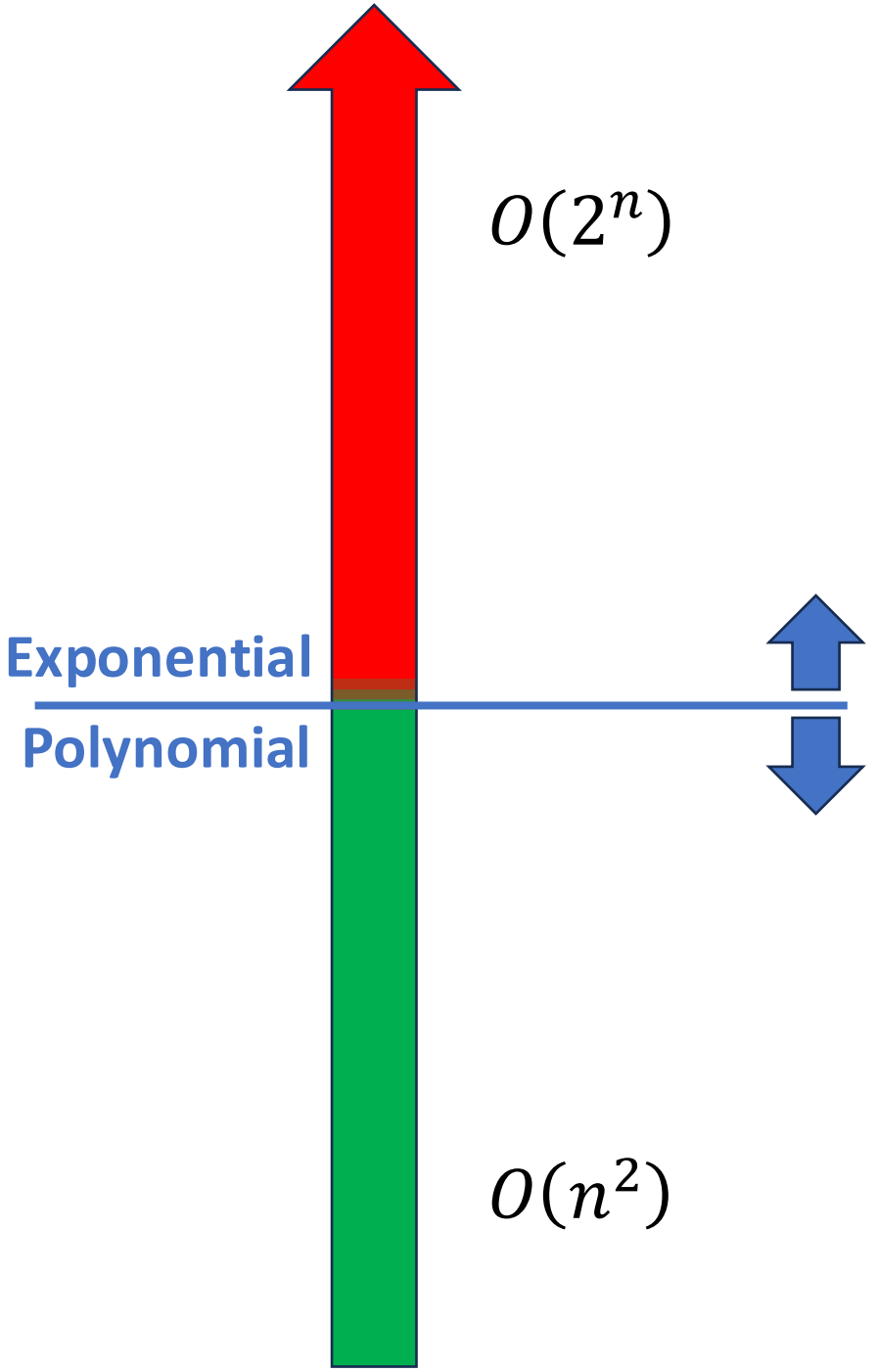
$$T \quad T \quad T$$

$$x_1 = false$$
$$x_2 = true$$

$$T$$

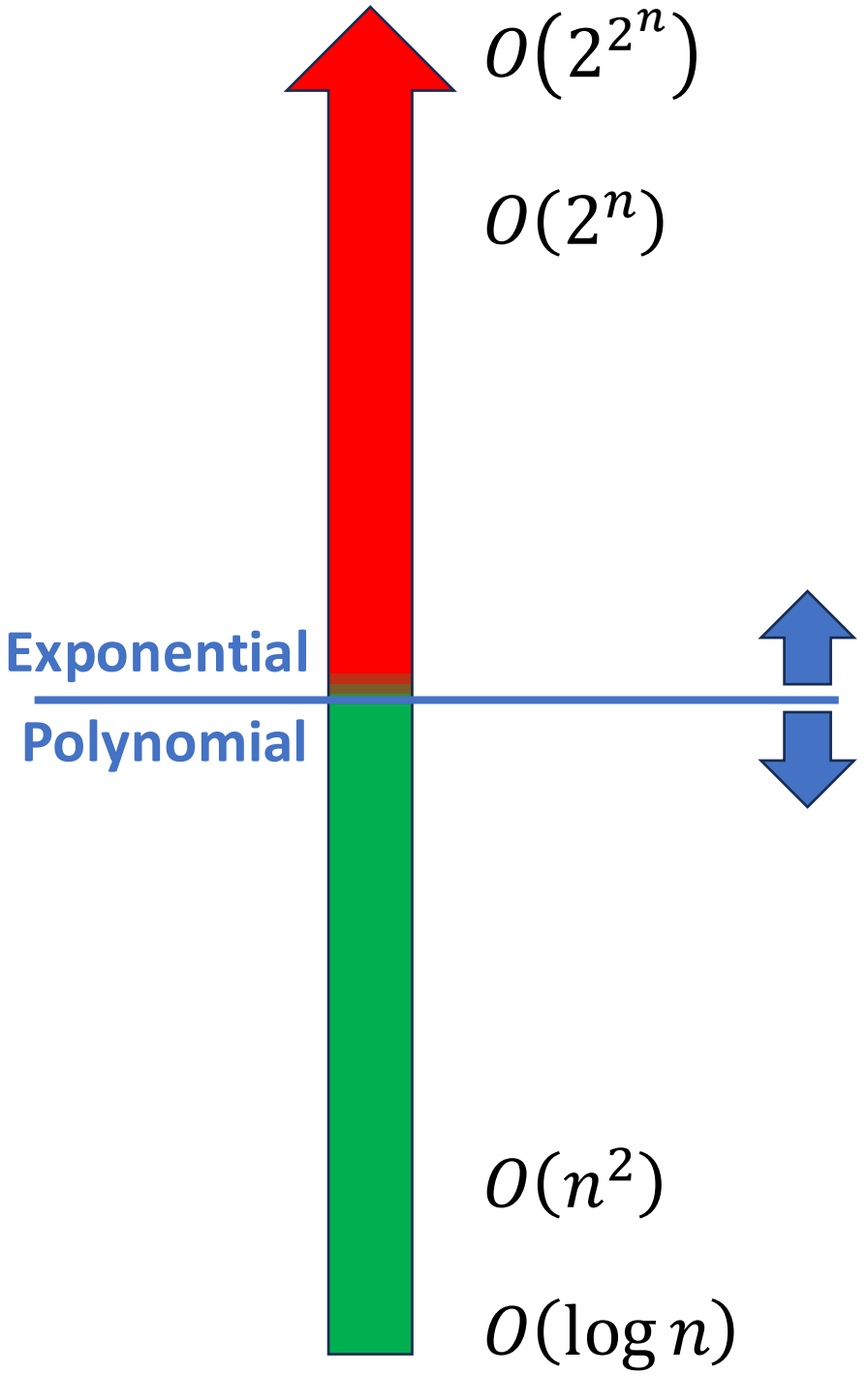
Theoretical CS News

Exponential Time Hypothesis (ETH):
3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).



