Aula 17 – Busca Binária e Herança

Norton Trevisan Roman

17 de maio de 2013

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas
  - Paramos a busca assim que uma das condições forem satisfeitas

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas
  - Paramos a busca assim que uma das condições forem satisfeitas
    - ★ Encontrarmos o elemento buscado

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas
  - Paramos a busca assim que uma das condições forem satisfeitas
    - ★ Encontrarmos o elemento buscado
    - ★ Chegarmos ao fim do arranjo

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas
  - Paramos a busca assim que uma das condições forem satisfeitas
    - ★ Encontrarmos o elemento buscado
    - ★ Chegarmos ao fim do arranjo
    - ★ (Diferencial!) Encontrarmos um elemento maior que o buscado

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas
  - Paramos a busca assim que uma das condições forem satisfeitas
    - ★ Encontrarmos o elemento buscado
    - ★ Chegarmos ao fim do arranjo
    - ★ (Diferencial!) Encontrarmos um elemento maior que o buscado
- Ainda assim, no pior caso, teremos que olhar o arranjo inteiro, quando:

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas
  - Paramos a busca assim que uma das condições forem satisfeitas
    - ★ Encontrarmos o elemento buscado
    - ★ Chegarmos ao fim do arranjo
    - ★ (Diferencial!) Encontrarmos um elemento maior que o buscado
- Ainda assim, no pior caso, teremos que olhar o arranjo inteiro, quando:
  - O elemento buscado for o último

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas
  - Paramos a busca assim que uma das condições forem satisfeitas
    - ★ Encontrarmos o elemento buscado
    - ★ Chegarmos ao fim do arranjo
    - ★ (Diferencial!) Encontrarmos um elemento maior que o buscado
- Ainda assim, no pior caso, teremos que olhar o arranjo inteiro, quando:
  - O elemento buscado for o último
  - O elemento buscado não estiver no arranjo, mas for maior que o último

Busca em Arranjo Ordenado

- Vimos que se o arranjo estiver ordenado, buscas ficam mais rápidas
  - Paramos a busca assim que uma das condições forem satisfeitas
    - ★ Encontrarmos o elemento buscado
    - ★ Chegarmos ao fim do arranjo
    - ★ (Diferencial!) Encontrarmos um elemento maior que o buscado
- Ainda assim, no pior caso, teremos que olhar o arranjo inteiro, quando:
  - O elemento buscado for o último
  - O elemento buscado não estiver no arranjo, mas for maior que o último
- Não teria um modo melhor?

Busca Binária

- Algoritmo:

Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo

Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior

Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - Se for, repita a busca na metade direita do arranjo

Busca Binária

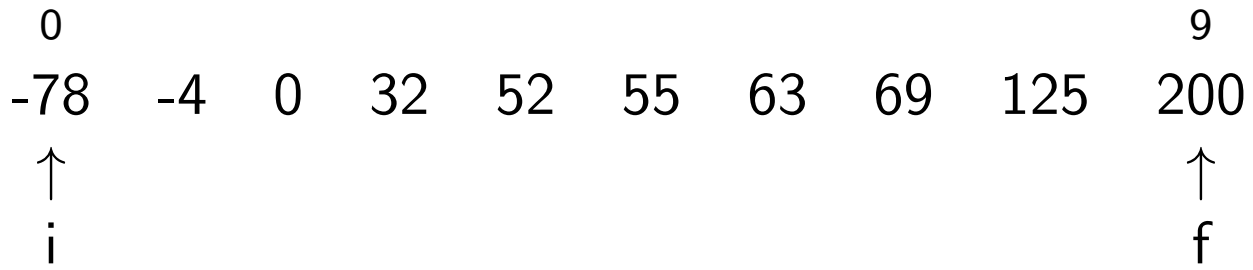
- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - Se for, repita a busca na metade direita do arranjo
    - Se não for, repita a busca na metade esquerda do arranjo

Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ★ Se for, repita a busca na metade direita do arranjo
    - ★ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 52

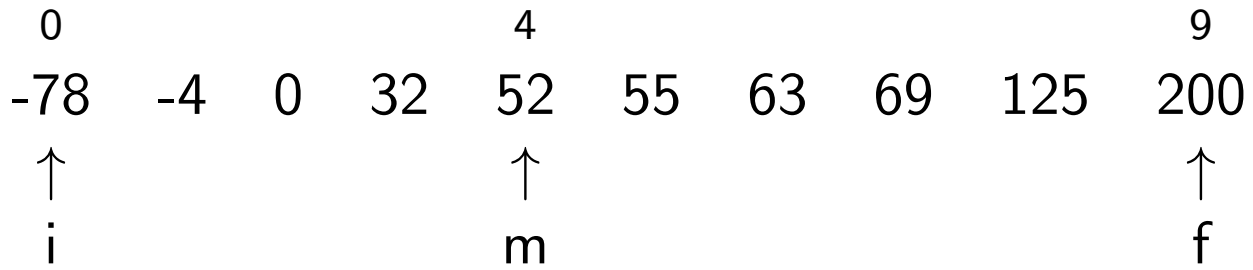
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 52



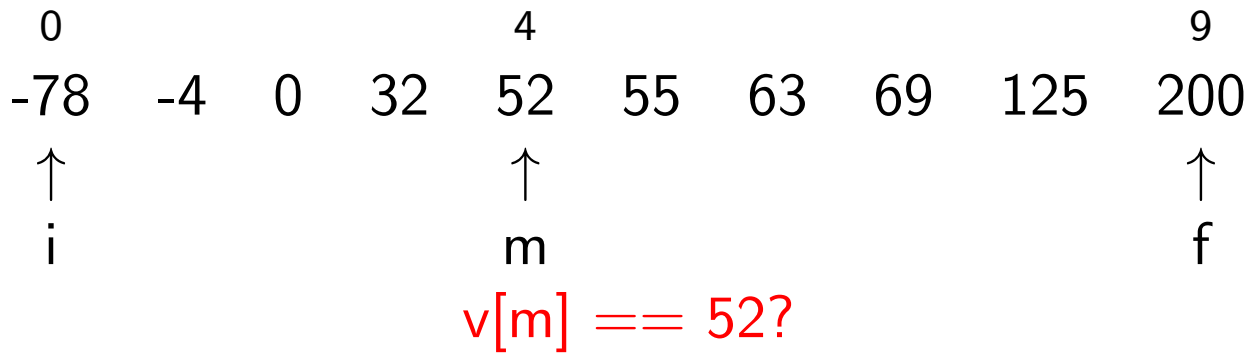
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 52



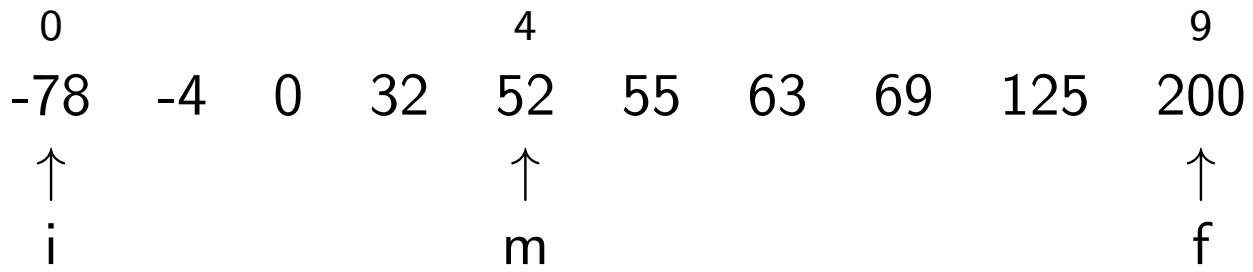
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ★ Se for, repita a busca na metade direita do arranjo
    - ★ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 52



Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ★ Se for, repita a busca na metade direita do arranjo
    - ★ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 52



v[m] == 52? Achou com apenas 1 acesso

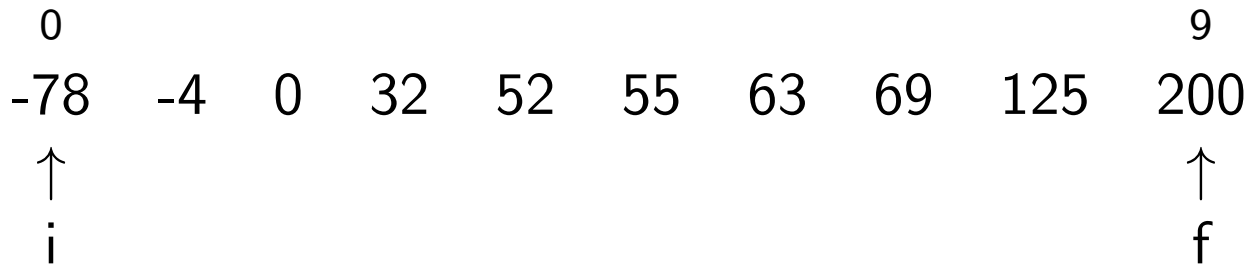
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ★ Se for, repita a busca na metade direita do arranjo
    - ★ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55

0									9
-78	-4	0	32	52	55	63	69	125	200

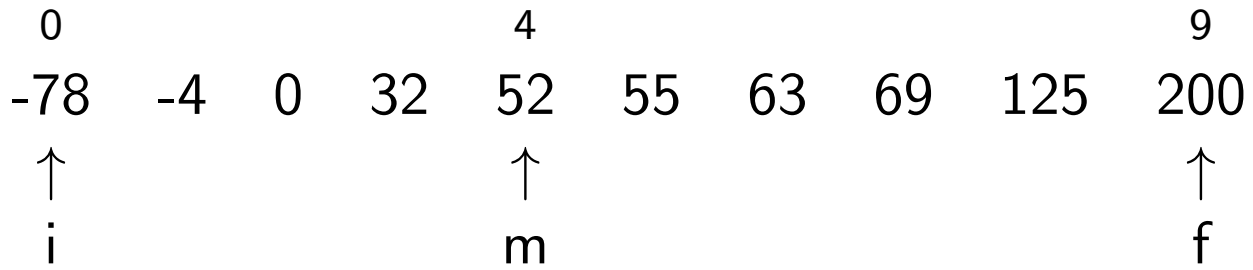
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



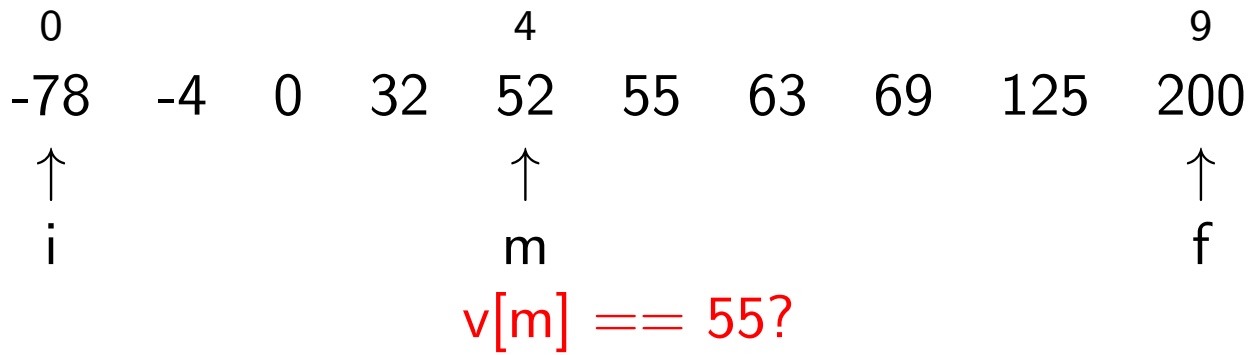
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



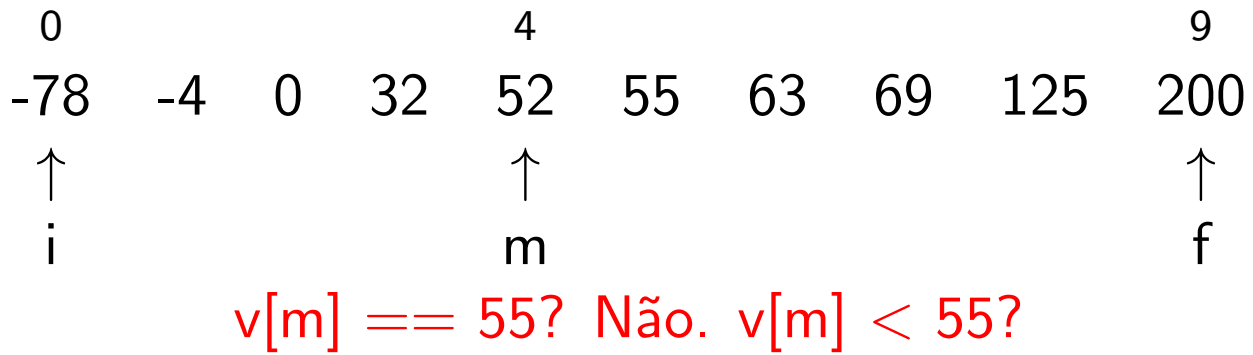
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ★ Se for, repita a busca na metade direita do arranjo
    - ★ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



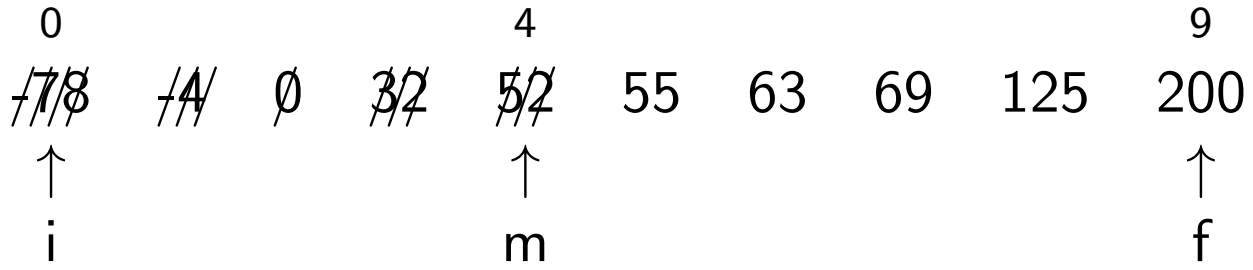
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ★ Se for, repita a busca na metade direita do arranjo
    - ★ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



Busca Binária

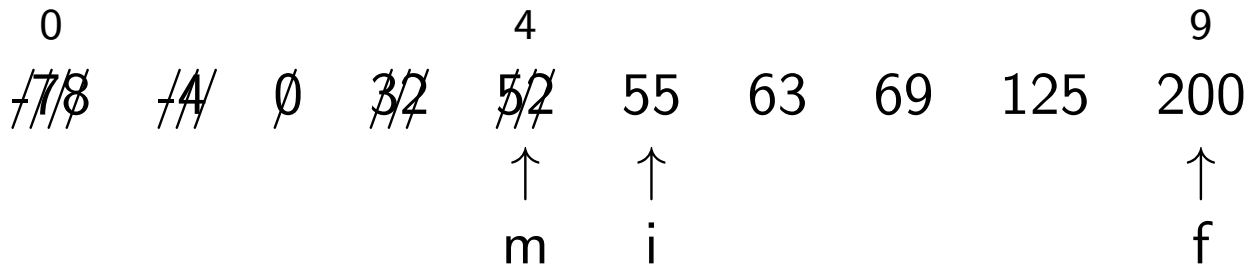
- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



v[m] == 55? Não. v[m] < 55? Sim.

