Projeto e Análise de Algoritmos

Prof. Dr. Ednaldo B. Pizzolato

www.dc.ufscar.br/~ednaldo

ednaldo@dc.ufscar.br

Decremento e Conquista

- Introdução
- Insertion sort
- Exercícios
- Algoritmos para gerar objetos combinatoriais
  - Permutações
  - Subconjuntos
- Exercícios

Decremento e Conquista

- Decremento de um fator constante
  - Busca binária
  - Moeda falsa
  - Multiplicação russa
- Decremento de tamanho variável

Decremento e conquista

- Introdução
  - A técnica de decremento e conquista é baseada na exploração da relação entre a solução de uma data instância de um problema e a solução de uma instância menor. Depois de descoberta (a relação), a técnica pode ser aplicada tanto bottom up como top down.
- A abordagem top down leva naturalmente à uma solução recursiva enquanto que a bottom up normalmente gera soluções iterativas (abordagem incremental).

Decremento e conquista

- Existem 3 grandes tipos de decremento e conquista:
  - Decremento por uma constante;
  - Decremento por um fator constante;
  - Decremento de tamanho variável.
- Decremento por uma constante
  - O tamanho da uma instancia é reduzido pela mesma constante em cada iteração. Normalmente a constante é 1.

Decremento e conquista - exemplo

- Exponenciação a^n com $a \neq 0$ e n inteiro não negativo.
  - $a^n = a^{n-1} . a$

$$f(n) = \begin{cases} f(n-1).a \\ 1 \end{cases}$$

Alguns podem argumentar que é similar ao processo de força bruta... Sim, mas usamos outro pensamento para se chegar a esta solução.

Decremento e conquista

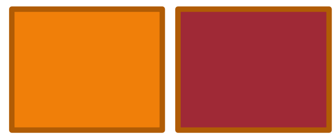
Decremento e conquista

- Decremento de um fator constante
  - *Esta técnica sugere que um problema seja reduzido a uma instância menor por um fator constante em cada iteração. Na maioria dos casos o fator é 2.*
  - *Exemplo:*

$$a^n = a^{n/2} . a^{n/2}$$

(funciona apenas se n for par)

Decremento e conquista



Decremento e conquista

- Para o problema de an temos:
  - Se n for par e positivo:

    $a^n = (a^{n/2})^2$
  - Se n for ímpar:

    $a^n = (a^{(n-1)/2})^2.a$
  - Se n for zero:

    $a^n = 1$

Decremento e conquista

- Decremento de um fator variável
  - Nos casos de fator variável, o fator varia de uma iteração para outra. Um exemplo é o algoritmo de Euclides para calcular o MDC.

    MDC(m,n) = MDC(n, m mod n)

Decremento e Conquista

- Introdução
- Insertion sort
- Exercícios
- Algoritmos para gerar objetos combinatoriais
  - Permutações
  - Subconjuntos
- Exercícios

Decremento e Conquista

- Insertion sort
  - Assume-se que há uma solução para o problema de tamanho n-1. Assim, tendo um vetor de n-1 posições ordenado, o problema de ordenar n elementos se resume a achar a posição correta para inserir (por isso insertion) o n-ésimo elemento.

Decremento e conquista

- Algoritmo

```
InsertionSort(A[0..n-1])
para i ← 1 até n-1 faça
    v ← A[i]
    j  ← i-1
    enquanto j ≥ 0 E A[j] > v faça
        A[j+1] ← A[j]
        j ← j + 1
    A[j+1] ← v
```

Decremento e conquista

- Algoritmo

InsertionSort(A[0..n-1])

para i ← 1 até n-1 faça

v ← A[i]

j ← i-1

enquanto j ≥ 0 E A[j] > v faça

A[j+1] ← A[j]

j ← j + 1

A[j+1] ← v

No pior caso (ordem decrescente) temos o maior número de comparações!

Decremento e conquista

- Algoritmo

InsertionSort(A[0..n-1])

para i ← 1 até n-1 faça

v ← A[i]

j ← i-1

enquanto j ≥ 0 E A[j] > v faça

A[j+1] ← A[j]

j ← j + 1

A[j+1] ← v

$$C_{pior}(n) = \sum_{i=1}^{n-1}\sum_{j=0}^{i-1} 1 = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} \in \Theta(n^2)$$

Decremento e conquista

- Algoritmo

InsertionSort(A[0..n-1])

para i ← 1 até n-1 faça

v ← A[i]

j ← i-1

enquanto j ≥ 0 E A[j] > v faça

A[j+1] ← A[j]

j ← j + 1

A[j+1] ← v

O melhor caso ocorre quando o vetor já está ordenado.

$$C_{melhor}(n) = \sum_{i=1}^{n-1} 1 = n - 1 \in \Theta(n)$$

Decremento e Conquista

- Introdução
- Insertion sort
- Exercícios
- Algoritmos para gerar objetos combinatoriais
  - Permutações
  - Subconjuntos
- Exercícios

Exercícios

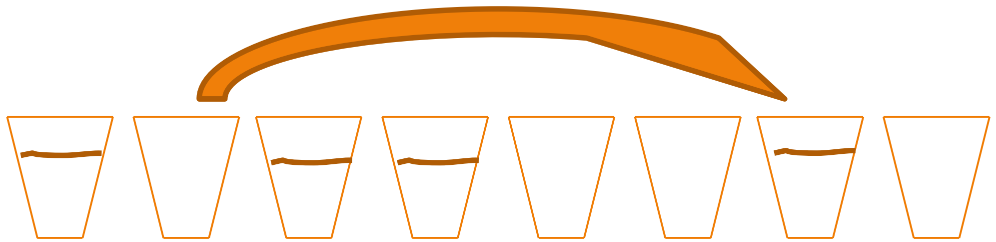
- Existem 2.n copos (um ao lado do outro), sendo que os n primeiros estão cheios e os n últimos vazios. Faça com que (no menor número de movimentos possível) os copos fiquem em uma formação alternada (cheio-vazio-cheio-vazio...).

Exercícios

- Dica: considere que jogar o líquido de um copo cheio em um vazio (qualquer) seja um movimento.

Exercícios

Para o exemplo em questão, um movimento possível seria colocar o conteúdo do segundo copo no penúltimo...



Exercícios

Fazendo isso, temos que nos preocupar com o conjunto de copos que estão entre o segundo e o penúltimo... Mas isso é a repetição do problema em menor tamanho...

Exercícios

- Se n for par, o número de movimentos é n/2; se for ímpar, é (n-1)/2.
- Assim, a número total de movimentações é $\lfloor n/2 \rfloor$

Exercícios

- Seja o conjunto A[0..n-1] um vetor de n elementos ordenáveis (para deixar o problema mais simples, assuma que são elementos distintos). Um par A[i], A[j] é chamado de inverso se i < j e A[i] > A[j].
  - Quais vetores de tamanho n tem o maior número de inversões e qual é este número?
  - Quais tem o menor número de inversões e, novamente, qual é este número?

Exercícios

- O maior número de inversões ocorre quando o vetor está em ordem decrescente, pois i < j e A[i] > A[j]! O número de inversões é n.(n-1)/2
- O menor número de inversões é 0, visto que se o vetor estiver ordenado em ordem crescente, teremos para cada i < j ➔ A[i] < A[j].

Decremento e Conquista

- Introdução
- Insertion sort
- Exercícios
- Algoritmos para gerar objetos combinatoriais
  - Permutações
  - Subconjuntos
- Exercícios

Decremento e conquista

- Gerando Permutações
  - Para efeitos de simplificação das ideias, consideremos apenas a permutação de inteiros (1 a n).

- Raciocínio do decremento de 1 unidade:
  - Resolvido o problema de n-1 permutações, basta inserir o valor n (ou índice n) em cada uma delas.
  - Como inserir?

Decremento e conquista

- Como?

- Resposta: esquerda p/ direita e direita p/ esquerda

Exemplo: 1

12 21

123 132 312

321 231 213

Decremento e conquista

- É possível gerar as permutações de n elementos sem ter que gerar explicitamente permutações para valores menores de n. Isso é possível se associarmos uma direção para cada elemento na permutação.

$$\overset{\rightarrow}{3}\,\overset{\leftarrow}{2}\,\overset{\rightarrow}{4}\,\overset{\leftarrow}{1}$$

- Considere um elemento k móvel. O elemento k é definido como sendo móvel se sua seta aponta para um número menor adjacente a ele. No exemplo acima, 3 aponta para 2 e 4 aponta para 1. Portanto 3 e 4 são móveis enquanto que 1 e 2 não!