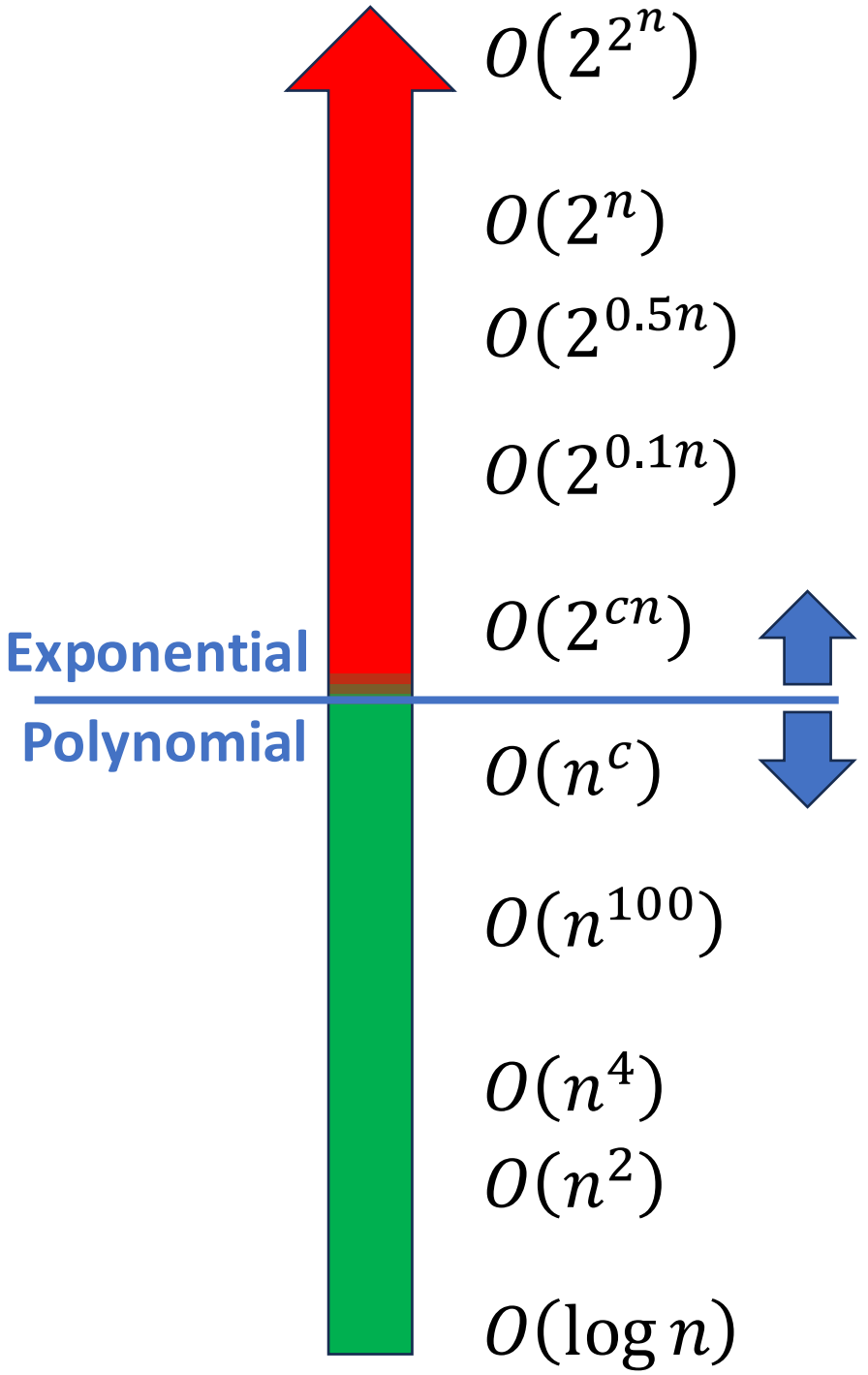
Theoretical CS News

Exponential Time Hypothesis (ETH): 3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).



Theoretical CS News

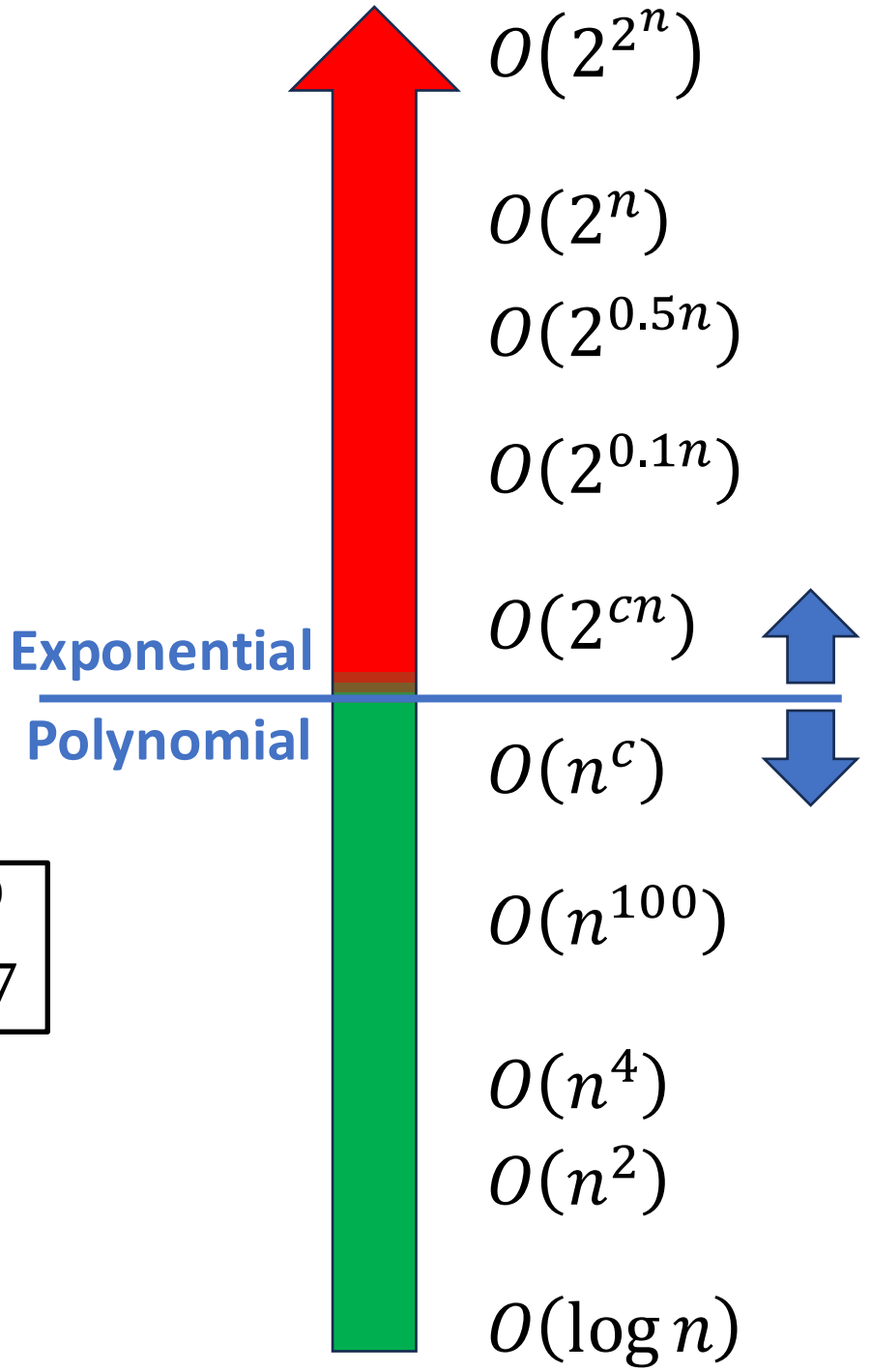
Exponential Time Hypothesis (ETH): 3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).



