UFMS DIGITAL

Algoritmos e Programação II

Prof. MSc. Samuel Benjoino Ferraz Aquino

UFMS DIGITAL

Módulo 5 - Noções de eficiência e complexidade de algoritmos

Prof. MSc. Samuel Benjoino Ferraz Aquino

Roteiro

- Unidade 1: Noções de eficiência de algoritmos: pior e melhor caso
- Unidade 2: Complexidade assintótica
- Resumo

UFMS DIGITAL

Unidade I

Noções de eficiência de algoritmos: pior e melhor caso

Noções de Eficiência

- Tempo de execução é uma característica importante de um algoritmo

- Como medir?
  - Diferentes máquinas
  - Diferentes entradas

Noções de Eficiência

- Nosso objetivo: encontrar uma maneira analítica de medir o tempo de execução de um algoritmo
- Independente de máquina
- Análise de *complexidade*

Noções de Eficiência

- A *complexidade* de um algoritmo reflete seu comportamento e desempenho em cenários específicos
  - Complexidade de *melhor caso*
  - Complexidade de *pior caso*
  - Complexidade de *caso médio*

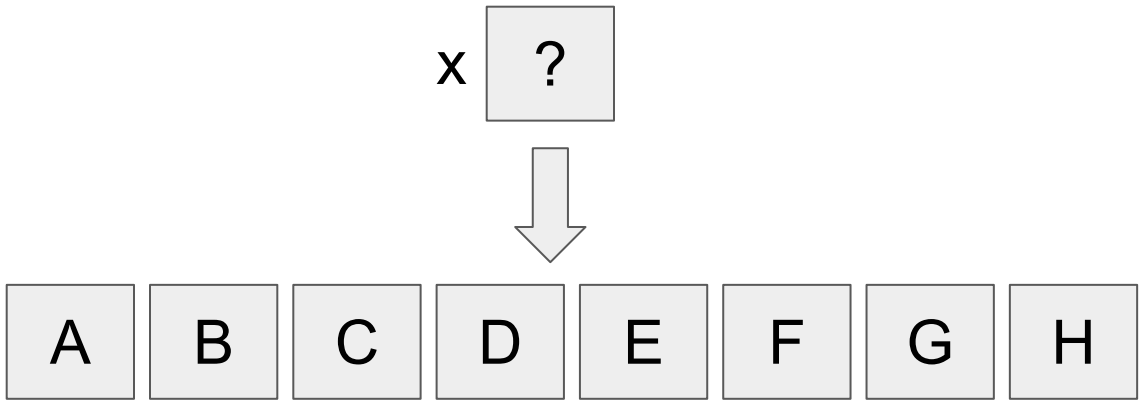
- Determinada em função do algoritmo e do padrão das suas entradas

Complexidade de melhor caso

- Comportamento do algoritmo no melhor cenário possível
- Padrão de entrada que gera a menor quantidade de iterações do algoritmo
- Exemplos: busca sequencial, busca binária e algoritmos de ordenação

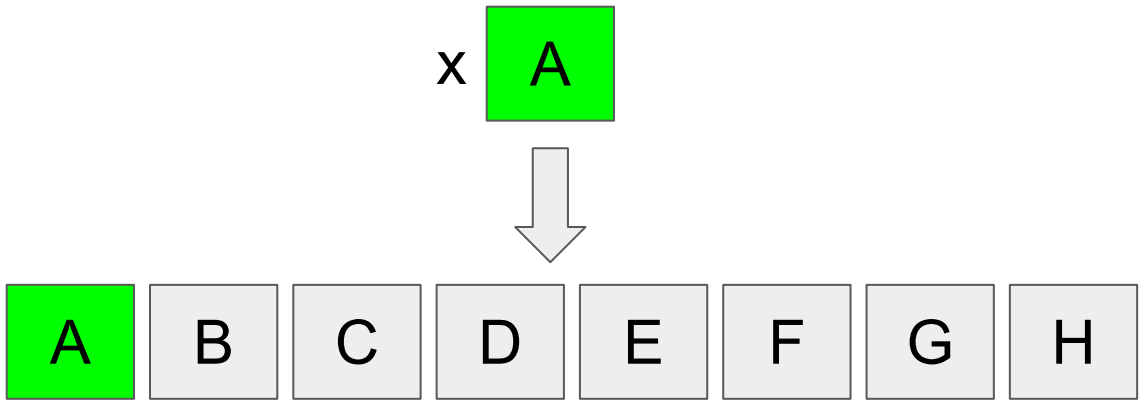
Complexidade de melhor caso

- Busca sequencial
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?



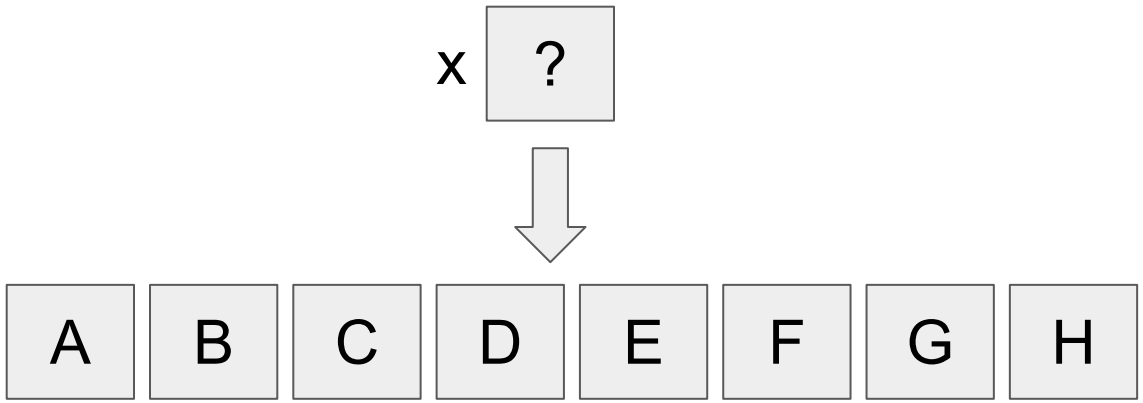
Complexidade de melhor caso

- Busca sequencial
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - *x* na primeira posição!



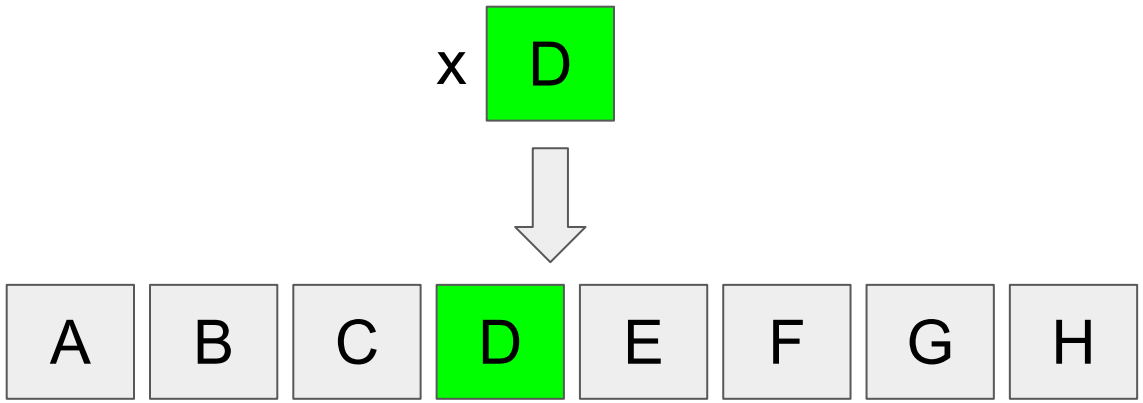
Complexidade de melhor caso

- Busca binária
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?