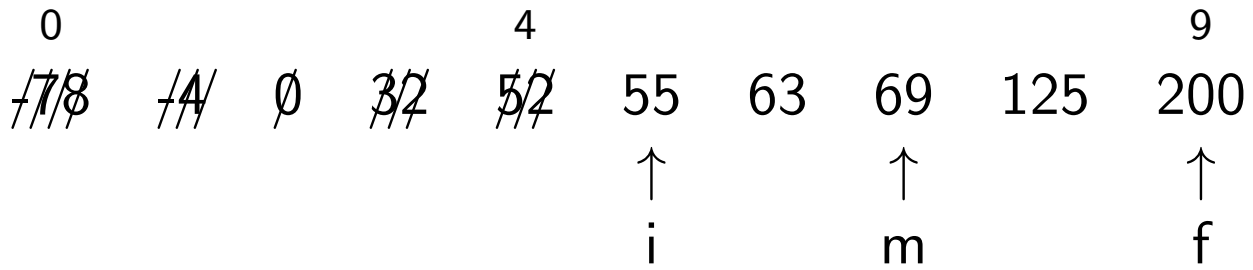
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



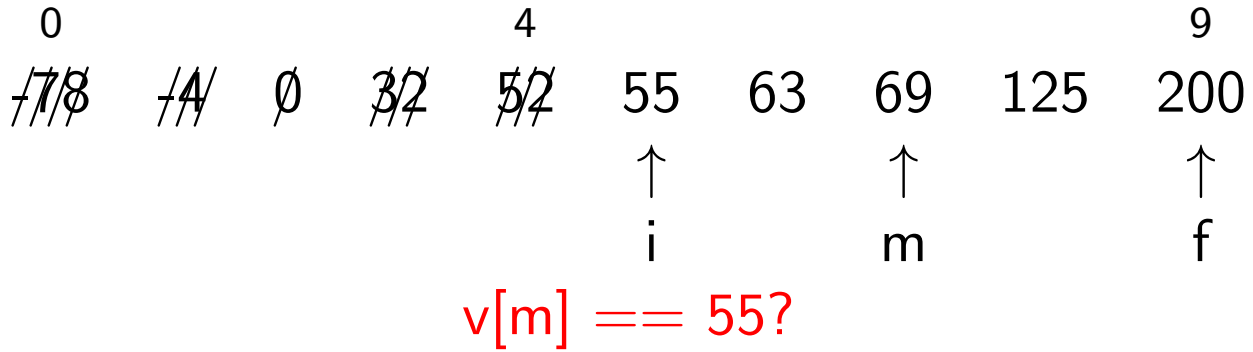
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



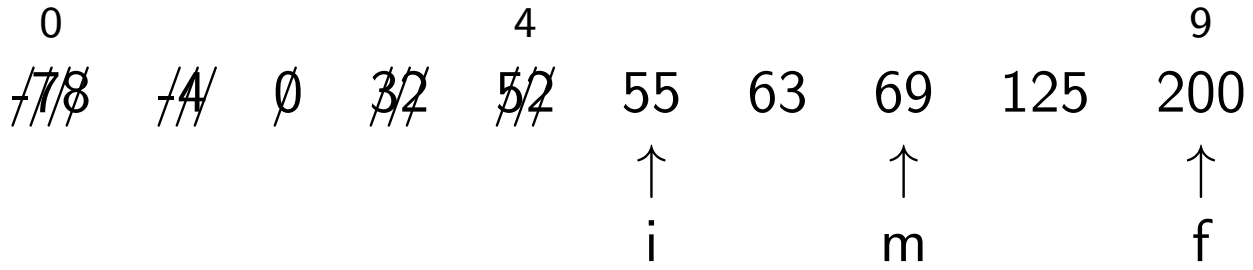
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



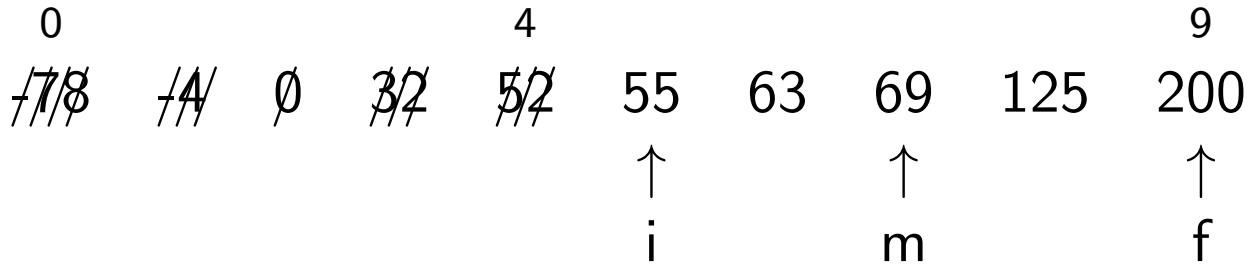
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



Busca Binária

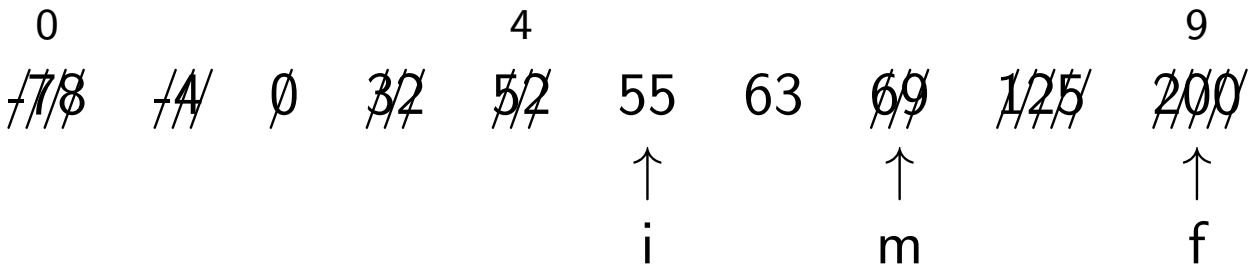
- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



v[m] == 55? Não. v[m] < 55? Não.

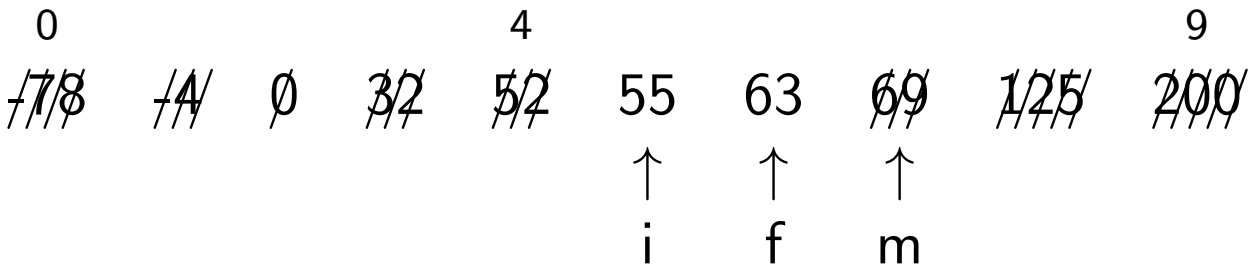
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



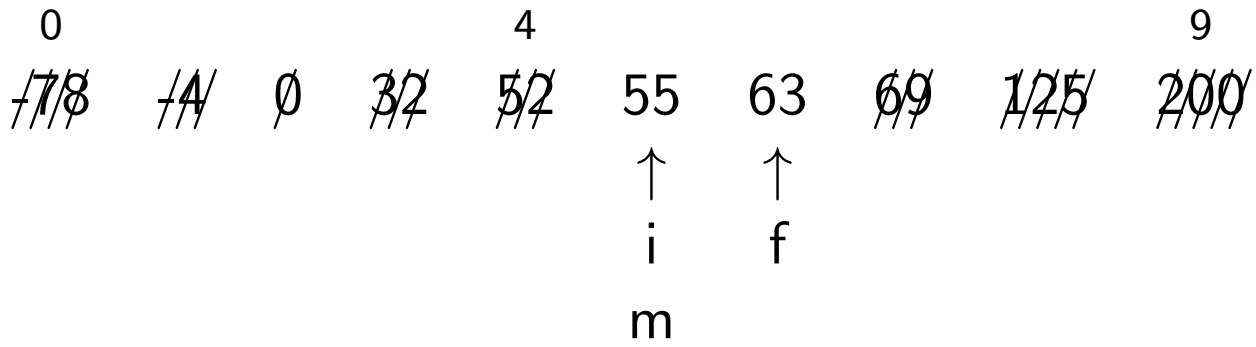
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



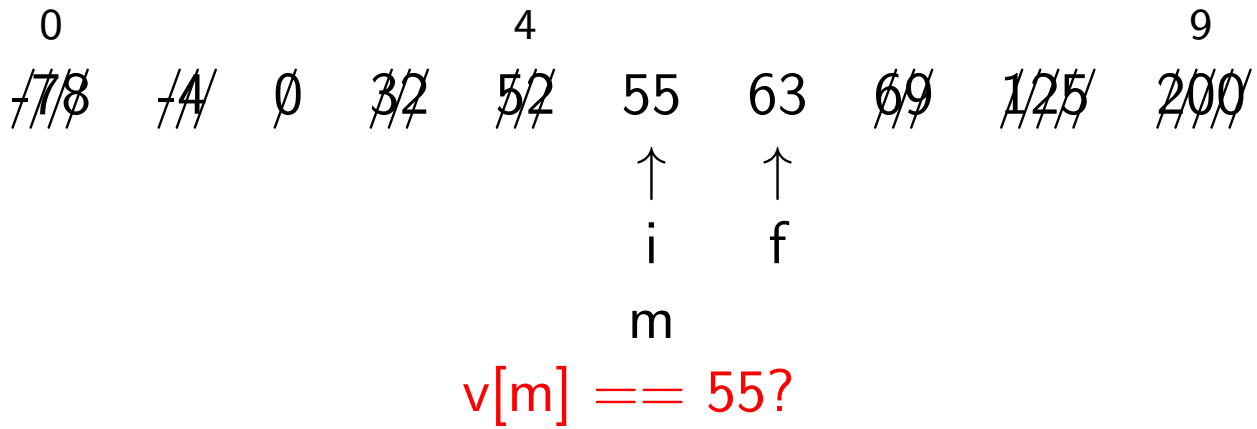
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



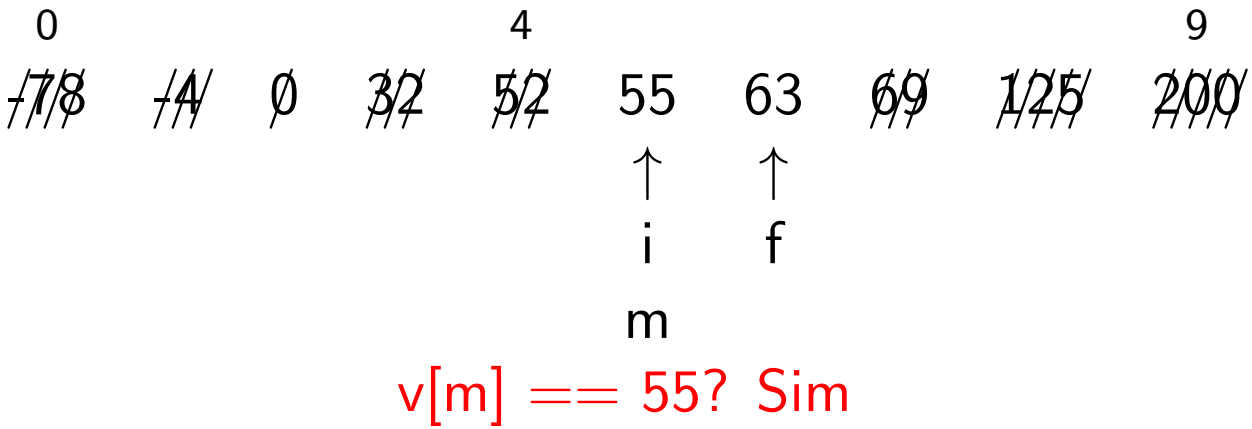
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



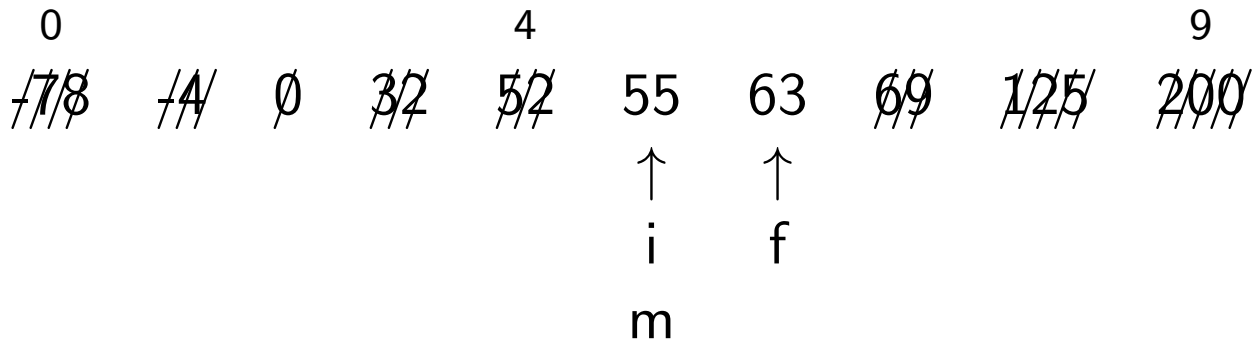
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 55



