



Curso **Discussão** Progresso Anotações Calendário Datas

☰ Todos os tópicos

Adicionar uma publicação

Pesquisar todas as postagens **Busca**

O acesso à auditoria expira a partir de 5 de 2020

Você perde todo o acesso a este curso, incluindo seu progresso, a partir de 5 de 2020 .

[Tutorial] Classes de complexidade e seus gráficos: nos bastidores

discussão postada about 21 hours ago por [Kiara-Elizabeth](#) (Community TA)

Computational Complexity! : D



COMPLEXITY CLASSES AND THEIR GRAPHS

Bem-vinda

Oi! Bem-vindo a esta seção sobre **Classes de complexidade e seus gráficos** .
Primeiro de tudo, **parabéns por essas semanas de trabalho duro !!** :) Você fez um trabalho fantástico e estamos tão felizes que você chegou até aqui.

- Você aprendeu na palestra que os algoritmos têm diferentes tempos de execução e como as ordens de crescimento indicam a eficiência de um algoritmo, considerando dados muito grandes. A ordem do crescimento é uma

aproximação, um limite superior de quanto um algoritmo pode levar para executar, o pior cenário.

- Primeiro, teremos uma rápida introdução aos gráficos 2D e, em seguida, discutiremos e ilustraremos a lógica por trás de cada complexidade que você aprendeu (constante, logarítmica, linear, loglinear, polinomial e exponencial).

O gráfico

As classes de complexidade são representadas com **um gráfico** .

Os gráficos têm dois eixos:

- O **eixo X** , uma linha horizontal (representada abaixo com uma linha laranja).
- O **eixo Y** , uma linha vertical (representada abaixo com uma linha verde).

Cada ponto no gráfico possui duas coordenadas, uma coordenada **x** e uma **coordenada y**, e o próprio ponto é representado como `(<x-coordinate>, <y-coordinate>)` .

- O **ponto em que o eixo x e o eixo y se cruzam** é a **origem** , o ponto **(0, 0)** .
- Começamos a contar **números positivos** da origem para a direita no eixo X e da origem acima no eixo Y.
- Começamos a contar **números negativos** da origem à esquerda no eixo X e da origem abaixo no eixo Y. (Você pode visualizar isso no gráfico em branco abaixo do primeiro diagrama)

Eixo X

- No nosso caso, para complexidade algorítmica, os números no eixo x representam o tamanho da entrada.