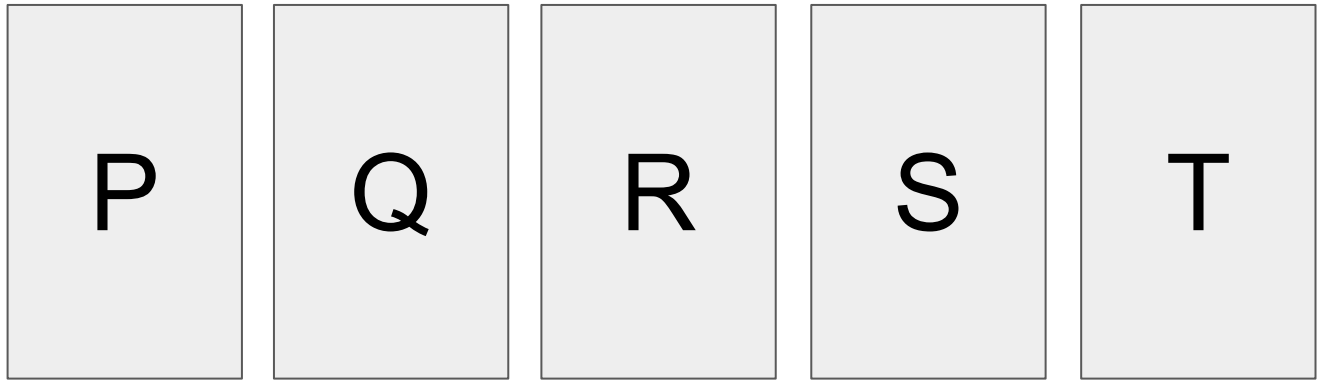
Complexidade de melhor caso

- Busca binária
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - *x* no meio do vetor!



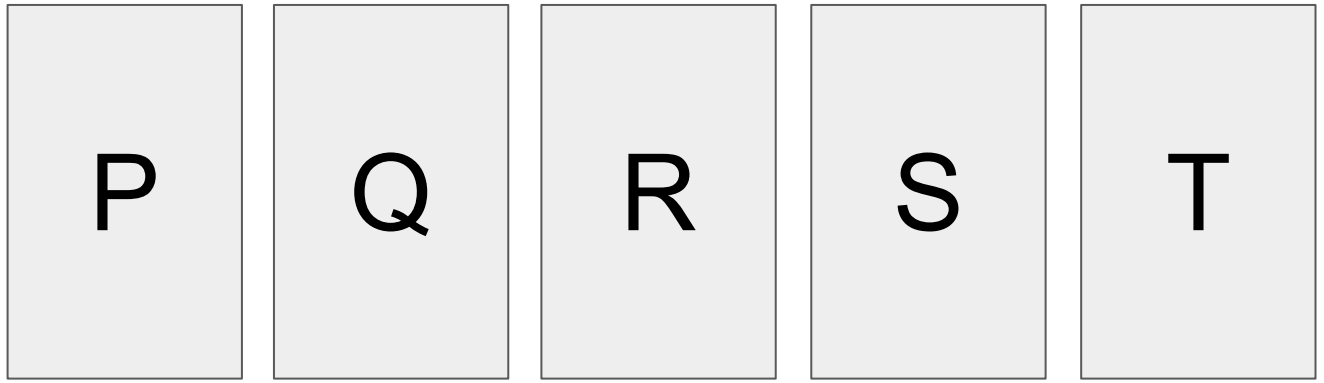
Complexidade de melhor caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?



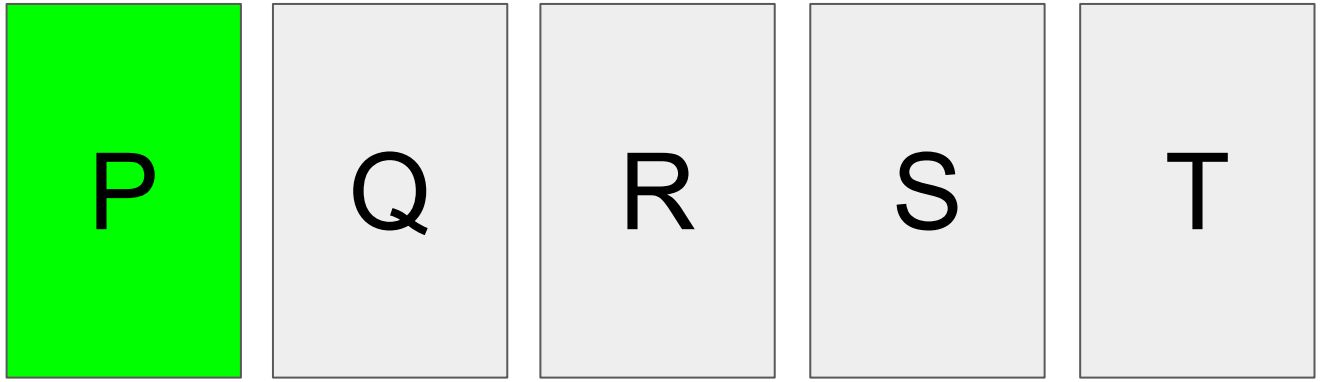
Complexidade de melhor caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor já estiver ordenado!



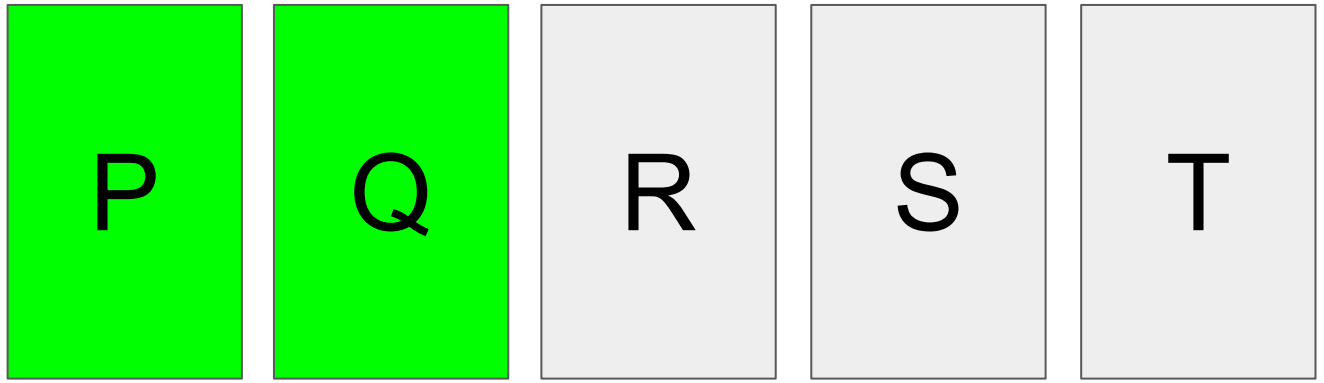
Complexidade de melhor caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor já estiver ordenado!



