Universidade Cruzeiro do Sul

Estruturas de Dados I

Conteúdos

Exercícios sobre:

- Vetores.
- Aleatorização de vetores.
- Busca sequencial. Busca binária.
- Métodos recursivos.
- Métodos de ordenação.

Elaboração

Prof. Manuel F. Paradela Ledón

Exercício 1 - para entregar

para entregar

Na Aula 03 estudamos o algoritmo de Fisher-Yates para "embaralhar" aleatoriamente os elementos de um vetor, utilizando a classe pronta ArrayList:

public void **aleatorizar** (ArrayList lista)

Utilizando a mesma lógica deste algoritmo, crie um método que permita "aleatorizar" um vetor comum (utilizando somente vetores), com o cabeçalho:

public void **aleatorizarVetor** (double vet[])

Como sugestão, para eliminar um elemento do vetor original, poderia utilizar o método **deslocaEsquerda**, também estudado na Aula 03:

```
void  deslocaEsquerda (double vet[], int de, int ate) {
            if(de>ate)return;
            if(de<=0)de=1;
            if(ate > vet.length-1)ate=vet.length-1;
            // elimina o item na posição (d-1) e desloca os restantes
            for (int i = (de - 1); i < ate; i++) vet[i] = vet[i+1];
            vet[ate] = 0;  // só para marcar o item final
}
```

Pode usar como base o projeto em **Ex_random_base.zip.**

A resposta estará disponível em Ex_random_mais_completo.java

Veja também uma solução, sem utilizar o algoritmo de Fisher-Yates, no projeto NetBeans em Embaralhar_outra_solução.

Exercício 2

Precisamos implementar algoritmos para efetuar uma **busca** de um país dentro de um vetor de Strings. Analise as características dos vetores mostrados e responda V ou F (verdadeiro ou falso) para cada uma das assertivas a seguir.

```
String paisesA [] = { "Ucrânia", "Turquia", "Suíça", "México", "França", "Espanha",
                      "Chile", "Brasil , "Argentina" };
String paisesB [] = { "Argentina", "Brasil", "Chile", "Dinamarca", "Espanha", "França",
                      "Inglaterra", "Turquia", "Uruguai" };
String paisesC [] = { "Canadá", "Áustria", "Chile", "Itália", "Portugal", "Grécia",
                      "Angola", "Moçambique", "Rússia" };
```

1.___ Para buscar um país em paisesA[] o método mais eficiente seria o da busca binária.
2.___ Para buscar um país em paisesB[] o método mais eficiente seria o da busca binária.
3.___ Para buscar um país em paisesC[] seria adequado o método de busca binária.
4.___ Para buscar um país em paisesC[] seria adequado o método de busca sequencial.
5.___ Seria possível buscar um país em paisesB[] com o método de busca sequencial.
6.___ Com o método de busca sequencial, no pior caso, encontrar um país tem complexidade O(n).
7.___ Com o método de busca binária, no pior caso, encontrar um país tem complexidade O(log n).
8.___ Com o método de busca sequencial, no pior caso, encontrar um país tem complexidade $O(n^2)$.

Exercício 2 - respostas

Precisamos implementar algoritmos para efetuar uma busca de um país dentro de um vetor de Strings. Analise as características dos vetores mostrados e responda V ou F (verdadeiro ou falso) para cada uma das assertivas a seguir.

```
String paisesA [] = { "Ucrânia", "Turquia", "Suíça", "México", "França", "Espanha",
                      "Chile", "Brasil , "Argentina" };
String paisesB [] = { "Argentina", "Brasil", "Chile", "Dinamarca", "Espanha", "França",
                      "Inglaterra", "Turquia", "Uruguai" };
String paisesC [] = { "Canadá", "Áustria", "Chile", "Itália", "Portugal", "Grécia",
                      "Angola", "Moçambique", "Rússia" };
```

1. V Para buscar um país em paisesA[] o método mais eficiente seria o da busca binária.
2. V Para buscar um país em paisesB[] o método mais eficiente seria o da busca binária.
3. F Para buscar um país em paisesC[] seria adequado o método de busca binária.
4. V Para buscar um país em paisesC[] seria adequado o método de busca sequencial.
5. V Seria possível buscar um país em paisesB[] com o método de busca sequencial.
6. V Com o método de busca sequencial, no pior caso, encontrar um país tem complexidade O(n).
7. V Com o método de busca binária, no pior caso, encontrar um país tem complexidade O(log n).
8. F Com o método de busca sequencial, no pior caso, encontrar um país tem complexidade $O(n^2)$.