Theoretical CS News

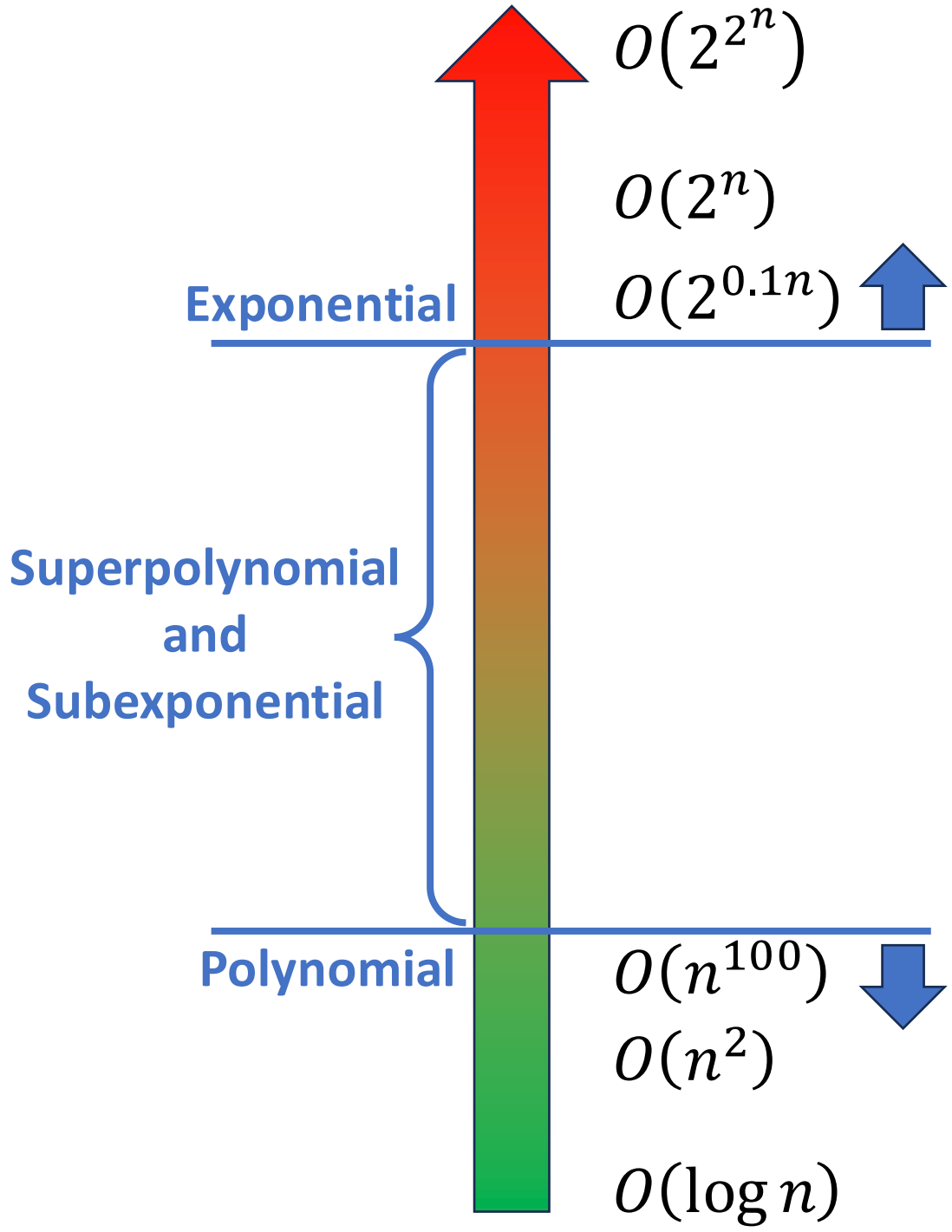
Exponential Time Hypothesis (ETH): 3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

$2^{0.1n}$ exceeds n^{100} around $n = 13{,}747$



Theoretical CS News

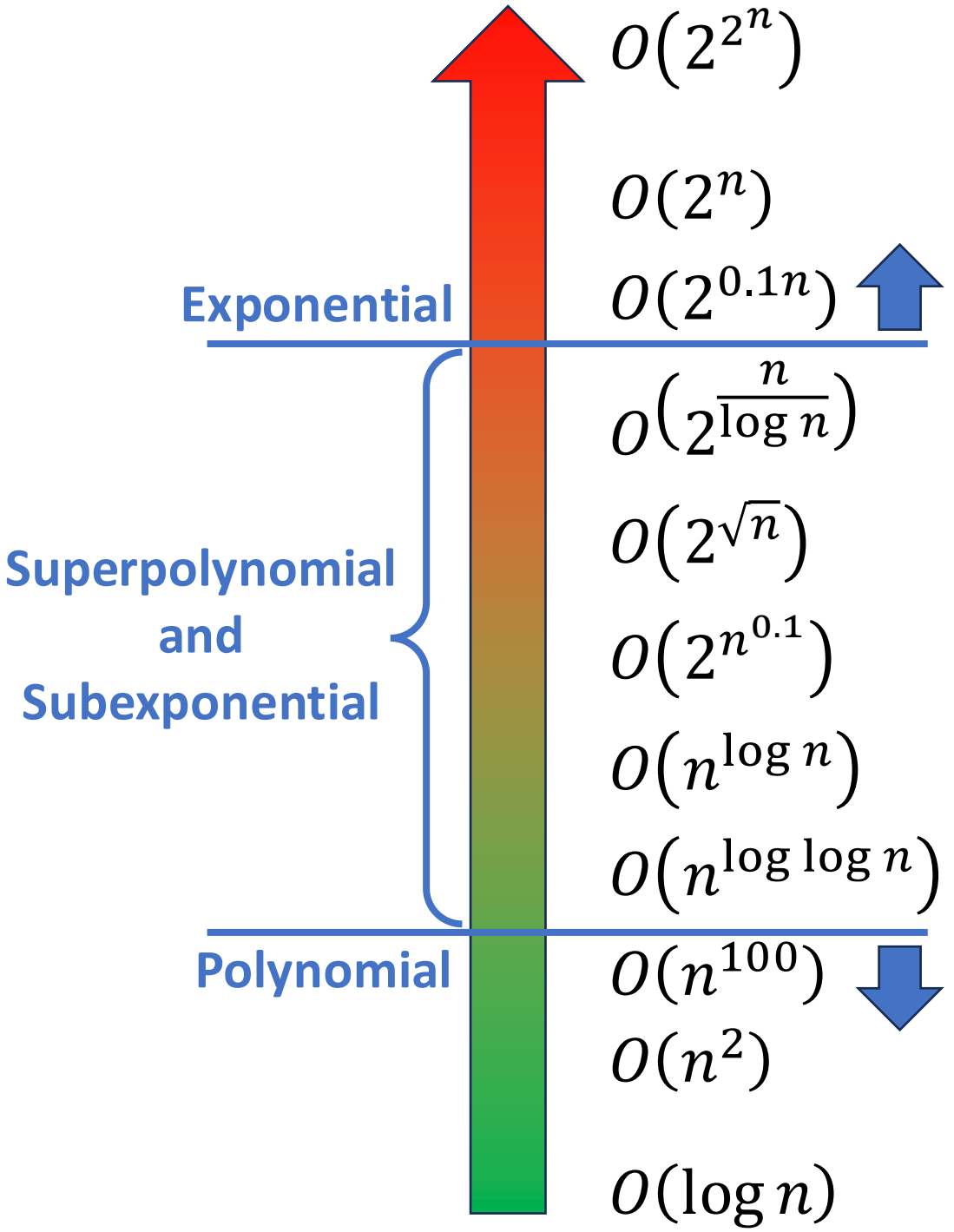
Exponential Time Hypothesis (ETH): 3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).



