Teoria dos Grafos e Computabilidade

— Computational cost —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Feb 2025

Teoria dos Grafos e Computabilidade

— Computational Tractability —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Feb 2025

What is Algorithm Analysis?

- Measure resource requirements: how does the **amount of time and space** an algorithm uses scale with increasing input size?

What is Algorithm Analysis?

- Measure resource requirements: how does the amount of time and space an algorithm uses scale with increasing input size?
- How do we put this notion on a concrete footing?

What is Algorithm Analysis?

- Measure resource requirements: how does the amount of time and space an algorithm uses scale with increasing input size?
- How do we put this notion on a concrete footing?
- What does it mean for one function to grow faster or slower than another?

What is Algorithm Analysis?

- Measure resource requirements: how does the amount of time and space an algorithm uses scale with increasing input size?
- How do we put this notion on a concrete footing?
- What does it mean for one function to grow faster or slower than another?

Develop algorithms that provably run quickly and use low amounts of space.

Worst-case Running Time

- We will measure worst-case running time of an algorithm.

Worst-case Running Time

- We will measure worst-case running time of an algorithm.
- Bound the largest possible running time the algorithm over all inputs of size n, as a function of n.

Worst-case Running Time

- We will measure worst-case running time of an algorithm.
- Bound the largest possible running time the algorithm over all inputs of size n, as a function of n.
- Why worst-case? Why not average-case or on random inputs?

Worst-case Running Time

- We will measure worst-case running time of an algorithm.
- Bound the largest possible running time the algorithm over all inputs of size n, as a function of n.
- Why worst-case? Why not average-case or on random inputs?
- Input size $=$ number of elements in the input. Values in the input do not matter.

Worst-case Running Time

- We will measure worst-case running time of an algorithm.
- Bound the largest possible running time the algorithm over all inputs of size n, as a function of n.
- Why worst-case? Why not average-case or on random inputs?
- Input size = number of elements in the input. Values in the input do not matter.
- Assume all elementary operations take unit time: assignment, arithmetic on a fixed-size number, comparisons, array lookup, following a pointer, etc.

Polynomial Time

- Brute force algorithm: Check every possible solution.

Polynomial Time

- Brute force algorithm: Check every possible solution.
- What is a brute force algorithm for sorting: given n numbers, permute them so that they appear in increasing order?

Polynomial Time

- Brute force algorithm: Check every possible solution.
- What is a brute force algorithm for sorting: given n numbers, permute them so that they appear in increasing order?
  - Try all possible $n!$ permutations of the numbers.
  - For each permutation, check if it is sorted.

Polynomial Time

- Brute force algorithm: Check every possible solution.
- What is a brute force algorithm for sorting: given n numbers, permute them so that they appear in increasing order?
  - Try all possible $n!$ permutations of the numbers.
  - For each permutation, check if it is sorted.
  - Running time is $nn!$. Unacceptable in practice!

Polynomial Time

- Brute force algorithm: Check every possible solution.
- What is a brute force algorithm for sorting: given n numbers, permute them so that they appear in increasing order?
  - Try all possible $n!$ permutations of the numbers.
  - For each permutation, check if it is sorted.
  - Running time is $nn!$. Unacceptable in practice!
- Desirable scaling property: when the input size doubles, the algorithm should only slow down by some constant factor c.

Polynomial Time

- Brute force algorithm: Check every possible solution.
- What is a brute force algorithm for sorting: given n numbers, permute them so that they appear in increasing order?
  - Try all possible $n!$ permutations of the numbers.
  - For each permutation, check if it is sorted.
  - Running time is $nn!$. Unacceptable in practice!
- Desirable scaling property: when the input size doubles, the algorithm should only slow down by some constant factor c.
- An algorithm has a **polynomial** running time if there exist constants $c > 0$ and $d > 0$ such that on every input of size n, the running time of the algorithm is bounded by cn^d steps.

Polynomial Time

- Brute force algorithm: Check every possible solution.
- What is a brute force algorithm for sorting: given n numbers, permute them so that they appear in increasing order?
  - Try all possible $n!$ permutations of the numbers.
  - For each permutation, check if it is sorted.
  - Running time is $nn!$. Unacceptable in practice!
- Desirable scaling property: when the input size doubles, the algorithm should only slow down by some constant factor c.
- An algorithm has a **polynomial** running time if there exist constants $c > 0$ and $d > 0$ such that on every input of size n, the running time of the algorithm is bounded by cn^d steps.