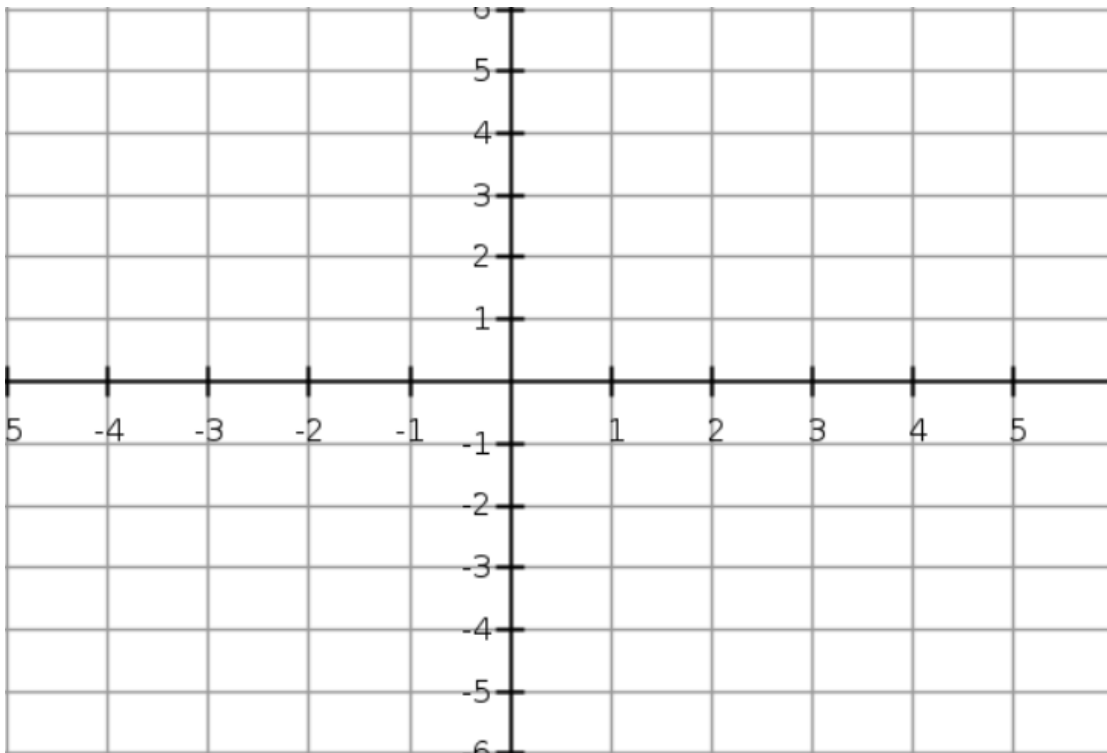
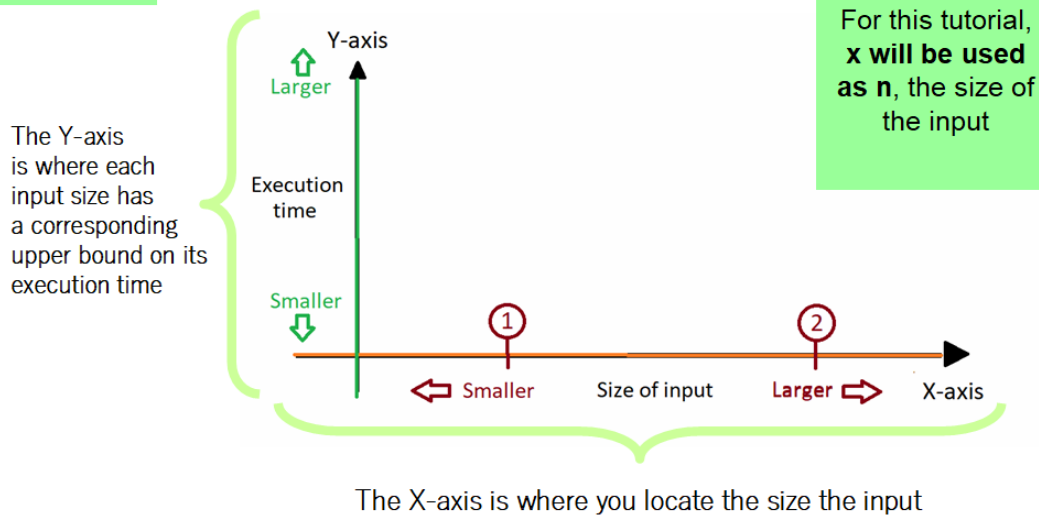
Eixo Y

- No nosso caso, para complexidade algorítmica, os números no eixo y representam o limite superior para o tempo de execução de um algoritmo, considerando o tamanho da entrada que corresponde a esse valor y no eixo x.
-

The Graph! : D



• Elements



Mais sobre gráficos

Um gráfico bidimensional (bidimensional) é uma linha (pode ser uma curva).

- Cada ponto nesta linha tem duas coordenadas (x, y) . Para nossos propósitos, **cada tamanho de entrada no eixo x terá uma coordenada y correspondente**, o limite superior em seu tempo de execução e será representado como $(\text{inputSize}, \text{executionTime})$.

No exemplo abaixo, temos um ponto que é representado como um ponto muito grosso. Este ponto tem duas coordenadas $(\text{input1}, a)$.

Vamos verificar isso:

- Se traçarmos uma linha vertical do ponto ao eixo x, alcançaremos `input1` (representado como uma bolha vermelha para fins de ilustração). Esta entrada1 para este exemplo é um tamanho de entrada arbitrário.
- Se traçarmos uma linha horizontal do ponto ao eixo y, alcançaremos `a`.