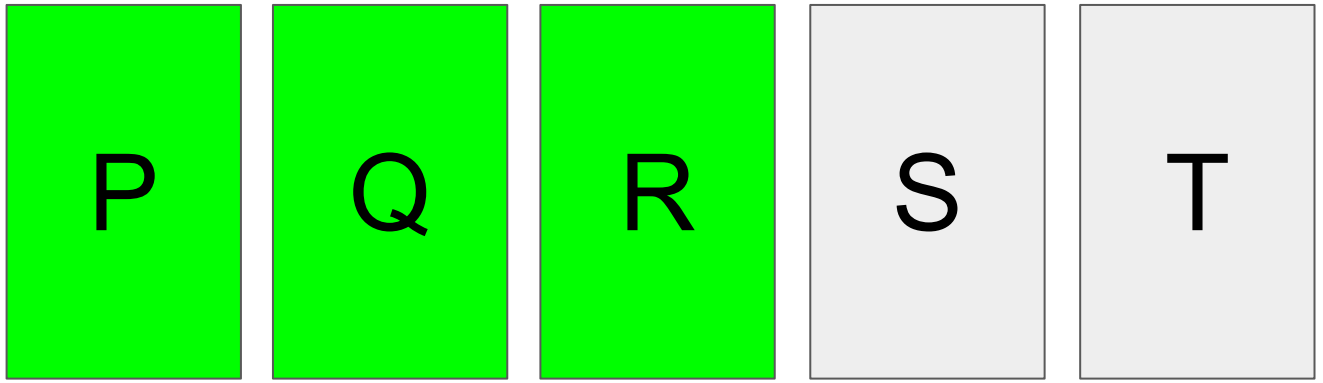
Complexidade de melhor caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor já estiver ordenado!



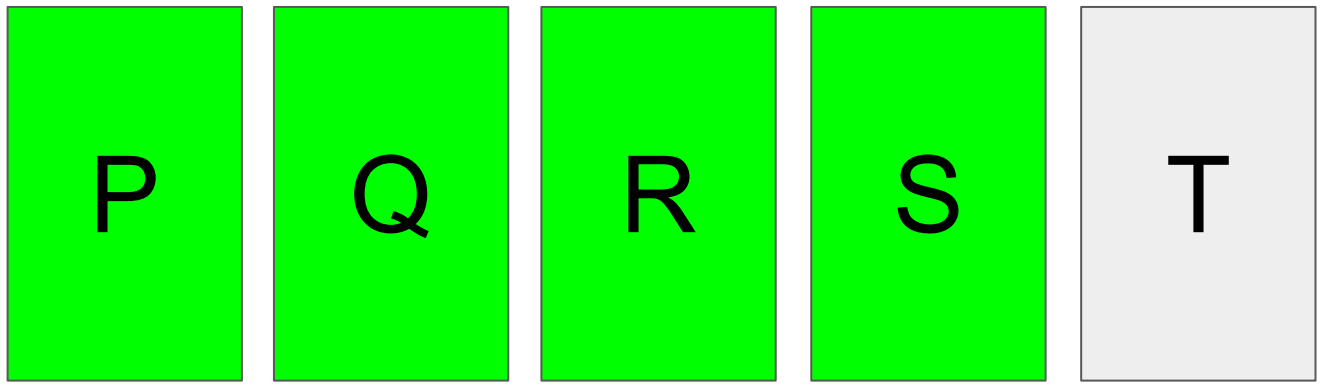
Complexidade de melhor caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor já estiver ordenado!



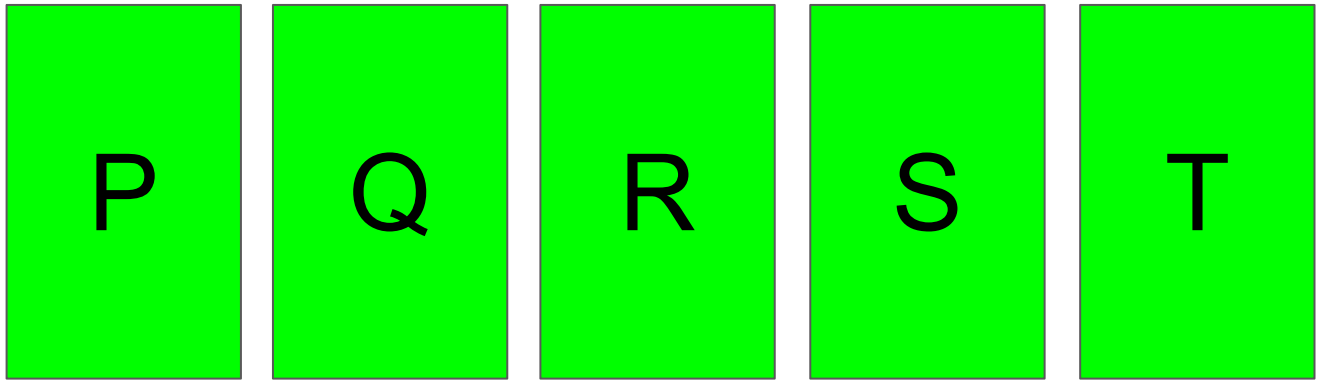
Complexidade de melhor caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor já estiver ordenado!



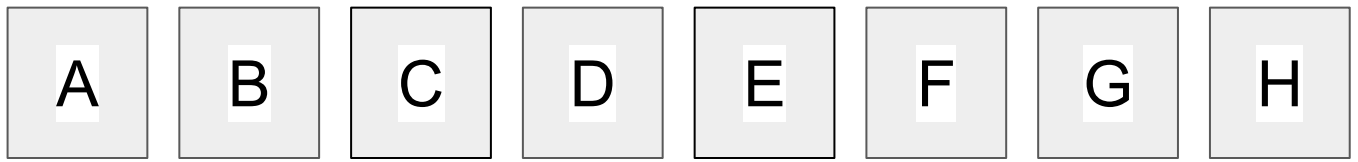
Complexidade de melhor caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor já estiver ordenado!