Decremento e conquista

- Algoritmo JohnsonTrotter(n)

Entrada: número n positivo

Saída: lista de permutações de {1,...,n}

Inicializar a primeira permutação $\overleftarrow{1}\,\overleftarrow{2}\ldots\overleftarrow{n}$

enquanto a última permutação tiver um elemento móvel faça

ache o maior elemento móvel k

troque k com o elemento adjacente apontado por k

troque a direção de todos os elementos maiores que k

adicione a nova permutação à lista

Decremento e conquista

- Exemplo:

$$\overleftarrow{1}\;\overleftarrow{2}\;\overleftarrow{3}$$

$$\overleftarrow{1}\;\overleftarrow{2}\;\boxed{\overleftarrow{3}}$$

$$\overleftarrow{1}\;\boxed{\overleftarrow{3}}\;\overleftarrow{2}$$

$$\overleftarrow{3}\;\overleftarrow{1}\;\overleftarrow{2}$$

elemento móvel 3

Decremento e conquista

- Exemplo:

$$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}$$

$$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}$$

$$\overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2}$$

$$\overleftarrow{3}\,\overleftarrow{1}\,\boxed{\overleftarrow{2}}$$

$$\overleftarrow{3}\,\boxed{\overleftarrow{2}}\,\overleftarrow{1}$$

elemento móvel 2

Decremento e conquista

- Exemplo:

$$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}$$

$$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}$$

$$\overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2}$$

$$\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}$$

$$\overleftarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}$$

$$\boxed{\overrightarrow{3}}\,\overleftarrow{2}\,\overleftarrow{1}$$

$$\overleftarrow{2}\,\boxed{\overrightarrow{3}}\,\overleftarrow{1}$$

$$\overleftarrow{2}\,\overleftarrow{1}\,\boxed{\overrightarrow{3}}$$

elemento móvel 3

Decremento e conquista

- Exemplo:

$$\overleftarrow{1}\;\overleftarrow{2}\;\overleftarrow{3}$$
$$\overleftarrow{1}\;\overleftarrow{3}\;\overleftarrow{2}$$
$$\overleftarrow{3}\;\overleftarrow{1}\;\overleftarrow{2}$$
$$\overleftarrow{3}\;\overleftarrow{2}\;\overleftarrow{1}$$
$$\overleftarrow{2}\;\overrightarrow{3}\;\overleftarrow{1}$$
$$\overleftarrow{2}\;\overleftarrow{1}\;\overrightarrow{3}$$

Decremento e conquista

- Para estudar
  - As permutações geradas pelo algoritmo de Johnson-Trotter não são lexográficas (não estão na ordem natural que esperamos). Pesquise um algoritmo que faça a permutação lexográfica.

Decremento e Conquista

- Introdução
- Insertion sort
- Exercícios
- Algoritmos para gerar objetos combinatoriais
  - Permutações
  - Subconjuntos
- Exercícios

Decremento e conquista

- Subconjuntos
  - Considere todos os subconjuntos de $A = \{a_1, ..., a_n\}$ e suponha que possa ser dividido em 2 grupos: aqueles com a_n e aqueles sem o a_n. O grupo de todos <u>sem</u> o a_n na verdade são todos os subconjuntos $\{a_1,...,a_{n-1}\}$.
  - Com a lista dos subconjuntos que não contêm a_n, aí basta incluir a_n para se ter os subconjuntos $\{a_1,...a_n\}$.

Decremento e conquista

- Considere $\{a_1, a_2, a_3\}$.
- $\{\ \}$
- $\{a_1\}$ $\{a_2\}$ $\{a_3\}$
- $\{a_1,a_2\}$ $\{a_1,a_3\}$ $\{a_2,a_3\}$
- $\{a_1,a_2,a_3\}$

Decremento e Conquista

- Introdução
- Insertion sort
- Exercícios
- Algoritmos para gerar objetos combinatoriais
  - Permutações
  - Subconjuntos
- Exercícios

Exercícios

- Projete um algoritmo (baseado no decremento de uma unidade) para gerar o conjunto potência de um conjunto de n elementos. Lembrando que o conjunto potência de um conjunto S é o conjunto de todos os subconjuntos de S, incluindo o conjunto vazio.

Exercícios

se n=0

retorna lista L(0) *// tem somente conjunto vazio*

senão

criar recursivamente lista L(n−1) *// com todos os subconjuntos de* $\{a_1, ..., a_{n-1}\}$

concatene a_n *a cada elemento de L(n−1) para obter a lista T*

retorna L(n) obtido pela concatenação de L(n−1) e T

Exercícios

- Exemplo

{1,2,3}

criar L(2)

{1,2}

criar L(1)

{1}

criar L(0)

{ }

concatene 1 ao { }

T = {ϕ,1}

L(1) = {ϕ,1}

concatene 2 a {ϕ,1}

T = {{ϕ,2},{1,2}}

L(2) = {ϕ,1,2,{1,2}}

concatene 3 a {ϕ,1,2,{1,2}}

...

Decremento e conquista

Considere $\{a_1, a_2, a_3\}$.

$\{\ \}$

$\{a_1\}$ $\{a_2\}$ $\{a_3\}$

$\{a_1,a_2\}$ $\{a_1,a_3\}$ $\{a_2,a_3\}$

$\{a_1,a_2,a_3\}$

000

100 010 001

110 101 011

111

Decremento e Conquista

- Introdução
- Insertion sort
- Exercícios
- Algoritmos para gerar objetos combinatoriais
  - Permutações
  - Subconjuntos
- Exercícios

Decremento e Conquista

- Decremento de um fator constante
  - Busca binária
  - Moeda falsa
  - Multiplicação russa
- Decremento de tamanho variável

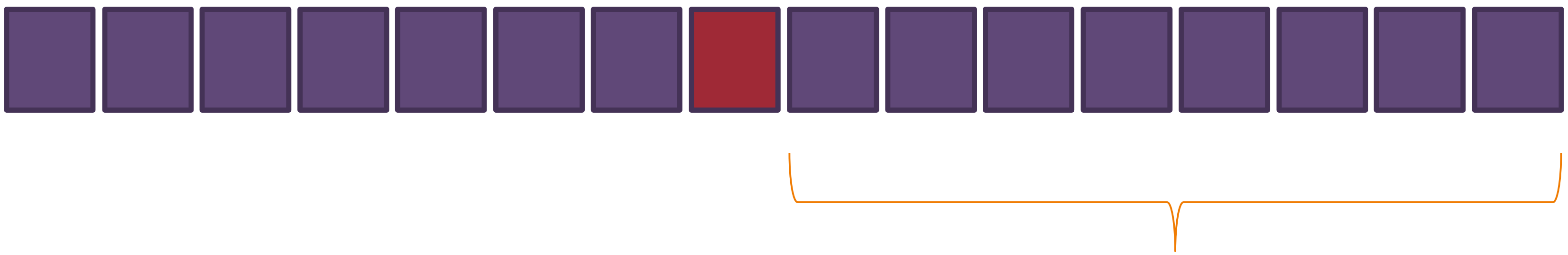
Decremento e conquista

- Busca binária:

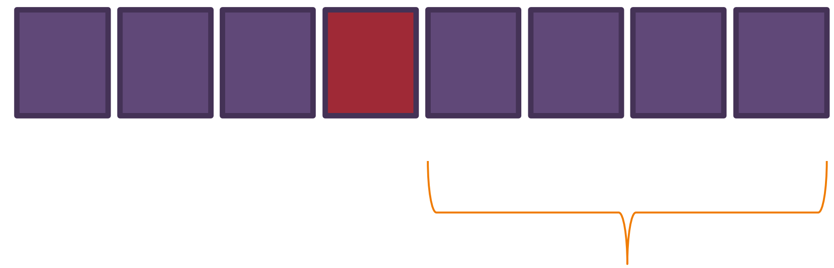
$$a_0 \ldots a_{m-1} \; \boxed{a_m} \; a_{m+1} \ldots a_{n-1}$$