Theoretical CS News

Exponential Time Hypothesis (ETH): 3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

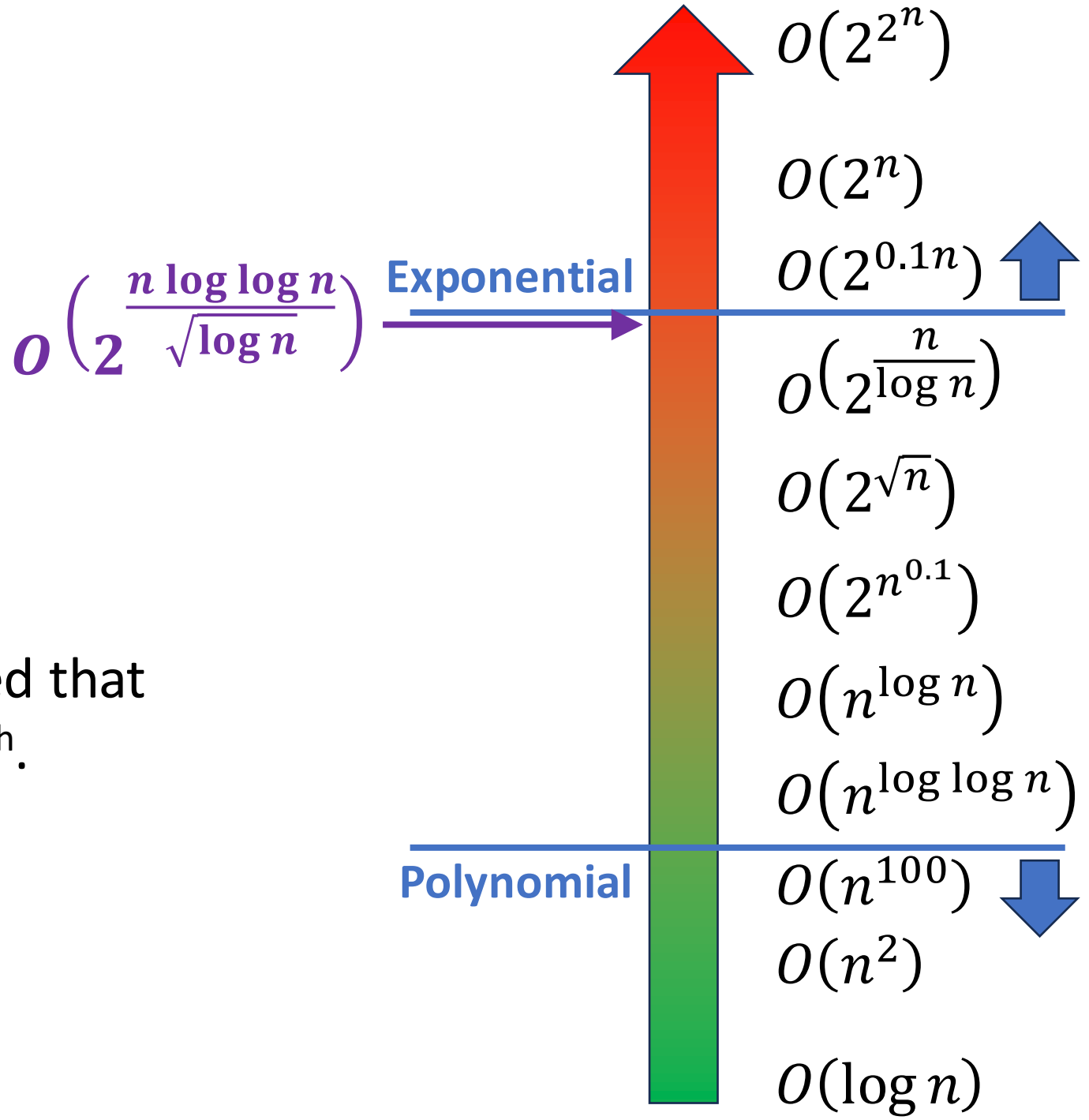
There is a continuum of asymptotic functions: Given any distinct functions f and g (such that g grows faster than f), we can make a new function that grows faster than f and slower than g.



Theoretical CS News

Exponential Time Hypothesis (ETH): 3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

On October 26th, OpenAI announced that they disproved ETH on October 16th.



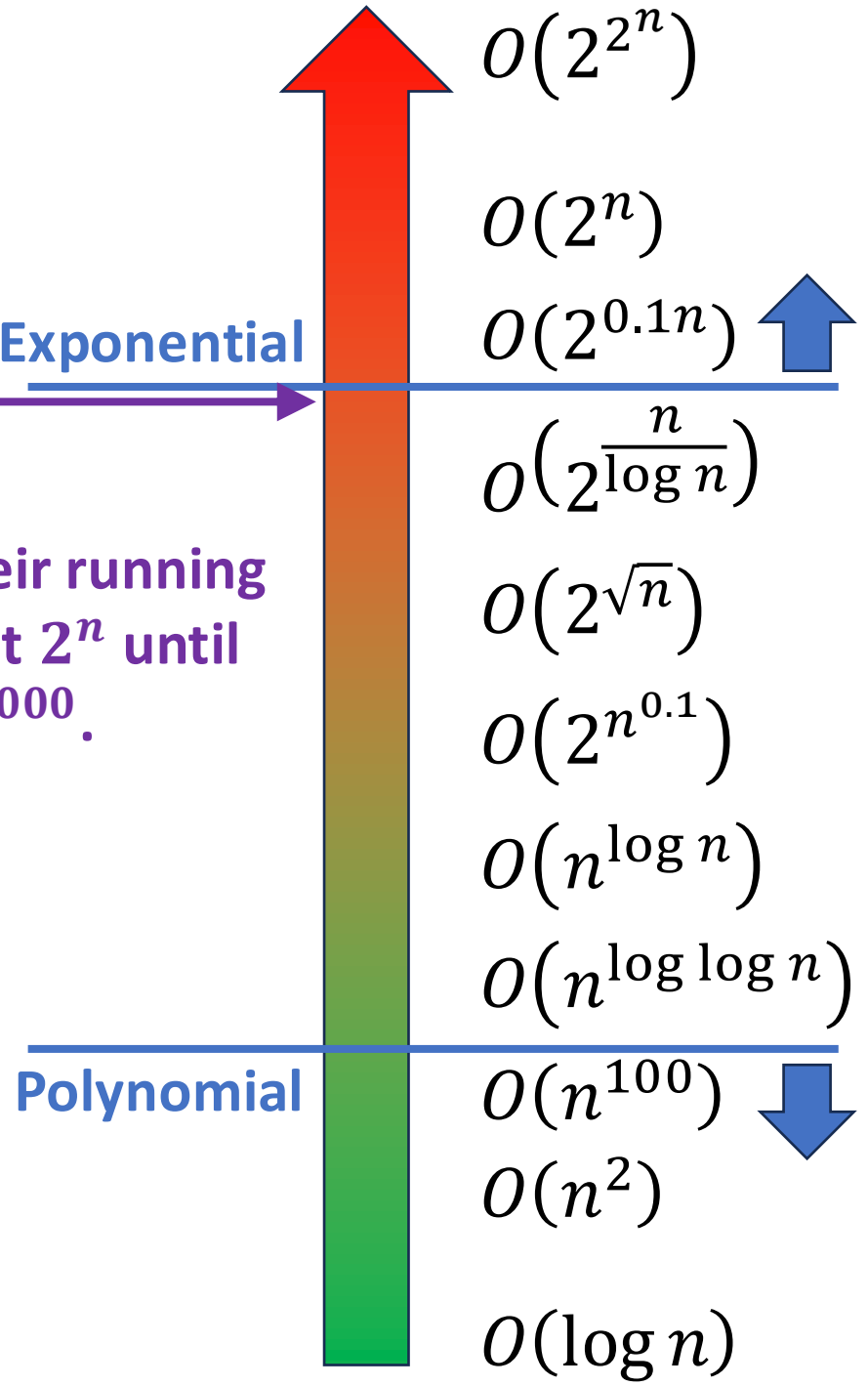
Theoretical CS News

Exponential Time Hypothesis (ETH): 3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

$O\left(2^{\frac{n \log\log n}{\sqrt{\log n}}}\right)$

Estimated that their running time does not beat 2^n until around $n \cong 10^{10,000}$.

