JOGO EDUCATIVO DE ORDENAÇÃO

Instituto Federal do Paraná - IFPR

Alexandre Raphael Marques De Freitas

Ciro Guilherme Nass

Luan Mickael da Rocha

Miguel Martins Costa

Nicolas Lourenço Dos Santos

Curso: Bacharelado em Ciência da Computação

Disciplina: Estrutura de Dados

Professor: Marcelo Maia

Turma: BCC-3

Novembro de 2025

Resumo Executivo

Este relatório apresenta o desenvolvimento de um jogo educativo interativo para o ensino do algoritmo de ordenação Bubble Sort. O projeto foi desenvolvido utilizando tecnologias web (HTML5, CSS3 e JavaScript), proporcionando uma experiência de aprendizagem prática e engajante para alunos de Estrutura de Dados.

O jogo implementa um sistema de validação de passos que verifica se o jogador executa as trocas de elementos corretas segundo o algoritmo bubble sort, com sistema de vidas e feedback em tempo real. A interface gráfica intuitiva facilita o entendimento do funcionamento do algoritmo através da prática.

Palavras-chave: Algoritmo Bubble Sort, Jogo Educativo, Estrutura de Dados, HTML5, JavaScript, Aprendizagem Interativa.

Conteúdo

1 Introdução

1.1 Contextualização

A compreensão de algoritmos de ordenação é fundamental para a formação de um cientista da computação. Porém, o aprendizado teórico frequentemente carece do engajamento do aluno. Métodos tradicionais, como aulas expositivas e listas de exercícios, nem sempre são suficientes para manter o interesse e promover aprendizagem profunda.

A gamificação de conceitos educacionais é uma estratégia comprovadamente eficaz para aumentar engajamento e retenção de conhecimento. Este projeto explora essa abordagem ao criar um jogo interativo onde o aluno não apenas observa o algoritmo em funcionamento, mas participa ativamente de sua execução.

1.2 Objetivo Geral

Desenvolver um aplicativo educativo web-based que permita aos alunos da disciplina de Estrutura de Dados compreender e praticar o algoritmo Bubble Sort através de um jogo interativo com validação de passos corretos.

1.3 Objetivos Específicos

- Implementar validação automática das trocas de elementos efetuadas pelo jogador
- Fornecer feedback visual e de texto em tempo real
- Criar interface amigável e intuitiva que promova engajamento
- Facilitar expansão futura para outros algoritmos de ordenação

2 Fundamentação Teórica

2.1 Algoritmo Bubble Sort

O Bubble Sort é um algoritmo de ordenação simples que funciona repetidamente percorrendo a lista, comparando elementos adjacentes e trocando-os se estiverem na ordem errada. O algoritmo recebe seu nome porque elementos menores “borbulham” para o topo da lista.

2.1.1 Funcionamento Básico

```
Para i = 0 at  n-1:
    Para j = 0 at  n-i-2:
        Se array[j] > array[j+1]:
            Trocar array[j] com array[j+1]
```

2.1.2 Características

Propriedade	Valor
Complexidade Temporal (pior caso)	$O(n^2)$
Complexidade Temporal (melhor caso)	$O(n)$
Estável	Sim
In-place	Sim

Tabela 1: Características do Algoritmo Bubble Sort

2.2 Motivação Pedagógica

O Bubble Sort é ideal para fins educacionais por:

1. Ser conceitualmente simples
2. Facilitar compreensão visual de como comparações e trocas funcionam
3. Servir como base para entender algoritmos mais complexos
4. Permitir análise clara de desempenho

3 Arquitetura do Sistema

3.1 Visão Geral

O aplicativo segue uma arquitetura de cliente (frontend) única, sem necessidade de servidor backend. A estrutura compreende três componentes principais:

1. **HTML5** - Estrutura e semântica do documento
2. **CSS3** - Apresentação visual e responsividade
3. **JavaScript (ES6+)** - Lógica de jogo e interatividade

3.2 Estrutura de Dados

3.2.1 Variáveis de Estado Principal

```
gameState = {
    initialArray: [],           // Array embaralhado inicial
    playerArray: [],            // Array que o jogador est  ordenando
    referenceArray: [],         // Array totalmente ordenado
    referenceSteps: [],         // Todos os passos corretos do
       algoritmo
```

```
    currentStep: 0,              //  ndice  do passo atual
    lives: 3,                    // Vidas restantes
    arraySize: 5,                // Tamanho selecionado (5, 8, 10)
    isGameOver: false,           // Flag de fim de jogo
    isWon: false,                // Flag de vit ria
    totalMoves: 0,               // Total de movimentos feitos
    correctMoves: 0,             // Movimentos corretos
    hintsUsed: 0,                // Dicas utilizadas
    selectedIndices: [],         //  ndices  selecionados pelo jogador
    currentReferenceStep: 0      // Passo atual da refer ncia
}
```

3.2.2 Justificativa da Escolha de Arrays

Arrays foram escolhidos porque:

- **Simplicidade:** Melhor alinha com o conceito educativo
- **Performance:** Acesso direto aos elementos por índice
- **Visualização:** Representação visual clara e intuitiva
- **Alinhamento com Bubble Sort:** O algoritmo trabalha naturalmente com arrays

4 Decisões Técnicas

4.1 Estrutura do Projeto

```
bubble-sort-game/
- index.html        # Estrutura HTML e layout
- style.css         # Estilos e tema visual
- script.js         # Lógica do jogo
- README.md         # Documentação
```

Vantagem do arquivo único: Facilita distribuição e hospedagem em GitHub Pages.