K

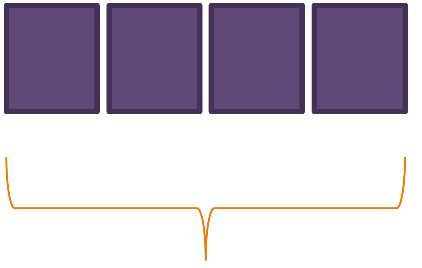
Decremento e conquista



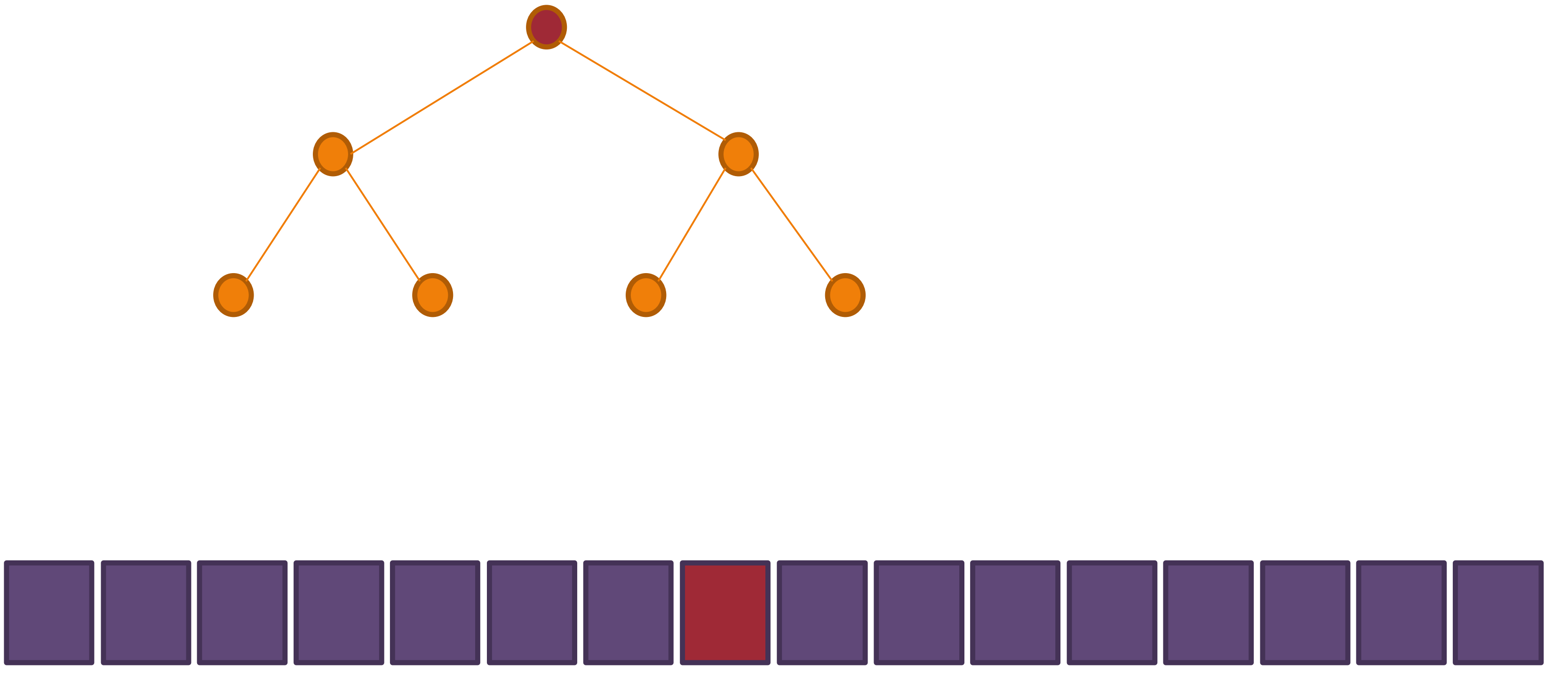
Decremento e conquista



Decremento e conquista



Decremento e conquista



Decremento e conquista

Algoritmo

$l \leftarrow 0$

$r \leftarrow n - 1$

enquanto $l \leq r$ *faça*

$\quad m \leftarrow \lfloor (l+r)/2 \rfloor$

$\quad$ *se* $k = a[m]$ *retorna* m

$\quad$ *senão*

$\quad\quad$ *se* $k < a[m]$ $r \leftarrow m - 1$

$\quad\quad$ *senão* $l \leftarrow m + 1$

retorna -1

- Pergunta: qual é o custo do pior caso?

Decremento e conquista

Algoritmo

$l \leftarrow 0$

$r \leftarrow n - 1$

enquanto $l \leq r$ *faça*

    $m \leftarrow \lfloor (l+r)/2 \rfloor$

    se $k = a[m]$ *retorna* m

    senão

        se $k < a[m]$ $r \leftarrow m - 1$

        senão $l \leftarrow m + 1$

retorna -1

$C_{pior}(n) = C_{pior}(\lfloor n/2 \rfloor) + 1$

$C_{pior}(1) = 1$

$$C_{pior}(n) = \lfloor \log_2 n \rfloor + 1$$

$$\in \Theta(\log n)$$

Decremento e Conquista

- Decremento de um fator constante
  - Busca binária
  - Moeda falsa
  - Multiplicação russa
- Decremento de tamanho variável

Decremento e conquista

- Moeda falsa
  - Dentre um conjunto de n moedas todas similares, uma é falsa (mais leve). Com uma balança pode-se comparar quaisquer 2 conjuntos de moedas.

Decremento e conquista

- Moeda falsa

Decremento e Conquista

- Decremento de um fator constante
  - Busca binária
  - Moeda falsa
  - Multiplicação russa
- Decremento de tamanho variável

Decremento e conquista

- Multiplicação a la russa
  - Sejam n e m 2 inteiros positivos, cujo produto desejamos calcular. Se n é par temos:

$$n.m = \frac{n}{2} 2.m$$

  - Se n é ímpar, temos:

$$n.m = \frac{n-1}{2} 2.m + m$$

Decremento e conquista

- Como caso trivial temos:

  1.m = m

Exemplo:	50.65	
	50	65
	25	130
	12	260 (+130)
	6	520
	3	1040
	1	2080 (+1040)
		2080 + 130 + 1040 = 3250

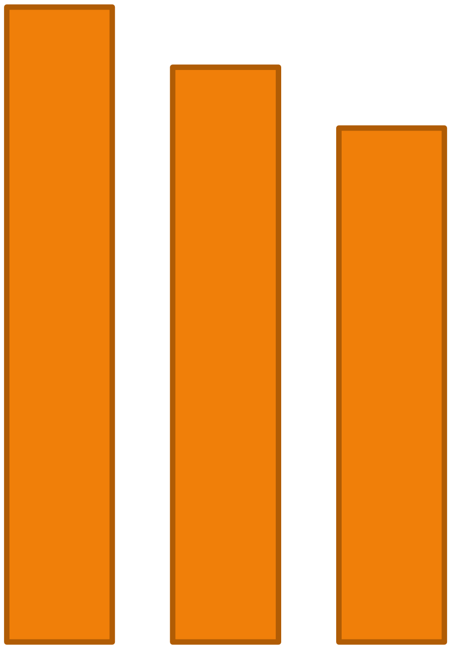
Decremento e Conquista

- Decremento de um fator constante
  - Busca binária
  - Moeda falsa
  - Multiplicação russa
- Decremento de tamanho variável

Decremento e conquista

- Recaptulando...

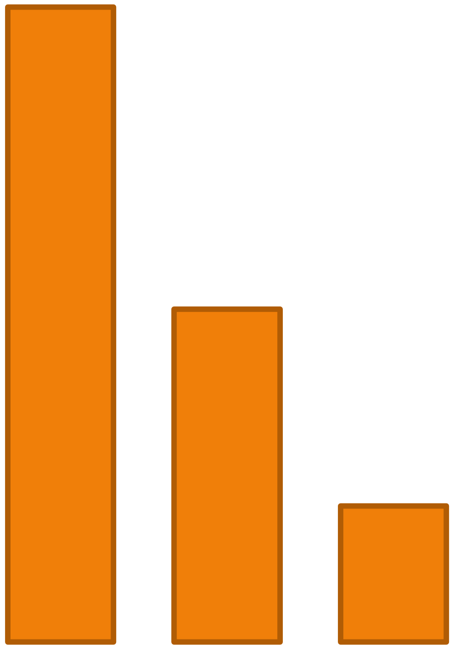
Decremento por uma constante



Decremento e conquista

- Recaptulando...

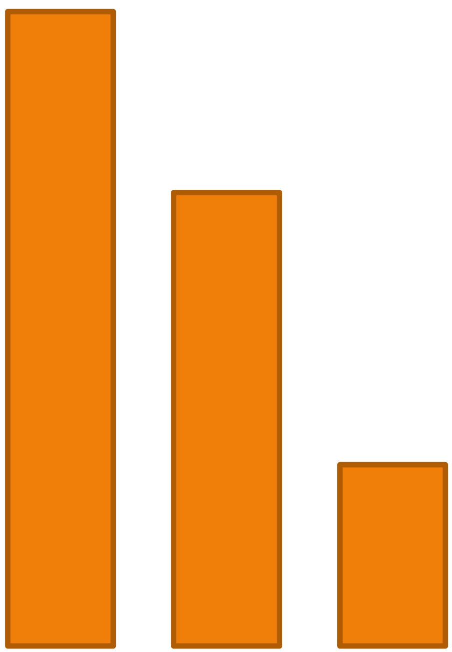
Decremento por um fator constante



Decremento e conquista

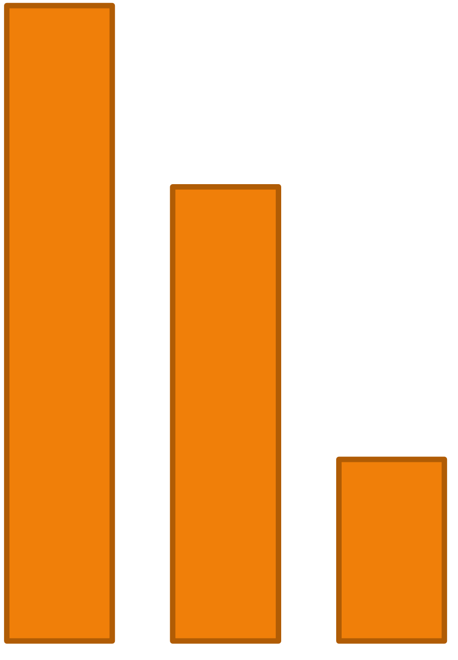
- E agora...

