USJT - UC Gestão de Qualidade de Software - Turma: ADS1AN-BUC1-6272430

RA 823159742 - Igor Cordeiro de Souza Pereira - 823159742@ulife.com.br
RA 823217461 - Lucca Palmieri Dittrich - 823217561@ulife.com.br
RA 823123930 - Eduardo Vieira de Jesus - 823123930@ulife.com.br
RA 82426451 - Eduardo Filipe Silva S. Santos - 82426451@ulife.com.br

Exercício prático 1:

- Tema:
  Plano de Testes;
- Objetivo:
  Realizar um plano de testes para o exercício prático 1 da aula 04;
- Entrega:
  Até o prazo estipulado pela atividade proposta pelo *Google Classroom*, criar um documento contendo os resultados solicitados e os seguintes dados dos integrantes do grupo:
  - Nome para o Grupo; e
  - RA, Nome completo, E-mail, Turma e Curso de cada integrante do Grupo.
- Observação:
  Gerar um PDF desse documento e, cada integrante do grupo, devolver como atividade, uma cópia desse documento.

Para a realização do plano de testes para o exemplo de busca binária deverá ser feito um documento ou mapa onde sejam identificadas as necessidades do projeto.

A ferramenta de testes escolhida será o JUnit, que é uma ferramenta para automatizar os testes feitos na linguagem Java.

O público alvo serão os desenvolvedores, testadores e o gerente do projeto, a fim de que todas as partes envolvidas possam garantir o bom funcionamento e a qualidade do software para entrega final.

Objetivos

O objetivo é garantir que o código funcione corretamente em todas as situações esperadas e verificar se o código suporta grandes volumes de dados.

O algoritmo deve encontrar corretamente o índice de um valor no array ou retornar -1 se o valor não estiver presente.

O tempo de execução deve se manter dentro da complexidade esperada.

Cronograma

Atividade	Descrição	Tempo	Dependência
1. Preparação dos Casos de Teste	Definir os casos de teste detalhadamente.	1 dia	Nenhuma
2. Implementação dos Testes	Desenvolver scripts de teste automatizados (usando JUnit).	2 dias	Preparação dos Casos de Teste
3. Execução dos Testes Funcionais	Rodar os testes funcionais para validar o comportamento esperado.	1 dia	Implementação dos Testes
4. Execução dos Testes de Desempenho	Avaliar o desempenho com grandes dados.	1 dia	Execução dos Testes Funcionais
5. Análise dos Resultados	Revisar os resultados dos testes.	1 dia	Execução dos Testes
6. Correção de Erros (Se houver)	Corrigir possíveis erros ou otimizar o código.	2 dias	Análise dos Resultados
7. Relatório Final	Documentar o processo de testes e os resultados obtidos.	1 dia	Análise dos Resultados

Exercício prático 2:

- Tema:
  Roteiro de Testes;

- Objetivo:
  Realizar um roteiro de testes para o exercício prático 1 da aula 04;

- Entrega:
  Até o prazo estipulado pela atividade proposta pelo *Google Classroom*, criar um documento contendo os resultados solicitados e os seguintes dados dos integrantes do grupo:
  - Nome para o Grupo; e
  - RA, Nome completo, E-mail, Turma e Curso de cada integrante do Grupo.

- Observação:
  Gerar um PDF desse documento e, cada integrante do grupo, devolver como atividade, uma cópia desse documento.

Caso de Teste	Entrada	Valor Buscado	Resultado Esperado	Observações
Teste 1: Element	[1, 3, 5, 7]	8	-1	O valor 8 não está presente no array.
Teste 2: Element	[1, 3, 5, 7]	3	1	O valor 3 está na primeira metade do array.
Teste 3: Element	[1, 3, 5, 7]	7	3	O valor 7 está na segunda metade do array.
Teste 4: Array va	[]	3	-1	O array está vazio, o método deve retornar -1.
Teste 5: Array cc	[3]	3	0	O array contém apenas um elemento que é o valor buscado.
Teste 6: Element	[1, 3, 5, 7, 9]	5	2	O valor 5 está no meio do array.

Exercício prático 3:

- Tema:
  Plano de Testes;

- Objetivo:
  Realizar um plano de testes para o exercício prático 2 da aula 04;

- Entrega:
  Até o prazo estipulado pela atividade proposta pelo *Google Classroom*, criar um documento contendo os resultados solicitados e os seguintes dados dos integrantes do grupo:
  - Nome para o Grupo; e
  - RA, Nome completo, E-mail, Turma e Curso de cada integrante do Grupo.