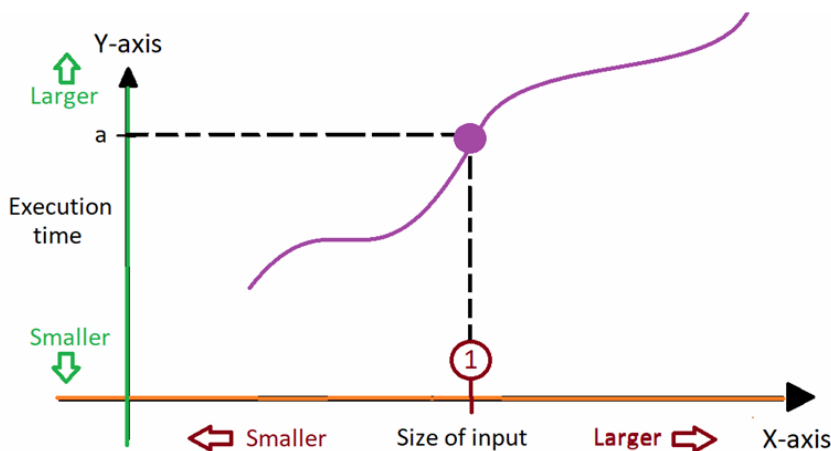
Determinamos que este ponto possui essas coordenadas `(input1, a)`.


Os mesmos princípios se aplicam a cada ponto do gráfico e é assim que obtemos uma linha contínua.

The Graph! : D



- Each element on the X-axis has an element on the Y-axis



In this case  has a corresponding upper bound on execution time `a`

Como ler um gráfico

Como começamos na origem, (0, 0) onde os dois eixos se cruzam, **os números nos eixos x e y aumentam à medida que avançamos para a direita e para cima, respectivamente.**

- Se considerarmos dois tamanhos de entrada no eixo x, o da esquerda será o menor e o da direita será o maior.
- Se tomarmos dois tempos de execução no eixo y, o que está localizado mais alto é o maior e o abaixo é o menor.

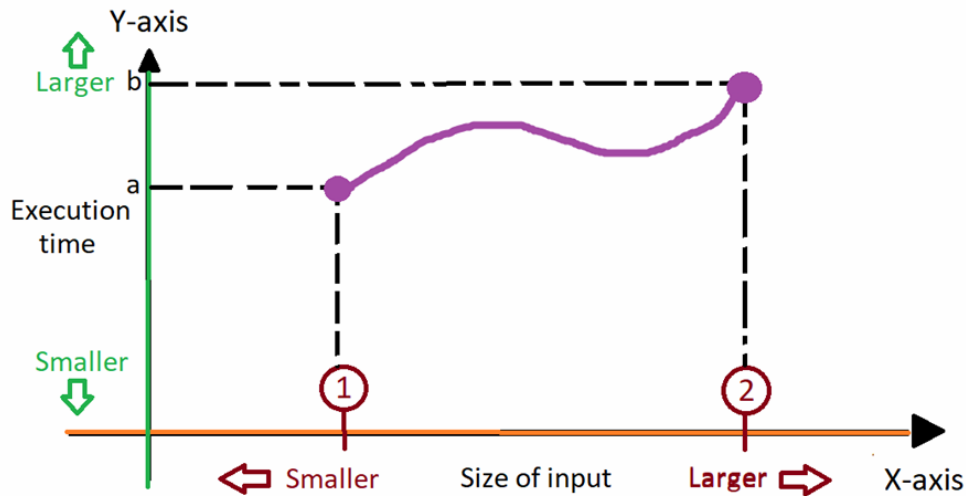
Nesse caso, pegamos `input1` e `input2` (as duas bolhas vermelhas no eixo x) e encontramos seus pontos correspondentes na linha (representados como pontos grossos para fins de ilustração). Esses pontos têm `a` e `b` como coordenadas y, respectivamente.

- O que o gráfico está nos dizendo é que `input1` é menor em tamanho `input2` e `input1` leva menos tempo para ser executado do que `input2` porque sua coordenada y é menor.

How to Read a Graph



• Example



Classes de complexidade no contexto!

Agora vamos visualizar e analisar cada uma das complexidades algorítmicas que iremos estudar.

Complexidade constante

Este é o nosso melhor amigo! Algoritmos com essa complexidade são **os mais eficientes**. Um algoritmo com essa complexidade sempre leva aproximadamente a **mesma quantidade de tempo / etapas para executar, independentemente do tamanho da entrada**.

Como você pode ver nos gráficos abaixo, temos `input1` e `input2` (as duas bolhas vermelhas no eixo x).

Um gráfico de complexidade constante é representado como **uma linha horizontal (linha azul abaixo) porque, à medida que a entrada aumenta, o tempo necessário para executar é aproximadamente o mesmo**.