Decremento por tamanho variável



Decremento e conquista

- Exemplo: MDC



Decremento e Conquista

- Decremento de um fator constante
  - Busca binária
  - Moeda falsa
  - Multiplicação russa
- Decremento de tamanho variável

Exercícios

- O que você acha de implementar um algoritmo que gere todas as permutações de 25 elementos em seu computador?
- E todos os subconjuntos de tal conjunto?

Exercícios

- O que você acha de implementar um algoritmo que gere todas as permutações de 25 elementos em seu computador?

R.: Gerar todas as permutações implica em ter que fazer 25! operações, o que levaria um tempo da ordem de 1.5×10^{25}. Muito, muito tempo...

- E todos os subconjuntos de tal conjunto?

R.: Isso custaria 2^{25} operações, o que é aproximadamente 3.3×10^{7}.(menos de 1s)

Exercícios

- Gere todas as permutações de {1,2,3,4} pelo algoritmo de Johnson-Trotter.
- Lembrand que teremos 4! = 4.3.2.1 = 24 permutações

Inicializar a primeira permutação {$\overleftarrow{1}$, $\overleftarrow{2}$, $\overleftarrow{3}$, $\overleftarrow{4}$}

enquanto a última permutação tiver um elemento móvel faça

ache o maior elemento móvel k

troque k com o elemento adjacente apontado por k

troque a direção de todos os elementos maiores que k

adicione a nova permutação à lista

Exercícios

1	2	3	4		1	2	4	3		1	4	2	3		4	1	2	3
4	1	3	2		1	4	3	2		1	3	4	2		1	3	2	4
3	1	2	4		3	1	4	2		3	4	1	2		4	3	1	2
4	3	2	1		3	4	2	1		3	2	4	1		3	2	1	4
2	3	1	4		2	3	4	1		2	4	3	1		4	2	3	1
4	2	1	3		2	4	1	3		2	1	4	3		2	1	3	4

Exercícios

- Um palito de n centímetros precisa ser cortado em n palitinhos de 1 centímetro cada. Desenhe um algoritmo que faça isso em um número mínimo de cortes, considerando que vários podem ser cortados ao mesmo tempo.

Exercícios

Decremento de uma unidade (força bruta)

```
Cortar (n)
se n = 1
    finalizar
se n > 1
    dividir o palito em 2 partes A e B sendo B de 1cm
    Cortar (n-1) // A tem tamanho n-1
```

Exercícios

para i $\leftarrow$ 1 ate n faça

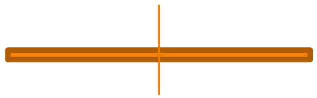
        cortar i

Temos aqui n cortes

Exercícios

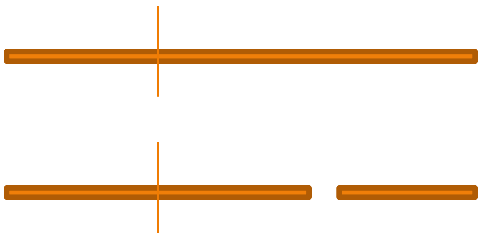
- Método de decrementar por um fator constante

Exercícios



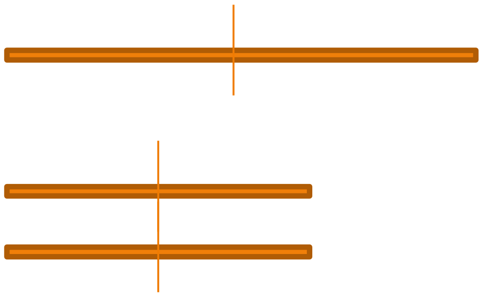
- Com n = 2cm temos 1 corte ao meio

Exercícios



- Com n = 3cm temos 1 corte de 1cm e outro corte ao meio (2cm)

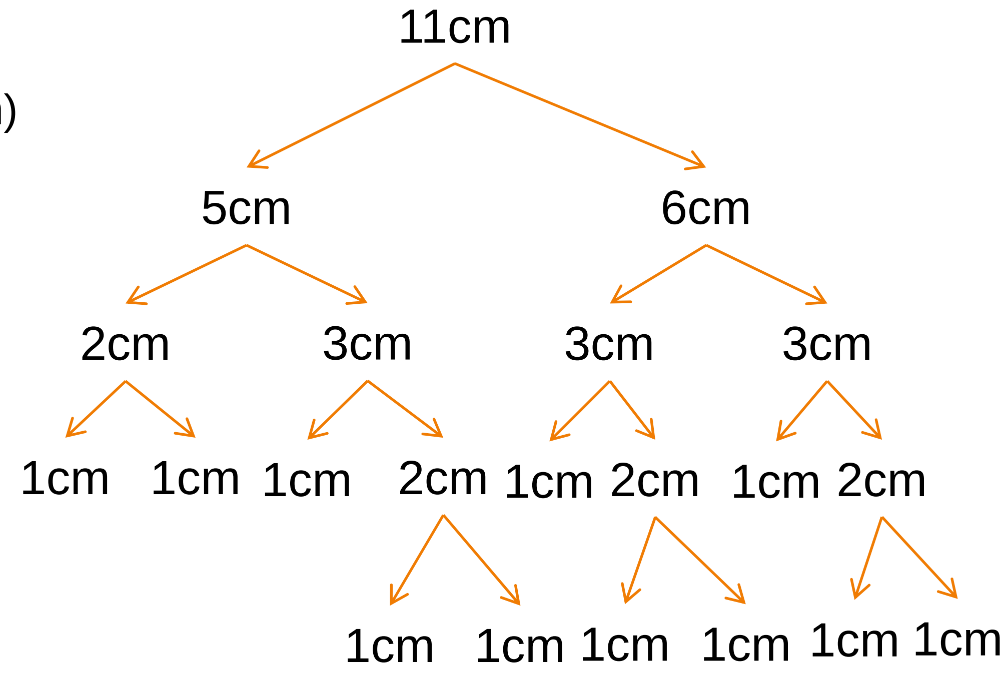
Exercícios



- Com n = 4cm temos 1 corte ao meio e um corte em 2 palitos de 2cm.

Exercícios

- Para n = 11
- 1 corte (A=5cm e B=6m)
- 1 corte em A (3m) e um corte em B (3cm)
- 1 corte em A (2cm) e em B (2cm)
- 1 corte em 2 de 4cm (produzindo 4 de 2cm)
- 1 corte em 4 de 2cm (finalizando)



Exercícios

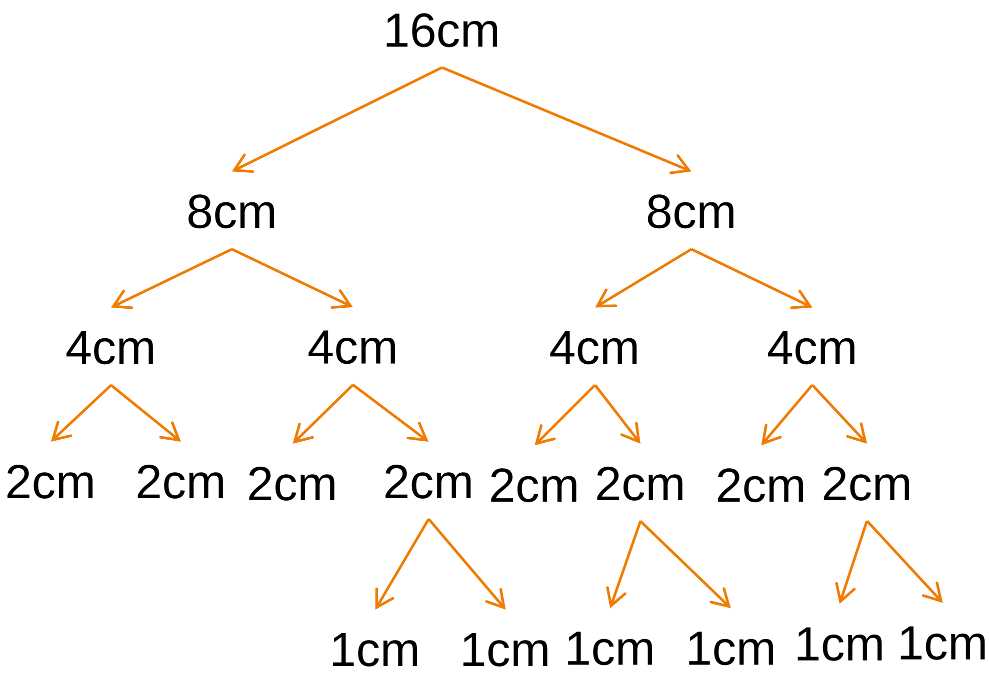
para n = 1 → 0 cortes		para n = 9 → 4 cortes	
para n = 2 → 1 corte		para n = 10 → 4 cortes	
para n = 3 → 2 cortes		para n = 11 → 4 cortes	
para n = 4 → 2 cortes		para n = 12 → 4 cortes	
para n = 5 → 3 cortes		...	
para n = 6 → 3 cortes		para n = 15 → 4 cortes	
para n = 7 → 3 cortes			
para n = 8 → 3 cortes			

Exercícios

- Para n = 16
- 1 corte (A=8cm e B=8m)

...

