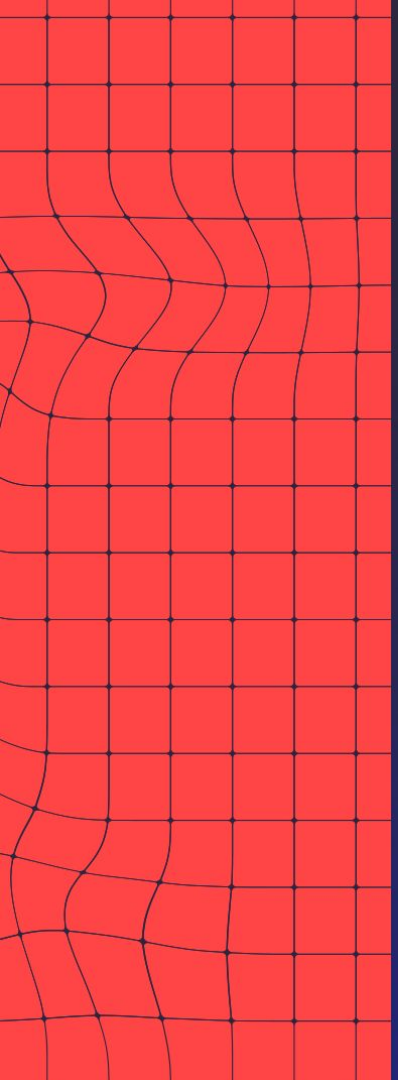
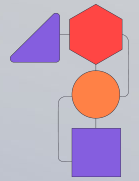


inteli

# Busca e Ordenação

Fillipe Resina

inteli



# Busca Sequencial

# Filtro de Spam

## Confiáveis

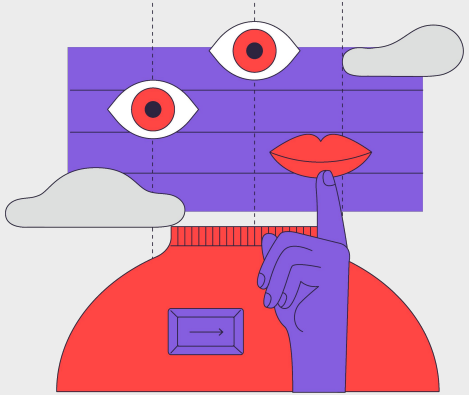
dobqi
xwnzb
dqwak
lnuqv
czpwx
bshla
idhld
utfyw
hafah
tsirv

## Testes

xwnzb
lnuqv
lnuqv
czpwx
czpwx
dqwak
idhld
dobqi
dobqi
tsirv
dqwak
dobqi
idhld
dqwak
dobqi
lnuqv
xwnzb

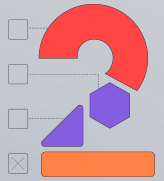
**Pergunta**

# Por que, no teste empírico, analisamos T(N) com T(N/2)?



inteli

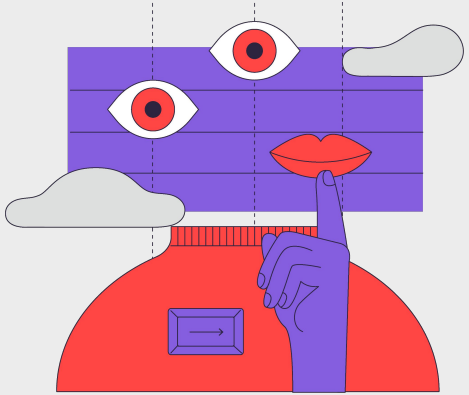
inteli



# Busca Binária

**Pergunta**

# O que é e qual o pré-requisito?



inteli

Definição

# Busca Binária

"É uma busca que tem como pré-requisito o vetor estar previamente ordenado. Com isso, posso começar a busca verificando o elemento que está no meio e, se não encontrar o elemento-alvo, consigo reduzir meu espaço de busca pela metade (no grupo dos maiores ou menores do que o elemento verificado)."