- Observação:
  Gerar um PDF desse documento e, cada integrante do grupo, devolver como atividade, uma cópia desse documento.

Plano de Testes para Busca Binária
Ferramenta de Testes:
A ferramenta escolhida para a automação dos testes será o JUnit, que é amplamente utilizado para a execução de testes automatizados em sistemas desenvolvidos em Java.

Com o uso do JUnit, é possível garantir a repetibilidade e a consistência dos testes realizados no algoritmo de busca binária.

Público-alvo:
Os testes serão direcionados aos desenvolvedores, testadores e ao gerente do projeto, garantindo que todas as partes envolvidas possam monitorar e garantir o bom funcionamento do software para sua entrega final.

Objetivos
O objetivo principal é garantir que o algoritmo de busca binária funcione corretamente em todas as situações esperadas. Os testes buscarão verificar se o código consegue:

Encontrar o índice correto de um valor no array.
Retornar -1 quando o valor não estiver presente no array.
Manter o tempo de execução dentro da complexidade esperada para um algoritmo de busca binária (O(log n)).
Validar o comportamento do código para diferentes tamanhos de entrada, incluindo grandes volumes de dados.

Cronograma de Testes
Dia 1 - Análise do Algoritmo e Ambiente de Testes:

Definir o ambiente de testes e verificar dependências.
Revisão do código-fonte para entendimento do algoritmo.
Criação da estrutura de testes automatizados no JUnit.
Dia 2 - Criação e Execução de Casos de Teste Básicos:

Implementar os casos de teste para entradas comuns (elementos no início, meio e fim do array).
Executar os testes automatizados e validar os resultados.
Dia 3 - Testes para Casos Limites e Grandes Volumes de Dados:

Implementar casos de teste para arrays vazios, arrays com um elemento e arrays grandes.
Executar testes com volumes de dados maiores para verificar o desempenho.
Dia 4 - Validação de Resultados e Relatórios de Desempenho:

Analisar os resultados obtidos dos testes.
Gerar um relatório detalhado com o tempo de execução, consumo de recursos e comportamento do algoritmo.
Casos de Uso para Testes Automatizados
Caso de Uso: Busca Binária em Arrays Ordenados
Análise dos Requisitos:

Entrada e Pré-condições:

Um array ordenado e um valor de busca devem ser fornecidos ao algoritmo.
O array pode ser de qualquer tamanho (inclusive vazio) e conter qualquer valor numérico.
Passos para Execução:

O sistema deve receber o array ordenado e o valor a ser buscado.
O algoritmo deve realizar a busca binária no array para encontrar o índice correspondente ao valor buscado.
Comparar o valor buscado com o valor médio da porção relevante do array.
Ajustar o limite superior (iAlto) ou o limite inferior (iBaixo), conforme a posição do valor buscado no array.
Validação de Resultados:

Se o valor for encontrado, o algoritmo deve retornar o índice do elemento no array.
Se o valor não for encontrado, o algoritmo deve retornar -1.

Exercício prático 4:

- Tema:
  Roteiro de Testes;

- Objetivo:
  Realizar um roteiro de testes para o exercício prático 2 da aula 04;

- Entrega:
  Até o prazo estipulado pela atividade proposta pelo *Google Classroom*, criar um documento contendo os resultados solicitados e os seguintes dados dos integrantes do grupo:
  - Nome para o Grupo; e
  - RA, Nome completo, E-mail, Turma e Curso de cada integrante do Grupo.

- Observação:
  Gerar um PDF desse documento e, cada integrante do grupo, devolver como atividade, uma cópia desse documento.

Caso de Teste	Entrada	Valor Buscado	Resultado Esperado	Observações
Teste 1: Elemento não encontrado	[1, 3, 5, 7]	8	-1	O valor 8 não está presente no array.
Teste 2: Elemento na primeira metade	[1, 3, 5, 7]	3	1	O valor 3 está na primeira metade do array.
Teste 3: Elemento na segunda metade	[1, 3, 5, 7]	7	3	O valor 7 está na segunda metade do array.
Teste 4: Array vazio	[]	3	-1	O array está vazio, o método deve retornar -1.
Teste 5: Array com um único elemento	[3]	3	0	O array contém apenas um elemento, e este é o valor buscado.
Teste 6: Elemento no meio	[1, 3, 5, 7, 9]	5	2	O valor 5 está no meio do array.