On October 26th, OpenAI announced that they disproved ETH on October 16th.



Theoretical CS News

Exponential Time Hypothesis (ETH):
3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

On October 26th, OpenAI announced that they disproved ETH on October 16th.



$O(2^{2^n})$

$O(2^n)$

$O\left(2^{\frac{n \log}{\sqrt{\log}}}\right.$

$O(n^{\log \log n})$

Polynomial

$O(n^{100})$

$O(n^2)$

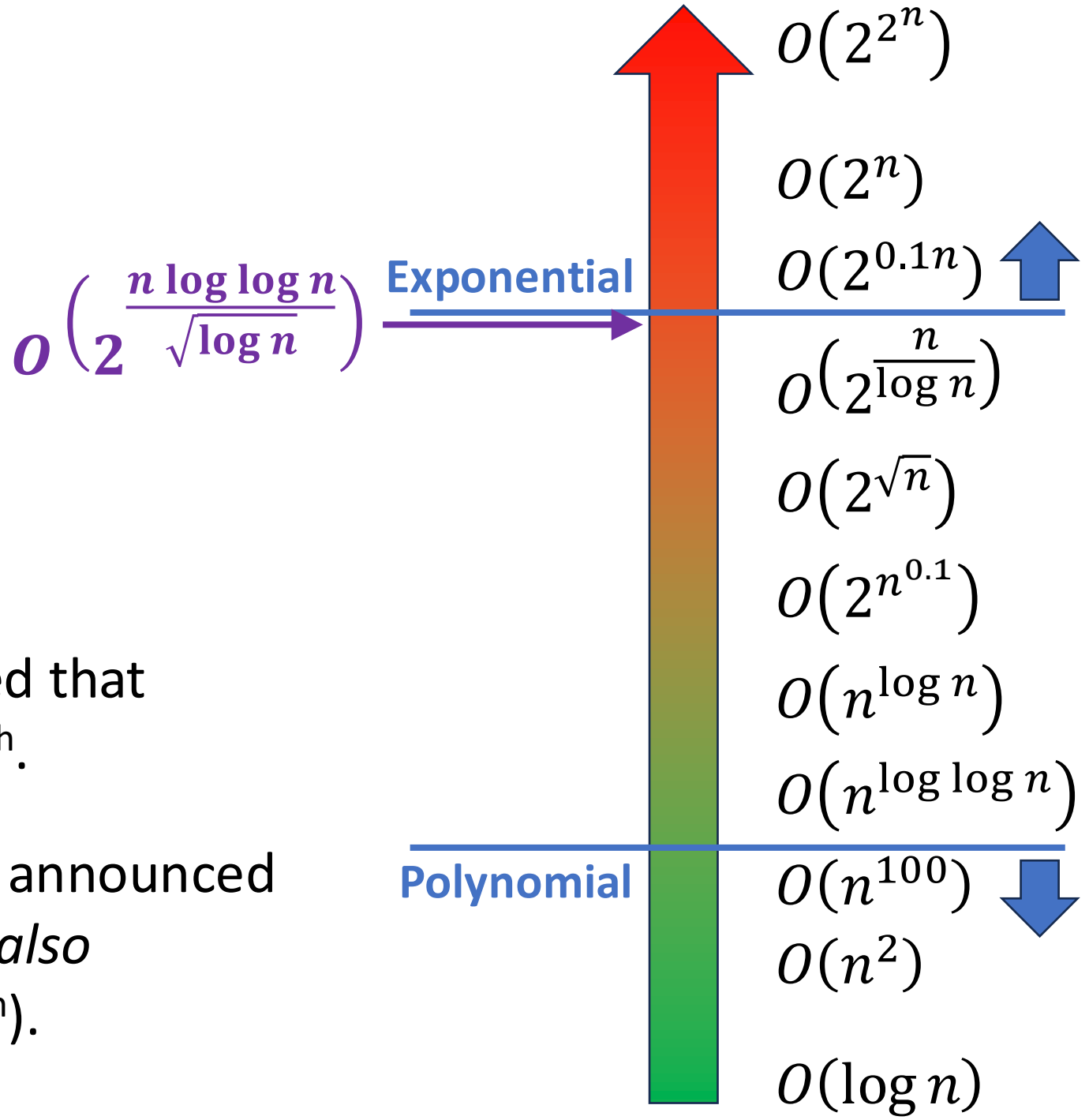
$O(\log n)$

Theoretical CS News

Exponential Time Hypothesis (ETH):
3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

On October 26th, OpenAI announced that they disproved ETH on October 16th.

About 6 hours later, Anthropic announced the same thing (i.e., that they *also* disproved ETH on October 16th).

