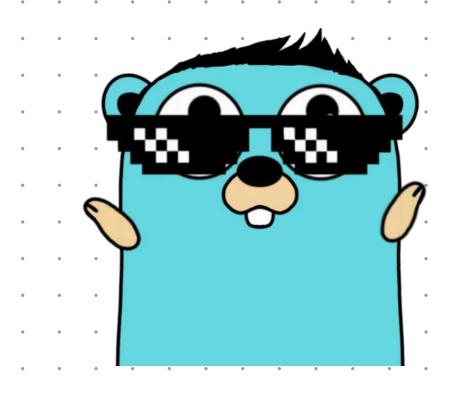


#### Análise de algoritmos

# BUSCA BINÁRIA

Baseado no livro Entendendo Algoritmos, Um guia ilustrado para programadores e outros curiosos. Aditya Y. Bhargava, Novatec, 2017.





### Análise de algoritmos

# BUSCA BINÁRIA

Consiste em repartir uma lista ordenada até chegar no conteúdo buscado com o menor número de tentativas, chutando um conteúdo encontrado no meio da lista e eliminando metade das possbilidades a cada vez que a estrutura for repartida.

O tempo de execução na notação Big O é de O(log n), pois cresce logaritmicamente com o tamanho da entrada, aumentando minimamente o tempo de execução para cada item adicional na lista ordenada.

#### **APLICAÇÕES**

- Busca em tabelas e catálogos;
- Otimização e algoritmos;
- Jogos e simulações;
- Compressão de dados;
- Sistemas embarcados e dispositivos móveis;
- Busca em intervalos, seleção de elementos e verificação de membros.



```
package main
import (
 "fmt"
 "strings"
 "time"
type Searcher interface {
 Search(target string, data []Name) int
type BinarySearch struct{}
type Name struct {
 FullName string
```



```
func main() {
 sortedNames := generateRandomNames()
 fmt.Println("the size of the ordered list is:", len(sortedNames))
 targetName := "Davi Costa"
 startTime := time.Now()
 binarySearcher := BinarySearch{}
 index := binarySearcher.Search(targetName, sortedNames)
 endTime := time.Now()
 executionTimeInNanosecounds := endTime.Sub(startTime)
 executionTimeInMilliseconds := float64(executionTimeInNanosecounds) / float64(time.Millisecond)
 if index != -1 {
  fmt.Printf("Execution time: %.3f milliseconds\n", executionTimeInMilliseconds)
 } else {
  fmt.Println("Name not found in the list:")
  fmt.Printf("Execution time: %.3f milliseconds\n", executionTimeInMilliseconds)
```



```
func generateRandomNames() []Name {
 fullNames := "Arthur Silva, Bernardo Santos, Davi Costa, Enzo Oliveira,
Gabriel Souza, Heitor Rodrigues, Lorenzo Nunes, Miguel Ferreira, Pedro
Pereira, Rafael Oliveira, Lucas Gomes, Matheus Silva..."
 splitedNames := strings.Split(fullNames, ",")
 names := []Name{}
 for _, value := range splitedNames {
  trimmedNames := strings.TrimLeft(value, " ")
  names = append(names, Name{FullName: trimmedNames})
 return names
```



```
func (b BinarySearch) Search(target string, data []Name) int {
 start := 0
 end := len(data) - 1
 middle := (end + start) / 2
 operation := 0
 for start <= end {
  if data[middle].FullName == target {
   fmt.Println("Name found in the list:", data[middle].FullName)
   return middle
  } else if data[middle].FullName > target {
   end = middle - 1
  } else {
   start = middle + 1
  middle = (end + start) / 2
  operation++
  fmt.Println("Number of binary search operations performed:", operation)
 return -1
```



## Q SAÍDA DO PROGRAMA

#### **CENÁRIO OTIMISTA**

the size of the ordered list is: 100 number of binary search operations performed: 1 number of binary search operations performed: 2 number of binary search operations performed: 3 number of binary search operations performed: 4 name found in the list: Davi Costa execution time: 0.541 milliseconds



## Q SAÍDA DO PROGRAMA

#### **CENÁRIO PESSIMISTA**

the size of the ordered list is: 100 number of binary search operations performed: 1 number of binary search operations performed: 2 number of binary search operations performed: 3 number of binary search operations performed: 4 number of binary search operations performed: 5 number of binary search operations performed: 6 number of binary search operations performed: 7 name not found in the list: execution time: 0.519 milliseconds



### Análise de algoritmos

# CONSIDERAÇÕES

Para o exemplo apresentado, seguiu-se as boas práticas de desenvolvimento com clean architecture, clean code e SOLID, na tentativa de simular um cenário otimista.

Essa iniciativa vai de encontro com a ideia de trazer conteúdos relevantes altamente abordados em processos seletivos e desmitificar a ideia de algoritmos e estrutura de dados. Espero que seja de bom proveito e bons estudos.