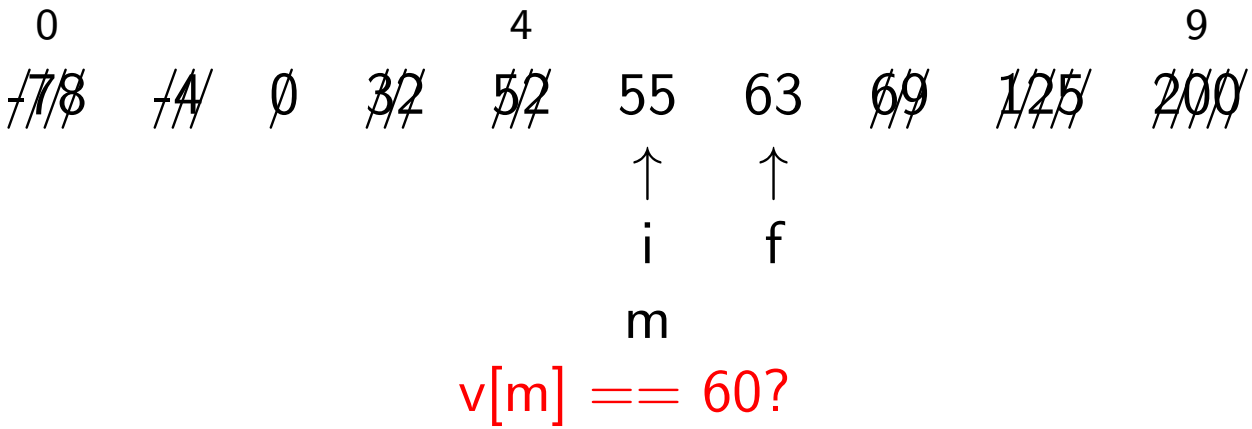
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



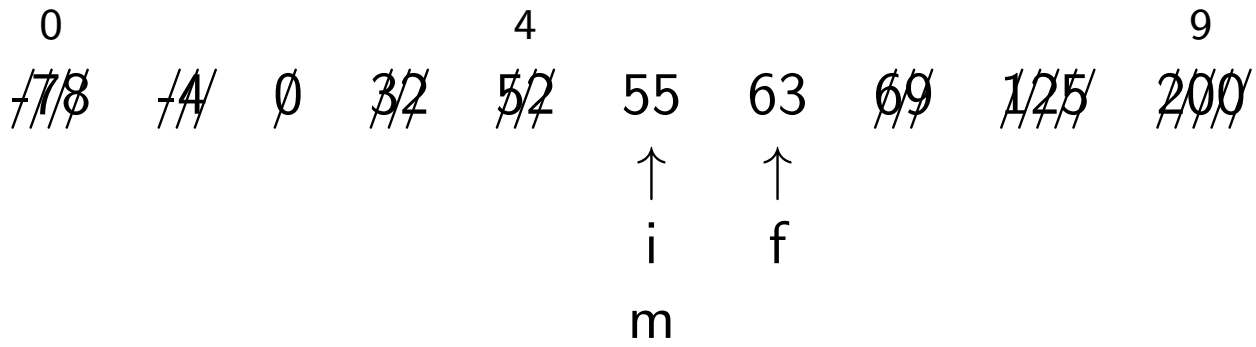
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



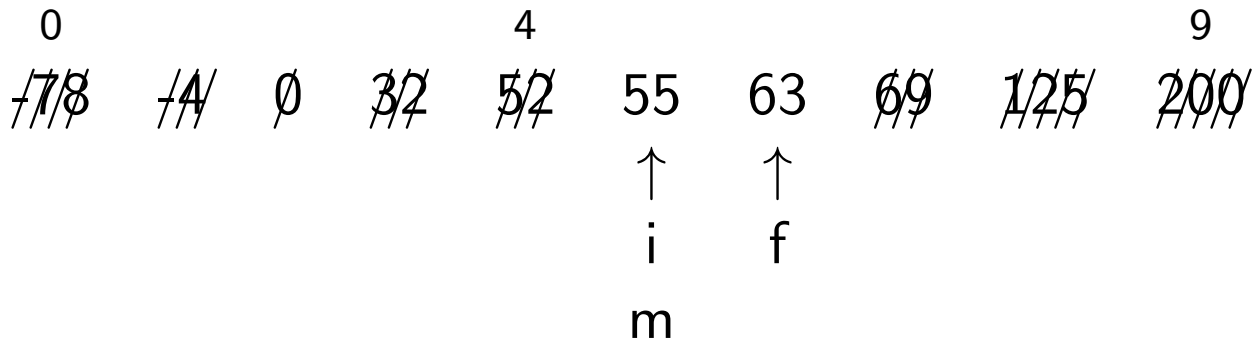
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



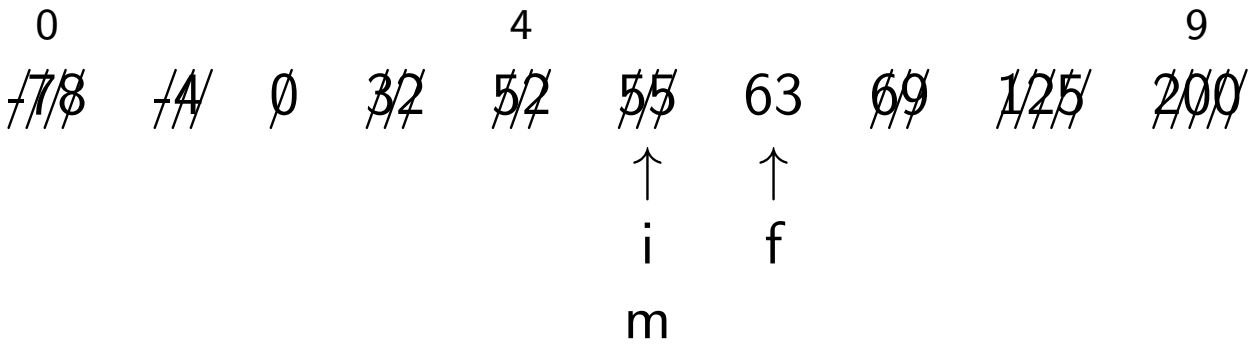
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



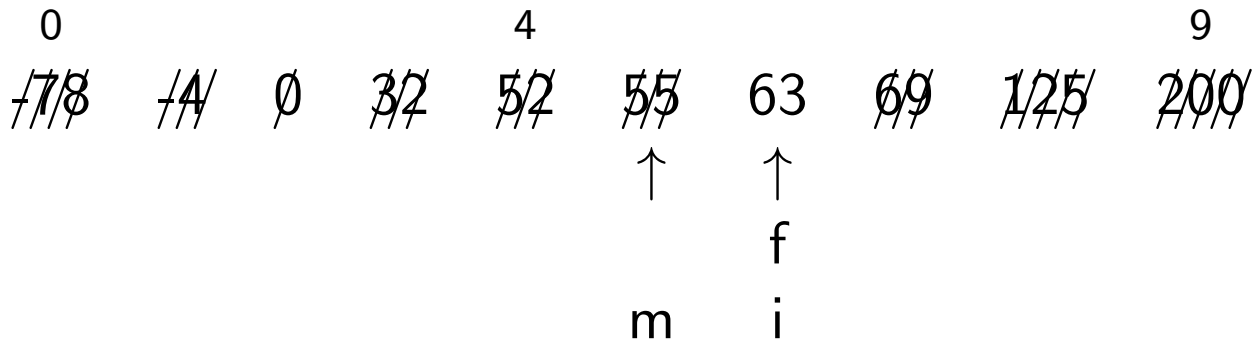
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



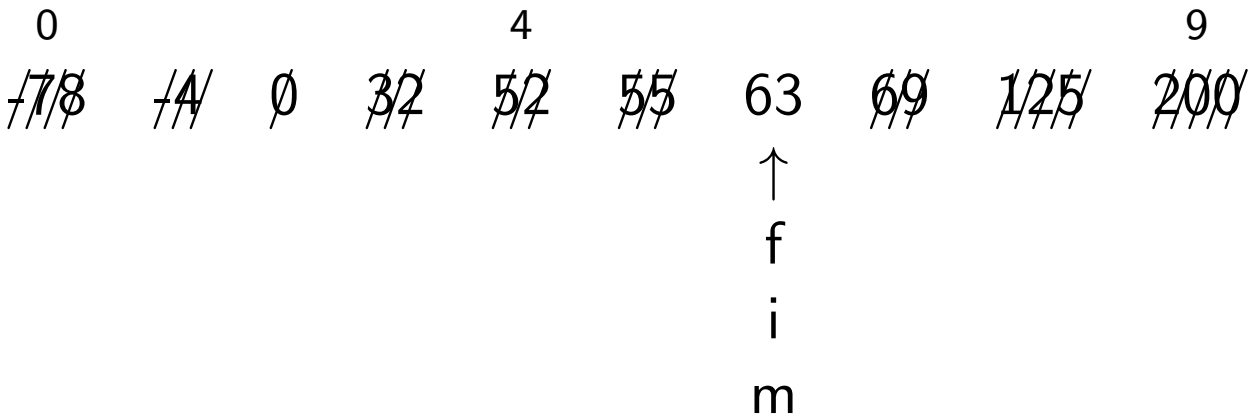
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



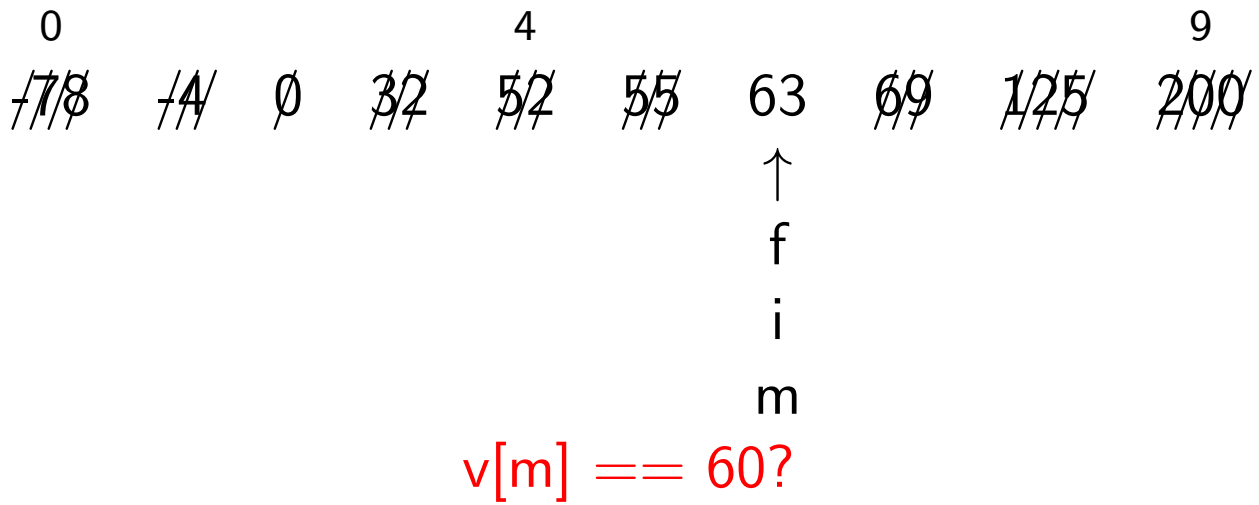
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



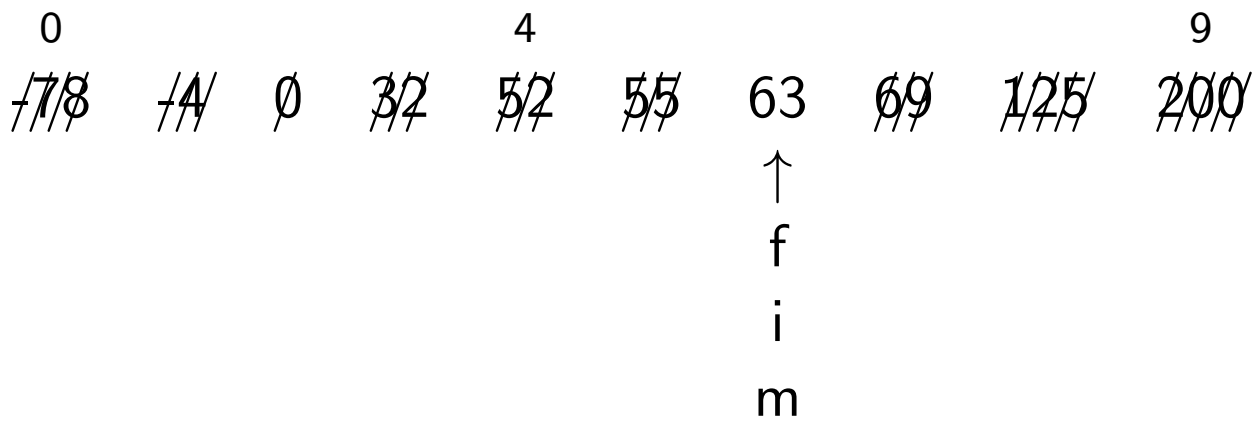
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



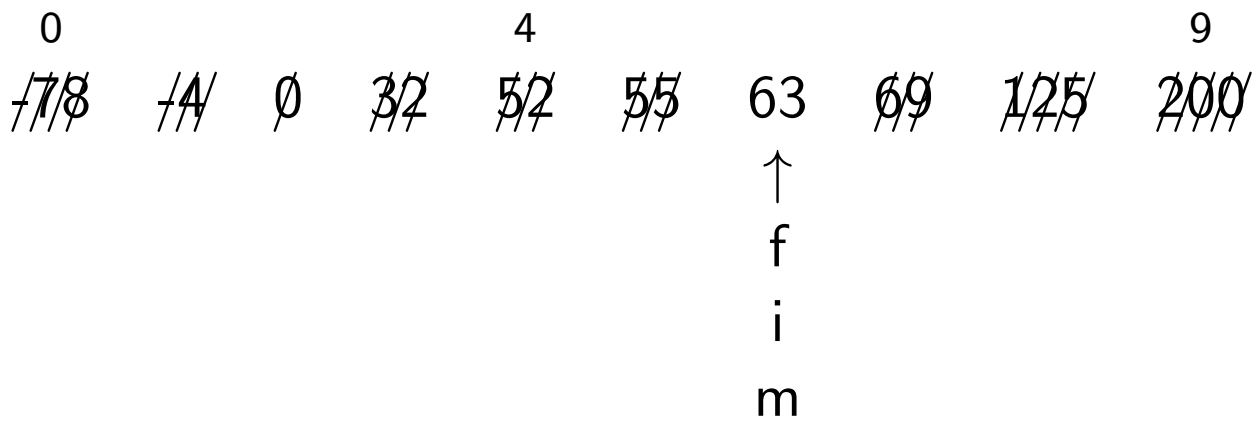
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