Exercício 3 - para entregar

para entregar

Implemente (ou adapte os algoritmos estudados) os métodos de **busca sequencial** e **busca binária**, para efetuar a busca de um país em exemplos de vetores como mostrados a seguir. Teste os métodos com os vetores exemplos mostrados a seguir.

Efetue chamadas para testar seus métodos, por exemplo, com os vetores:

```
String paisesA [] = { "Ucrânia", "Turquia", "Suíça", "México", "França", "Espanha",
                      "Chile", "Brasil , "Argentina" };
String paisesB [] = { "Argentina", "Brasil", "Chile", "Dinamarca", "Espanha", "França",
                      "Inglaterra", "Turquia", "Uruguai" };
String paisesC [] = { "Canadá", "Áustria", "Chile", "Itália", "Portugal", "Grécia",
                      "Angola", "Moçambique", "Rússia" };
```

Pode usar como base o projeto em **BuscasBinariasIterativas_base.zip.**

Tarefas:

- Criar um método: int **buscaSequencial**(String vet[], String buscado).
- Criar um método: int **buscaBinariaEmListaCrescente** (String vet[], String buscado).
- Criar um método: int **buscaBinariaEmListaDecrescente** (String vet[], String buscado).
- Testar os métodos anteriores. Entregar um projeto NetBeans com todos os métodos e testes dentro.

Exercício 4 (veja antes o próximo slide)

O método Merge Sort utiliza uma lógica para fundir (fusionar, misturar) os elementos de duas partes de um vetor para que o trecho total fique ordenado.

Utilizando uma lógica de fusão semelhante, construir um método simples:

mergePaises (String a[], String b[], String res[])

que recebe dois vetores de Strings **a** e **b**, ordenados em ordem crescente, e retorne o vetor **res** com todos os países (dos dois vetores) também em ordem crescente. Considere que os vetores poderiam ter diferentes tamanhos e que poderiam ter países repetidos (repita os mesmos na cópia).

Pode usar como base o projeto em **MergePaisesBase.zip.**

Teste seu método, por exemplo, fusionando os vetores:

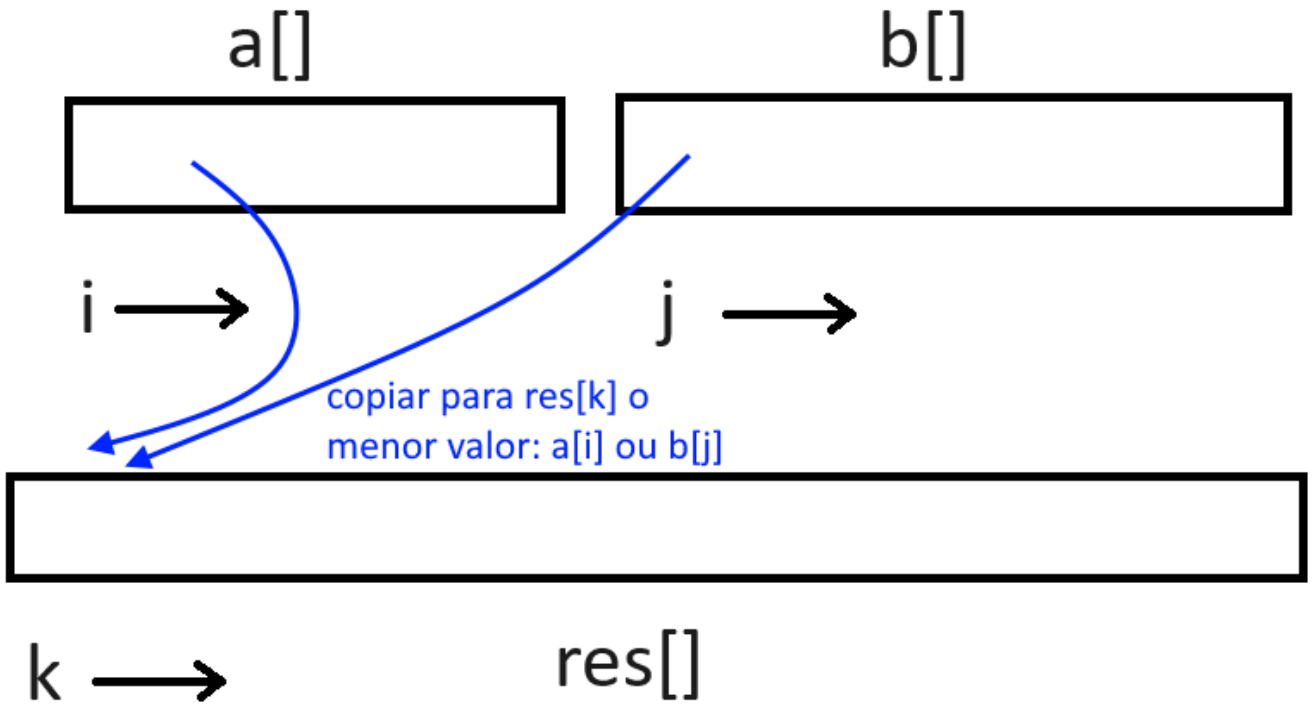
```
String veta[] = { "Angola", "Chile", "Grécia", "Itália", "Moçambique", "Portugal",
                  "Rússia", "Suécia" };
String vetb[] = { "Argentina", "Brasil", "Chile", "Dinamarca", "Espanha", "França",
                  "Inglaterra", "Turquia", "Uruguai" };
```