An algorithm is efficient if it has a polynomial running time.

Teoria dos Grafos e Computabilidade

— Exercises —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Exercises

Solve the exercises related to computational cost.

The idea is to compute the number of operations of each part of the code.

Teoria dos Grafos e Computabilidade

— Recurrences —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & if \;\; n > \;1 \\ 0, & otherwise \end{cases}$$

$$T(n) \quad = $$

Recurrences

$$T(n) = \begin{cases} T(n-1) + 1, & \textit{if } \; n > \; 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll} T(n) & = & T(n-1) & +1 \\ T(n-1) & = & T(n-2) & +1 \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \textit{if } \ n > \ 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{ccccc} T(n) & = & T(n-1) & & +1 \\ T(n-1) & = & T(n-2) & & +1 \\ \vdots & & \vdots & & \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n-1) + 1, & \textit{if } \ n > \ 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{ccccc} T(n) & = & T(n-1) & +1 \\ T(n-1) & = & T(n-2) & +1 \\ \vdots & & \vdots & \\ T(n-i) & = & T(n-i-1) & +1 \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \textit{if } \; n > \; 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{ccccc} T(n) & = & T(n-1) & +1 \\ T(n-1) & = & T(n-2) & +1 \\ \vdots & & \vdots & \\ T(n-i) & = & T(n-i-1) & +1 \\ \vdots & & \vdots & \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \text{if } n > 1 \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{array}{ccccc}
T(n) & = & T(n-1) & +1 \\
T(n-1) & = & T(n-2) & +1 \\
\vdots & & \vdots & \\
T(n-i) & = & T(n-i-1) & +1 \\
\vdots & & \vdots & \\
T(2) & = & T(1) & +1
\end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \textit{if } \ n > \ 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{ccccc}
T(n) & = & T(n-1) & +1 \\
T(n-1) & = & T(n-2) & +1 \\
\vdots & & \vdots & \\
T(n-i) & = & T(n-i-1) & +1 \\
\vdots & & \vdots & \\
T(2) & = & T(1) & +1 \\
T(1) & = & & 0
\end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n-1) + 1, & \text{if } n > 1 \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n-0) & = & \cancel{T(n-1)} & +1 \\
\cancel{T(n-1)} & = & \cancel{T(n-2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n-i)} & = & \cancel{T(n-i-1)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0
\end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcrr}
T(n-0) & = & \cancel{T(n-1)} & +1 \\
\cancel{T(n-1)} & = & \cancel{T(n-2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n-i)} & = & \cancel{T(n-i-1)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0
\end{array}$$

What's the range for i?

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \textit{if } \; n > \; 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcrr}
T(n-0) & = & \cancel{T(n-1)} & +1 \\
\cancel{T(n-1)} & = & \cancel{T(n-2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n-i)} & = & \cancel{T(n-i-1)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \mathit{if}\ \ n > \ 1 \\ 0, & \mathit{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n-0) & = & \cancel{T(n-1)} & +1 \\
\cancel{T(n-1)} & = & \cancel{T(n-2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n-i)} & = & \cancel{T(n-i-1)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl}
n-i & = & 2 \\
i & = & n-2
\end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \text{if } n > 1 \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n-0) & = & \cancel{T(n-1)} & +1 \\
\cancel{T(n-1)} & = & \cancel{T(n-2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n-i)} & = & \cancel{T(n-i-1)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl} n - i & = & 2 \\ i & = & n-2 \end{array}$$

$$\Downarrow$$

So, $i \in [0, n-2]$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & if \;\; n > \;\; 1 \\ 0, & otherwise \end{cases}$$

$$\begin{array}{rcll}
T(n-0) & = & \cancel{T(n-1)} & +1 \\
\cancel{T(n-1)} & = & \cancel{T(n-2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n-i)} & = & \cancel{T(n-i-1)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0 \\
T(n) & = & \sum_{i=0}^{n-2} 1 & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl} n-i & = & 2 \\ i & = & n-2 \end{array}$$