Busca Binária

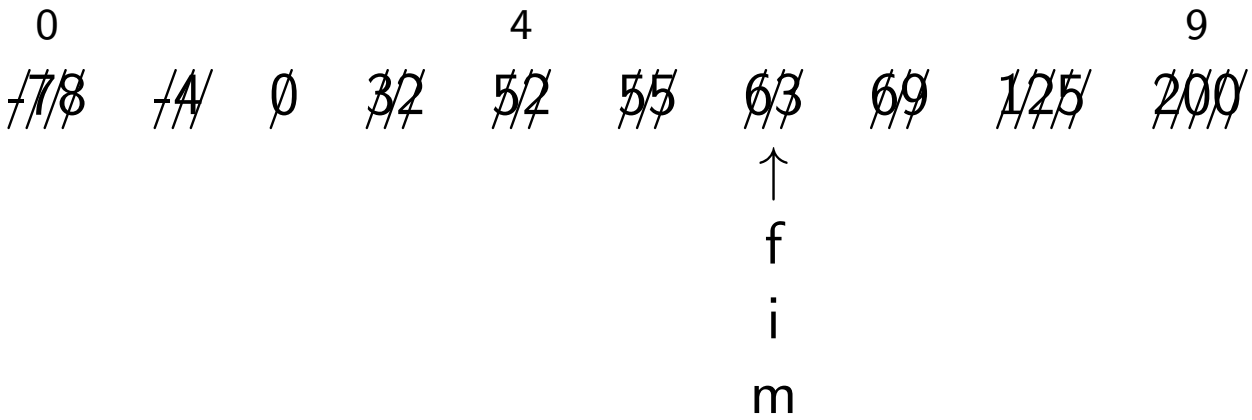
- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



v[m] == 60? Não. v[m] < 60? Não

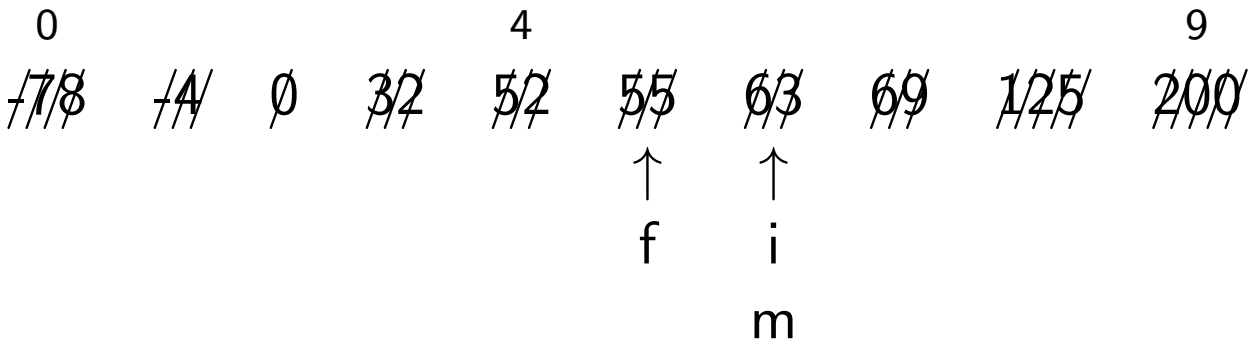
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



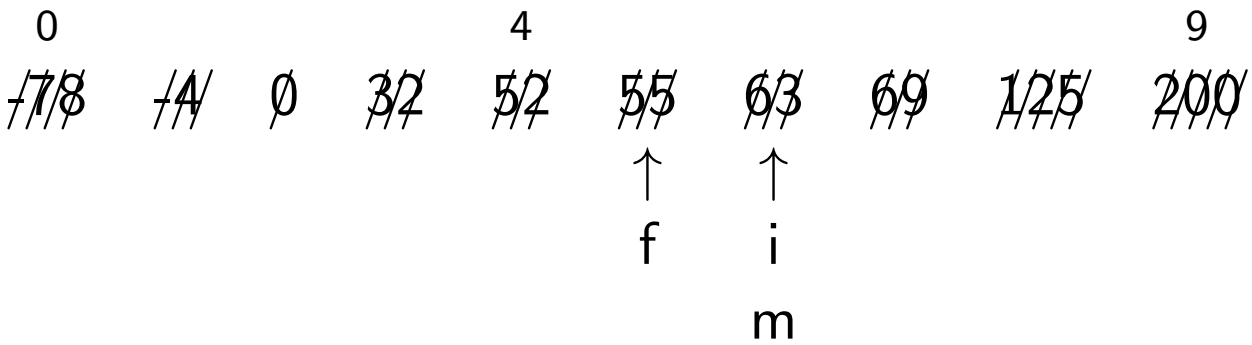
Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



Busca Binária

- Algoritmo:
  - Verifique se o elemento buscado é o do meio do arranjo
  - Se não for, verifique se é maior
    - ⋆ Se for, repita a busca na metade direita do arranjo
    - ⋆ Se não for, repita a busca na metade esquerda do arranjo
- Ex: Buscando 60



Inconsistência. O elemento não está no arranjo

Busca Binária

- Então...

```
static int buscaBin(int[] arr, int el) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio] < el) ini = meio + 1;
        else
            if (arr[meio] > el) fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

Busca Binária

- Então...
- Note que retornamos a posição no arranjo do elemento buscado, ou -1 em caso de erro

```
static int buscaBin(int[] arr, int el) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio] < el) ini = meio + 1;
        else
            if (arr[meio] > el) fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

Busca Binária

- Então...
- Note que retornamos a posição no arranjo do elemento buscado, ou -1 em caso de erro
- E sua versão com objetos?

```
static int buscaBin(int[] arr, int el) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio] < el) ini = meio + 1;
        else
            if (arr[meio] > el) fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

Busca Binária

- Então...
- Note que retornamos a posição no arranjo do elemento buscado, ou -1 em caso de erro
- E sua versão com objetos?

```
static int buscaBin(int[] arr, int el) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio] < el) ini = meio + 1;
        else
            if (arr[meio] > el) fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

```
static int buscaBin(Residencia[] arr,
                                double area) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio].area() < area)
            ini = meio + 1;
        else
            if (arr[meio].area() > area)
                fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

Busca Binária

- Então...
- Note que retornamos a posição no arranjo do elemento buscado, ou -1 em caso de erro
- E sua versão com objetos?
  - Note os nomes iguais

```
static int buscaBin(int[] arr, int el) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio] < el) ini = meio + 1;
        else
            if (arr[meio] > el) fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

```
static int buscaBin(Residencia[] arr,
                                double area) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio].area() < area)
            ini = meio + 1;
        else
            if (arr[meio].area() > area)
                fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

Busca Binária

- Então...
- Note que retornamos a posição no arranjo do elemento buscado, ou -1 em caso de erro
- E sua versão com objetos?
  - Note os nomes iguais
    - ★ Funciona porque os parâmetros são diferentes

```
static int buscaBin(int[] arr, int el) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio] < el) ini = meio + 1;
        else
            if (arr[meio] > el) fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

```
static int buscaBin(Residencia[] arr,
                                double area) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio].area() < area)
            ini = meio + 1;
        else
            if (arr[meio].area() > area)
                fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

Busca Binária

- Então...
- Note que retornamos a posição no arranjo do elemento buscado, ou -1 em caso de erro
- E sua versão com objetos?
  - Note os nomes iguais
    - ★ Funciona porque os parâmetros são diferentes
  - Nesse caso, procuramos alguma residência com aquela área

```
static int buscaBin(int[] arr, int el) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio] < el) ini = meio + 1;
        else
            if (arr[meio] > el) fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

```
static int buscaBin(Residencia[] arr,
                                double area) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio].area() < area)
            ini = meio + 1;
        else
            if (arr[meio].area() > area)
                fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

Busca Binária

- Então...
- Note que retornamos a posição no arranjo do elemento buscado, ou -1 em caso de erro
- E sua versão com objetos?
  - Note os nomes iguais
    - ★ Funciona porque os parâmetros são diferentes
  - Nesse caso, procuramos alguma residência com aquela área
  - Para residências específicas, teríamos que criar um atributo id, único, para elas

```
static int buscaBin(int[] arr, int el) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio] < el) ini = meio + 1;
        else
            if (arr[meio] > el) fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

```
static int buscaBin(Residencia[] arr,
                                double area) {
    int fim = arr.length-1;
    int ini = 0;
    while (ini <= fim) {
        int meio = (fim + ini)/2;
        if (arr[meio].area() < area)
            ini = meio + 1;
        else
            if (arr[meio].area() > area)
                fim = meio - 1;
            else return(meio);
    }
    return(-1);
}
```

Busca Binária

- Sabemos que fazemos um máximo de n comparações com busca seqüencial

Busca Binária

- Sabemos que fazemos um máximo de n comparações com busca seqüencial
  - Onde n é o número de elementos do arranjo

Busca Binária

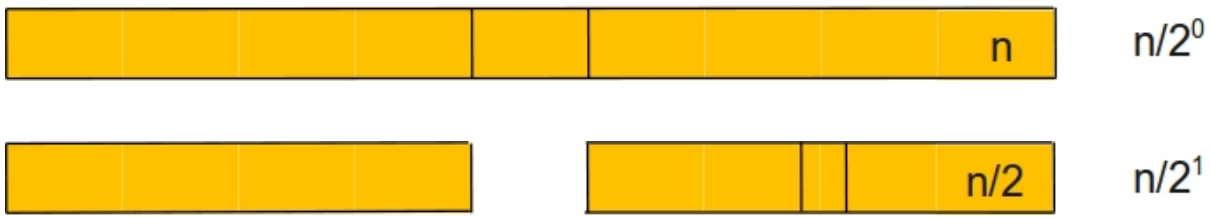
- Sabemos que fazemos um máximo de n comparações com busca seqüencial
  - Onde n é o número de elementos do arranjo
- E quantas fazemos no caso da binária?

Busca Binária

- Sabemos que fazemos um máximo de n comparações com busca seqüencial
  - Onde n é o número de elementos do arranjo
- E quantas fazemos no caso da binária?

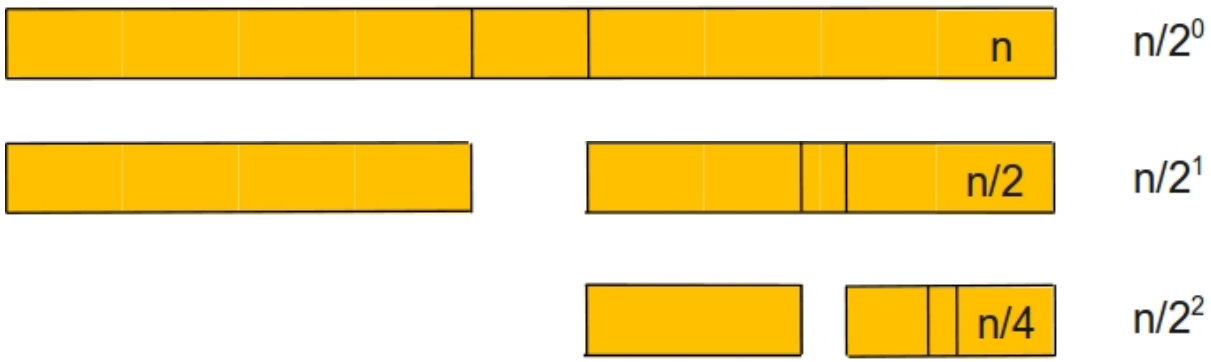
Busca Binária

- Sabemos que fazemos um máximo de n comparações com busca seqüencial
  - Onde n é o número de elementos do arranjo
- E quantas fazemos no caso da binária?



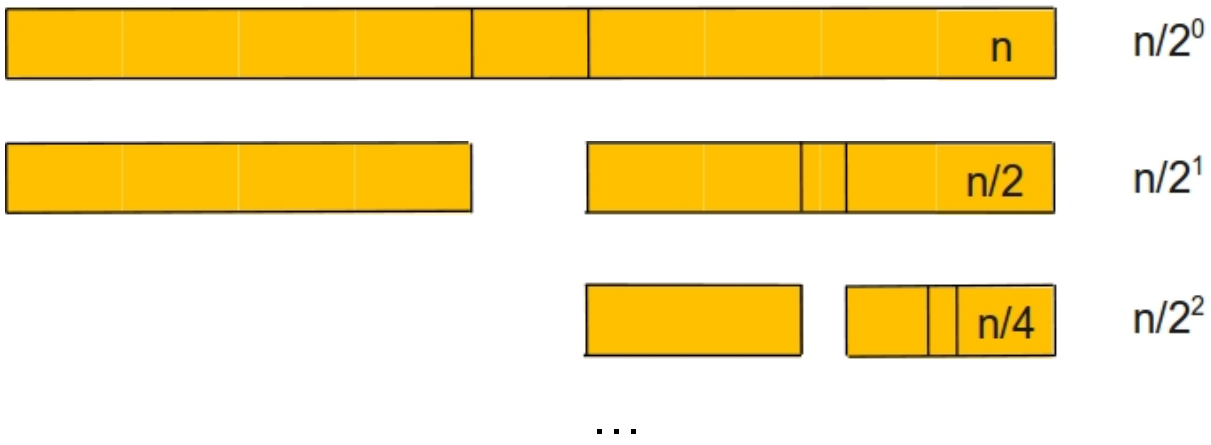
Busca Binária

- Sabemos que fazemos um máximo de n comparações com busca seqüencial
  - Onde n é o número de elementos do arranjo
- E quantas fazemos no caso da binária?



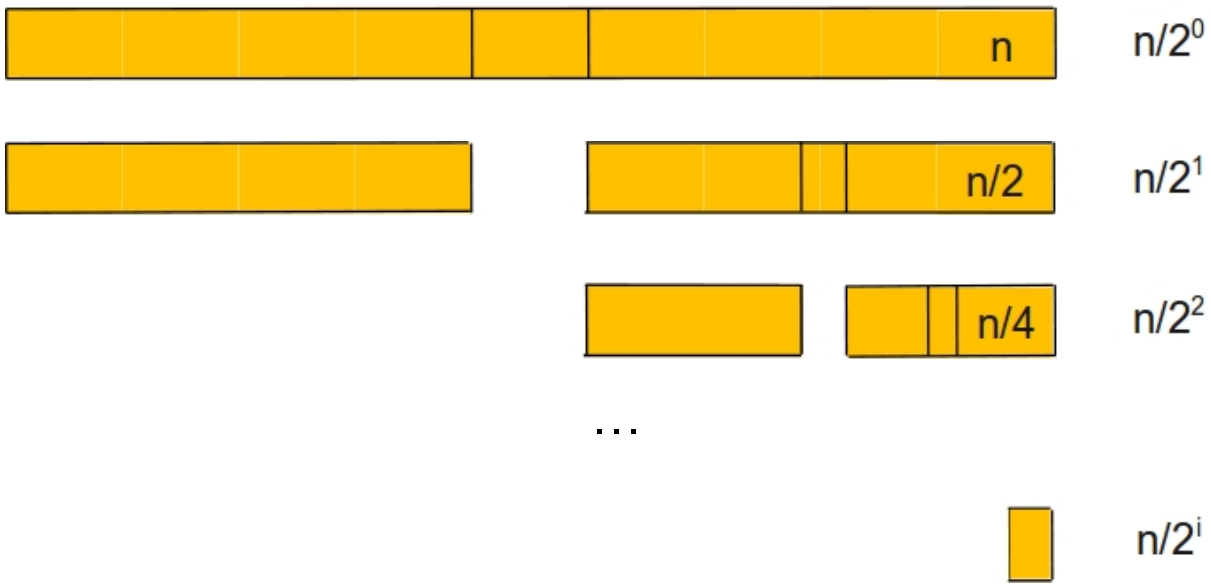
Busca Binária

- Sabemos que fazemos um máximo de n comparações com busca seqüencial
  - Onde n é o número de elementos do arranjo
- E quantas fazemos no caso da binária?

