

**Curso: Bacharelado em Sistemas de Informação**

**Disciplina: Estrutura de Dados I**

**Professor: Marcelo Fernando Rauer**

**Trabalho de Estrutura de Dados: Jogo em Java baseado em árvore Ternária contando uma história baseada em Robin Hood, alinhado ao ODS 1 e 4 da ONU.**

*Jonathan Bryan Kanssaw Gomez*

Balneário Camboriú, SC – 25/06/2025

**1. Conceitos e Terminologias Fundamentais sobre Árvores**

Árvores são estruturas de dados não lineares que organizam elementos de forma hierárquica. Elas são compostas por elementos chamados **nós**, que representam os dados armazenados, e ligações entre esses nós chamadas **arestas** ou **ponteiros**.

Os principais conceitos associados a árvores são:

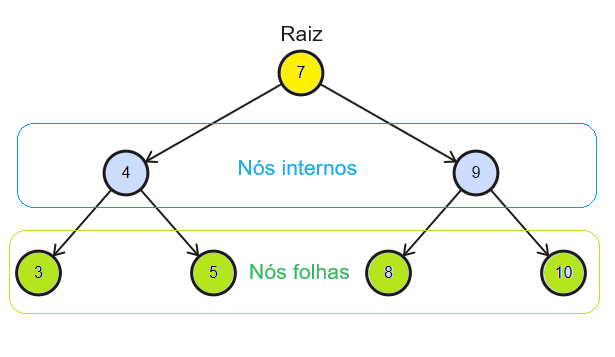
* **Nó (Node):** unidade básica da árvore, que contém um valor ou informação e referências para outros nós.
* **Raiz (Root):** é o nó principal da árvore, que não possui nenhum nó pai. É o ponto de partida para acessar toda a estrutura.
* **Filho (Child):** é um nó que está diretamente ligado a outro nó superior, chamado pai.
* **Pai (Parent):** nó que possui um ou mais filhos.
* **Irmãos (Siblings):** nós que compartilham o mesmo pai.
* **Folha (Leaf):** nó que não possui filhos, ou seja, está no final de um ramo da árvore.
* **Subárvore (Subtree):** qualquer nó da árvore, junto com todos os seus descendentes, forma uma subárvore.
* **Nível (Level):** a profundidade de um nó na árvore, geralmente o nó raiz está no nível 0, seus filhos no nível 1, e assim por diante.
* **Altura (Height):** é a maior profundidade entre todos os nós da árvore; ou seja, a distância máxima da raiz até uma folha.
* **Grau (Degree):** o número de filhos que um nó possui.

A estrutura em árvore permite representar hierarquias de forma eficiente, facilitando buscas, inserções e organização de dados complexos, como em arquivos, jogos de decisão, classificações, entre outros.

**2. Tipos de Árvores**

Neste tópico, vamos apresentar alguns dos tipos mais comuns de árvores usadas em estruturas de dados, explicando suas características, operações principais, exemplos práticos e desempenho.

**2.1 Árvore Binária**



**Descrição:**  
Árvore em que cada nó pode ter no máximo dois filhos, chamados filho esquerdo e filho direito.

**Principais características:**

* Simplicidade na estrutura.
* Pode ser usada para representar decisões binárias (sim/não).
* Não necessariamente balanceada.

**Principais operações:**

* Inserção de nós.
* Busca (em pré-ordem, em-ordem, pós-ordem).
* Remoção de nós.

**Exemplo de aplicação:**  
Árvores de decisão, expressões matemáticas.

**Desempenho:**

* Depende do balanceamento; em pior caso, a árvore pode se tornar uma lista (O(n)).

**2.2 Árvore Binária de Busca (BST)**

**Descrição:**  
Árvore binária onde os nós à esquerda têm valores menores que o nó pai, e os nós à direita têm valores maiores.

**Principais características:**

* Facilita buscas rápidas.
* Estrutura ordenada.

**Principais operações:**

* Inserção, busca, remoção, percursos (pré, em, pós).

**Exemplo de aplicação:**  
Bases de dados simples, dicionários.

**Desempenho:**

* Busca, inserção e remoção com complexidade média O(log n), pior caso O(n).

**2.3 Árvore AVL**

**Descrição:**  
Árvore binária de busca auto-balanceada que mantém o fator de balanceamento dos nós para garantir operações eficientes.

**Principais características:**

* Auto-balanceamento após inserção ou remoção.
* Garante altura O(log n).

**Principais operações:**

* Inserção com rotações para balancear.
* Busca rápida.

**Exemplo de aplicação:**  
Sistemas que exigem buscas rápidas e balanceamento constante.

**Desempenho:**

* Operações em O(log n).

**2.4 Árvore de Decisão (relacionada ao seu projeto)**

**Descrição:**  
Estrutura em árvore usada para representar decisões sequenciais com múltiplas opções em cada nó.

**Principais características:**

* Cada nó representa uma pergunta ou decisão.
* Ramo segue para um resultado ou próxima decisão.