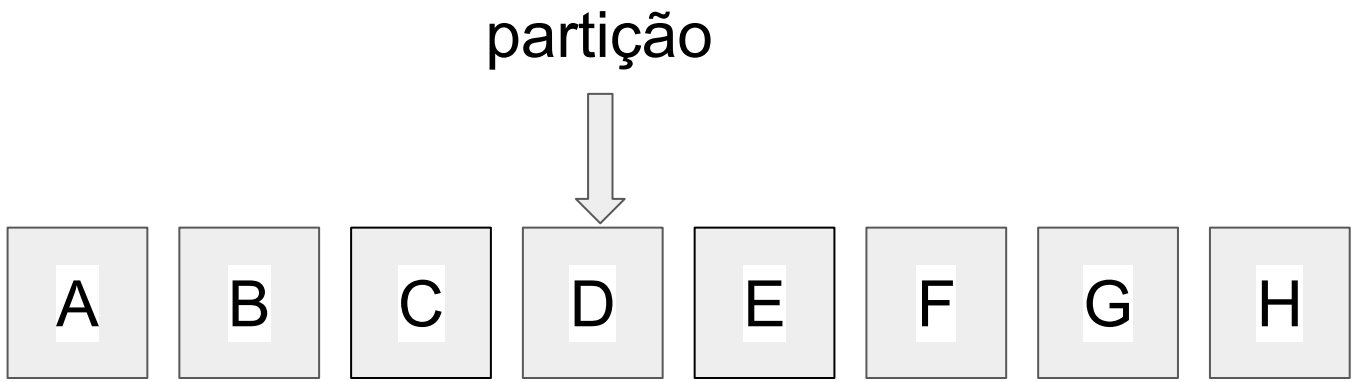
Complexidade de melhor caso

- Ordenação rápida
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?



Complexidade de melhor caso

- Ordenação rápida
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando a partição sempre dividir o vetor no meio.



Complexidade de melhor caso

- Útil para se ter uma primeira métrica inicial sobre o comportamento de um algoritmo

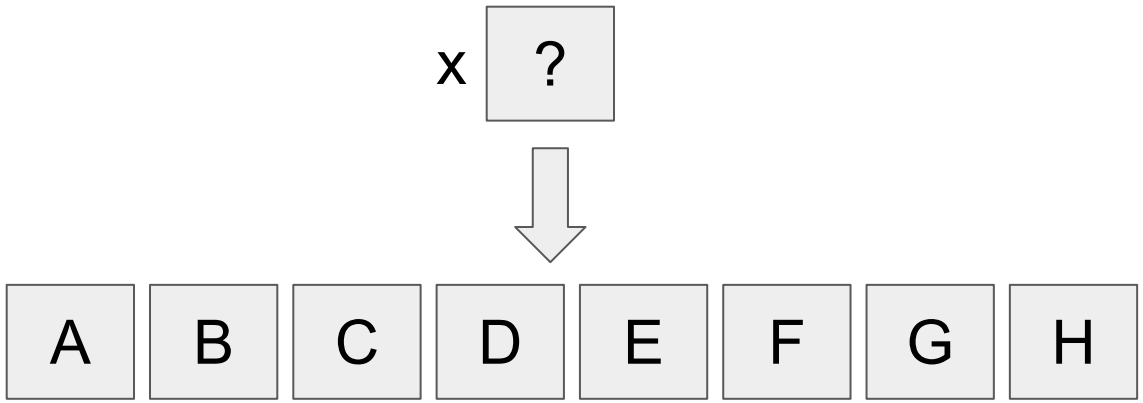
- Na prática, pouco utilizada, pois não oferece garantias de desempenho ao algoritmo

Complexidade de pior caso

- Comportamento do algoritmo no pior cenário possível
- Padrão de entrada que gera a maior quantidade de iterações do algoritmo
- Exemplos: busca sequencial, busca binária e algoritmos de ordenação

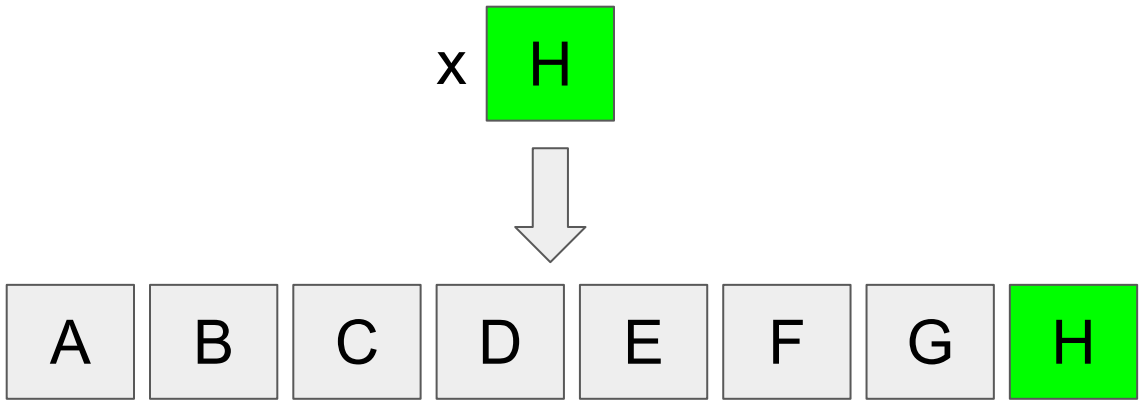
Complexidade de pior caso

- Busca sequencial
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?



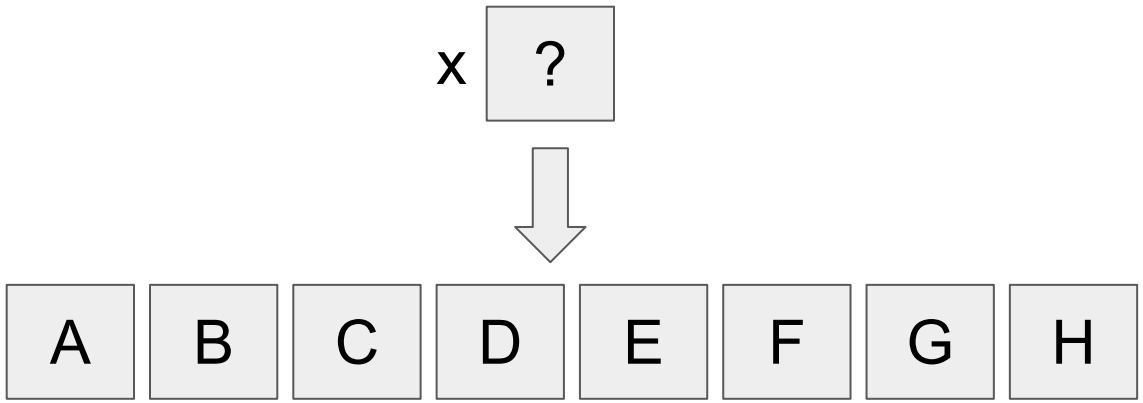
Complexidade de pior caso

- Busca sequencial
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?
  - *x* na última posição!