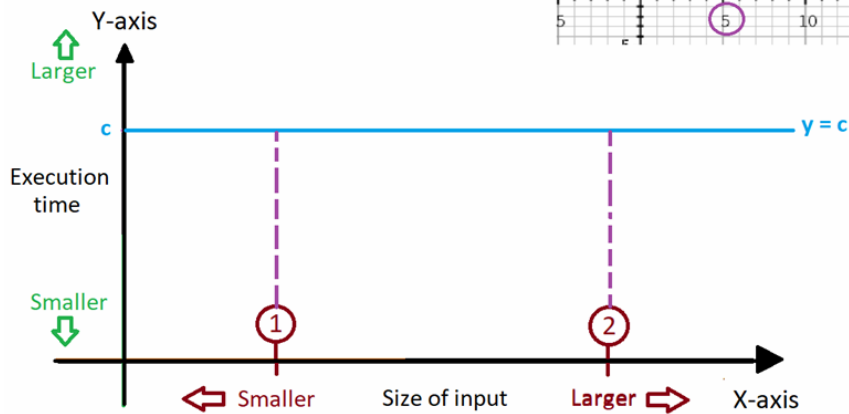
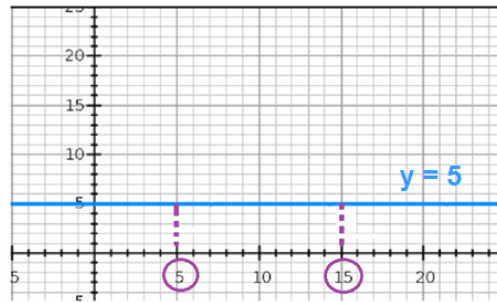
No gráfico no canto superior direito do diagrama, você pode ver um exemplo de um gráfico constante $y = 5$.

NOTA: Esses gráficos auxiliares no canto superior direito dos diagramas (fornecidos por [graphsketch.com](https://www.graphsketch.com)) ilustram o quanto a coordenada y muda em cada gráfico de complexidade quando triplicamos o tamanho da entrada. **Os números nos eixos xey (5 e 15 neste caso) não estão dentro do contexto de algoritmos. Estes são exemplos gerais desses tipos de funções representadas matematicamente)**

Constant Complexity $O(1)$



Always takes the same amount of time to execute no matter the size of the input



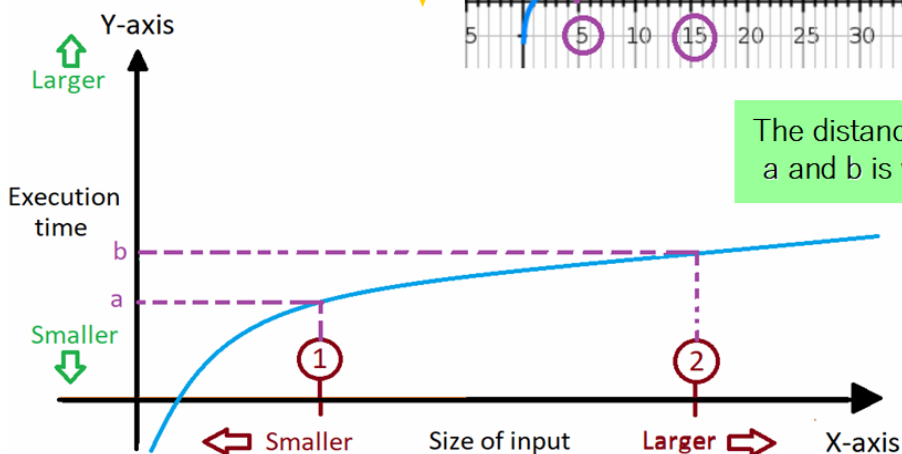
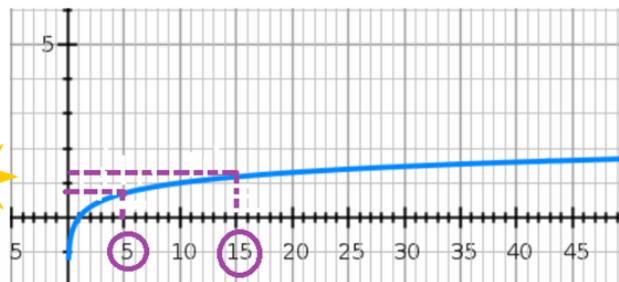
Complexidade logarítmica

Essa complexidade também é grande, o tempo de execução aumenta lentamente com o tamanho da entrada. Você pode ver como, se triplicarmos a entrada (5 a 15), a alteração na coordenada y (tempo de execução) é muito pequena, portanto o algoritmo é muito eficiente.