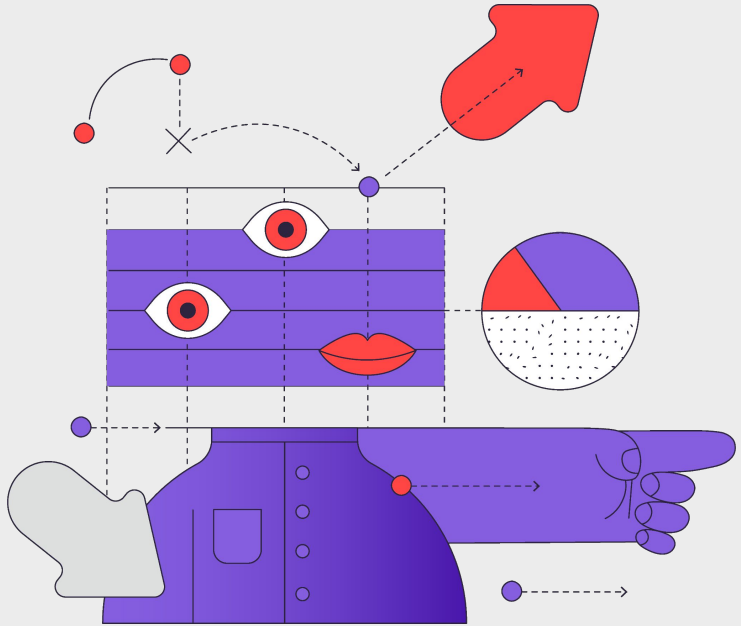
**Complexidade de Tempo:** O(lgn)

inteli

inteli

# Métodos de Ordenação

# Atividade em Grupos

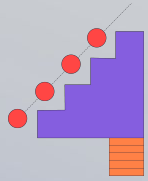


inteli

## Vetor a Ordenar

6 4 10 8 2

inteli



# Bubblesort

## Vetor a Ordenar

6 4 10 8 2

# Cerne da Implementação

```
for(i = 0; i<5; i++){
    for(int j = 0; j<4; j++){
        if(vet[j] > vet[j + 1]){
            aux = vet[j];
            vet[j] = vet[j+1];
            vet[j+1] = aux;
        }
    }
}
```