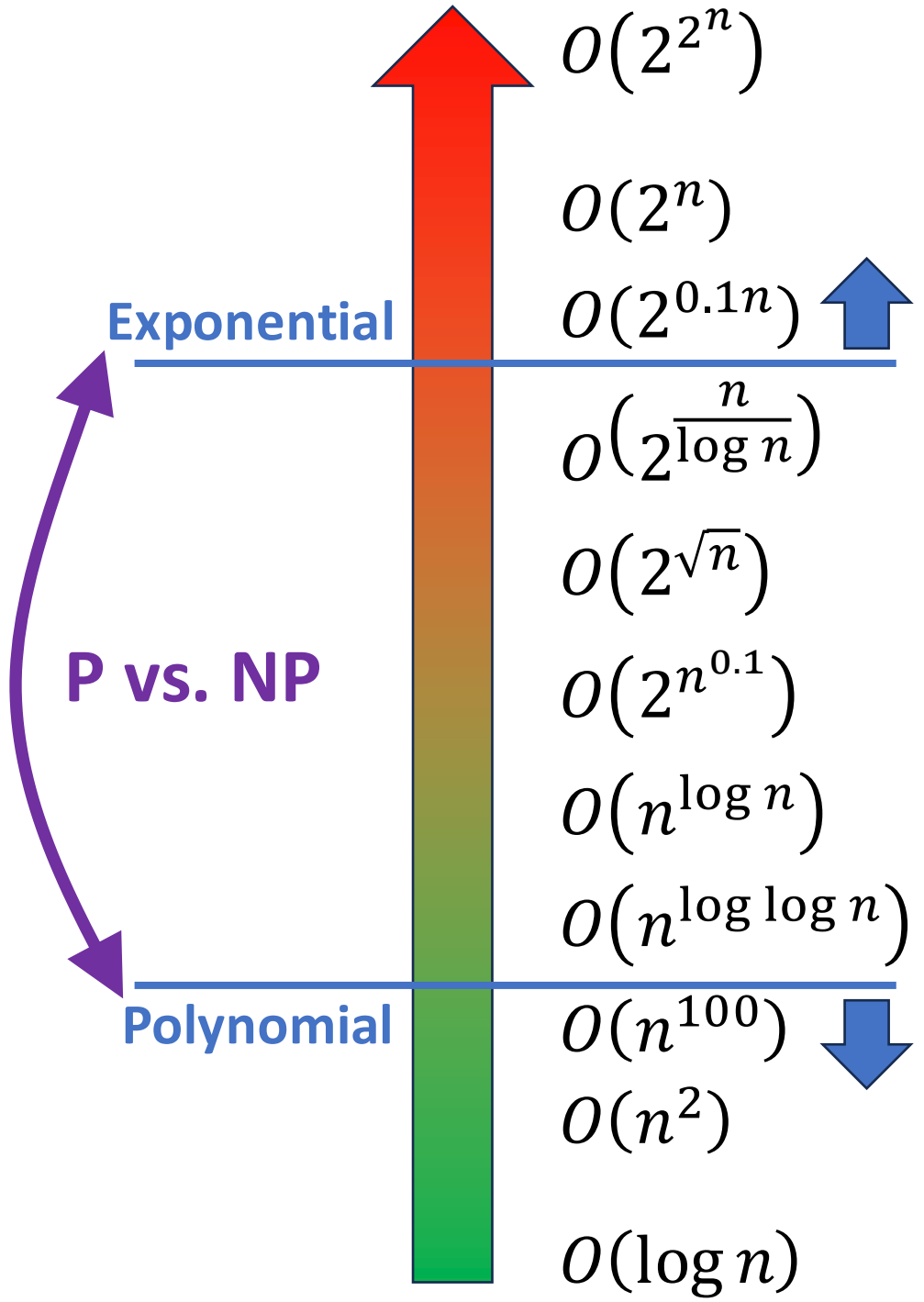


Theoretical CS News

Exponential Time Hypothesis (ETH):
3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

On October 26th, OpenAI announced that they disproved ETH on October 16th.

About 6 hours later, Anthropic announced the same thing (i.e., that they *also* disproved ETH on October 16th).

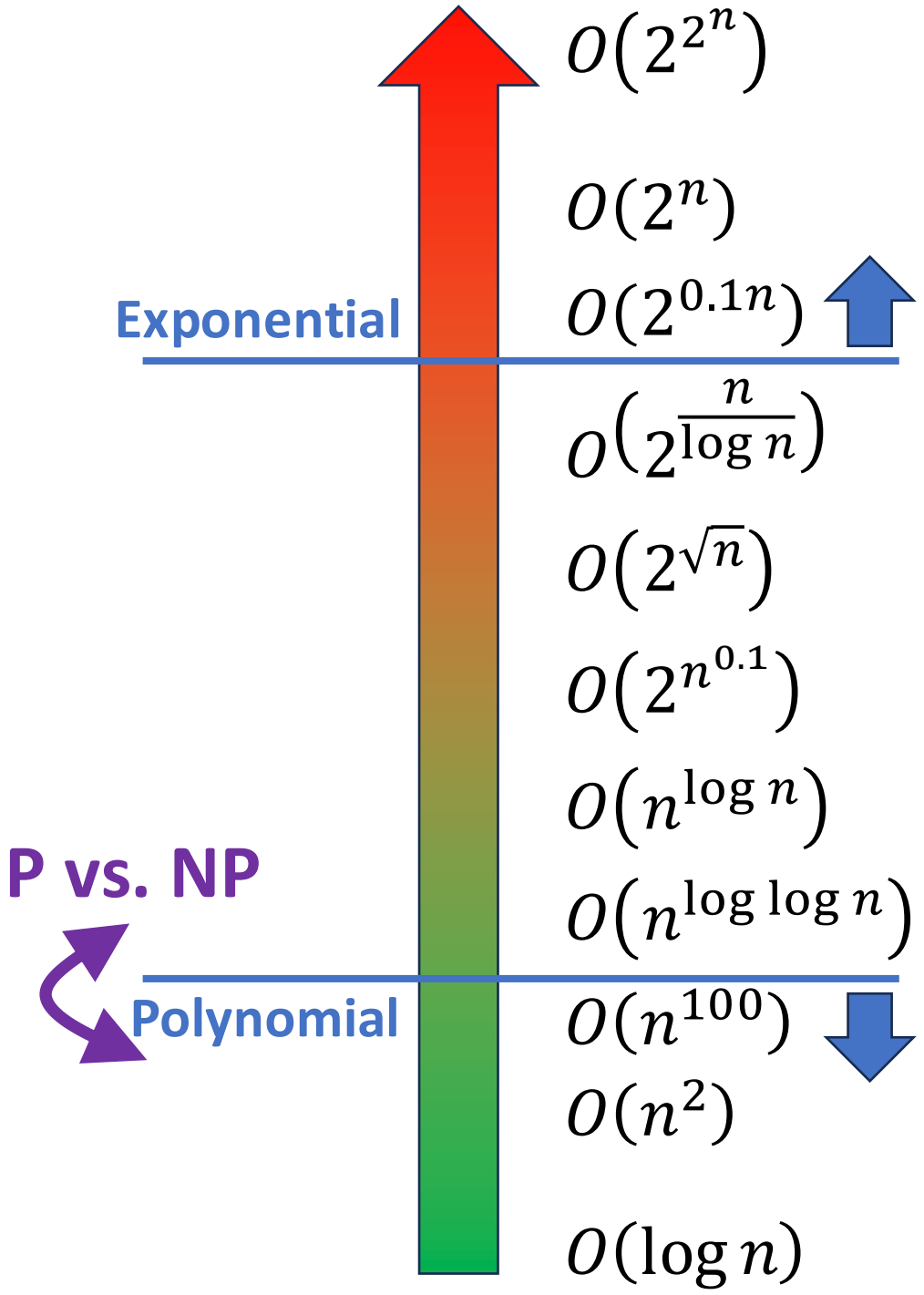


Theoretical CS News

Exponential Time Hypothesis (ETH): 3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

On October 26th, OpenAI announced that they disproved ETH on October 16th.

About 6 hours later, Anthropic announced the same thing (i.e., that they *also* disproved ETH on October 16th).