**Principais operações:**

* Navegação conforme escolha.
* Inserção de decisões.

**Exemplo de aplicação:**  
Jogos interativos, quizzes, assistentes virtuais (como seu RPG do Robin Hood).

**Desempenho:**

* Depende da profundidade da árvore; geralmente é eficiente para tomadas de decisão.

**3. Explorando possibilidades – outras estruturas de dados**

Além das árvores, existem outras estruturas de dados importantes amplamente utilizadas na computação. Abaixo, apresento as principais, com suas características, operações e exemplos práticos.

**3.1 Listas Encadeadas**

**Descrição:**  
Estrutura linear composta por nós onde cada nó contém um dado e uma referência para o próximo nó da lista.

**Principais características:**

* Fácil inserção e remoção em qualquer posição.
* Não permite acesso direto (andom access), a navegação é sequencial.

**Principais operações:**

* Inserção, remoção, busca sequencial.

**Exemplo de aplicação:**  
Implementação de filas, pilhas, e buffers.

**Desempenho:**

* Inserção e remoção em O(1) (se já tem o nó anterior).
* Busca em O(n).

**3.2 Pilhas (Stacks)**

**Descrição:**  
Estrutura do tipo LIFO (Last In, First Out), onde o último elemento inserido é o primeiro a ser removido.

**Principais características:**

* Operações restritas ao topo da pilha.
* Simples de implementar com listas ou arrays.

**Principais operações:**

* Push (inserir), pop (remover), peek (ver topo).

**Exemplo de aplicação:**  
Desfazer ações em editores, avaliação de expressões.

**Desempenho:**

* Operações em O(1).

**3.3 Filas (Queues)**

**Descrição:**  
Estrutura do tipo FIFO (First In, First Out), onde o primeiro elemento inserido é o primeiro a sair.

**Principais características:**

* Operações restritas na inserção no final e remoção no início.

**Principais operações:**

* Enqueue (inserir), dequeue (remover), peek (ver frente).

**Exemplo de aplicação:**  
Gerenciamento de tarefas, impressão de documentos.

**Desempenho:**

* Operações em O(1).

**3.4 Tabelas Hash (Hash Tables)**

**Descrição:**  
Estrutura que associa chaves a valores usando uma função hash para facilitar acesso rápido.

**Principais características:**

* Acesso quase imediato a dados com base na chave.
* Pode sofrer colisões, exigindo tratamento.

**Principais operações:**

* Inserção, busca, remoção por chave.

**Exemplo de aplicação:**  
Dicionários, caches.

**Desempenho:**

* Operações em média O(1), pior caso O(n).

**4 Contexto e Requisitos Atendidos**

**O projeto desenvolvido tem como proposta a criação de um jogo de RPG interativo em console, baseado na história de Robin Hood, no qual o jogador deve fazer escolhas em diferentes situações que afetam diretamente o rumo da narrativa. A aplicação está alinhada aos Objetivos de Desenvolvimento Sustentável (ODS) da ONU, especificamente:**

* **ODS 1 – Erradicação da pobreza: Ao longo da história, o jogador pode realizar ações que redistribuem riqueza, ajudam populações vulneráveis e enfrentam regimes opressores.**
* **ODS 4 – Educação de qualidade: A trama permite escolhas que favorecem a consciência social, resistência intelectual e empoderamento por meio da educação popular.**

**Entre os requisitos técnicos atendidos estão:**

* **Construção de uma árvore de decisão totalmente manual, sem uso de bibliotecas ou estruturas prontas;**
* **Interface simples via console, com opções claras de escolha;**
* **Operações de navegação e leitura da árvore conforme as decisões do usuário;**
* **Fluxo lógico com finais variados e múltiplos caminhos possíveis;**
* **Validação de entrada para garantir consistência durante o uso.**

**4.1 Tipo de Árvore Escolhida e Justificativa**

**Foi utilizada uma árvore de decisão com múltiplos filhos (neste caso, até 3 por nó: A, B e C). Esse modelo foi escolhido por ser ideal para jogos de múltipla escolha, permitindo que o jogador explore diferentes possibilidades e caminhos narrativos.**

**A árvore não possui balanceamento ou ordenação — o foco está em percorrer os nós com base nas escolhas do usuário, não em eficiência de busca.**

**4.2 Estrutura Interna e Estratégia de Uso**

**A árvore foi modelada com uma classe No, que contém:**

* **String texto — armazena o conteúdo da cena ou decisão;**
* **No A, No B, No C — apontam para os próximos nós (dependendo da escolha feita).**

**Exemplo da estrutura base:**

class No {

String texto;

No A;

No B;

No C;

No(String texto) {

this.texto = texto;

}

}

**A árvore é montada manualmente no método main, com cada cena sendo instanciada e conectada logicamente a outras. O programa utiliza uma estrutura de laço com scanner para navegar pela árvore:**