$$\Downarrow$$

So, $i \in [0, n-2]$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n-0) & = & \cancel{T(n-1)} & +1 \\
\cancel{T(n-1)} & = & \cancel{T(n-2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n-i)} & = & \cancel{T(n-i-1)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0 \\
T(n) & = & \sum_{i=0}^{n-2} 1 & +0 \\
T(n) & = & n-2-0+1 & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl} n-i & = & 2 \\ i & = & n-2 \end{array}$$

$$\Downarrow$$

So, $i \in [0, n-2]$

Recurrences

$$T(n) = \begin{cases} T(n-1)+1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n-0) & = & \cancel{T(n-1)} & +1 \\
\cancel{T(n-1)} & = & \cancel{T(n-2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n-i)} & = & \cancel{T(n-i-1)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0 \\
T(n) & = & \sum_{i=0}^{n-2} 1 & +0 \\
T(n) & = & n-2-0+1 & +0 \\
T(n) & = & n-1 &
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl} n-i & = & 2 \\ i & = & n-2 \end{array}$$

$$\Downarrow$$

So, $i \in [0, n-2]$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & if\ \ n > \ 1 \\ 0, & otherwise \end{cases}$$

$$T(n) \quad = $$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{lcll} T(n) & = & T(n/2) & +1 \\ T(n/2) & = & T(n/4) & +1 \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{ccccc} T(n) & = & T(n/2) & & +1 \\ T(n/2) & = & T(n/4) & & +1 \\ \vdots & & \vdots & & \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{ccll} T(n) & = & T(n/2) & +1 \\ T(n/2) & = & T(n/4) & +1 \\ \vdots & & \vdots & \\ T(n/2^i) & = & T(n/2^{i+1}) & +1 \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll} T(n) & = & T(n/2) & +1 \\ T(n/2) & = & T(n/4) & +1 \\ \vdots & & \vdots & \\ T(n/2^i) & = & T(n/2^{i+1}) & +1 \\ \vdots & & \vdots & \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll} T(n) & = & T(n/2) & +1 \\ T(n/2) & = & T(n/4) & +1 \\ \vdots & & \vdots & \\ T(n/2^i) & = & T(n/2^{i+1}) & +1 \\ \vdots & & \vdots & \\ T(2) & = & T(1) & +1 \end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{ccccc}
T(n) & = & T(n/2) & +1 \\
T(n/2) & = & T(n/4) & +1 \\
\vdots & & \vdots & \\
T(n/2^i) & = & T(n/2^{i+1}) & +1 \\
\vdots & & \vdots & \\
T(2) & = & T(1) & +1 \\
T(1) & = & & 0
\end{array}$$

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{ccccc}
T(n/2^0) & = & \cancel{T(n/2^1)} & +1 \\
\cancel{T(n/2^1)} & = & \cancel{T(n/2^2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n/2^i)} & = & \cancel{T(n/2^{i+1})} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1 + 1 + \cdots + 1}_{?} & +0
\end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n/2^0) & = & \cancel{T(n/2^1)} & +1 \\
\cancel{T(n/2^1)} & = & \cancel{T(n/2^2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n/2^i)} & = & \cancel{T(n/2^{i+1})} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1 + 1 + \cdots + 1}_{?} & +0
\end{array}$$

What's the range for i?

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n/2^0) & = & \cancel{T(n/2^1)} & +1 \\
\cancel{T(n/2^1)} & = & \cancel{T(n/2^2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n/2^i)} & = & \cancel{T(n/2^{i+1})} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1 + 1 + \cdots + 1}_{?} & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n/2^0) & = & \cancel{T(n/2^1)} & +1 \\
\cancel{T(n/2^1)} & = & \cancel{T(n/2^2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n/2^i)} & = & \cancel{T(n/2^{i+1})} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1 + 1 + \cdots + 1}_{?} & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl}
n/2^i & = & 2 \\
2^{i+1} & = & n \\
i & = & \log_2 n - 1
\end{array}$$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n/2^0) & = & \cancel{T(n/2^1)} & +1 \\
\cancel{T(n/2^1)} & = & \cancel{T(n/2^2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n/2^i)} & = & \cancel{T(n/2^{i+1})} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1 + 1 + \cdots + 1}_{?} & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl}
n/2^i & = & 2 \\
2^{i+1} & = & n \\
i & = & \log_2 n - 1
\end{array}$$