4 cortes



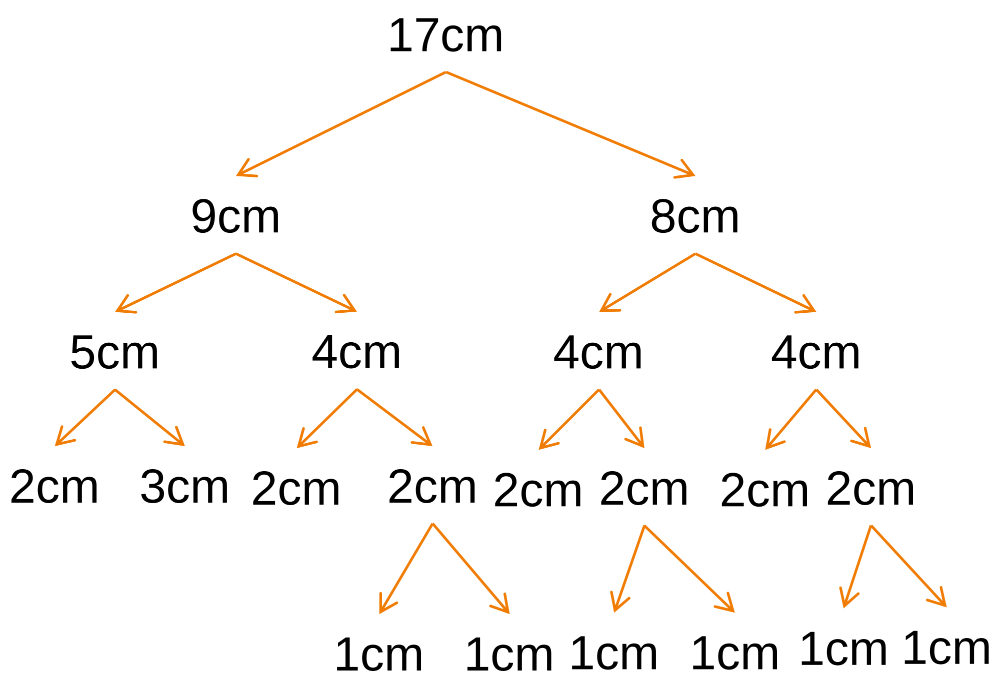
Exercícios

- Para n = 17
- 1 corte (A=9cm e B=8m)

...

5 cortes

$$= \lceil \log_2 n \rceil$$



Exercícios

- Estime quantas vezes mais rápida é, em média, a pesquisa binária em comparação com a pesquisa sequencial para buscas bem sucedidas em vetores ordenados de tamanho 1 milhão.

Exercícios

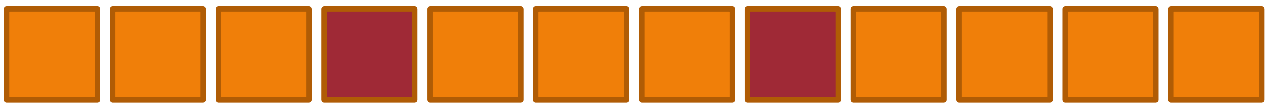
- Em média para pesquisas sequenciais teríamos 500.000 comparações.
- Para buscas binárias bem sucedidas teremos no máximo (pior caso) a altura da árvore binária, que é log n.
- Com 1.000.000 de registros, teríamos no máximo 20 comparações, pois 2^{20} = 1.048.576 e na média 10 comparações.
- Portanto, a busca binária para o caso em questão é da ordem de 50.000 vezes mais rápida.

Exercícios

- Desenhe a busca ternária.
- Em qual técnica ela é baseada?
- Compare a eficiência (complexidade) do algoritmo em relação à pesquisa binária.

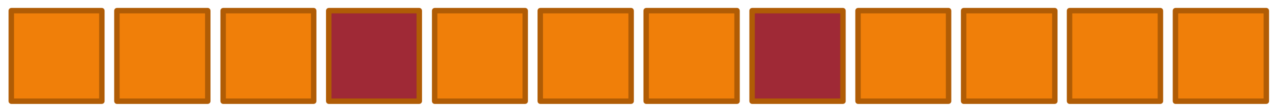
Exercícios

- Desenhe a busca ternária.



Em uma busca ternária em um vetor ordenado, o vetor é dividido em três partes aproximadamente iguais. As chaves nas posições aproximadamente 1/3n e 2/3n são comparadas com a chave que está sendo buscada para determinar em qual terço a busca vai prosseguir.

Exercícios



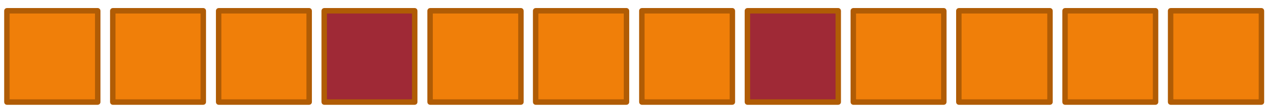
Assim, podemos ter no máximo 2 comparações antes de prosseguir.
A cada divisão, descartamos aproximadamente 2/3n elementos e mantemos vivos aproximadamente 1/3n dos elementos.

Exercícios

```
buscaTernaria(k,inicio,fim)
se inicio ≥ fim
    retorna -1
senão
    m1 ← (fim-inicio)/3
    m2 ← 2.m1
    se k ≤ m1
      buscaTernaria(k, inicio, m1)
    senão
      se k ≤ m2
          buscaTernaria(k,m1+1,m2)
      senão
          buscaTernaria(k,m2+1,fim)
```

Exercícios

- Desenhe a busca ternária.



$T(n) = T(n/2) + 1 = \log_2 n$, para a busca binária

$T(n) = T(n/3) + 2 = 2 * \log_3 n$, para a busca ternária

Exercícios

- Em qual técnica ela é baseada?

R.: Decremento de um fator constante (3)

Exercícios

- Escreva o algoritmo para o problema das moedas falsas baseado no conceito de dividir em 3 partes.
- Garanta que seu algoritmo funcione corretamente em todas as situações e não apenas para aquelas em que n é múltiplo de 3.

Exercícios

se n = 9

A = 3

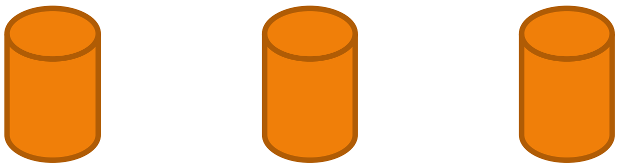
B = 3

C = 3

Exercícios

se n = 10 ?

Exercícios



se n = 10

A = 3

B = 3

C = 4

Exercícios

se n = 11

A = 3

B = 4

C = 4

Exercícios

se n = 100

A = 33

B = 33

C = 34

Exercícios

se n = 4

A = 1

B = 1

C = 2

Exercícios

- Aplique o algoritmo de multiplicação russa para os números 26 e 47.

Lembrando...

- Sejam n e m 2 inteiros positivos, cujo produto desejamos calcular. Se n é par temos:

$$n.m = \frac{n}{2} 2.m$$

- Se n é ímpar, temos:

$$n.m = \frac{n-1}{2} 2.m + m$$

Exercícios

- Aplique o algoritmo de multiplicação russa para os números 26 e 47.