**Complexidade de Tempo:** $O(n^2)$

inteli

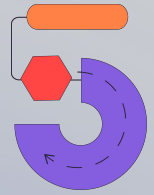
**Pergunta**

**Como provar que ao final estará ordenado?**

**Resp:** identificando e provando um invariante (propriedade que não varia ao longo do loop principal).

inteli

inteli



# Insertion Sort

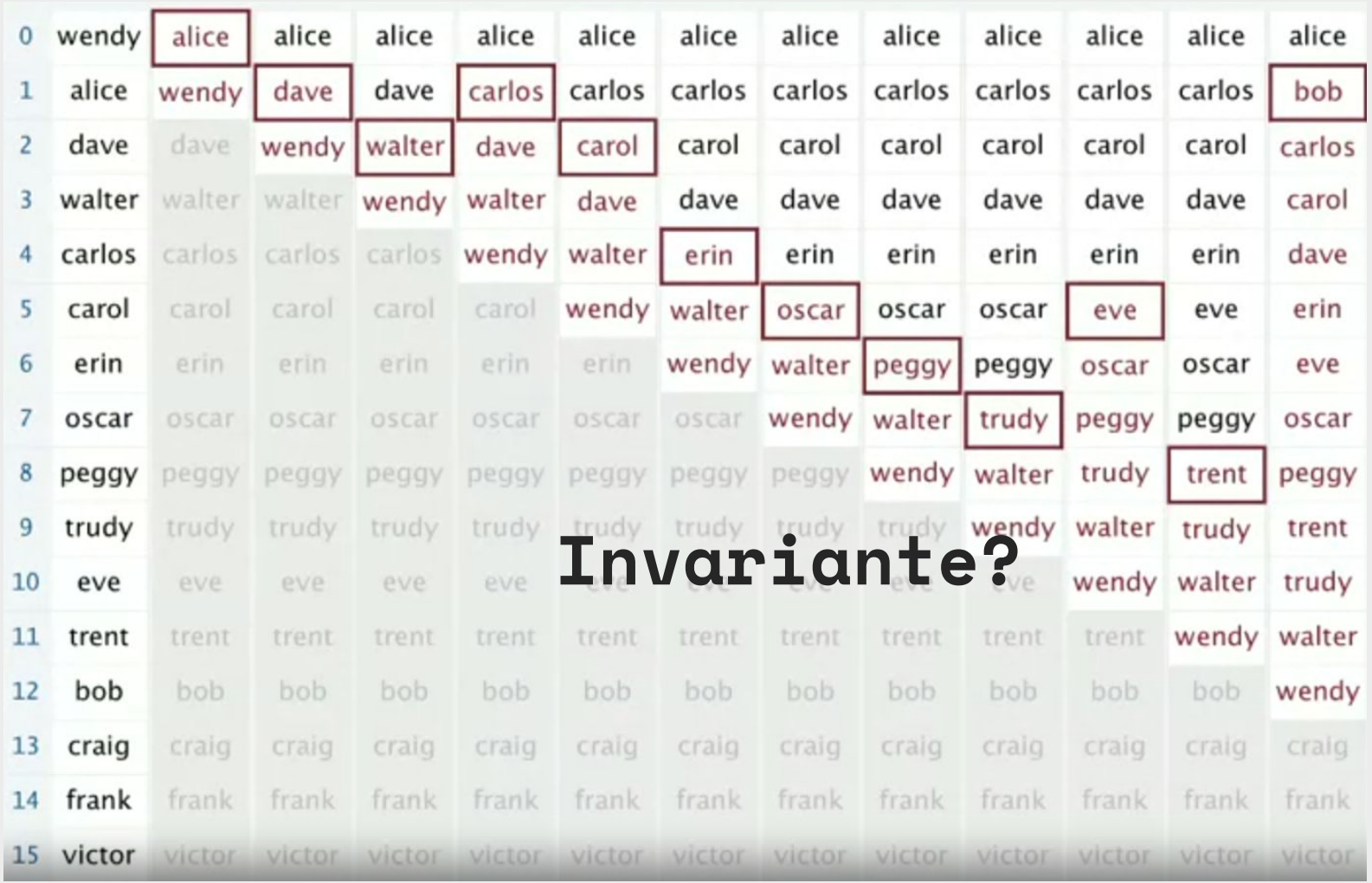
## Vetor a Ordenar

6 4 10 8 2

**Pergunta**

# Como provar que ao final estará ordenado?

inteli




# Invariante?

inteli

inteli

# Mergesort

## Vetor a Ordenar

6 4 10 8 2

# Cerne da Implementação

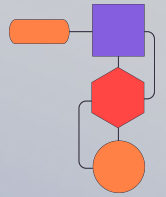
```
public static void sort(String[] a, int lo, int hi)
{  // Sort a[lo, hi).
   int N = hi - lo;
   if (N <= 1) return;
   int mid = lo + N/2;
   sort(a, lo, mid);
   sort(a, mid, hi);
   merge(a, lo, mid, hi);
}
```

DIVISÃO E CONQUISTA

**Complexidade de Tempo:** $\Theta(n \lg n)$

inteli

inteli



# Quicksort

## Vetor a Ordenar

6 4 10 8 2

# Implementação

```
private static int partition(Comparable[] a, int lo, int hi)
{
   int i = lo, j = hi+1;
   while (true)
   {
      while (less(a[++i], a[lo]))
         if (i == hi) break;

      while (less(a[lo], a[--j]))
         if (j == lo) break;

      if (i >= j) break;
      exch(a, i, j);
   }

   exch(a, lo, j);
   return j;
}
```

find item on left to swap

find item on right to swap

check if pointers cross

swap

swap with partitioning item

return index of item now known to be in place

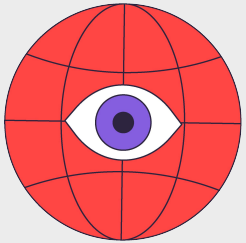
```
public class Quick
{
   private static int partition(Comparable[] a, int lo, int hi)
   {  /* see previous slide */  }

   public static void sort(Comparable[] a)
   {
      StdRandom.shuffle(a);
      sort(a, 0, a.length - 1);
   }

   private static void sort(Comparable[] a, int lo, int hi)
   {
      if (hi <= lo) return;
      int j = partition(a, lo, hi);
      sort(a, lo, j-1);
      sort(a, j+1, hi);
  }
}
```

**Complexidade de Tempo (caso médio):** O(n lgn)

inteli




# Fillipe Resina

fillipe.resina@prof.inteli.edu.br


inteli


inteli