* **Apresenta o texto da cena atual;**
* **Recebe entrada do jogador (A, B, C);**
* **Move o ponteiro para o nó correspondente;**
* **Se o nó atual não possui filhos, o jogo termina.**

**O fluxo completo cobre 33 finais diferentes.**

**4.3 Instruções de Execução**

1. **Certifique-se de ter o JDK instalado.**
2. **Salve o código em um arquivo chamado Arvores\_exercicio\_1.java.**
3. **Compile o programa com:**

javac Arvores\_exercicio\_1.java

1. **Execute com:**

java Arvores\_exercicio\_1

1. **Use o menu para iniciar o jogo.**
2. **Durante a jogatina, digite A, B ou C para fazer suas escolhas.**

**5. Conclusões e Discussão**

A construção do jogo RPG do Robin Hood utilizando uma estrutura de árvore foi uma experiência desafiadora e enriquecedora, tanto do ponto de vista técnico quanto criativo. Ao longo do desenvolvimento, foi possível aplicar na prática os principais conceitos teóricos de estruturas de dados vistas em sala de aula, como **nós, níveis, folhas, percursos e ramificações**.

**Integração entre teoria e prática**

A árvore de decisão permitiu explorar de forma concreta a ideia de ramificações múltiplas e caminhos alternativos. Cada escolha do jogador corresponde a uma transição de nós na árvore, o que reforçou o entendimento sobre conexões, profundidade e estrutura hierárquica. A estrutura desenvolvida também demonstrou como árvores são úteis para representar **fluxos lógicos complexos**, como os de um jogo narrativo.

**Principais desafios e soluções**

Um dos maiores desafios foi **modelar uma árvore grande e profunda** sem o uso de Programação Orientada a Objetos. Toda a árvore precisou ser construída manualmente, criando centenas de objetos e conectando-os um a um. Para contornar a repetição, foi necessário organização cuidadosa do código e padronização das nomenclaturas.

Outro desafio foi implementar um **sistema robusto de entrada de dados**, garantindo que o programa reagisse corretamente a entradas inválidas. Isso foi resolvido com loops de validação dentro da estrutura de navegação.

**Lições de desempenho e escalabilidade**

Apesar de funcional e didática, a árvore construída é estática e não possui balanceamento ou inserção dinâmica. Isso significa que, embora seja perfeita para um jogo com estrutura fixa, não seria escalável para aplicações com dados variáveis. Essa limitação, no entanto, foi adequada ao objetivo do projeto.

**Impacto no desenvolvimento profissional**

Este projeto foi fundamental para o desenvolvimento da lógica de programação, especialmente na manipulação de estruturas dinâmicas como árvores. Além disso, incentivou a criatividade e o senso crítico na criação de narrativas educativas com impacto social.

**Possíveis extensões e trabalhos futuros**

* Implementação de **interface gráfica** para tornar a experiência mais imersiva.
* Criação de **versão com persistência de dados**, para salvar o progresso do jogador.
* Adição de um **contador de escolhas** ou **trilha de decisões** para gerar estatísticas.
* Migração para uma **estrutura orientada a objetos**, com classes separadas para cenas, jogadores e ações.
* Inclusão de mais ODS da ONU, como igualdade de gênero, justiça ou meio ambiente.

**Considerações finais**

A árvore binária de decisão deixou de ser apenas um conceito abstrato e se transformou em uma ferramenta de criação interativa. O projeto conseguiu unir programação, narrativa e impacto social, provando que estruturas de dados podem ir muito além do técnico — elas podem contar histórias, ensinar valores e inspirar mudanças.

1. **Referências**

ASSUNÇÃO, Leandro. *Estruturas de Dados em Java*. 2. ed. São Paulo: Novatec, 2020.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. *Algoritmos: teoria e prática*. 3. ed. São Paulo: GEN LTC, 2012.

WIKIPÉDIA. Árvore de decisão. Disponível em: <https://pt.wikipedia.org/wiki/%C3%81rvore_de_decis%C3%A3o>. Acesso em: 25 jun. 2025.

WIKIPÉDIA. Árvore binária. Disponível em: <https://pt.wikipedia.org/wiki/%C3%81rvore_bin%C3%A1ria>. Acesso em: 25 jun. 2025.

UNESCO. *Objetivos de Desenvolvimento Sustentável*. Disponível em: <https://brasil.un.org/pt-br/sdgs>. Acesso em: 25 jun. 2025.

FREECODECAMP. Tudo sobre estruturas de dados em árvore. Disponível em: <https://www.freecodecamp.org/portuguese/news/tudo-o-que-voce-precisa-saber-sobre-estruturas-de-dados-em-arvore/>. Acesso em: 25 jun. 2025.

ORACLE. *Java SE Documentation*. Disponível em: <https://docs.oracle.com/javase/8/docs/>. Acesso em: 25 jun. 2025.

RAUBER, Marcelo Fernando. *Material da disciplina de Estrutura de Dados I*. Instituto Federal Catarinense - IFC. Notas de aula, 2025.