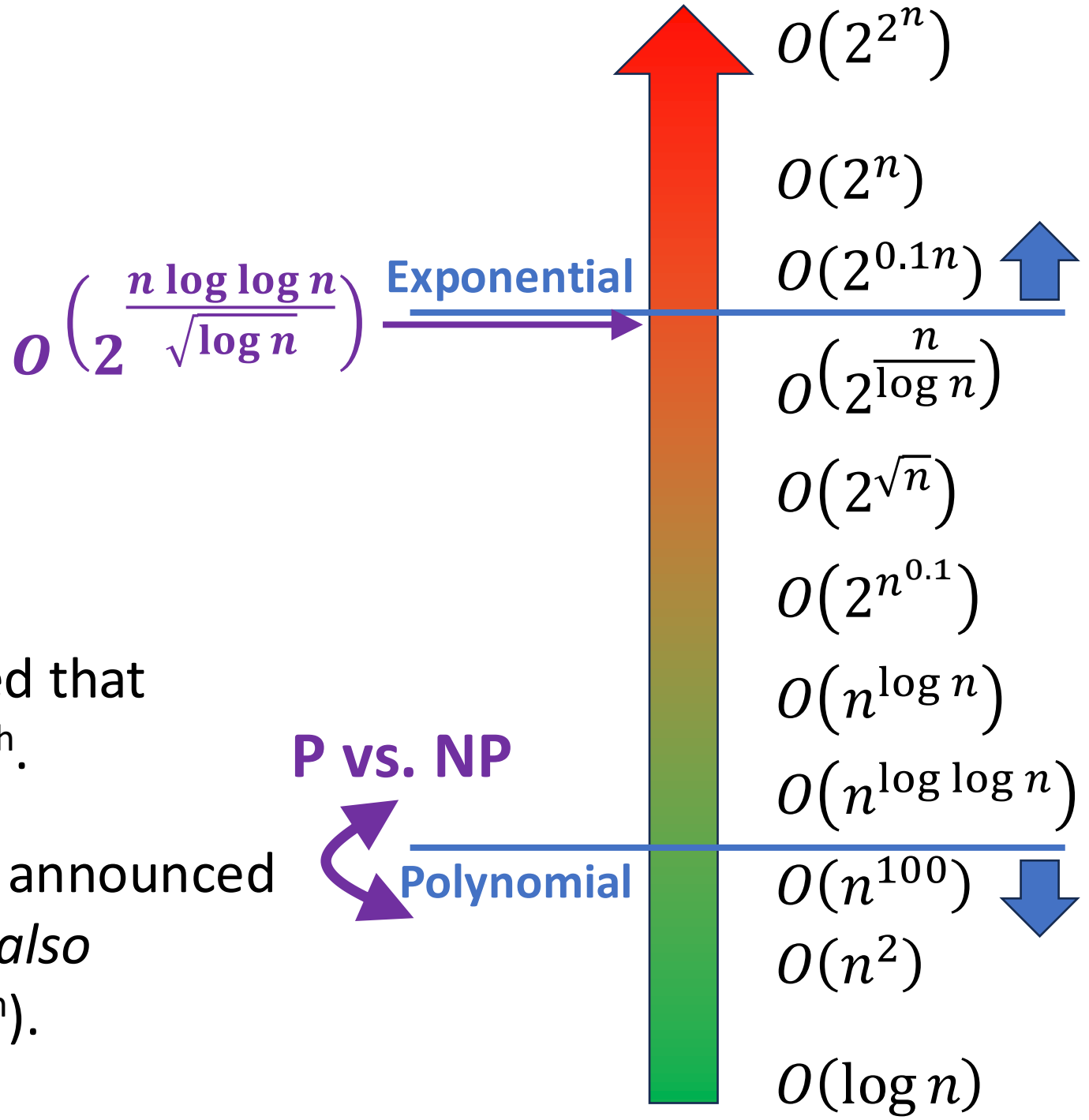


Theoretical CS News

Exponential Time Hypothesis (ETH):
3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some constant c).

On October 26th, OpenAI announced that they disproved ETH on October 16th.

About 6 hours later, Anthropic announced the same thing (i.e., that they *also* disproved ETH on October 16th).



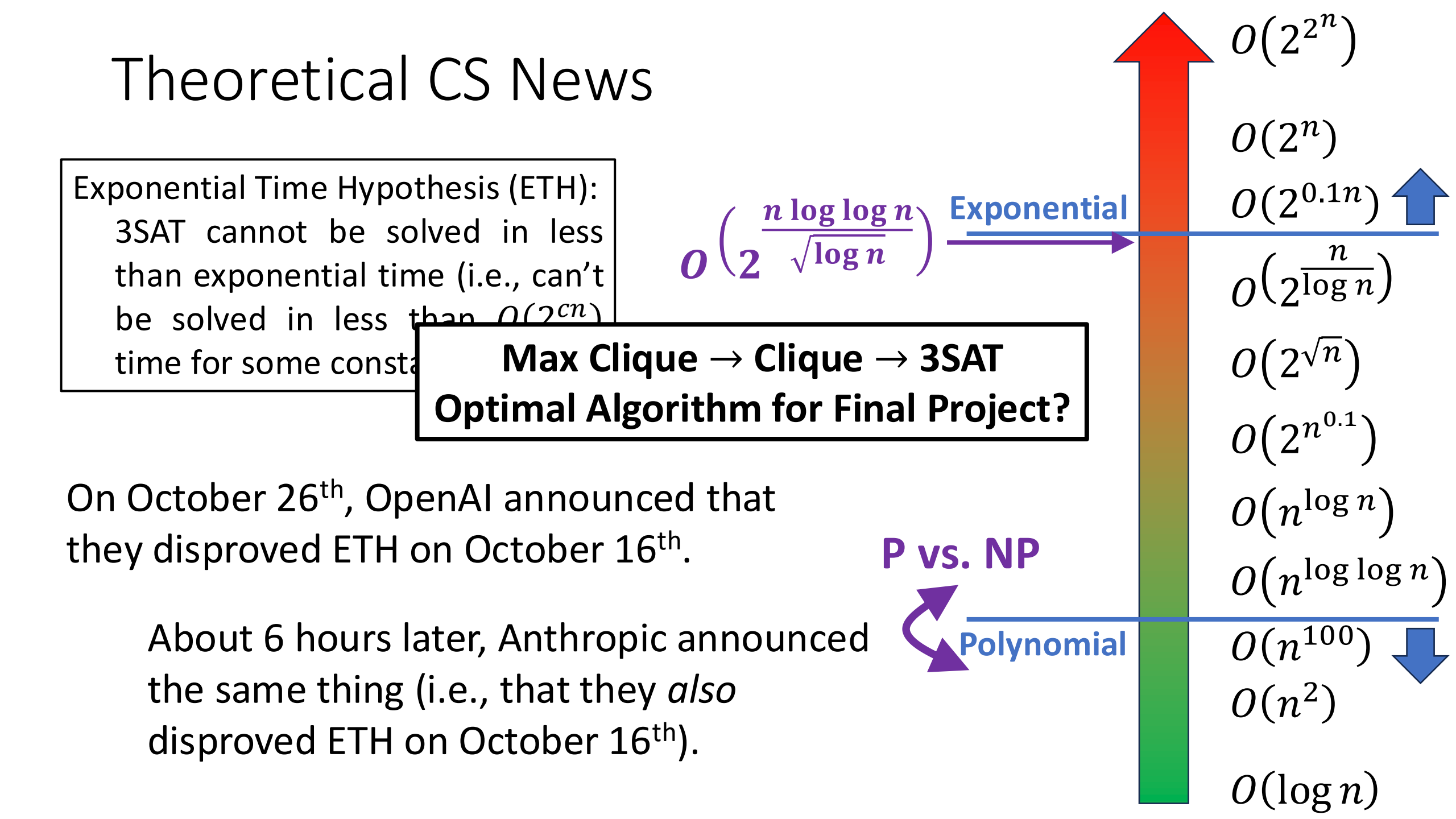
Theoretical CS News

Exponential Time Hypothesis (ETH):
3SAT cannot be solved in less than exponential time (i.e., can't be solved in less than $O(2^{cn})$ time for some consta

Max Clique → Clique → 3SAT
Optimal Algorithm for Final Project?

On October 26th, OpenAI announced that they disproved ETH on October 16th.

About 6 hours later, Anthropic announced the same thing (i.e., that they *also* disproved ETH on October 16th).



Test 2 Logistics

Focus your studying on:

- Change making
- Rod cutting
- Weighted Independent Set

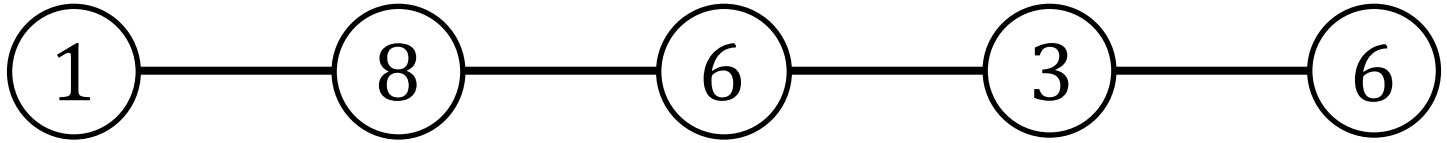
1. During class on Friday 10/30.
2. You can bring your book and any notes you would like, but no electronic devices.
3. You may assume anything proven in class or in workshops.
4. Three questions (12 points):
    1) Fill out a DP table and identify a calculation (5 points).
    2) Provide recursive function (5 points).
    3) Provide recursive function (2 points).

Code

Input: n node graph, $G = (V, E)$, with vertex weights.