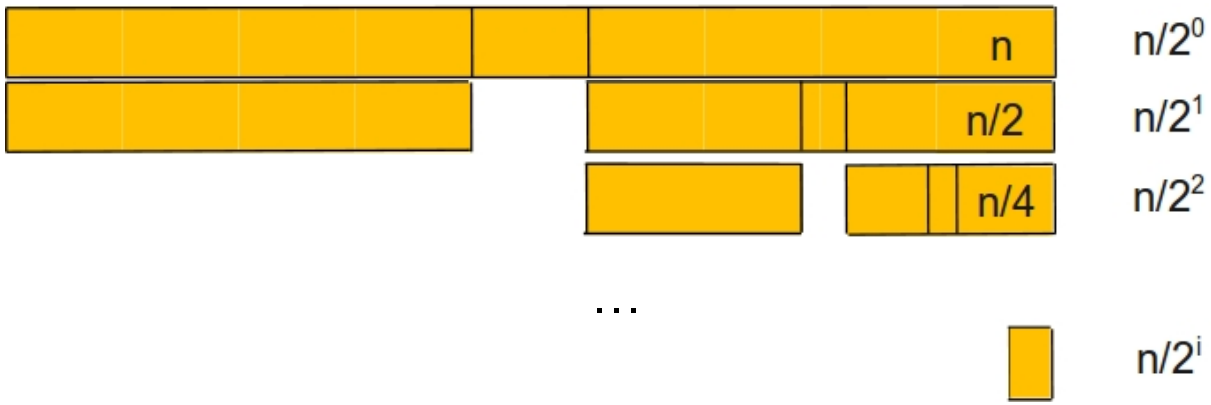


Busca Binária

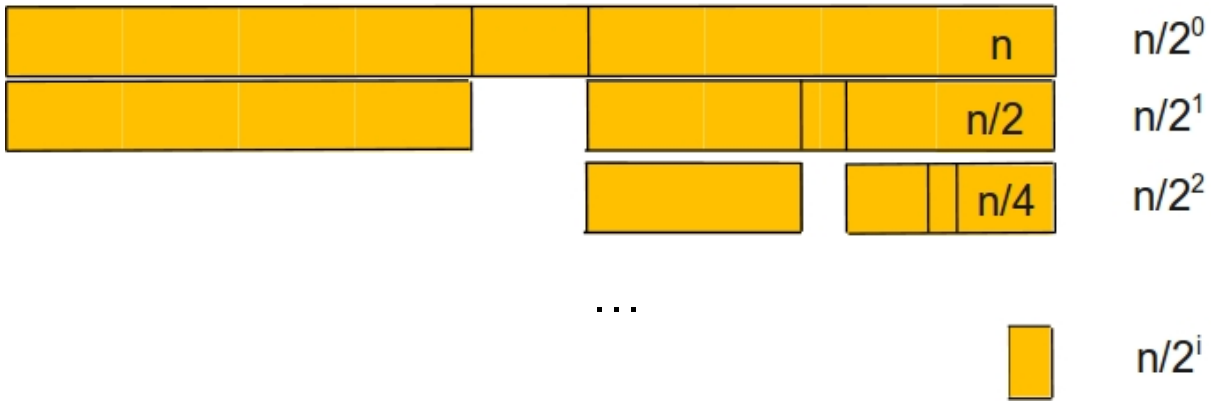
- Sabemos que fazemos um máximo de n comparações com busca seqüencial
  - Onde n é o número de elementos do arranjo
- E quantas fazemos no caso da binária?



Busca Binária

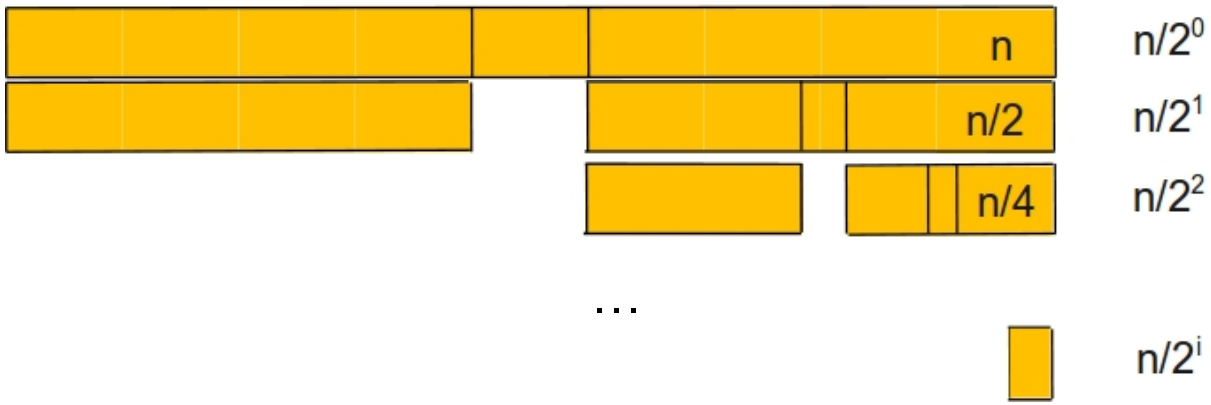


Busca Binária



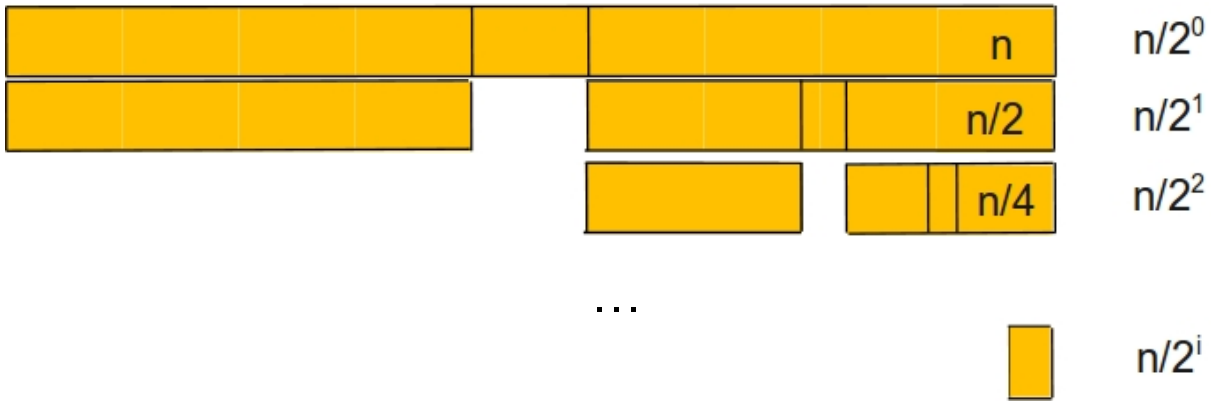
- Temos $i + 1$ comparações, sendo a última feita com o arranjo de tamanho 1

Busca Binária



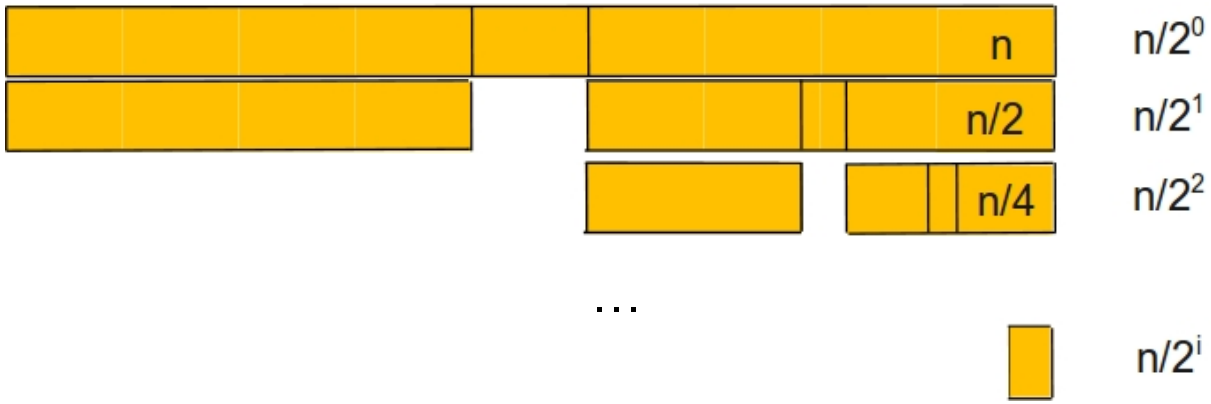
- Temos $i + 1$ comparações, sendo a última feita com o arranjo de tamanho 1
  - A relação entre n e i é tal que, após cada comparação, o arranjo terá $n/2^i$ elementos.

Busca Binária



- Temos $i + 1$ comparações, sendo a última feita com o arranjo de tamanho 1
  - A relação entre n e i é tal que, após cada comparação, o arranjo terá $n/2^i$ elementos.
  - Como no último nível há 1 elemento, então $n/2^i = 1 \Rightarrow n = 2^i \Rightarrow lg_2(n) = i$

Busca Binária



- Temos $i + 1$ comparações, sendo a última feita com o arranjo de tamanho 1
  - A relação entre n e i é tal que, após cada comparação, o arranjo terá $n/2^i$ elementos.
  - Como no último nível há 1 elemento, então $n/2^i = 1 \Rightarrow n = 2^i \Rightarrow lg_2(n) = i$
- Assim temos $lg_2(n) + 1$ comparações

Busca Seqüencial × Busca Binária

- Seqüencial
- Binária

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
- Binária

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ⋆ 1 comparação (arranjo ordenado ou não)
- Binária

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
- Binária
  - Melhor caso: O elemento é o do meio

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
- Binária
  - Melhor caso: O elemento é o do meio
    - ★ 1 comparação

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
  - Pior caso: O elemento não está no arranjo e é maior que todos
- Binária
  - Melhor caso: O elemento é o do meio
    - ★ 1 comparação

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
  - Pior caso: O elemento não está no arranjo e é maior que todos
    - ★ n comparações (arranjo ordenado ou não)
- Binária
  - Melhor caso: O elemento é o do meio
    - ★ 1 comparação

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
  - Pior caso: O elemento não está no arranjo e é maior que todos
    - ★ n comparações (arranjo ordenado ou não)
- Binária
  - Melhor caso: O elemento é o do meio
    - ★ 1 comparação
  - Pior caso: O elemento não está no arranjo

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
  - Pior caso: O elemento não está no arranjo e é maior que todos
    - ★ n comparações (arranjo ordenado ou não)
- Binária
  - Melhor caso: O elemento é o do meio
    - ★ 1 comparação
  - Pior caso: O elemento não está no arranjo
    - ★ $log_2(n) + 1$ comparações

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
  - Pior caso: O elemento não está no arranjo e é maior que todos
    - ★ n comparações (arranjo ordenado ou não)
- Binária
  - Melhor caso: O elemento é o do meio
    - ★ 1 comparação
  - Pior caso: O elemento não está no arranjo
    - ★ $log_2(n) + 1$ comparações

- $log_2(n) + 1 < n$ para $n \geq 3$ (para 1 e 2 $log_2(n) + 1 = n$)

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
  - Pior caso: O elemento não está no arranjo e é maior que todos
    - ★ n comparações (arranjo ordenado ou não)
- Binária
  - Melhor caso: O elemento é o do meio
    - ★ 1 comparação
  - Pior caso: O elemento não está no arranjo
    - ★ $log_2(n) + 1$ comparações

- $log_2(n) + 1 < n$ para $n \geq 3$ (para 1 e 2 $log_2(n) + 1 = n$)
  - No pior caso, a busca binária é pelo menos tão boa quanto a seqüencial, mas apenas para arranjos de tamanho mínimo.

Busca Seqüencial × Busca Binária

- Seqüencial
  - Melhor caso: O elemento é o primeiro
    - ★ 1 comparação (arranjo ordenado ou não)
  - Pior caso: O elemento não está no arranjo e é maior que todos
    - ★ n comparações (arranjo ordenado ou não)
- Binária
  - Melhor caso: O elemento é o do meio
    - ★ 1 comparação
  - Pior caso: O elemento não está no arranjo
    - ★ $log_2(n) + 1$ comparações

- $log_2(n) + 1 < n$ para $n \geq 3$ (para 1 e 2 $log_2(n) + 1 = n$)
  - No pior caso, a busca binária é pelo menos tão boa quanto a seqüencial, mas apenas para arranjos de tamanho mínimo.
  - Para os demais, ela é melhor

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso
- Importância: decisão de qual algoritmo usar dependendo do requisito do seu sistema

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso
- Importância: decisão de qual algoritmo usar dependendo do requisito do seu sistema
- Exemplo:

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso
- Importância: decisão de qual algoritmo usar dependendo do requisito do seu sistema
- Exemplo:
  - Lista telefônica de São Paulo: 18 milhões de entradas

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso
- Importância: decisão de qual algoritmo usar dependendo do requisito do seu sistema
- Exemplo:
  - Lista telefônica de São Paulo: 18 milhões de entradas
  - Se cada comparação (a um elemento do vetor) gasta 10 μs (10 milionésimos de segundo), como ficam os piores casos?

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso
- Importância: decisão de qual algoritmo usar dependendo do requisito do seu sistema
- Exemplo:
  - Lista telefônica de São Paulo: 18 milhões de entradas
  - Se cada comparação (a um elemento do vetor) gasta 10 μs (10 milionésimos de segundo), como ficam os piores casos?
    - ★ Busca sequencial:

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso
- Importância: decisão de qual algoritmo usar dependendo do requisito do seu sistema
- Exemplo:
  - Lista telefônica de São Paulo: 18 milhões de entradas
  - Se cada comparação (a um elemento do vetor) gasta 10 μs (10 milionésimos de segundo), como ficam os piores casos?
    - ★ Busca sequencial: 10/1000000 * 18000000 = 180s = 3 minutos

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso
- Importância: decisão de qual algoritmo usar dependendo do requisito do seu sistema
- Exemplo:
  - Lista telefônica de São Paulo: 18 milhões de entradas
  - Se cada comparação (a um elemento do vetor) gasta 10 μs (10 milionésimos de segundo), como ficam os piores casos?
    - ★ Busca sequencial: 10/1000000 * 18000000 = 180s = 3 minutos
    - ★ Busca binária:

Complexidade Computacional

- Estudo do esforço computacional despendido para que o algoritmo seja executado
- Pode ser avaliado o melhor ou o pior caso, ou ainda o caso médio.
  - Normalmente é considerado o pior caso
- Importância: decisão de qual algoritmo usar dependendo do requisito do seu sistema
- Exemplo:
  - Lista telefônica de São Paulo: 18 milhões de entradas
  - Se cada comparação (a um elemento do vetor) gasta 10 μs (10 milionésimos de segundo), como ficam os piores casos?
    - ★ Busca sequencial: 10/1000000 * 18000000 = 180s = 3 minutos
    - ★ Busca binária: 10/1000000 * log2 18000000 = 0.000241s = 0.24 milisegundos

Quando Usar Busca Binária?