A resposta estará depois disponível em MergePaises.zip

Exercício 4 mergePaises(String a[], String b[], String res[])

O algoritmo Merge Sort utiliza uma lógica para fundir (fusionar, misturar) os elementos de dois trechos de um vetor, de forma que os trechos fundidos fiquem finalmente ordenados dentro das mesmas posições de um vetor auxiliar.

Este Exercício 4 utilizará um algoritmo de fusão "semelhante" ao utilizado no método merge do Merge Sort, fusionando os vetores ordenados a[] e b[] no vetor ordenado res[].



Exercício 5

Adapte o método **Quick Sort** estudado, considerando as alterações:

- Seleção do pivô utilizando o algoritmo de **mediana de três valores**.
- Ordenar valores **inteiros** em **ordem decrescente**.

Teste seu algoritmo com algum vetor de valores inteiros, por exemplo:

```
int x[] = {3, 5, -12, 34, 91, 81, 91, 2, 0, 180, 21, 76, 22, 20, 19, 43, -15, 1, 65};
```

A resposta estará disponível no projeto NetBeans OrdenacaoQuickSortMediana3

Bibliografia (oficial) para a disciplina

BIBLIOGRAFIA BÁSICA	BIBLIOGRAFIA COMPLEMENTAR
CORMEN, T. H.; et al. Algoritmos: teoria e prática. 3. ed. Rio de Janeiro: Elsevier, 2012. GOODRICH, M. T.; TAMASSIA, R. Estrutura de dados e algoritmos em java. 5. ed. Porto Alegre: Bookman, 2013. (livro físico e e-book) CURY, T. E., BARRETO, J. S., SARAIVA, M. O., et al. Estrutura de Dados (1. ed.) ISBN 9788595024328, Porto Alegre: SAGAH, 2018 (e-book)	ASCENCIO, A. F. G.; ARAÚJO, G. S. Estruturas de Dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson Prentice Hall, 2010. (eBook) PUGA, S.; RISSETTI, G. Lógica de programação e estruturas de dados, com aplicações em Java. 3. ed. São Paulo: Pearson Education do Brasil, 2016. (eBook) DEITEL, P.;DEITEL, H. Java como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017. (eBook) BARNES, D. J.; KOLLING, M. Programação Orientada a Objetos com Java: uma introdução prática usando o Blue J. São Paulo: Pearson Prentice Hall, 2004. (eBook) BORIN, V. POZZOBON. Estrutura de Dados. ISBN: 9786557451595, Edição: 1ª . Curitiba: Contentus, 2020 (e-book)