Logarithmic Complexity $O(\log(n))$



Execution time increases slowly with input size



The distance between a and b is very small

Complexidade Linear

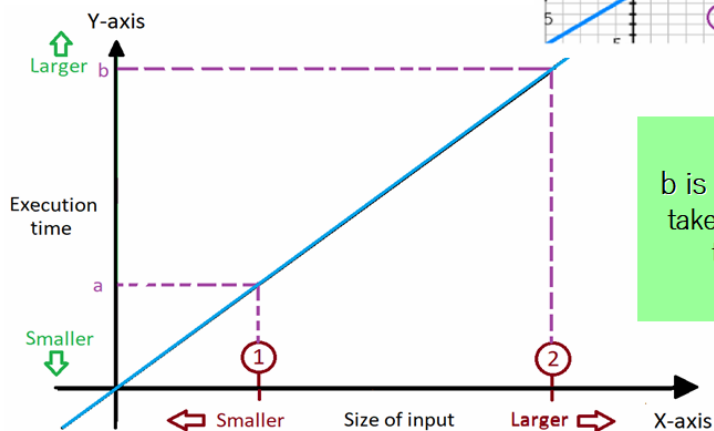
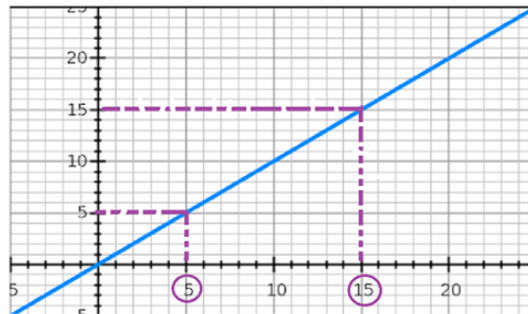
Agora vamos à complexidade linear. Essa complexidade começa a ser menos eficiente porque, se multiplicarmos o tamanho da entrada por uma constante (por exemplo, duplicar, triplicar o tamanho da entrada), o algoritmo levará vezes em dobro, em triplo ou em c (ca constante) o tempo necessário para n (o tamanho da entrada inicial) a ser executado.

Linear Complexity $O(n)$



The execution time grows linearly with input size.

(e.g If you triple the input size, the execution time will approximately triple)



b is larger than a , larger inputs take more time to be executed than the smaller inputs

Complexidade linear

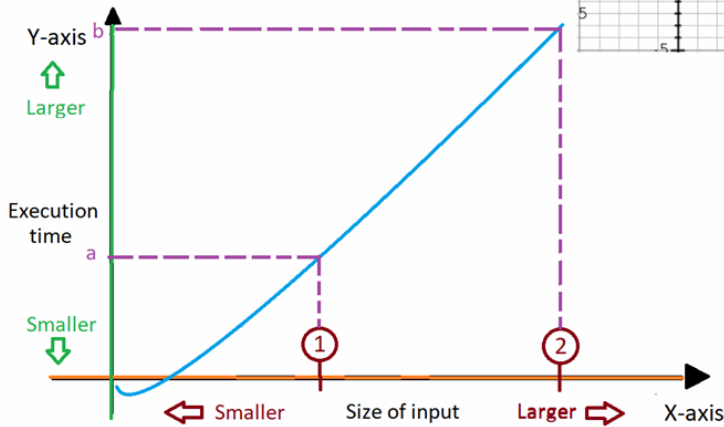
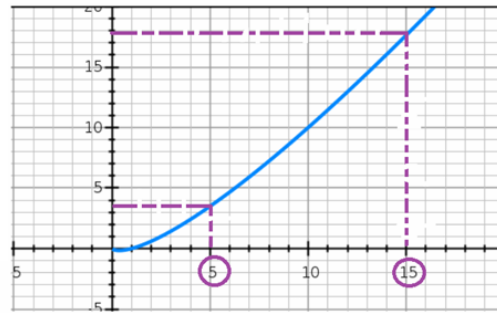
Essa complexidade começa a ser ineficiente, pois o tempo de execução cresce mais rápido do que linearmente. Você pode ver no gráfico que, se triplicarmos o tamanho da entrada (de 5 a 15), a coordenada y será um pouco mais do que tripla.