R.: 13 x 2.47

6 x 2.94+94

3 x 2.188+94

1 x 2.376+94+376

= 1222

$$n.m = \frac{n}{2}2.m$$

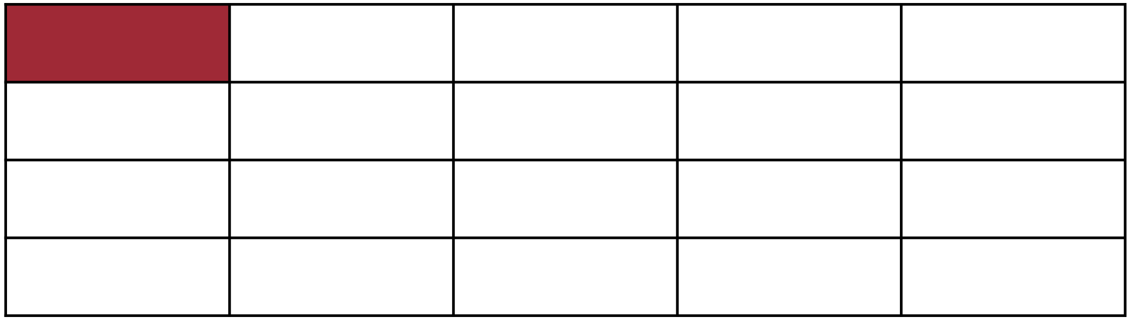
$$n.m = \frac{n-1}{2}2.m + m$$

Exercícios

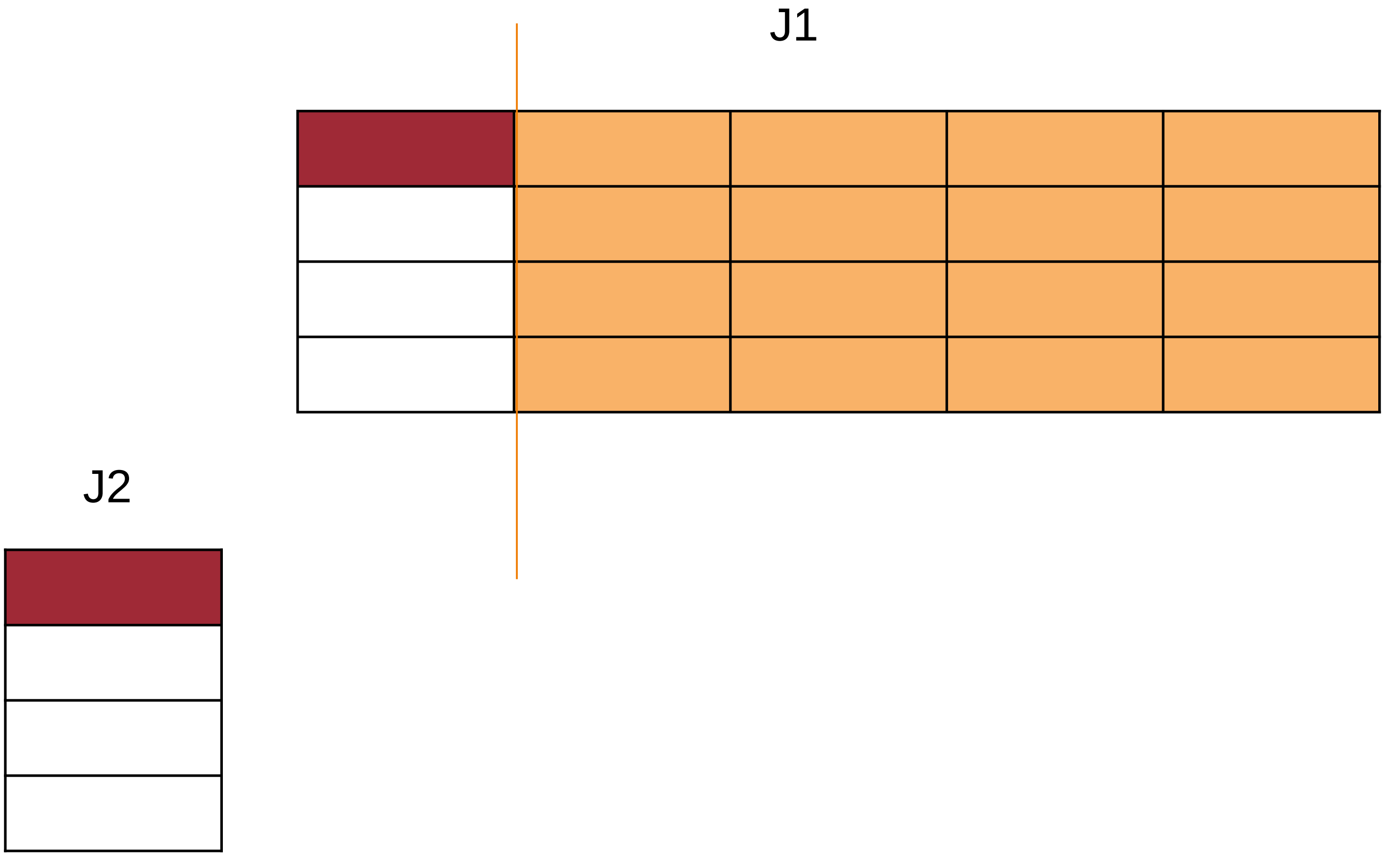
- Duas pessoas brincam de quebrar uma barra de chocolate de m x n quadradinhos, que tem um deles estragado. Quebra-se a barra por inteiro nos espaços que separam os quadradinhos. Depois de quebrar a barra, o jogador come o pedaço que não contem o quadradinho estragado. O jogador que tiver que comer o quadradinho estragado perde o jogo. Começar o jogo é vantagem ou não?