Output: Weight of the heaviest vertex subset that do not share edges.

```
maxWeight(G = (V, E))
    A[0,...,n] = [0,w_1,0,...,0]
    for i = 2,...,n
      A[i] = max(A[i - 1], A[i - 2] + w_i)
    return A[n]
```



$A[i]$ – weight of the optimal solution for V_i.

$$A[i] = \max\begin{pmatrix} A[i-1] \\ A[i-2] + w_i \end{pmatrix}$$

A[n] =

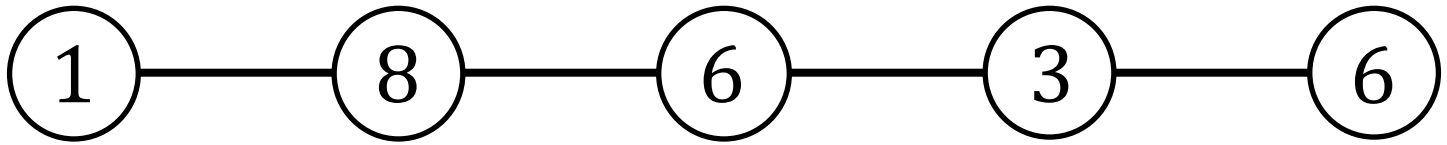
0	1	2	3	4	5
0	0	0	0	0	0

Weighted IS on Line

Input: n node graph, $G = (V, E)$, with vertex weights.

Output: Weight of the heaviest vertex subset that do not share edges.

```
maxWeight(G = (V, E))
    A[0,...,n] = [0,w1,0,...,0]
    for i = 2,...,n
      A[i] = max(A[i - 1], A[i - 2] + wi)
    return A[n]
```



$A[i]$ – weight of the optimal solution for V_i.

A[n] =

0	1	2	3	4	5
0	1	8	8	11	14

$$\max\begin{bmatrix}\mathbf{11}\\ \mathbf{8+6}\end{bmatrix}$$

$$A[i] = \max\begin{pmatrix}A[i-1]\\ A[i-2]+w_i\end{pmatrix}$$

Recursive Function

Suppose that $G[k]$ is the max/min___(something related to k)___.
Recursively define $G[k]$. Maybe a hint (or not).

Dynamic Programming Algorithm

Input: n node graph, $G = (V, E)$, with vert

Suppose that $G[k]$ is the max/min (something related to k).
Recursively define $G[k]$.

Output: Subset of vertices such that they do not share any edges with each other, and the sum of their weights (w_i) is maximized.

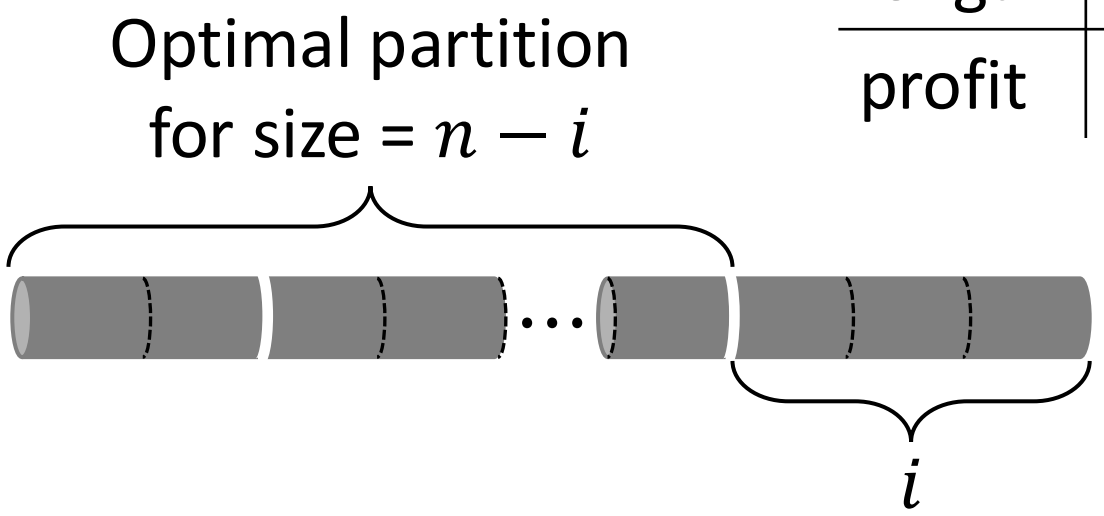
Optimal solution to V_i is either the optimal for V_{i-1} (with vertex v_i excluded), or the optimal for V_{i-2} (with vertex v_i included).

Let $A[i]$ be the weight of the optimal solution for V_i.

$$A[i] = \max \begin{pmatrix} A[i-1] \\ A[i-2] + w_i \end{pmatrix}$$

Rod Cutting

length	1	2	3	4	…	n
profit	\$1	\$5	\$8	\$9	…	\$?



$M[n]$ = optimal profit from partitioning rod of length n.

p_i = profit for rod of length i.

$$M[n] = \max_{1 \le i \le n}(M[n-i] + p_i)$$

Check all possibilities!

Suppose that $G[k]$ is the max/min (something related to k).
Recursively define $G[k]$.

Making Change

In general, suppose a country has denominations:

$$1 = d_1 < d_2 < \cdots < d_k$$

(US coins: $d_1 = 1, d_2 = 5, d_3 = 10, d_4 = 25$)

$C(p)$ – minimum number of coins to make p cents.

x – value (e.g. \$0.25) of a coin used in the optimal solution.

So, we'll check all possibilities.

But we don't know what actual coin is in the optimal solution.

$$C(p) = \min_{d_i \leq p} C(p - d_i) + 1$$

Suppose that $G[k]$ is the max/min (something related to k).
Recursively define $G[k]$.

Max Nonadjacent Sum

Suppose you have a list of numbers that you will select from. You are not allowed to select adjacent entries. Your goal is to select numbers from the list such that the sum of selected values is maximized.

Let $S[i]$ be the maximum sum of the of the first i entries in the list.
Recursively define $S[i]$, given that v_i is the value of entry i.
Hint: What are the options for entry i being selected?

Suppose that $G[k]$ is the max/min (something related to k) .
Recursively define $G[k]$.

Max Nonadjacent Sum

Suppose you have a list of numbers that you will select from. You are not allowed to select adjacent entries. Your goal is to select numbers from the list such that the sum of selected values is maximized.

Let $S[i]$ be the maximum sum of the of the first i entries in the list.
Recursively define $S[i]$, given that v_i is the value of entry i.
Hint: What are the options for entry i being selected?

If entry i is used.

Suppose that $G[k]$ is the max/min (something related to k) .
Recursively define $G[k]$.

If entry i is not used.

Max Nonadjacent Sum

Suppose you have a list of numbers that you will select from. You are not allowed to select adjacent entries. Your goal is to select numbers from the list such that the sum of selected values is maximized.

Let $S[i]$ be the maximum sum of the of the first i entries in the list.
Recursively define $S[i]$, given that v_i is the value of entry i.
Hint: What are the options for entry i being selected?

If entry i is used.

$$S[i-2] + v_i$$

If entry i is not used.

Suppose that $G[k]$ is the max/min (something related to k) .
Recursively define $G[k]$.

Max Nonadjacent Sum

Suppose you have a list of numbers that you will select from. You are not allowed to select adjacent entries. Your goal is to select numbers from the list such that the sum of selected values is maximized.

Let $S[i]$ be the maximum sum of the of the first i entries in the list.
Recursively define $S[i]$, given that v_i is the value of entry i.
Hint: What are the options for entry i being selected?

If entry i is used.

$$S[i-2] + v_i$$

$$S[i-1]$$

If entry i is not used.

Suppose that $G[k]$ is the max/min (something related to k).
Recursively define $G[k]$.

Max Nonadjacent Sum

Suppose you have a list of numbers that you will select from. You are not allowed to select adjacent entries. Your goal is to select numbers from the list such that the sum of selected values is maximized.

Let $S[i]$ be the maximum sum of the of the first i entries in the list.
Recursively define $S[i]$, given that v_i is the value of entry i.
Hint: What are the options for entry i being selected?

Suppose that $G[k]$ is the max/min (something related to k).
Recursively define $G[k]$.

$$S[i] = \max\begin{pmatrix} S[i-2] + v_i \\ S[i-1] \end{pmatrix}$$

If entry i is used.

If entry i is not used.