- Qual o problema da busca binária?

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?
  - Quando há muitas buscas

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?
  - Quando há muitas buscas
  - Quando os dados sofrem pouca alteração na chave de busca

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?
  - Quando há muitas buscas
  - Quando os dados sofrem pouca alteração na chave de busca
    - ★ A chave de busca é o que se quer buscar

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?
  - Quando há muitas buscas
  - Quando os dados sofrem pouca alteração na chave de busca
    - ★ A chave de busca é o que se quer buscar
    - ★ Ex: no caso das residências, era a área

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?
  - Quando há muitas buscas
  - Quando os dados sofrem pouca alteração na chave de busca
    - ★ A chave de busca é o que se quer buscar
    - ★ Ex: no caso das residências, era a área
    - ★ Se a área ficar mudando, muda a ordem e o arranjo deve ser reordenado

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?
  - Quando há muitas buscas
  - Quando os dados sofrem pouca alteração na chave de busca
    - ★ A chave de busca é o que se quer buscar
    - ★ Ex: no caso das residências, era a área
    - ★ Se a área ficar mudando, muda a ordem e o arranjo deve ser reordenado
    - ★ Gera um custo extra que acaba se sobrepondo ao ganho com a busca

Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?
  - Quando há muitas buscas
  - Quando os dados sofrem pouca alteração na chave de busca
    - ★ A chave de busca é o que se quer buscar
    - ★ Ex: no caso das residências, era a área
    - ★ Se a área ficar mudando, muda a ordem e o arranjo deve ser reordenado
    - ★ Gera um custo extra que acaba se sobrepondo ao ganho com a busca
  - Quando as inserções/deleções não são freqüentes

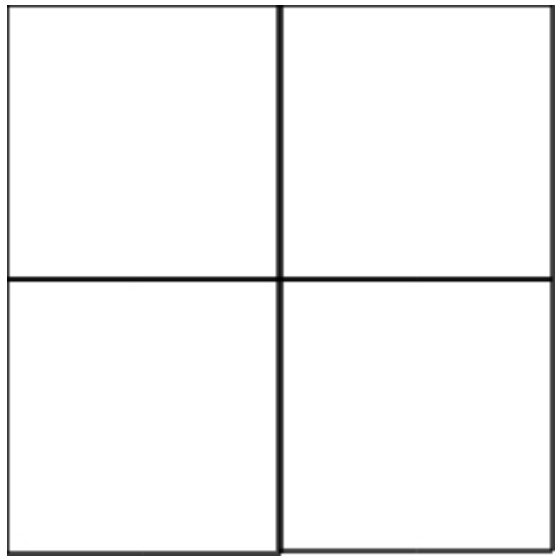
Quando Usar Busca Binária?

- Qual o problema da busca binária?
  - Precisa que o vetor esteja ordenado
- Portanto, quando ela realmente ajuda?
  - Quando há muitas buscas
  - Quando os dados sofrem pouca alteração na chave de busca
    - ★ A chave de busca é o que se quer buscar
    - ★ Ex: no caso das residências, era a área
    - ★ Se a área ficar mudando, muda a ordem e o arranjo deve ser reordenado
    - ★ Gera um custo extra que acaba se sobrepondo ao ganho com a busca
  - Quando as inserções/deleções não são freqüentes
    - ★ Em suma, quando a ordem não é mudada com freqüência

Herança

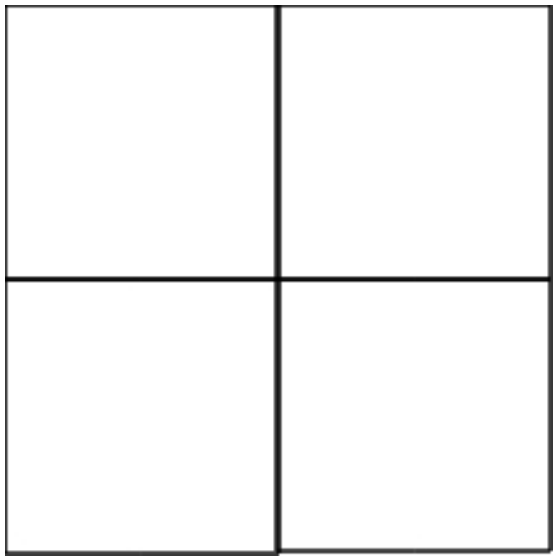
Superclasses e Subclasses

- Temos agora outro tipo de cabana, com outro formato.



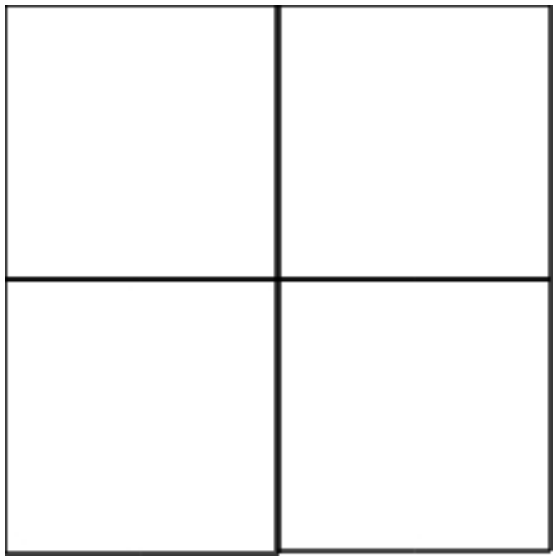
Superclasses e Subclasses

- Temos agora outro tipo de cabana, com outro formato.
- Queremos calcular também a área e custo



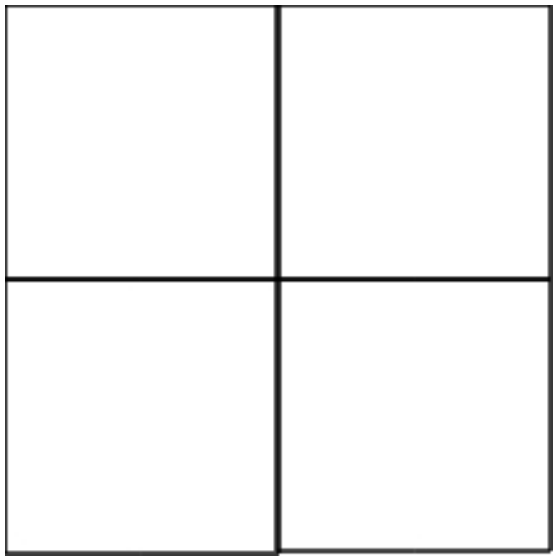
Superclasses e Subclasses

- Temos agora outro tipo de cabana, com outro formato.
- Queremos calcular também a área e custo
- Quais os dados necessários?



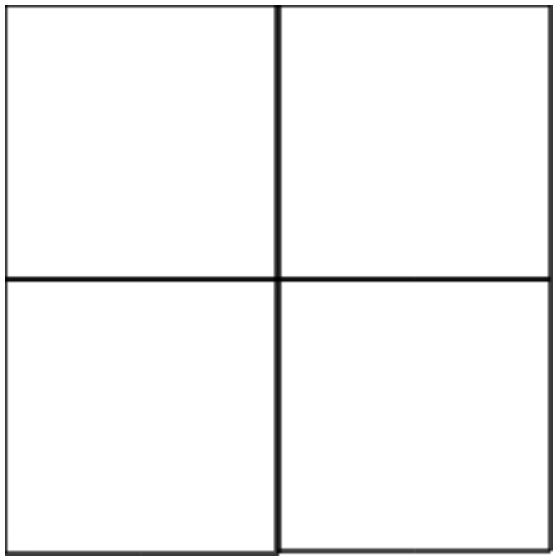
Superclasses e Subclasses

- Temos agora outro tipo de cabana, com outro formato.
- Queremos calcular também a área e custo
- Quais os dados necessários?
  - Lateral da cabana



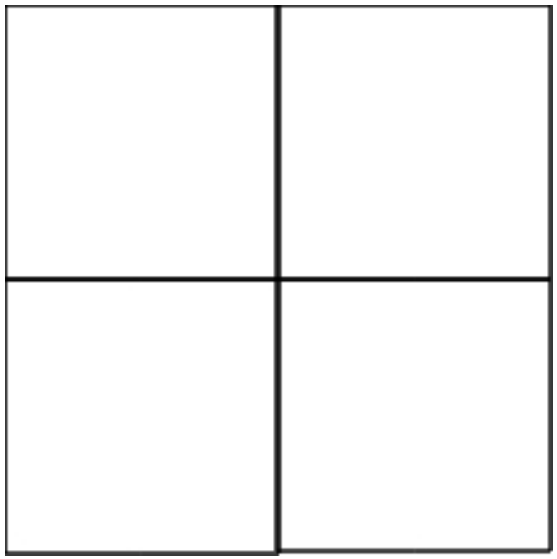
Superclasses e Subclasses

- Temos agora outro tipo de cabana, com outro formato.
- Queremos calcular também a área e custo
- Quais os dados necessários?
  - Lateral da cabana
  - Preço do m^2



Superclasses e Subclasses

- Temos agora outro tipo de cabana, com outro formato.
- Queremos calcular também a área e custo
- Quais os dados necessários?
  - Lateral da cabana
  - Preço do m^2
- Que fazer?



Herança

- Podemos construir uma classe para essa casa

Herança

- Podemos construir uma classe para essa casa
  - CasaQuad

Herança

- Podemos construir uma classe para essa casa
  - CasaQuad

```
class CasaQuad {
    double valorM2 = 1500;
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double lateral) {
        this.lateral = lateral;
    }

    CasaQuad(double lateral, double valorM2) {
        this(lateral);
        this.valorM2 = valorM2;
    }

    double area() {
        double areat=-1; // área total

        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }

    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

Herança

- Podemos construir uma classe para essa casa
  - CasaQuad
- Note que não há construtor do tipo:

```
CasaQuad(double valorM2) {
    this.valorM2 = valorM2;
}
```

```
class CasaQuad {
    double valorM2 = 1500;
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double lateral) {
        this.lateral = lateral;
    }

    CasaQuad(double lateral, double valorM2) {
        this(lateral);
        this.valorM2 = valorM2;
    }

    double area() {
        double areat=-1; // área total

        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }

    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

Herança

- Podemos construir uma classe para essa casa
  - CasaQuad
- Note que não há construtor do tipo:

```
CasaQuad(double valorM2) {
    this.valorM2 = valorM2;
}
```

- Se houvesse, ao compilarmos teríamos:

Linha de Comando

```
$ javac CasaQuad.java
CasaQuad.java:24: CasaQuad(double) is
          already defined in CasaQuad
    CasaQuad(double valorM2) {
    ^
1 error
```

```
class CasaQuad {
    double valorM2 = 1500;
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double lateral) {
        this.lateral = lateral;
    }

    CasaQuad(double lateral, double valorM2) {
        this(lateral);
        this.valorM2 = valorM2;
    }

    double area() {
        double areat=-1; // área total

        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }

    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

Comparando AreaCasa e CasaQuad

```
class AreaCasa {
    double valorM2 = 1500;
    double lateral = 10;
    double cquarto = 10;

    ... (construtores) ...

    double area() {
        double areat=-1;

        if (this.lateral>=0 && this.cquarto>=0) {
            areat = this.lateral*this.lateral;
            areat += this.cquarto*this.lateral;
        }
        return(areat);
    }

    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

```
class CasaQuad {
    double valorM2 = 1500;
    double lateral = 10;


    ... (construtores) ...

    double area() {
        double areat=-1; // área total

        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }


    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

Comparando AreaCasa e CasaQuad

```
class AreaCasa {
    double valorM2 = 1500;
    double lateral = 10;
    double cquarto = 10;

    ... (construtores) ...

    double area() {
        double areat=-1;

        if (this.lateral>=0 && this.cquarto>=0) {
            areat = this.lateral*this.lateral;
            areat += this.cquarto*this.lateral;
        }
        return(areat);
    }

    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

```
class CasaQuad {
    double valorM2 = 1500;
    double lateral = 10;


    ... (construtores) ...

    double area() {
        double areat=-1; // área total

        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }


    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

- Há vários pontos em comum

Comparando AreaCasa e CasaQuad

```
class AreaCasa {
    double valorM2 = 1500;
    double lateral = 10;
    double cquarto = 10;

    ... (construtores) ...

    double area() {
        double areat=-1;

        if (this.lateral>=0 && this.cquarto>=0) {
            areat = this.lateral*this.lateral;
            areat += this.cquarto*this.lateral;
        }
        return(areat);
    }

    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

```
class CasaQuad {
    double valorM2 = 1500;
    double lateral = 10;


    ... (construtores) ...

    double area() {
        double areat=-1; // área total

        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }


    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

- Há vários pontos em comum
- E pontos bastante semelhantes

Comparando AreaCasa e CasaQuad

```
class AreaCasa {
    double valorM2 = 1500;
    double lateral = 10;
    double cquarto = 10;

    ... (construtores) ...

    double area() {
        double areat=-1;

        if (this.lateral>=0 && this.cquarto>=0) {
            areat = this.lateral*this.lateral;
            areat += this.cquarto*this.lateral;
        }
        return(areat);
    }