Exercícios

J1

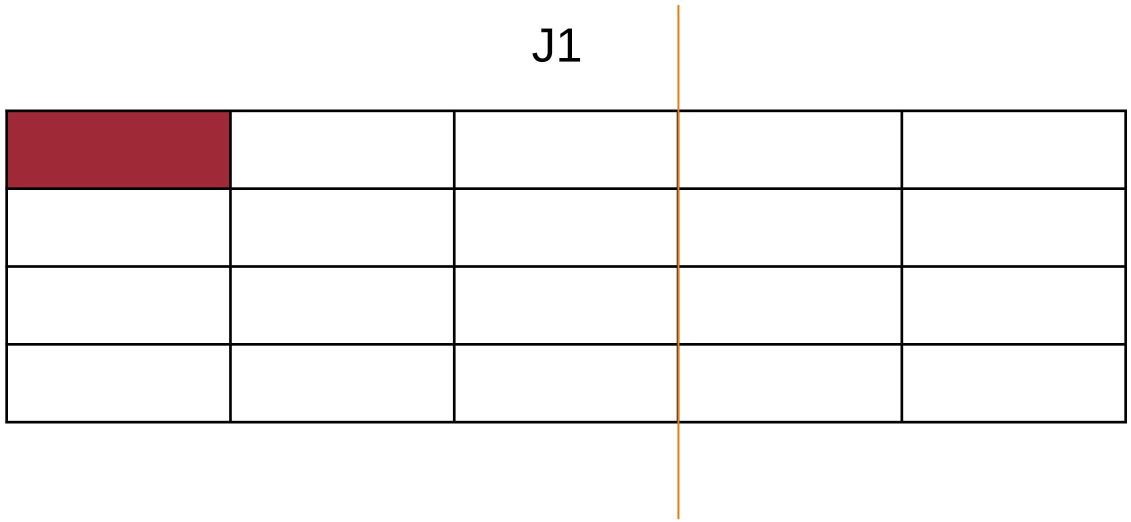


Exercícios

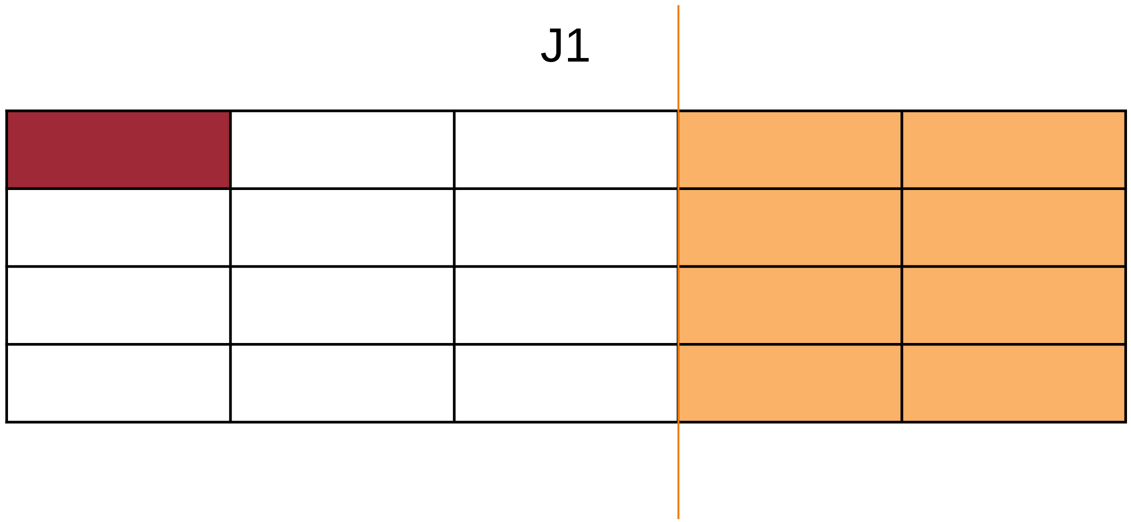


Exercícios

Exercícios

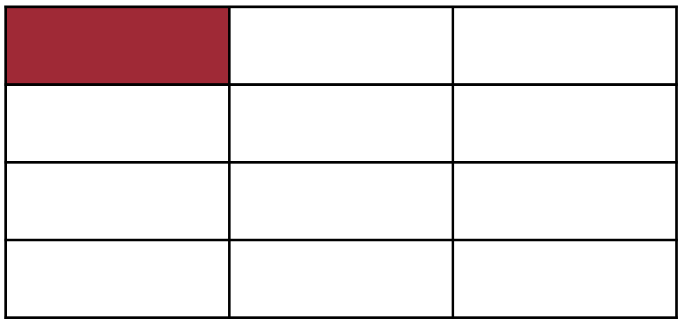


Exercícios



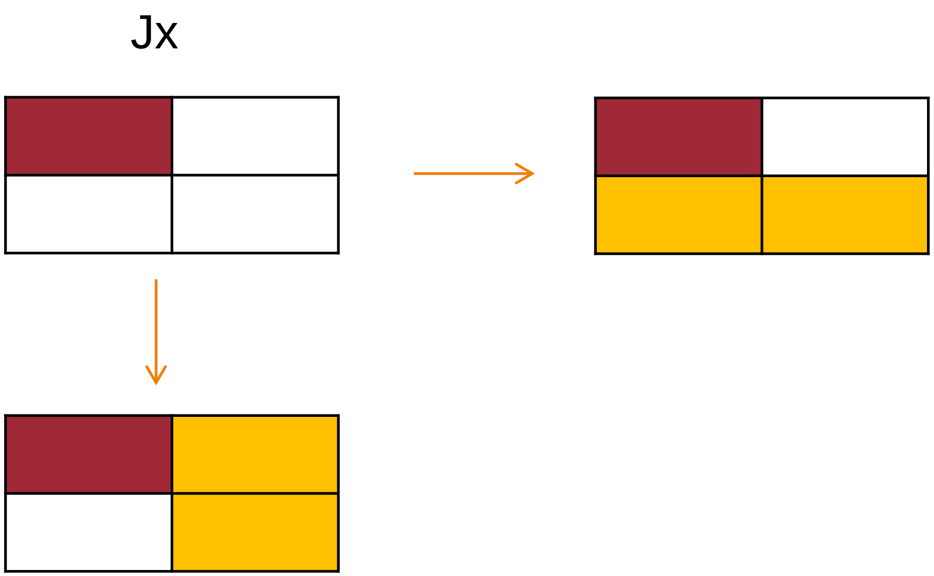
Exercícios

J2

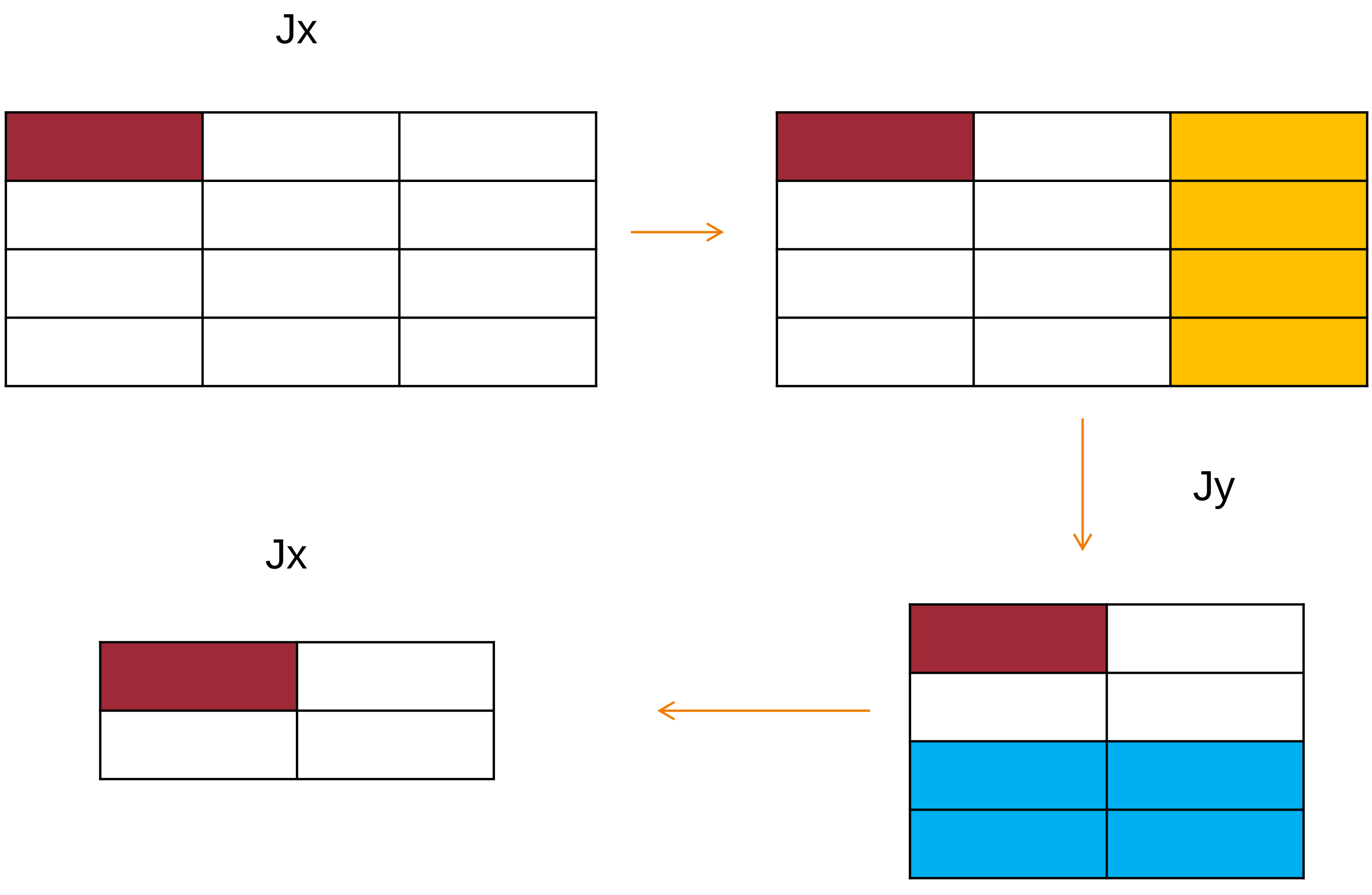


Exercícios

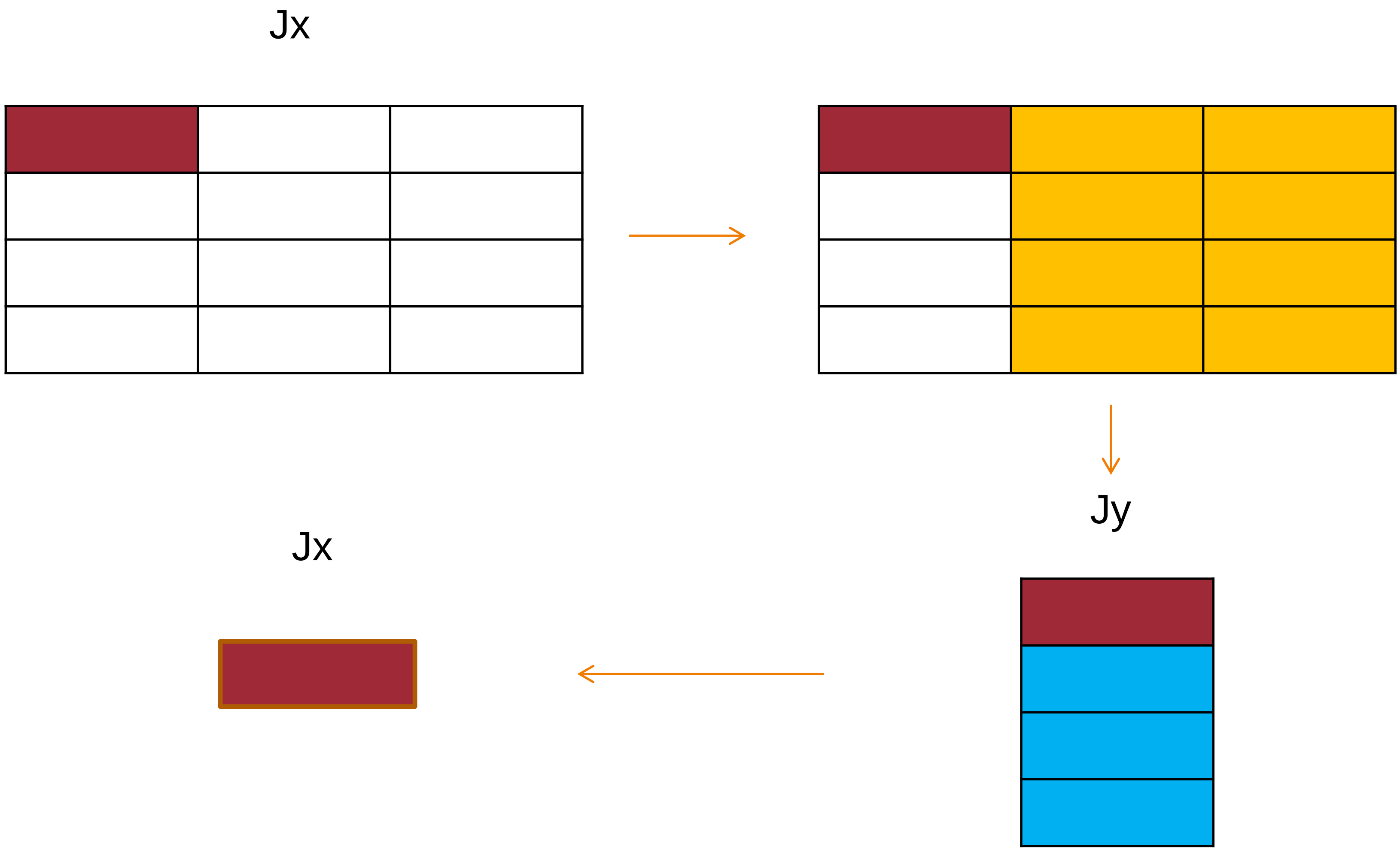
- Tente provar que se um jogador obtiver a barra a x a (barra quadrada de tamanho a), então ele perdeu. (veja 2 x 2)