$$\Downarrow$$

So, $i \in [0, \log_2 n - 1]$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \text{if } n > 1 \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{array}{rclr}
T(n/2^0) & = & \cancel{T(n/2^1)} & +1 \\
\cancel{T(n/2^1)} & = & \cancel{T(n/2^2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n/2^i)} & = & \cancel{T(n/2^{i+1})} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1+1+\cdots+1}_{?} & +0 \\
T(n) & = & \sum_{i=0}^{\log_2 n-1} 1 & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl}
n/2^i & = & 2 \\
2^{i+1} & = & n \\
i & = & \log_2 n - 1
\end{array}$$

$$\Downarrow$$

So, $i \in [0, \log_2 n - 1]$

Recurrences

$$T(n) = \begin{cases} T(n/2) + 1, & \textit{if } n > 1 \\ 0, & \textit{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n/2^0) & = & \cancel{T(n/2^1)} & +1 \\
\cancel{T(n/2^1)} & = & \cancel{T(n/2^2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n/2^i)} & = & \cancel{T(n/2^{i+1})} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1 + 1 + \cdots + 1}_{?} & +0 \\
T(n) & = & \sum_{i=0}^{\log_2 n - 1} 1 & +0 \\
T(n) & = & \log_2 n - 1 - 0 + 1 & +0
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl}
n/2^i & = & 2 \\
2^{i+1} & = & n \\
i & = & \log_2 n - 1
\end{array}$$

$$\Downarrow$$

So, $i \in [0, \log_2 n - 1]$

$$T(n) = \begin{cases} T(n/2) + 1, & \text{if } n > 1 \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{array}{rcll}
T(n/2^0) & = & \cancel{T(n/2^1)} & +1 \\
\cancel{T(n/2^1)} & = & \cancel{T(n/2^2)} & +1 \\
\vdots & & \vdots & \\
\cancel{T(n/2^i)} & = & \cancel{T(n/2^{i+1})} & +1 \\
\vdots & & \vdots & \\
\cancel{T(2)} & = & \cancel{T(1)} & +1 \\
\cancel{T(1)} & = & & 0 \\
\hline
T(n) & = & \underbrace{1 + 1 + \cdots + 1}_{?} & +0 \\
T(n) & = & \sum_{i=0}^{\log_2 n - 1} 1 & +0 \\
T(n) & = & \log_2 n - 1 - 0 + 1 & +0 \\
T(n) & = & \log_2 n &
\end{array}$$

What's the range for i?

$$i \in [0, x]$$

$$\Downarrow$$

$$\begin{array}{rcl}
n/2^i & = & 2 \\
2^{i+1} & = & n \\
i & = & \log_2 n - 1
\end{array}$$

$$\Downarrow$$

So, $i \in [0, \log_2 n - 1]$

Teoria dos Grafos e Computabilidade

— Asymptotic Order of Growth —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Feb 2025

Upper and Lower Bounds

Asymptotic lower bound: A function $f(n)$ is $\Omega(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $f(n) \geq cg(n)$.

Upper and Lower Bounds

Asymptotic lower bound: A function $f(n)$ is $\Omega(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $f(n) \geq cg(n)$.

Asymptotic upper bound: A function $f(n)$ is $O(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $f(n) \leq cg(n)$.

Upper and Lower Bounds

Asymptotic lower bound: A function $f(n)$ is $\Omega(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $f(n) \geq cg(n)$.

Asymptotic upper bound: A function $f(n)$ is $O(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $f(n) \leq cg(n)$.

Asymptotic tight bound: A function $f(n)$ is $\Theta(g(n))$ if $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$.

Upper and Lower Bounds

Asymptotic lower bound: A function $f(n)$ is $\Omega(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $f(n) \geq cg(n)$.

Asymptotic upper bound: A function $f(n)$ is $O(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, we have $f(n) \leq cg(n)$.

Asymptotic tight bound: A function $f(n)$ is $\Theta(g(n))$ if $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$.

- In these definitions, c is a constant independent of n.
- Abuse of notation: say $g(n) = O(f(n))$, $g(n) = \Omega(f(n))$, $g(n) = \Theta(f(n))$.

Properties of Asymptotic Growth Rates

Transitivity

- If $f = O(g)$ and $g = O(h)$, then $f = O(h)$.
- If $f = \Omega(g)$ and $g = \Omega(h)$, then $f = \Omega(h)$.
- If $f = \Theta(g)$ and $g = \Theta(h)$, then $f = \Theta(h)$.