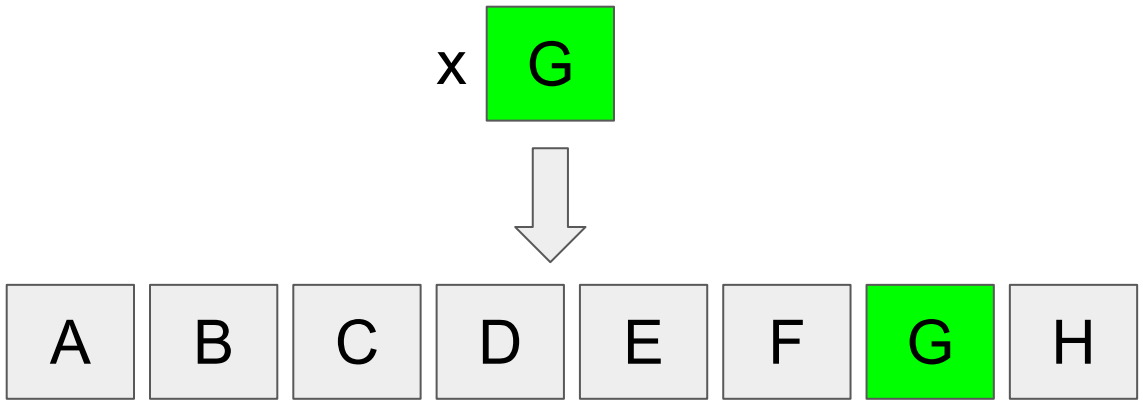
Complexidade de pior caso

- Busca binária
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?



Complexidade de pior caso

- Busca binária
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?
  - *x* realizando a maior quantidade de chamadas recursivas.

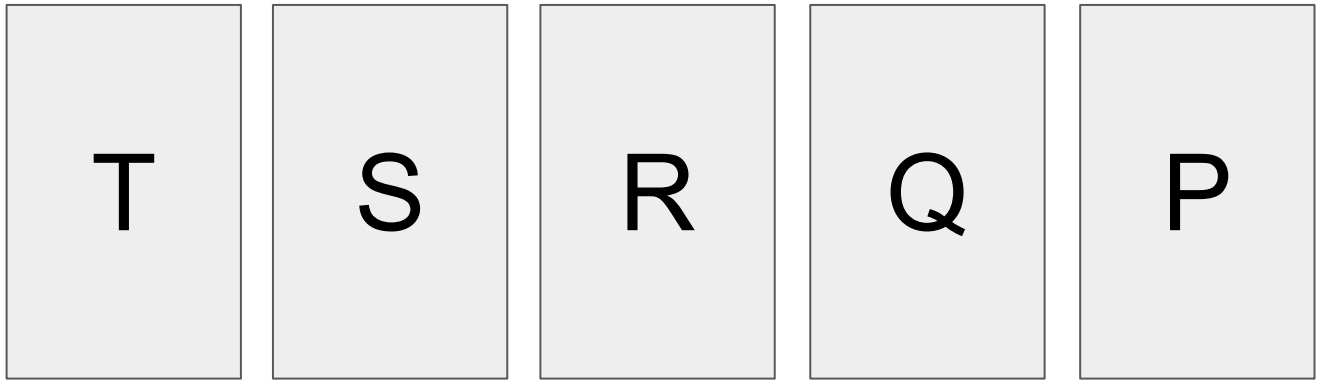


Complexidade de pior caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?

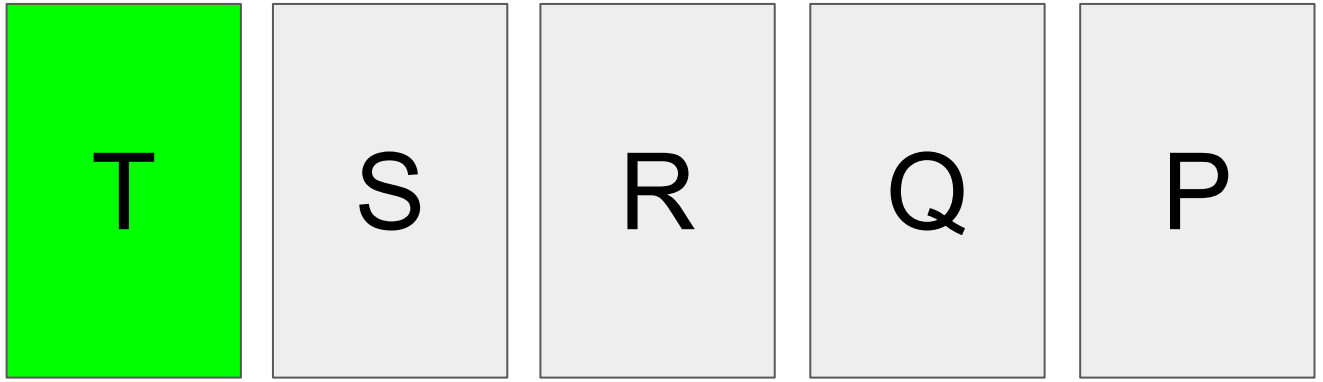
Complexidade de pior caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?
  - Quando o vetor estiver ordenado na ordem inversa!



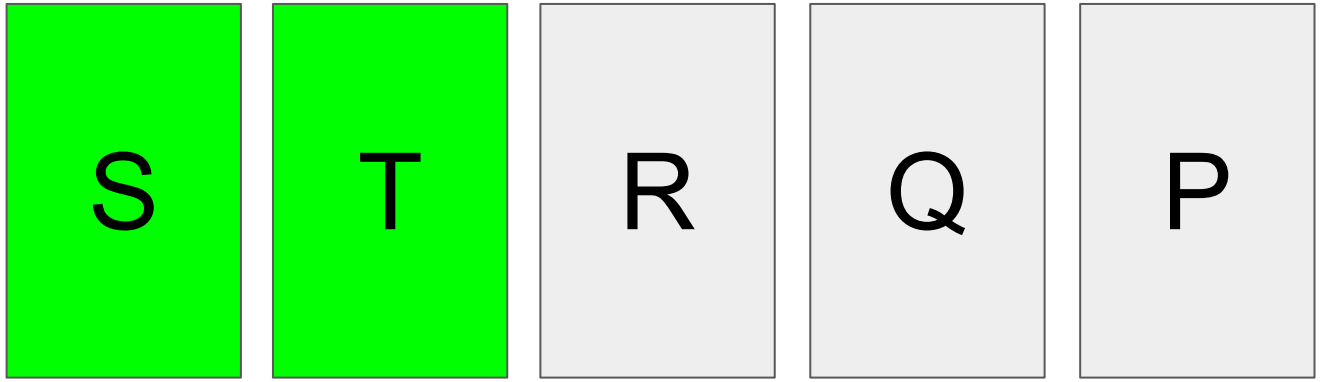
Complexidade de pior caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor estiver ordenado na ordem inversa!