Exercícios

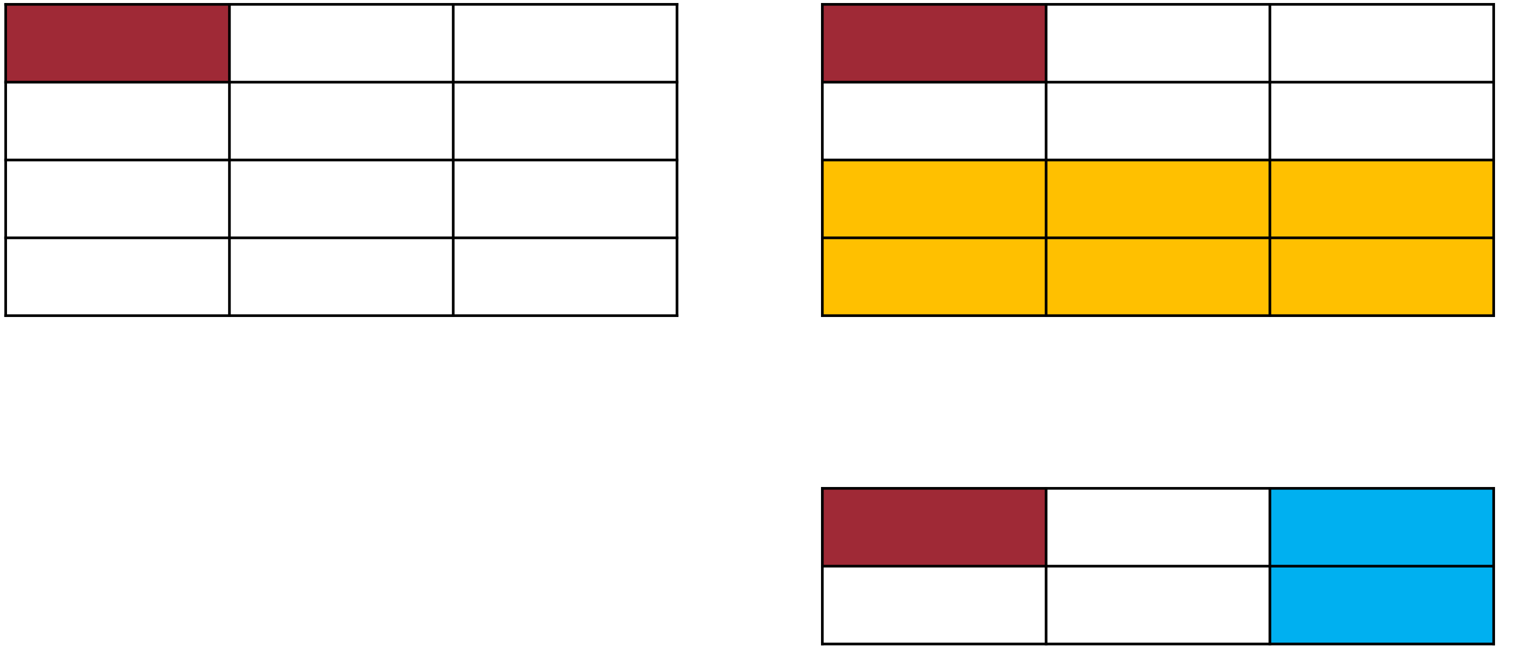


Exercícios



Exercícios

Jx



Exercícios

- Desenhe um algoritmo O(n) que faça uma busca em uma matriz n x n em que cada coluna e cada linha estão ordenadas em ordem crescente.

Exercícios

- Desenhe um algoritmo O(n) que faça uma busca em uma matriz n x n em que cada coluna e cada linha estão ordenadas em ordem crescente.

1	2	3	4	5
20	24	27	33	38
45	47	52	54	59
63	67	71	75	80
85	88	92	95	99

Exercícios

1	2	3	4	5
20	24	27	33	38
45	47	52	54	59
63	67	71	75	80
85	88	92	95	99

```
j ← 1
i ← n
encontrou ← falso
enquanto i > 0 E não encontrou faça
    se A[i,j] < K então
        m ← 1
        enquanto m ≤ n E não encontrou faça
            se A[i,m] = k então
                encontrou ← verdadeiro
            senão
                m ← m + 1
```

$O(n+n) = O(2.n) = O(n)$

Exercícios

- Um vetor A de tamanho n-1 (portanto de 0 a n-2), contem n-1 inteiros que vão de 1 a n em ordem crescente. Pelo enunciado já está claro que um elemento está faltando. Projete um algoritmo para achar o elemento que falta e indique sua eficiência (complexidade).

Exercícios

- Um vetor A de tamanho n-1 (portanto de 0 a n-2), contem n-1 inteiros que vão de 1 a n em ordem crescente. Pelo enunciado já está claro que um elemento está faltando. Projete um algoritmo para achar o elemento que falta e indique sua eficiência (complexidade).
- Supondo que a sequência deveria ser:

{0,1,2,3,4,5,6} mas você tem {0,1,3,4,5,6}

Exercícios

- Soma dos termos de uma PA:

$(a_1+a_n).n/2 = (0+6).7/2 = 42/2 = 21$

Assim, a soma da primeira sequencia é 21.

Basta somar os elementos da segunda sequência e fazer 21 – S para obter o valor que está faltando.

Isso nos leva a um algoritmo O(n).

Resumo

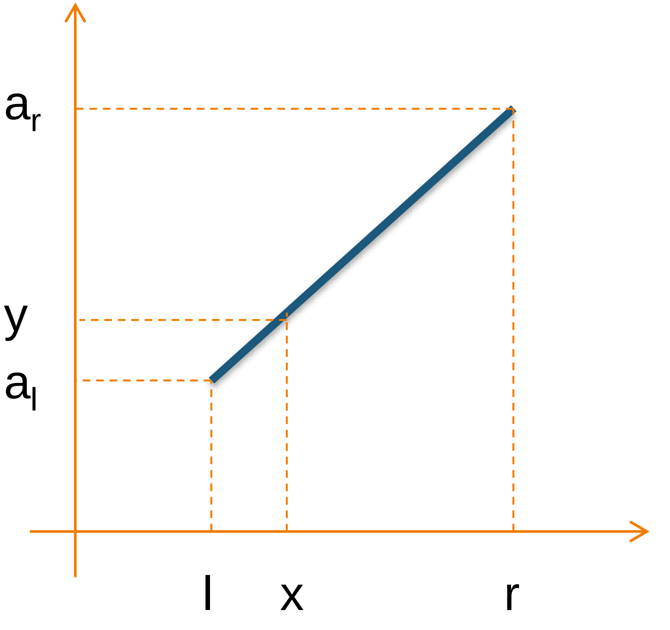
- Decremento e conquista é uma estratégia geral de resolução de problemas (algoritmos) baseada na relação entre a solução para uma dada instância de problema e uma instância menor do mesmo problema.
- Existem 3 variações principais:
  - Decremento por uma constante (geralmente 1);
  - Decremento por um fator constante (geralmente 2);
  - Decremento por um fator variável (MDC).

Resumo

- O algoritmo Insertion Sort é um exemplo de aplicação direta do decremento por uma constante (1) para o problema de ordenação.

Busca por interpolação

$$x = l + \left\lfloor \frac{(y - a_l)(r - l)}{a_r - a_l} \right\rfloor$$



THE END