LogLinear Complexity $O(n \log(n))$



The execution time grows faster than linearly with input size.

(e.g If you triple the input size, the execution time will approximately be more than triple)



Complexidade polinomial

Algoritmos com essa complexidade são muito ineficientes. Quanto maior o valor de c (a constante na qual aumentamos o tamanho da entrada), mais tempo leva para executar.

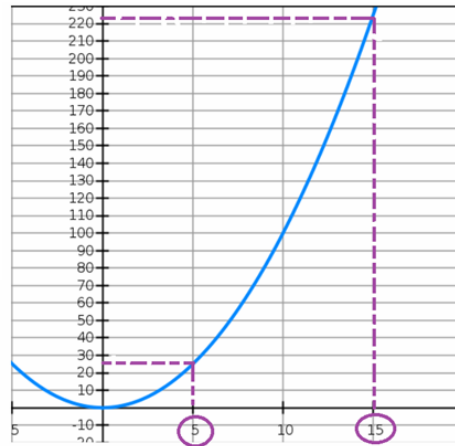
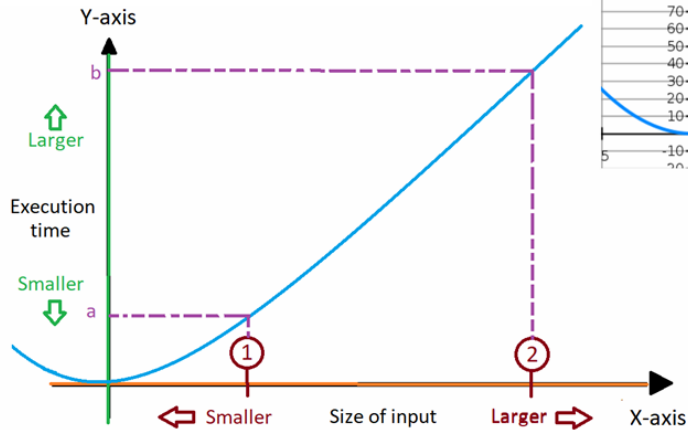
Nesse caso, temos um exemplo onde está $c = 2$ o gráfico $y = x^2$. Você pode ver que a entrada é quadrada e isso nos dá a coordenada y (tempo de execução), para que o gráfico cresça extremamente rápido e ilustra por que esse algoritmo leva muito para ser executado em entradas grandes.

Polynomial Complexity $O(n^c)$



The execution time grows much faster with input size.

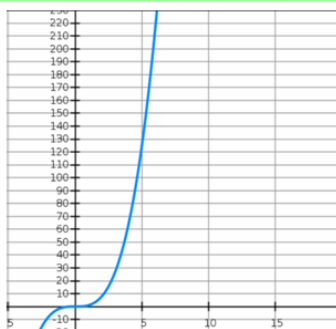
(e.g Larger the exponent, larger execution time)



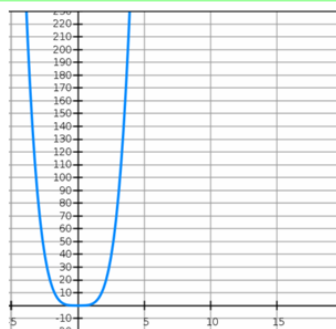
To illustrate this type of graph $y = x^2$ is used for this example.

Aqui você pode ver mais alguns gráficos polinomiais. Observe como eles crescem muito mais rápido à medida que aumentamos o valor de c , o expoente.