Additivity

- If $f = O(h)$ and $g = O(h)$, then $f + g = O(h)$.
- Similar statements hold for lower and tight bounds.

Properties of Asymptotic Growth Rates

Transitivity

- If $f = O(g)$ and $g = O(h)$, then $f = O(h)$.
- If $f = \Omega(g)$ and $g = \Omega(h)$, then $f = \Omega(h)$.
- If $f = \Theta(g)$ and $g = \Theta(h)$, then $f = \Theta(h)$.

Additivity

- If $f = O(h)$ and $g = O(h)$, then $f + g = O(h)$.
- Similar statements hold for lower and tight bounds.
- If k is a constant and there are k functions $f_i = O(h), 1 \leq i \leq k$,

Properties of Asymptotic Growth Rates

Transitivity

- If $f = O(g)$ and $g = O(h)$, then $f = O(h)$.
- If $f = \Omega(g)$ and $g = \Omega(h)$, then $f = \Omega(h)$.
- If $f = \Theta(g)$ and $g = \Theta(h)$, then $f = \Theta(h)$.

Additivity

- If $f = O(h)$ and $g = O(h)$, then $f + g = O(h)$.
- Similar statements hold for lower and tight bounds.
- If k is a constant and there are k functions $f_i = O(h), 1 \leq i \leq k$, then $f_1 + f_2 + \ldots + f_k = O(h)$.
- If $f = O(g)$, then $f + g =$

Properties of Asymptotic Growth Rates

Transitivity

- If $f = O(g)$ and $g = O(h)$, then $f = O(h)$.
- If $f = \Omega(g)$ and $g = \Omega(h)$, then $f = \Omega(h)$.
- If $f = \Theta(g)$ and $g = \Theta(h)$, then $f = \Theta(h)$.

Additivity

- If $f = O(h)$ and $g = O(h)$, then $f + g = O(h)$.
- Similar statements hold for lower and tight bounds.
- If k is a constant and there are k functions $f_i = O(h), 1 \leq i \leq k$, then $f_1 + f_2 + \ldots + f_k = O(h)$.
- If $f = O(g)$, then $f + g = \Theta(g)$.

Properties of Asymptotic Growth Rates

Transitivity

- If $f = O(g)$ and $g = O(h)$, then $f = O(h)$.
- If $f = \Omega(g)$ and $g = \Omega(h)$, then $f = \Omega(h)$.
- If $f = \Theta(g)$ and $g = \Theta(h)$, then $f = \Theta(h)$.

Additivity

- If $f = O(h)$ and $g = O(h)$, then $f + g = O(h)$.
- Similar statements hold for lower and tight bounds.
- If k is a constant and there are k functions $f_i = O(h), 1 \leq i \leq k$, then $f_1 + f_2 + \ldots + f_k = O(h)$.
- If $f = O(g)$, then $f + g = \Theta(g)$.

Prove that the properties for O are true!!!

Examples

- $f(n) = pn^2 + qn + r$ is

Examples

- $f(n) = pn^2 + qn + r$ is $\theta(n^2)$. Can ignore lower order terms.

Examples

- $f(n) = pn^2 + qn + r$ is $\theta(n^2)$. Can ignore lower order terms.
- Is $f(n) = pn^2 + qn + r = O(n^3)$?

Examples

- $f(n) = pn^2 + qn + r$ is $\theta(n^2)$. Can ignore lower order terms.
- Is $f(n) = pn^2 + qn + r = O(n^3)$?
- $f(n) = \sum_{0 \le i \le d} a_i n^i =$

Examples

- $f(n) = pn^2 + qn + r$ is $\theta(n^2)$. Can ignore lower order terms.
- Is $f(n) = pn^2 + qn + r = O(n^3)$?
- $f(n) = \sum_{0 \leq i \leq d} a_i n^i = O(n^d)$, if $d > 0$ is an integer constant and $a_d > 0$. Definition of polynomial time

Examples

- $f(n) = pn^2 + qn + r$ is $\theta(n^2)$. Can ignore lower order terms.
- Is $f(n) = pn^2 + qn + r = O(n^3)$?
- $f(n) = \sum_{0 \leq i \leq d} a_i n^i = O(n^d)$, if $d > 0$ is an integer constant and $a_d > 0$. Definition of polynomial time
- Is an algorithm with running time $O(n^{1.59})$ a polynomial-time algorithm?