    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

```
class CasaQuad {
    double valorM2 = 1500;
    double lateral = 10;


    ... (construtores) ...

    double area() {
        double areat=-1; // área total

        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }


    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

- Há vários pontos em comum
- E pontos bastante semelhantes
  - Mesma assinatura, mas comportamento ou interpretação diferentes

Herança

- Quase todo o código das classes ou está repetido ou muito semelhante

Herança

- Quase todo o código das classes ou está repetido ou muito semelhante
  - Repetição do trabalho

Herança

- Quase todo o código das classes ou está repetido ou muito semelhante
  - Repetição do trabalho
  - Desperdício de tempo de desenvolvimento

Herança

- Quase todo o código das classes ou está repetido ou muito semelhante
  - Repetição do trabalho
  - Desperdício de tempo de desenvolvimento
  - Desperdício de tempo de validação do código (teste das mesmas coisas....)

Herança

- Quase todo o código das classes ou está repetido ou muito semelhante
  - Repetição do trabalho
  - Desperdício de tempo de desenvolvimento
  - Desperdício de tempo de validação do código (teste das mesmas coisas....)
  - Mudanças em uma parte comum a elas devem ser feitas (e testadas novamente) nas duas classes!

Herança

- Quase todo o código das classes ou está repetido ou muito semelhante
  - Repetição do trabalho
  - Desperdício de tempo de desenvolvimento
  - Desperdício de tempo de validação do código (teste das mesmas coisas....)
  - Mudanças em uma parte comum a elas devem ser feitas (e testadas novamente) nas duas classes!
- Há uma solução?

Herança

- Quase todo o código das classes ou está repetido ou muito semelhante
  - Repetição do trabalho
  - Desperdício de tempo de desenvolvimento
  - Desperdício de tempo de validação do código (teste das mesmas coisas....)
  - Mudanças em uma parte comum a elas devem ser feitas (e testadas novamente) nas duas classes!
- Há uma solução?
  - Agrupar o que for comum (exatamente igual) em ambas em uma nova classe

Herança

- Quase todo o código das classes ou está repetido ou muito semelhante
  - Repetição do trabalho
  - Desperdício de tempo de desenvolvimento
  - Desperdício de tempo de validação do código (teste das mesmas coisas....)
  - Mudanças em uma parte comum a elas devem ser feitas (e testadas novamente) nas duas classes!
- Há uma solução?
  - Agrupar o que for comum (exatamente igual) em ambas em uma nova classe
  - Ambas então compartilhariam das definições (não necessariamente memória) contidas nessa nova classe

Herança

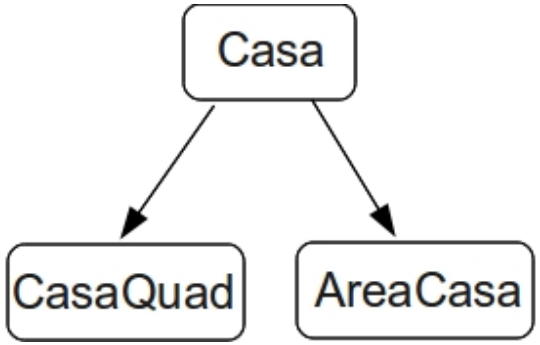
- Quase todo o código das classes ou está repetido ou muito semelhante
  - Repetição do trabalho
  - Desperdício de tempo de desenvolvimento
  - Desperdício de tempo de validação do código (teste das mesmas coisas....)
  - Mudanças em uma parte comum a elas devem ser feitas (e testadas novamente) nas duas classes!
- Há uma solução?
  - Agrupar o que for comum (exatamente igual) em ambas em uma nova classe
  - Ambas então compartilhariam das definições (não necessariamente memória) contidas nessa nova classe
    - ★ Ambas herdariam essas definições

Herança

- Esse esquema dá origem a uma hierarquia de classes

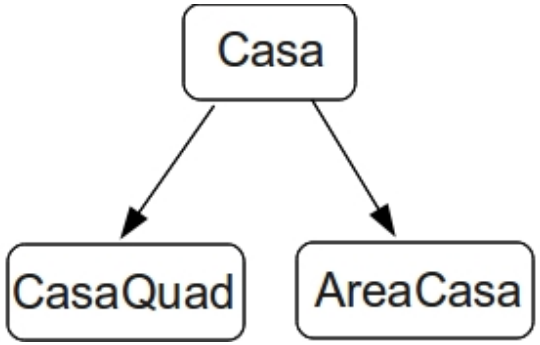
Herança

- Esse esquema dá origem a uma hierarquia de classes



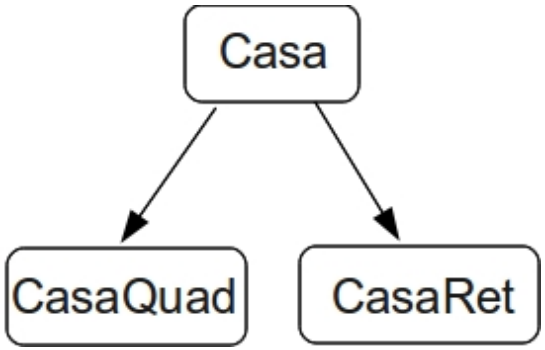
Herança

- Esse esquema dá origem a uma hierarquia de classes
  - Nela, a nova classe – Casa – está acima de CasaQuad e AreaCasa



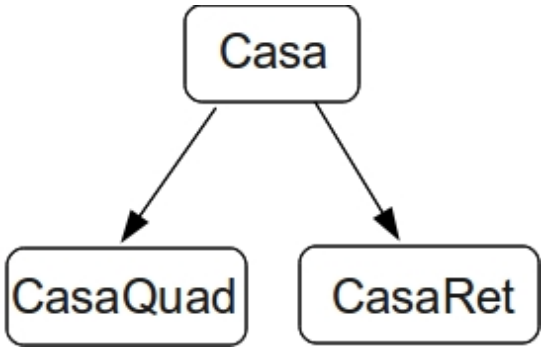
Herança

- Esse esquema dá origem a uma hierarquia de classes
  - Nela, a nova classe – Casa – está acima de CasaQuad e AreaCasa
  - Para manter o padrão, podemos rebatizar AreaCasa para CasaRet



Herança

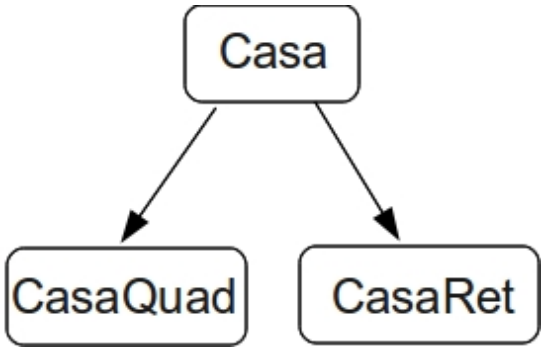
- Esse esquema dá origem a uma hierarquia de classes
  - Nela, a nova classe – Casa – está acima de CasaQuad e AreaCasa
  - Para manter o padrão, podemos rebatizar AreaCasa para CasaRet



- Diz-se que CasaQuad e CasaRet são subclasses de Casa

Herança

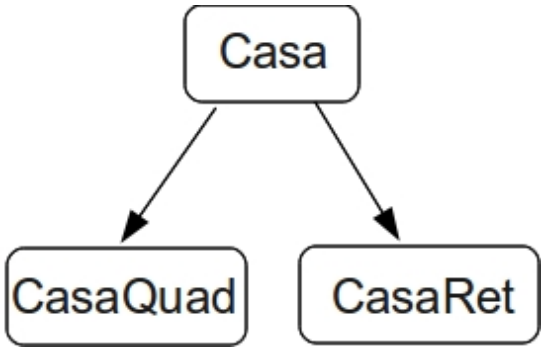
- Esse esquema dá origem a uma hierarquia de classes
  - Nela, a nova classe – Casa – está acima de CasaQuad e AreaCasa
  - Para manter o padrão, podemos rebatizar AreaCasa para CasaRet



- Diz-se que CasaQuad e CasaRet são subclasses de Casa
  - E que Casa é superclasse de CasaQuad e CasaRet

Herança

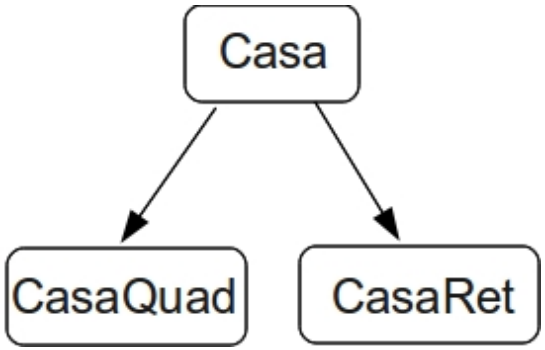
- Esse esquema dá origem a uma hierarquia de classes
  - Nela, a nova classe – Casa – está acima de CasaQuad e AreaCasa
  - Para manter o padrão, podemos rebatizar AreaCasa para CasaRet



- Diz-se que CasaQuad e CasaRet são subclasses de Casa
  - E que Casa é superclasse de CasaQuad e CasaRet
  - Classes mais especializadas (subclasses) herdam propriedades (atributos e código) da classe mais geral (superclasse)

Herança

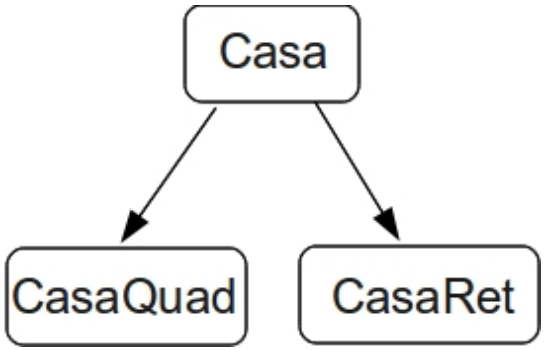
- Esse esquema dá origem a uma hierarquia de classes
  - Nela, a nova classe – Casa – está acima de CasaQuad e AreaCasa
  - Para manter o padrão, podemos rebatizar AreaCasa para CasaRet



- Diz-se que CasaQuad e CasaRet são subclasses de Casa
  - E que Casa é superclasse de CasaQuad e CasaRet
  - Classes mais especializadas (subclasses) herdam propriedades (atributos e código) da classe mais geral (superclasse)
  - Cria-se uma subclasse inserindo somente as diferenças desta para sua superclasse

Herança

- Esse esquema dá origem a uma hierarquia de classes
  - Nela, a nova classe – Casa – está acima de CasaQuad e AreaCasa
  - Para manter o padrão, podemos rebatizar AreaCasa para CasaRet



- Diz-se que CasaQuad e CasaRet são subclasses de Casa
  - E que Casa é superclasse de CasaQuad e CasaRet
  - Classes mais especializadas (subclasses) herdam propriedades (atributos e código) da classe mais geral (superclasse)
  - Cria-se uma subclasse inserindo somente as diferenças desta para sua superclasse
  - Identifica-se a possibilidade de herança por meio da seguinte expressão: "é um tipo de"

Classes e Subclasses

- Como criamos Casa?

Classes e Subclasses

- Como criamos Casa?
  - Separando os atributos e métodos idênticos em CasaQuad e CasaRet

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

Classes e Subclasses

- Como criamos Casa?
  - Separando os atributos e métodos idênticos em CasaQuad e CasaRet
- E CasaQuad?

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

Classes e Subclasses

- Como criamos Casa?
  - Separando os atributos e métodos idênticos em CasaQuad e CasaRet
- E CasaQuad?
  - Inserindo somente as diferenças

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                          area);
        return(-1);
    }
}
```

```
class CasaQuad extends Casa {
    /* comprimento da lateral (casa) */
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaQuad(double lateral, double valorM2) {
        this(valorM2);
        this.lateral = lateral;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- Como criamos Casa?
  - Separando os atributos e métodos idênticos em CasaQuad e CasaRet
- E CasaQuad?
  - Inserindo somente as diferenças
  - E dizendo ao compilador quem é a classe mãe (superclasse)

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                          area);
        return(-1);
    }
}
```

```
class CasaQuad extends Casa {
    /* comprimento da lateral (casa) */
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaQuad(double lateral, double valorM2) {
        this(valorM2);
        this.lateral = lateral;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- Como criamos Casa?
  - Separando os atributos e métodos idênticos em CasaQuad e CasaRet
- E CasaQuad?
  - Inserindo somente as diferenças
  - E dizendo ao compilador quem é a classe mãe (superclasse)

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

- Não cria um objeto extra para a superclasse

```
class CasaQuad extends Casa {
    /* comprimento da lateral (casa) */
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaQuad(double lateral, double valorM2) {
        this(valorM2);
        this.lateral = lateral;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- Como criamos Casa?
  - Separando os atributos e métodos idênticos em CasaQuad e CasaRet
- E CasaQuad?
  - Inserindo somente as diferenças
  - E dizendo ao compilador quem é a classe mãe (superclasse)

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

- Não cria um objeto extra para a superclasse
  - É como se copiasse os atributos desta para dentro da memória da subclasse

```
class CasaQuad extends Casa {
    /* comprimento da lateral (casa) */
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaQuad(double lateral, double valorM2) {
        this(valorM2);
        this.lateral = lateral;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- E por que lateral, embora comum a CasaRet e CasaQuad, fo colocada na subclasse?

```
class CasaQuad extends Casa {
    /* comprimento da lateral (casa) */
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaQuad(double lateral, double valorM2) {
        this(valorM2);
        this.lateral = lateral;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- E por que lateral, embora comum a CasaRet e CasaQuad, fo colocada na subclasse?
  - Porque tem significados diferentes: Lateral da casa e da sala

```
class CasaQuad extends Casa {
    /* comprimento da lateral (casa) */
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaQuad(double lateral, double valorM2) {
        this(valorM2);
        this.lateral = lateral;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- E por que lateral, embora comum a CasaRet e CasaQuad, fo colocada na subclasse?
  - Porque tem significados diferentes: Lateral da casa e da sala
  - Poderia também haver uma casa redonda, em que esse atributo não faria sentido

```
class CasaQuad extends Casa {
    /* comprimento da lateral (casa) */
    double lateral = 10;

    CasaQuad() {}

    CasaQuad(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaQuad(double lateral, double valorM2) {
        this(valorM2);
        this.lateral = lateral;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0) {
            areat = this.lateral*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- CasaRet sai de maneira semelhante:

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

```
class CasaRet extends Casa {
    double cquarto = 10;
    /* comprimento da lateral (sala) */
    double lateral = 10;

    CasaRet() {}

    CasaRet(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaRet(double lateral, double cquarto) {
        this.lateral = lateral;
        this.cquarto = cquarto;
    }

    CasaRet(double lateral, double cquarto,
                              double valorM2) {
        this(lateral, cquarto);
        this.valorM2 = valorM2;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0 && this.cquarto>=0) {
            areat = this.lateral*this.lateral;
            areat += this.cquarto*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- CasaRet sai de maneira semelhante:

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

  - CasaQuad é um tipo de Casa

```
class CasaRet extends Casa {
    double cquarto = 10;
    /* comprimento da lateral (sala) */
    double lateral = 10;

    CasaRet() {}

    CasaRet(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaRet(double lateral, double cquarto) {
        this.lateral = lateral;
        this.cquarto = cquarto;
    }

    CasaRet(double lateral, double cquarto,
                              double valorM2) {
        this(lateral, cquarto);
        this.valorM2 = valorM2;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0 && this.cquarto>=0) {
            areat = this.lateral*this.lateral;
            areat += this.cquarto*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- CasaRet sai de maneira semelhante:

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

  - CasaQuad é um tipo de Casa
  - CasaRet é um tipo de Casa

```
class CasaRet extends Casa {
    double cquarto = 10;
    /* comprimento da lateral (sala) */
    double lateral = 10;

    CasaRet() {}

    CasaRet(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaRet(double lateral, double cquarto) {
        this.lateral = lateral;
        this.cquarto = cquarto;
    }

    CasaRet(double lateral, double cquarto,
                              double valorM2) {
        this(lateral, cquarto);
        this.valorM2 = valorM2;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0 && this.cquarto>=0) {
            areat = this.lateral*this.lateral;
            areat += this.cquarto*this.lateral;
        }
        return(areat);
    }
}
```

Classes e Subclasses

- CasaRet sai de maneira semelhante:

```
class Casa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    double valor(double area) {
        if (area >= 0) return(this.valorM2*
                                        area);
        return(-1);
    }
}
```

  - CasaQuad é um tipo de Casa
  - CasaRet é um tipo de Casa
  - Identifica-se a possibilidade de herança por meio da expressão "é um tipo de"

```
class CasaRet extends Casa {
    double cquarto = 10;
    /* comprimento da lateral (sala) */
    double lateral = 10;

    CasaRet() {}

    CasaRet(double valorM2) {
        this.valorM2 = valorM2;
    }

    CasaRet(double lateral, double cquarto) {
        this.lateral = lateral;
        this.cquarto = cquarto;
    }

    CasaRet(double lateral, double cquarto,
                              double valorM2) {
        this(lateral, cquarto);
        this.valorM2 = valorM2;
    }

    double area() {
        double areat=-1;
        if (this.lateral>=0 && this.cquarto>=0) {
            areat = this.lateral*this.lateral;
            areat += this.cquarto*this.lateral;
        }
        return(areat);
    }
}
```

Herança

- E como usamos essa nova versão?