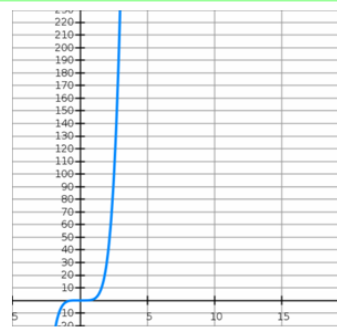
More Polynomial Graphs $O(n^c)$



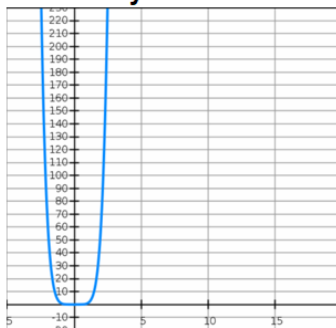
$$y = x^3$$



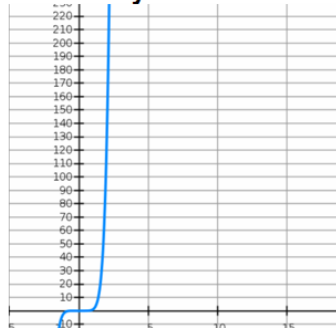
$$y = x^4$$



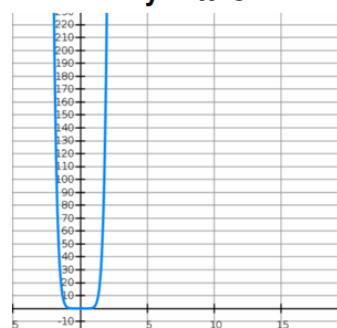
$$y = x^5$$



$$y = x^6$$



$$y = x^7$$



$$y = x^8$$

Complexidade Exponencial

AVISO! Algoritmos com complexidade exponencial levam uma quantidade extremamente grande de tempo para serem executados em entradas grandes.

Você pode ver como o tempo de execução no gráfico abaixo cresce de um número muito pequeno no eixo y para uma entrada de 5 a mais de 30000 quando triplicamos o tamanho da entrada.

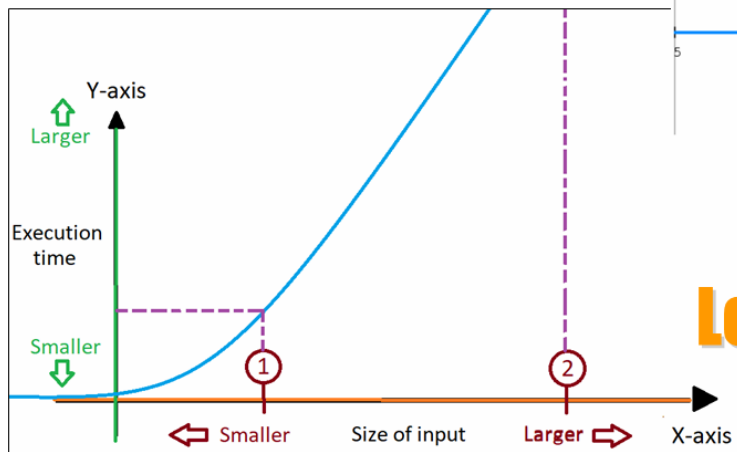
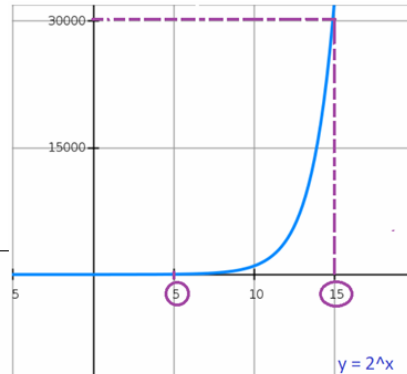
Maior a base (c), maior tempo de execução.

Exponential Complexity $O(c^x)$



The execution time grows much much much faster with input size!!!