Examples

- $f(n) = pn^2 + qn + r$ is $\theta(n^2)$. Can ignore lower order terms.
- Is $f(n) = pn^2 + qn + r = O(n^3)$?
- $f(n) = \sum_{0 \leq i \leq d} a_i n^i = O(n^d)$, if $d > 0$ is an integer constant and $a_d > 0$. Definition of **polynomial time**
- Is an algorithm with running time $O(n^{1.59})$ a polynomial-time algorithm?
- $O(\log_a n) = O(\log_b n)$ for any pair of constants $a, b > 1$.
- For every $x > 0$, $\log n = O(n^x)$.

Examples

- $f(n) = pn^2 + qn + r$ is $\theta(n^2)$. Can ignore lower order terms.
- Is $f(n) = pn^2 + qn + r = O(n^3)$?
- $f(n) = \sum_{0 \leq i \leq d} a_i n^i = O(n^d)$, if $d > 0$ is an integer constant and $a_d > 0$. Definition of polynomial time
- Is an algorithm with running time $O(n^{1.59})$ a polynomial-time algorithm?
- $O(\log_a n) = O(\log_b n)$ for any pair of constants $a, b > 1$.
- For every $x > 0$, $\log n = O(n^x)$.
- For every $r > 1$ and every $d > 0$, $n^d = O(r^n)$.

Teoria dos Grafos e Computabilidade

— Common Running Times —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Linear Time

- Running time is at most a constant factor times the size of the input.

Linear Time

- Running time is at most a constant factor times the size of the input.
- Finding the minimum, merging two sorted lists.

Linear Time

- Running time is at most a constant factor times the size of the input.
- Finding the minimum, merging two sorted lists.
- Sub-linear time.

Linear Time

- Running time is at most a constant factor times the size of the input.
- Finding the minimum, merging two sorted lists.
- Sub-linear time. Binary search in a sorted array of n numbers takes $O(\log n)$ time.

$O(n \log n)$ Time

- Any algorithm where the costliest step is sorting.

Quadratic Time

- Enumerate all pairs of elements.

Quadratic Time

- Enumerate all pairs of elements.
- Given a set of n points in the plane, find the pair that are the closest.

Quadratic Time

- Enumerate all pairs of elements.
- Given a set of n points in the plane, find the pair that are the closest. Surprising fact: can solve this problem in $O(n \log n)$ time later in the semester.

$O(n^k)$ Time

- Does a graph have an independent set of size k, where k is a constant, i.e. there are k nodes such that no two are joined by an edge?

$O(n^k)$ Time

- Does a graph have an independent set of size k, where k is a constant, i.e. there are k nodes such that no two are joined by an edge?
- Algorithm: For each subset S of k nodes, check if S is an independent set. If the answer is yes, report it.

$O(n^k)$ Time

- Does a graph have an independent set of size k, where k is a constant, i.e. there are k nodes such that no two are joined by an edge?
- Algorithm: For each subset S of k nodes, check if S is an independent set. If the answer is yes, report it.
- Running time is

$O(n^k)$ Time

- Does a graph have an independent set of size k, where k is a constant, i.e. there are k nodes such that no two are joined by an edge?
- Algorithm: For each subset S of k nodes, check if S is an independent set. If the answer is yes, report it.
- Running time is $O(k^2\binom{n}{k}) = O(n^k)$.

Beyond Polynomial Time

- What is the largest size of an independent set in a graph with n nodes?

- What is the largest size of an independent set in a graph with n nodes?
- Algorithm: For each $1 \leq i \leq n$, check if the graph has an independent size of size i. Output largest independent set found.

Beyond Polynomial Time

- What is the largest size of an independent set in a graph with n nodes?
- Algorithm: For each $1 \leq i \leq n$, check if the graph has an independent size of size i. Output largest independent set found.
- What is the running time?

Beyond Polynomial Time

- What is the largest size of an independent set in a graph with n nodes?
- Algorithm: For each $1 \leq i \leq n$, check if the graph has an independent size of size i. Output largest independent set found.
- What is the running time? $O(n^2 2^n)$.