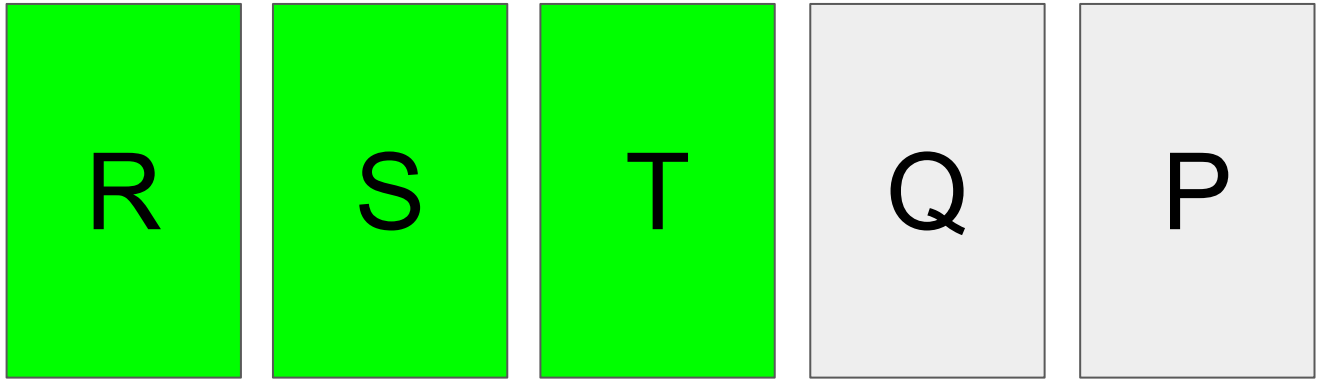
Complexidade de pior caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor estiver ordenado na ordem inversa!



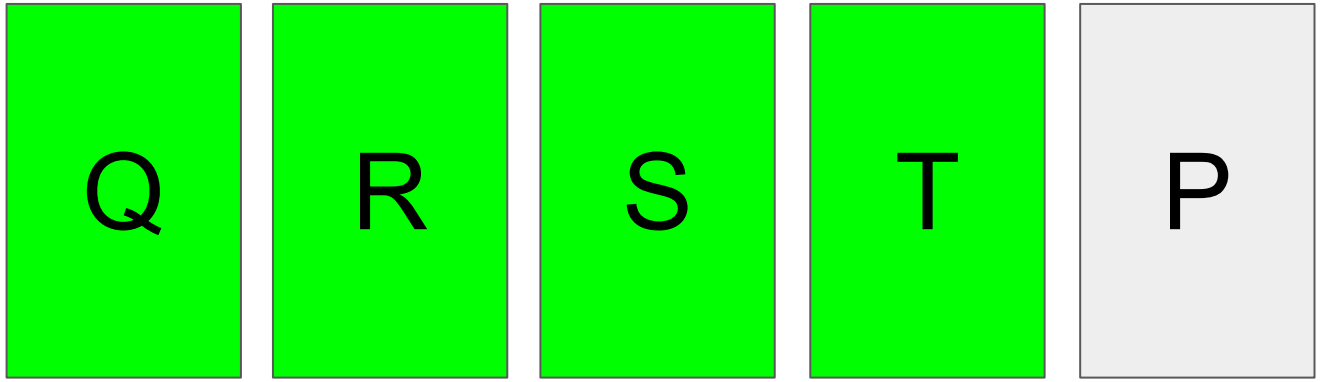
Complexidade de pior caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor estiver ordenado na ordem inversa!



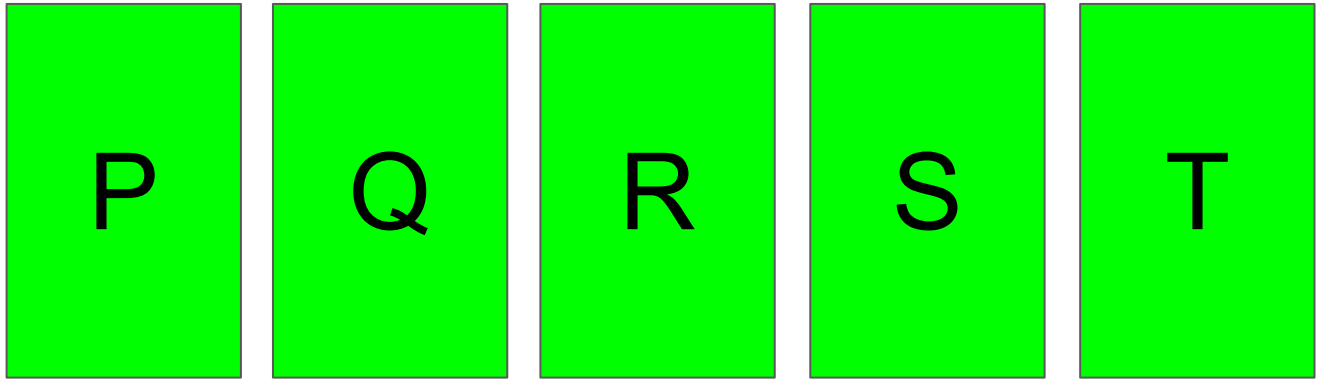
Complexidade de pior caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor estiver ordenado na ordem inversa!



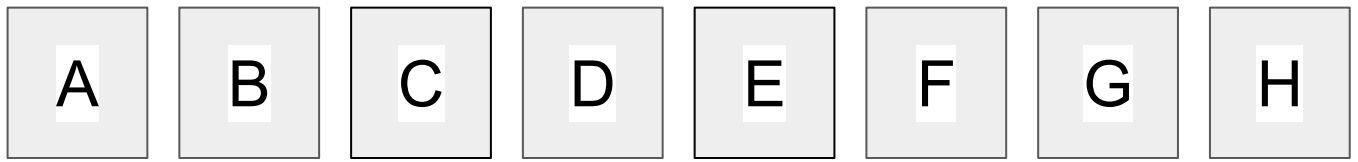
Complexidade de pior caso

- Ordenação por inserção
  - Qual entrada fará o algoritmo executar a menor quantidade de iterações?
  - Quando o vetor estiver ordenado na ordem inversa!



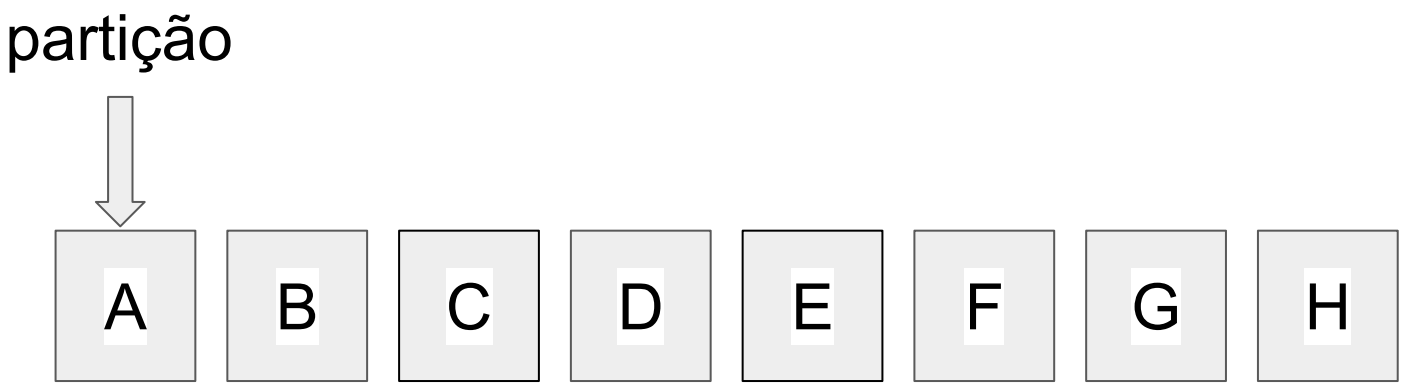
Complexidade de pior caso

- Ordenação rápida
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?