Herança

- E como usamos essa nova versão?
  - Como antes:

```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Área: "+cr.area());
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("M2: "+cr.valorM2);
    System.out.println("Valor: "+cr.valor(
                                    cr.area()));
    System.out.println();
    System.out.println("Área: "+cq.area());
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("M2: "+cq.valorM2);
    System.out.println("Valor: "+cq.valor(
                                    cq.area()));
}
```

Herança

- E como usamos essa nova versão?
  - Como antes:
- A saída será:

Linha de Comando

```
$ java Projeto
Quarto: 5.0
Área: 150.0
Lateral da sala: 10.0
M2: 1320.0
Valor: 198000.0

Área: 100.0
Lateral da casa: 10.0
M2: 1523.0
Valor: 152300.0
```

```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Área: "+cr.area());
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("M2: "+cr.valorM2);
    System.out.println("Valor: "+cr.valor(
                                    cr.area()));
    System.out.println();
    System.out.println("Área: "+cq.area());
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("M2: "+cq.valorM2);
    System.out.println("Valor: "+cq.valor(
                                    cq.area()));
}
```

Herança

- E como usamos essa nova versão?
  - Como antes:
- A saída será:

```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Área: "+cr.area());
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("M2: "+cr.valorM2);
    System.out.println("Valor: "+cr.valor(
                                    cr.area()));
    System.out.println();
    System.out.println("Área: "+cq.area());
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("M2: "+cq.valorM2);
    System.out.println("Valor: "+cq.valor(
                                    cq.area()));
}
```

Linha de Comando

```
$ java Projeto
Quarto: 5.0
Área: 150.0
Lateral da sala: 10.0
M2: 1320.0
Valor: 198000.0

Área: 100.0
Lateral da casa: 10.0
M2: 1523.0
Valor: 152300.0
```

  - Temos acesso a todos os métodos e atributos tanto da subclasse

Herança

- E como usamos essa nova versão?
  - Como antes:
- A saída será:

```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Área: "+cr.area());
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("M2: "+cr.valorM2);
    System.out.println("Valor: "+cr.valor(
                                    cr.area()));
    System.out.println();
    System.out.println("Área: "+cq.area());
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("M2: "+cq.valorM2);
    System.out.println("Valor: "+cq.valor(
                                    cq.area()));
}
```

Linha de Comando

```
$ java Projeto
Quarto: 5.0
Área: 150.0
Lateral da sala: 10.0
M2: 1320.0
Valor: 198000.0

Área: 100.0
Lateral da casa: 10.0
M2: 1523.0
Valor: 152300.0
```

  - Temos acesso a todos os métodos e atributos tanto da subclasse quanto da superclasse

Herança

- E como usamos essa nova versão?
  - Como antes:
- A saída será:

```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Área: "+cr.area());
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("M2: "+cr.valorM2);
    System.out.println("Valor: "+cr.valor(
                                    cr.area()));
    System.out.println();
    System.out.println("Área: "+cq.area());
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("M2: "+cq.valorM2);
    System.out.println("Valor: "+cq.valor(
                                    cq.area()));
}
```

Linha de Comando

```
$ java Projeto
Quarto: 5.0
Área: 150.0
Lateral da sala: 10.0
M2: 1320.0
Valor: 198000.0

Área: 100.0
Lateral da casa: 10.0
M2: 1523.0
Valor: 152300.0
```

- Temos acesso a todos os métodos e atributos tanto da subclasse quanto da superclasse
- Não há compartilhamento de memória (pois não há static) – cada um tem o seu

Herança e Memória

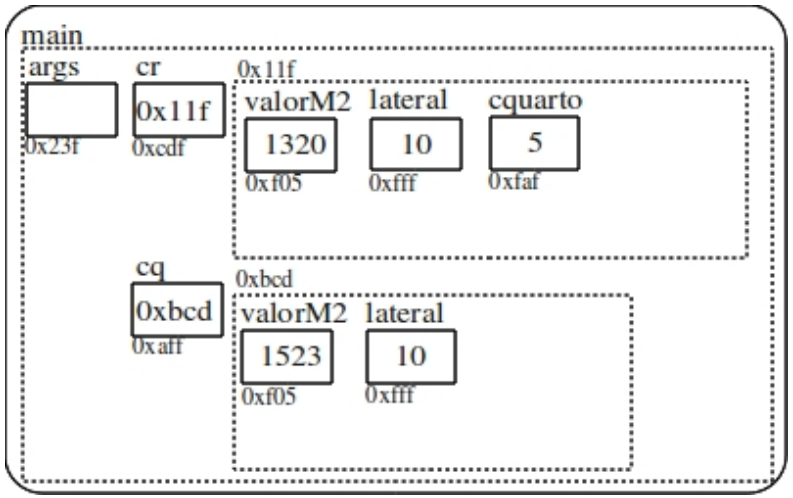
- Como se dá o funcionamento da memória nesse caso?

```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("Área: "+cr.area());
    System.out.println();
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("Área: "+cq.area());
}
```

Herança e Memória

- Como se dá o funcionamento da memória nesse caso?
  - Ao criarmos os objetos, alocamos espaço para atributos tanto de sua classe quanto superclasse

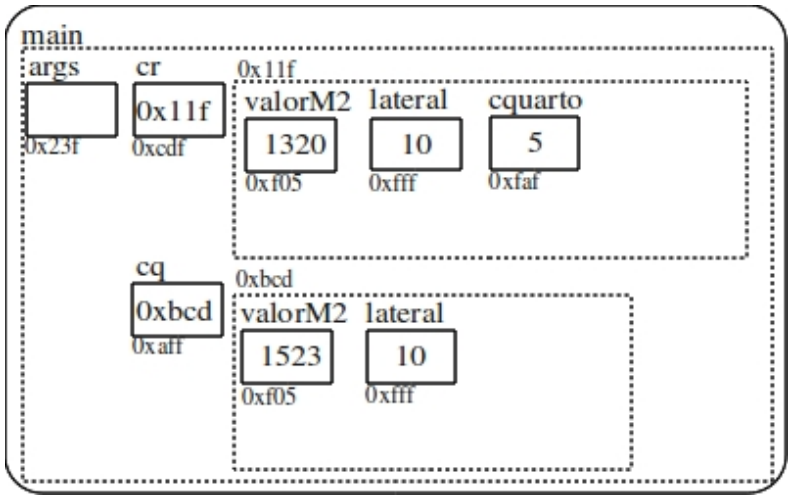
```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("Área: "+cr.area());
    System.out.println();
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("Área: "+cq.area());
}
```



Herança e Memória

- Como se dá o funcionamento da memória nesse caso?
  - Ao criarmos os objetos, alocamos espaço para atributos tanto de sua classe quanto superclasse
    - ★ O funcionamento dos construtores foi omitido para simplificar

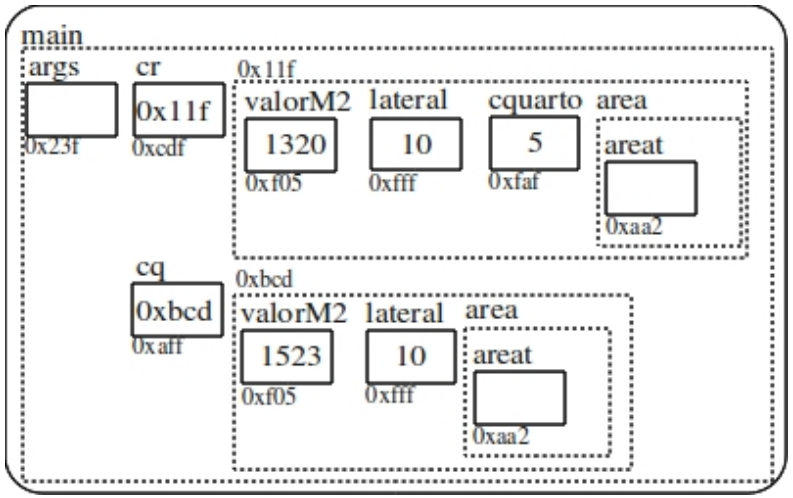
```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("Área: "+cr.area());
    System.out.println();
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("Área: "+cq.area());
}
```



Herança e Memória

- Como se dá o funcionamento da memória nesse caso?
  - Ao criarmos os objetos, alocamos espaço para atributos tanto de sua classe quanto superclasse
    - ★ O funcionamento dos construtores foi omitido para simplificar
  - Cada objeto tem seus próprios atributos e métodos

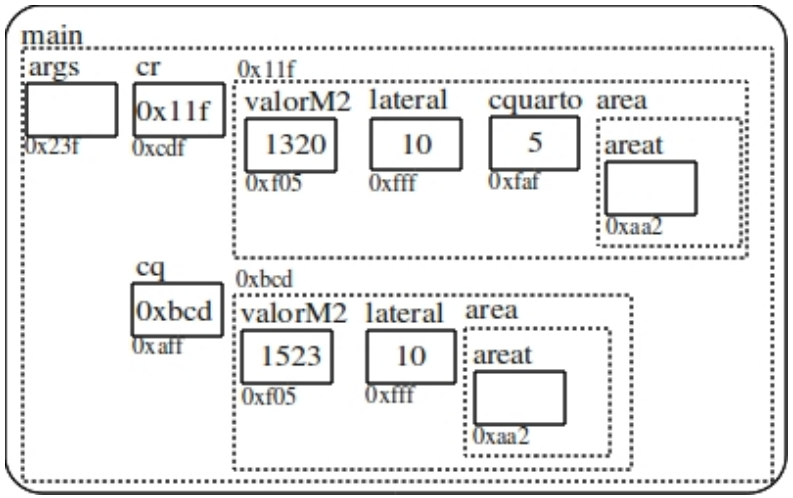
```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("Área: "+cr.area());
    System.out.println();
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("Área: "+cq.area());
}
```



Herança e Memória

- Como se dá o funcionamento da memória nesse caso?
  - Ao criarmos os objetos, alocamos espaço para atributos tanto de sua classe quanto superclasse
    - ★ O funcionamento dos construtores foi omitido para simplificar
  - Cada objeto tem seus próprios atributos e métodos
  - Não há compartilhamento de nada

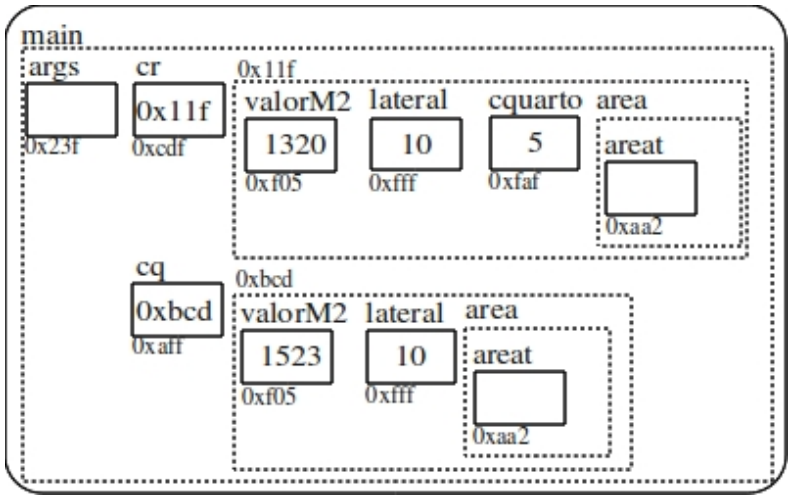
```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("Área: "+cr.area());
    System.out.println();
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("Área: "+cq.area());
}
```



Herança e Memória

- Como se dá o funcionamento da memória nesse caso?
  - Ao criarmos os objetos, alocamos espaço para atributos tanto de sua classe quanto superclasse
    - ★ O funcionamento dos construtores foi omitido para simplificar
  - Cada objeto tem seus próprios atributos e métodos
  - Não há compartilhamento de nada
    - ★ A menos que haja algum atributo ou método static

```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("Área: "+cr.area());
    System.out.println();
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("Área: "+cq.area());
}
```



Herança e Memória

- Como se dá o funcionamento da memória nesse caso?
  - Ao criarmos os objetos, alocamos espaço para atributos tanto de sua classe quanto superclasse
    - ★ O funcionamento dos construtores foi omitido para simplificar
  - Cada objeto tem seus próprios atributos e métodos
  - Não há compartilhamento de nada
    - ★ A menos que haja algum atributo ou método static
    - ★ Nesse caso, todos os objetos compartilharão do mesmo atributo/método na memória

```
public static void main(String[] args) {
    CasaRet cr = new CasaRet(10,5,1320);
    CasaQuad cq = new CasaQuad(10,1523);

    System.out.println("Quarto: "+cr.cquarto);
    System.out.println("Lateral da sala: "+
                                    cr.lateral);
    System.out.println("Área: "+cr.area());
    System.out.println();
    System.out.println("Lateral da casa: "+
                                    cq.lateral);
    System.out.println("Área: "+cq.area());
}
```