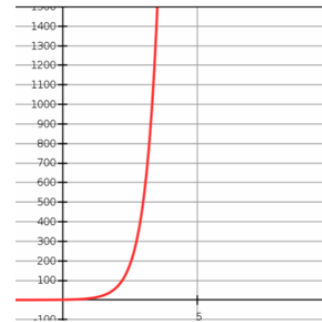
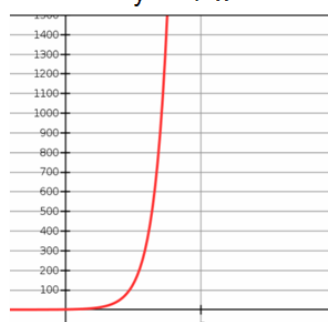
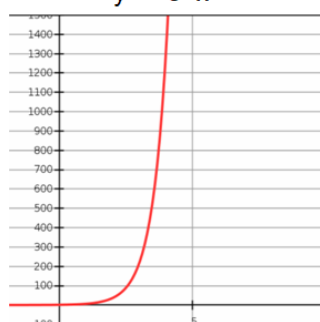
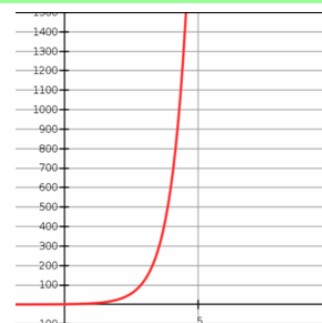
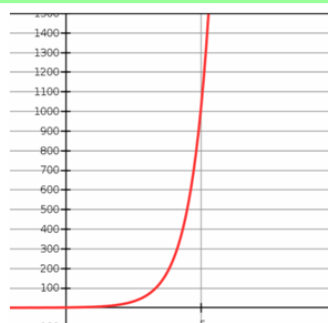
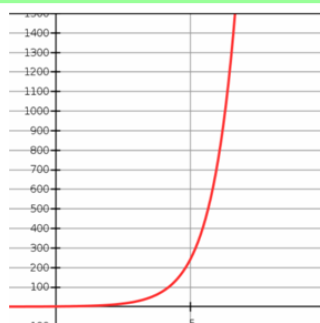
(e.g Larger base (c), longer execution time)



Warning!
Less efficient

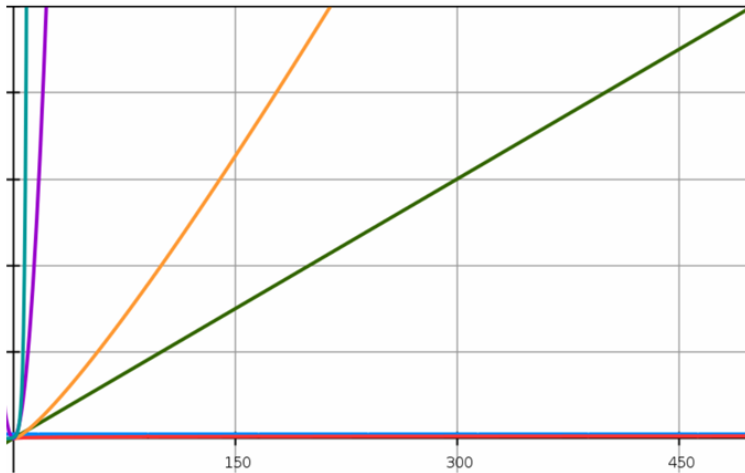
Aqui você pode ver mais gráficos exponenciais e como a base fica maior, o tempo de execução também cresce mais rápido.

More Exponential Graphs $O(c^n)$



Aqui estão todos os gráficos usados neste tutorial juntos, para que você possa ver a diferença entre cada classe de complexidade e como seus gráficos crescem com o tamanho da entrada.

All graphs together!



Graphs

- $y = 5$
- $y = \log(x)$
- $y = x$
- $y = x \cdot \log(x)$
- $y = x^2$
- $y = 2^x$

Espero que ajude!

Se você tiver alguma dúvida, não hesite em perguntar nos fóruns ou logo abaixo deste post. Os ATs da comunidade e seus colegas de classe estarão lá para ajudá-lo:)

Stephanie.

Relacionado a: [Aula 11 / Vídeo: Classes de Complexidade](#)

Esta postagem é visível para todos.

[Adicionar uma resposta](#)

1 resposta

[leopold_bernard_leo](#)

about 21 hours ago

obrigado !



Adicione um comentário

Exibindo todas as respostas

Adicione uma resposta:

Pré-visualizar

Enviar

Filtrar tópicos

Todas as Discussões

★ Publicações que estou seguindo

Geral

Instalar

Motoniveladoras

Recursos

Faça o download do Python

Filosofia

Lendo

Vídeo: Variáveis Globais

Leitura 1

Vídeo: Conhecimento

Vídeo: Idiomas

Exercício 7

[Sobre](#)
[edX for Business](#)

Legal

[Termos de Serviço e Código de Honra](#)
[Política de Privacidade](#)
[Política de Acessibilidade](#)

Conectar

[Blog](#)
[Contate-nos](#)
[Central de ajuda](#)



2020 edX Inc. Todos os direitos © reservados.
Karen Yu Bo Technology Co., Ltd. [ICP No. 17044299-2 de Guangdong](#)