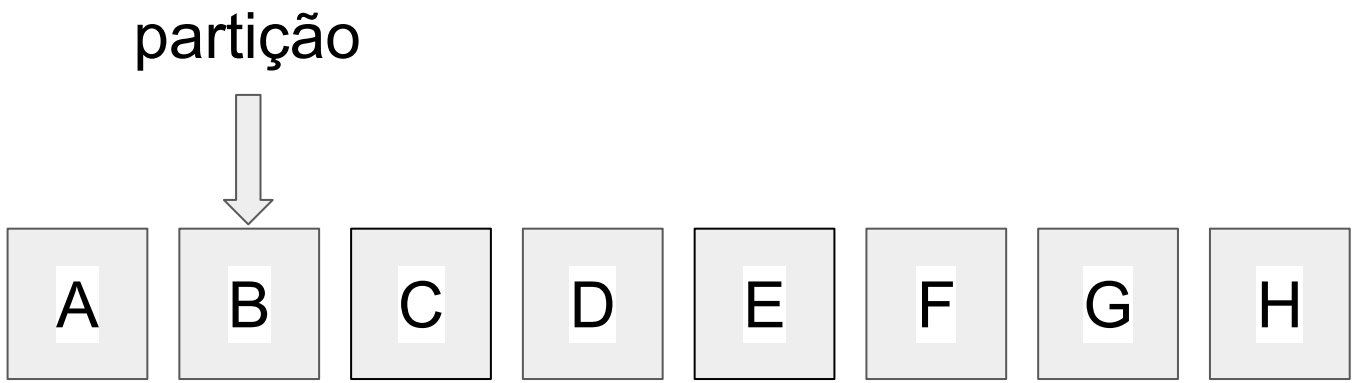
Complexidade de pior caso

- Ordenação rápida
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?
  - Quando a partição sempre dividir o vetor desbalanceado.
    - Vetor já ordenado!



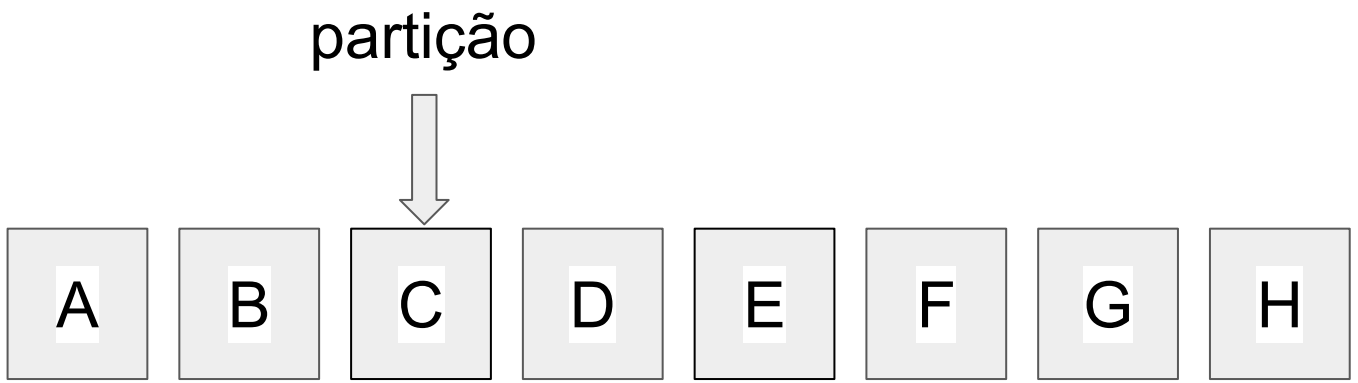
Complexidade de pior caso

- Ordenação rápida
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?
  - Quando a partição sempre dividir o vetor desbalanceado.
    - Vetor já ordenado!



Complexidade de pior caso

- Ordenação rápida
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?
  - Quando a partição sempre dividir o vetor desbalanceado.
    - Vetor já ordenado!



Complexidade de pior caso

- Útil para se ter uma métrica do pior cenário sobre o comportamento de um algoritmo

- Muito utilizada na prática, pois permite preparar o ambiente computacional para a pior situação possível

Noções de eficiência

- Neste unidade vocês aprenderão a avaliar o pior e o melhor caso de diferentes tipos de algoritmos

- Construirão uma visão crítica inicial em relação ao desempenho dos seus próprios algoritmos

UFMS DIGITAL

Unidade II

Complexidade assintótica

Complexidade assintótica

- Até o momento aprendemos a identificar o melhor e o pior cenário de diferentes algoritmos

- Entretanto, não aprendemos a **quantificar** esse desempenho

Complexidade assintótica

- A *complexidade assintótica* cria uma notação que permite gerar uma **fórmula** para identificar o desempenho de um algoritmo

- Três diferentes notações
  - Notação O
  - Notação Θ
  - Notação Ω

Complexidade assintótica

- A *complexidade assintótica* cria uma notação que permite gerar uma **fórmula** para identificar o desempenho de um algoritmo

- Três diferentes notações
  - **Notação *O***
  - Notação Θ
  - Notação Ω

Complexidade assintótica

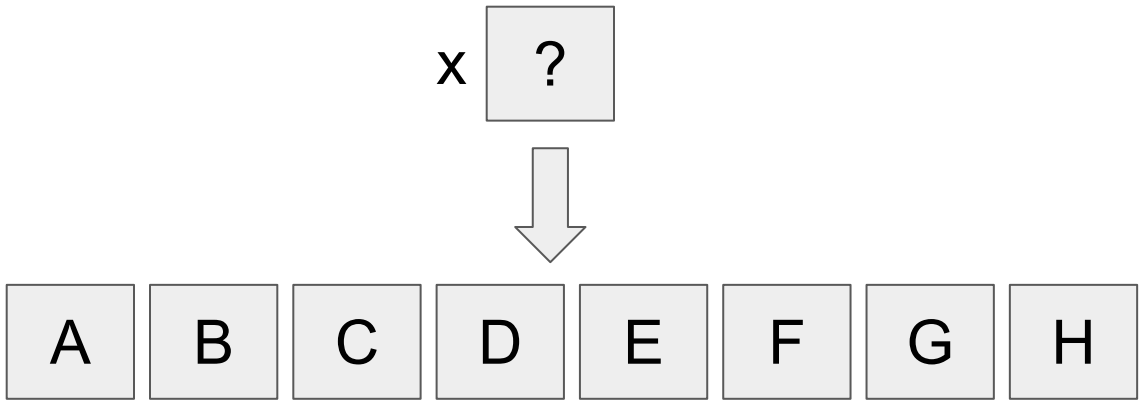
- A *complexidade assintótica* calculará a quantidade de iterações que um algoritmo desempenha para um determinado caso