5 Implementação Detalhada

5.1 Fluxo Principal do Jogo

O jogo segue uma sequência bem definida de estados, começando pela tela inicial, passando pela seleção de dificuldade, até o loop principal do jogo e, finalmente, a tela de resultados.

5.1.1 Geração do Array Embaralhado

```
function generateShuffledArray(size) {
    // 1. Cria array com valores  nicos  de 0 a 10
    // 2. Aplica algoritmo Fisher-Yates para embaralhamento
    // 3. Retorna array shuffled
}
```

Algoritmo Fisher-Yates:

- Garante distribuição uniforme de permutações
- Complexidade: $O(n)$
- Mais eficiente que naive shuffle

5.1.2 Geração dos Passos do Bubble Sort

```
function generateBubbleSortSteps(array) {
    // 1. Executa bubble sort enquanto rastreia cada estado
    // 2. Armazena snapshot ap s cada swap
    // 3. Retorna array de todos os estados intermedi rios
}
```

Benefício: Permite validação precisa de qualquer movimento do jogador contra a sequência correta.

5.1.3 Validação de Movimento

```
function isValidSwap(expectedArray, playerArray, pos1, pos2) {
    // 1. Cria c pia do array do jogador
    // 2. Realiza swap nas posi  es
    // 3. Compara com pr ximo passo esperado
    // 4. Retorna true se match, false caso contr rio
}
```

Lógica de Validação:

- O jogo não valida se a troca é "boa em geral"
- Valida se é o **próximo passo correto** específico
- Isto força o jogador a seguir o algoritmo passo-a-passo

5.2 Sistema de Vidas

- Começa com **3 vidas**
- Perde **1 vida** a cada erro
- **Game Over** quando vidas = 0
- Feedback visual imediato ao perder vida

6 Análise de Qualidade

6.1 Validação e Testes

6.1.1 Testes Funcionais

Caso	Entrada	Esperado	Status
Swap correto	Posições 0,1 em {6,2,...}	Troca validada	
Swap errado	Posições 0,1 quando não deve	Erro, perde vida	
Vitória	Array completamente ordenado	Tela de vitória	
Game Over	Vidas = 0	Tela de derrota	
Dica	Clica em "Dica"	Mostra próximo swap	

Tabela 2: Testes Funcionais do Jogo

6.2 Performance

Métrica	Valor	Status
Tempo de carregamento	< 500ms	
Geração de array (10 elem.)	< 10ms	
Validação de swap	< 5ms	
Renderização de tela	< 16ms	

Tabela 3: Métricas de Performance

7 Expansibilidade

7.1 Adição de Novos Algoritmos

O código foi estruturado para facilitar adição de outros algoritmos:

```
function generateInsertionSortSteps(array) {
    // Implementar insertion sort com rastreamento de passos
}

function generateQuickSortSteps(array) {
    // Implementar quick sort com rastreamento de passos
}
```

Mudanças necessárias:

1. Criar função `generateXxxSortSteps()` para novo algoritmo
2. Adicionar opção de seleção na tela inicial
3. Adicionar explicação do algoritmo na seção "Como Jogar"

7.2 Melhorias Futuras

- **Leaderboard:** Armazenar scores em localStorage
- **Modo Competitivo:** Multiplayer (dois jogadores simultâneos)
- **Temas Personalizados:** Diferentes visuais
- **Modo Automático:** Ver algoritmo resolver antes de jogar
- **Análise Detalhada:** Gráficos de comparações e trocas
- **Suporte a Mobile:** Toque em vez de clique

8 Conclusões

Este projeto demonstrou a viabilidade e efetividade de gamificação para ensino de algoritmos de estrutura de dados. A escolha de tecnologias web proporcionou:

1. **Interface Rica:** Visualização clara do algoritmo
2. **Engajamento:** Mecânica de jogo motiva aprendizado
3. **Acessibilidade:** Fácil compartilhamento e acesso
4. **Manutenibilidade:** Código simples e extensível
5. **Escalabilidade:** Preparado para novos algoritmos

O jogo cumpre seu objetivo de permitir que alunos pratiquem e compreendam o algoritmo Bubble Sort de forma interativa e lúdica.

9 Trabalhos Futuros

- Implementar Selection Sort, Insertion Sort, e Quick Sort
- Adicionar visualização animada do algoritmo
- Criar versão mobile nativa (React Native)
- Integrar com plataforma de aprendizagem (Moodle)
- Adicionar sistema de challenges com rankings
- Expandir para 50+ algoritmos diferentes

Referências

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.

[2] Knuth, D. E. (1998). *The art of computer programming: Volume 3 - Sorting and searching* (2nd ed.). Addison-Wesley.

[3] W3C. (2023). HTML Living Standard. Retrieved from https://html.spec.whatwg.org/

[4] MDN Web Docs. (2023). JavaScript Reference. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript

[5] Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

[6] Prensky, M. (2007). Digital game-based learning. *Computers in Entertainment*, 5(1), 21.