- Com essa informação, gerará uma *fórmula* para representar esse desempenho

- Notação *O* extrairá o padrão de comportamento dessa fórmula, gerando um número de desempenho para o algoritmo

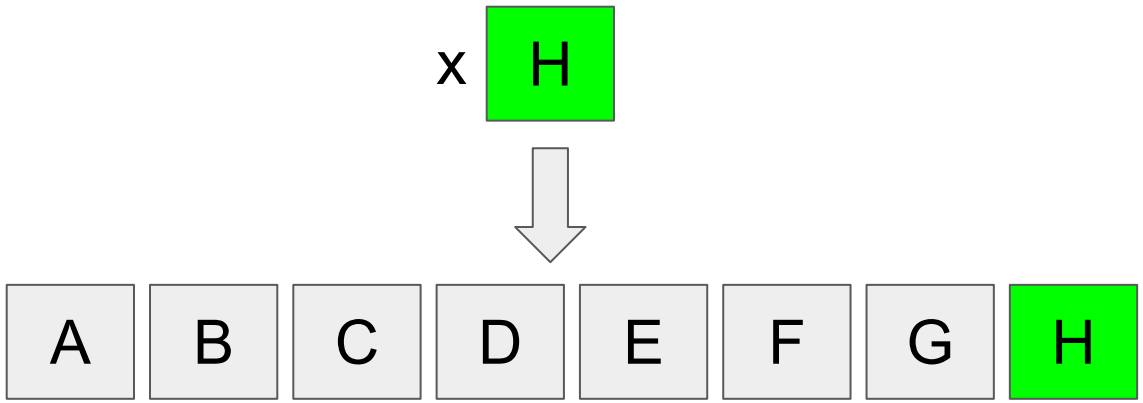
Complexidade assintótica

- Exemplo: busca sequencial
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?



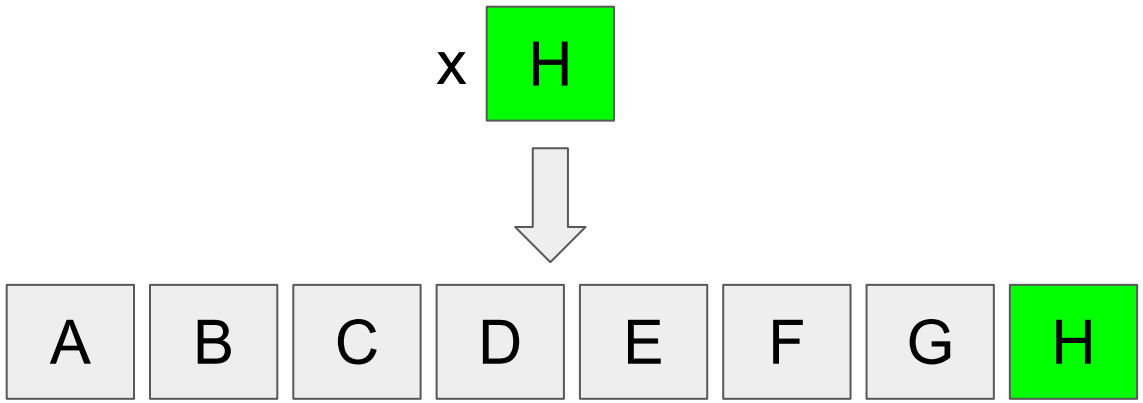
Complexidade assintótica

- Exemplo: busca sequencial
  - Qual entrada fará o algoritmo executar a maior quantidade de iterações?
  - Qual a complexidade assintótica do pior caso?



Complexidade assintótica

- Exemplo: busca sequencial
  - Qual a complexidade assintótica do pior caso?
    - Vetor *v* com *n* entradas e *x* na última posição
    - *n* iterações



Complexidade assintótica

- Exemplo: busca sequencial

```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```

Complexidade assintótica

- Exemplo: busca sequencial
  - Contaremos a quantidade de iterações por linha

```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```

Complexidade assintótica

- Exemplo: busca sequencial
  - Contaremos a quantidade de iterações por linha

```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```

1

Complexidade assintótica

- Exemplo: busca sequencial
  - Contaremos a quantidade de iterações por linha

```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```

1
n+1

Complexidade assintótica

- Exemplo: busca sequencial
  - Contaremos a quantidade de iterações por linha

```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```

1
n+1
n

Complexidade assintótica

- Exemplo: busca sequencial
  - Contaremos a quantidade de iterações por linha

```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```

1
n+1
n
n

Complexidade assintótica

- Exemplo: busca sequencial
  - Contaremos a quantidade de iterações por linha

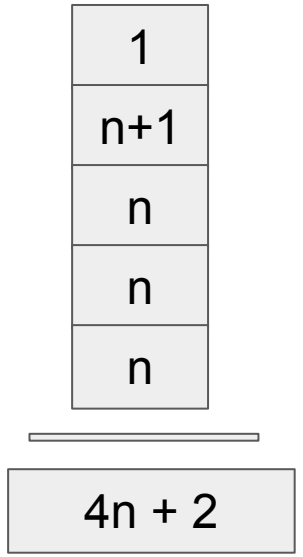
```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```

1
n+1
n
n
n

Complexidade assintótica

- Exemplo: busca sequencial
  - Contaremos a quantidade de iterações por linha

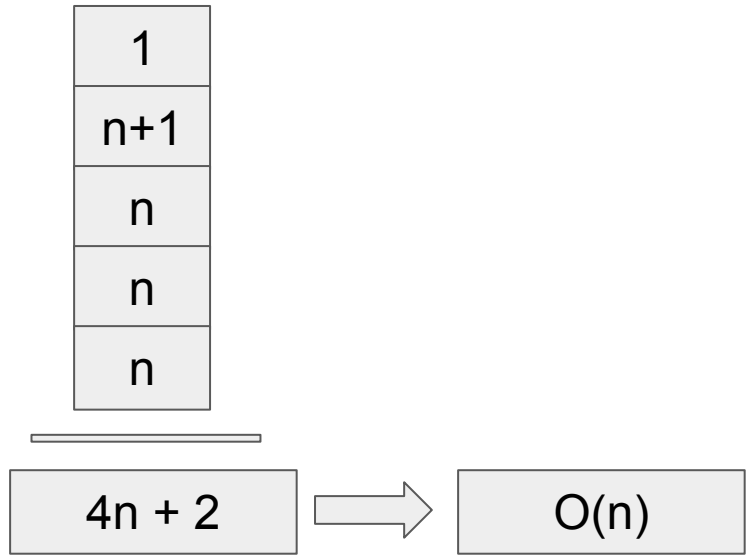
```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```



Complexidade assintótica

- Exemplo: busca sequencial
  - Contaremos a quantidade de iterações por linha

```
encontrei = False
for i in range(len(v)):
    if v[i] == x:
        encontrei = True
        break
```



Complexidade assintótica

- Permite criar uma notação que extrai a parte mais custosa de um algoritmo

- Quantifica o desempenho de algoritmo, independente da máquina de execução

Roteiro

- Unidade 1: Noções de eficiência de algoritmos: pior e melhor caso
- Unidade 2: Complexidade assintótica
- Resumo