Universidade Federal de Mato Grosso do Sul
Engenharia de Computação
Estruturas de Dados
Prof. Dr. Anderson Bessa da Costa

Trabalho Prático 2024-1

1 Introdução

No contexto de aplicações computacionais onde dados são processados em larga escala, frequentemente nos deparamos com desafios associados ao gerenciamento eficiente de grandes volumes de informações. Uma das questões centrais é a capacidade de armazenamento e recuperação rápida de dados a partir de conjuntos extensos que não se acomodam integralmente na memória principal. Isso requer o uso de memória secundária, onde o acesso aos dados pode ser substancialmente mais lento devido ao tempo necessário para leitura e escrita de informações.

A necessidade de otimizar o tempo de acesso a estes dados, especialmente em operações de busca, inserção e remoção, torna imperativo o emprego de estruturas de dados avançadas e eficientes. Nesse contexto, a árvore B emerge como uma solução robusta, por sua capacidade de manter os dados balanceados e permitir operações eficientes mesmo quando os dados estão distribuídos em memória secundária.

Este trabalho tem como objetivo desenvolver uma implementação de uma árvore B utilizando a linguagem C, visando explorar suas propriedades e benefícios em cenários de manipulação de grandes volumes de dados. Através desta implementação, procuraremos demonstrar como a árvore B pode ser utilizada para melhorar significativamente o desempenho de sistemas que dependem de acesso rápido e eficaz a grandes tabelas de dados em memória secundária.

2 Árvore B

A árvore B, utilizando o recurso de manter mais de uma chave em cada nó da estrutura, proporciona uma organização de ponteiros tal que as operações mencionadas são executadas rapidamente. Além disso, sua construção assegura que as folhas se encontram todas em um mesmo nível, não importando a ordem de entrada de dados.

Este tipo de árvore é largamente utilizado como forma de armazenamento em memória secundária. Diversos sistemas comerciais de bancos de dados, por exemplo, as empregam.

Seja d um número natural. Uma *árvore B de ordem d* é uma árvore ordenada que é vazia, ou que satisfaz as seguintes condições:

(i) a raiz é uma folha ou tem no mínimo dois filhos;

(ii) cada nó diferente da raiz e das folhas possui no mínimo $d + 1$ filhos;

(iii) cada nó tem no máximo $2d + 1$ filhos;

(iv) todas as folhas estão no mesmo nível.

Um nó de uma árvore B é chamado *página*. Uma página armazena então diversos nós da tabela original. A estrutura apresentada satisfaz um conjunto de propriedades. A Figura 1 ilustra uma árvore B de ordem 2. Observe que todas as propriedades apresentadas são satisfeitas: (i) a raiz é uma folha ou tem no mínimo dois filhos; (ii) cada nó diferente da raiz e das folhas

possui no mínimo 3 filhos; (iii) cada nó tem no máximo 5 filhos; e por fim (iv) todas as folhas estão no mesmo nível.

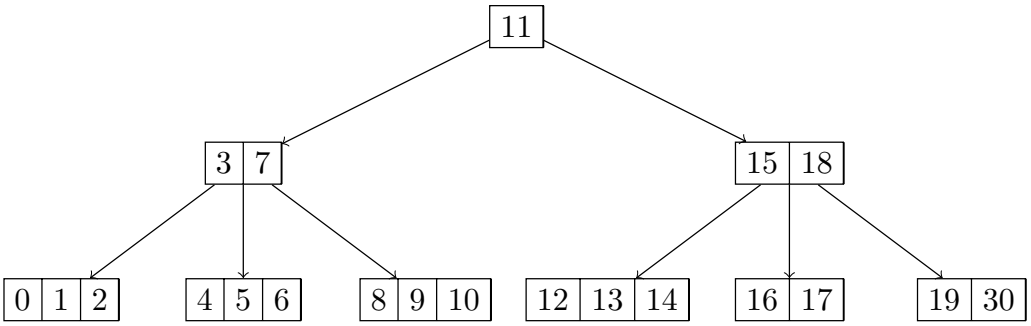

Figura 1: Árvore B de ordem 2.

Uma descrição detalhada sobre Árvores B pode ser encontrada em [Szwarcfiter and Markenzon, 2010], no Capítulo 5, Seção 5.5 "Árvores B".

3 Instruções

Você deverá ler sobre Árvores B em [Szwarcfiter and Markenzon, 2010], no Capítulo 5, Seção 5.5 "Árvores B". Deverá seguir as instruções contidos na referência. Será avaliado a razoabilidade das decisões tomadas em sua implementação. A implementação deve ser feita em linguagem C ANSI, e somente utilizando as bibliotecas padrão do C podem ser utilizadas.

Será necessário declarar um tipo registro (`struct`) representando um nó. A chave de cada nó será uma *string*. Como árvores B são interessantes quando se tem uma grande quantidade de dados, você irá populá-la utilizando a sua operação de inserção.

Uma vez implementada a sua árvore B, você irá preenchê-la (de forma automatizada) com a lista de nomes de pokémons fornecida no AVA Moodle ("pokemon_names.txt"). Insira todos os 809 pokémons (a lista não contém todos, porém a maioria).

Após popular a sua árvore B, seu programa começará de fato, com a apresentação do menu a seguir:

```
// ----- // ----- // ÁRVORE B // ----- // ----- //"
"[1] - Buscar"
"[2] - Inserir"
"[3] - Remover"
"[9] - Finalizar"
"-------------------------------"
"Entre com a sua opção: ");
```

Abaixo é apresentado uma descrição breve do comportamento esperado em cada uma das opções:

Buscar: faz a leitura de uma chave x, e na sequência realiza a busca de x na árvore B;

Inserir: insere uma nova chave x na árvore B;

Remover: remove uma chave x da árvore B;

Finalizar: finaliza o programa.

4 Entrega do Trabalho

O trabalho deve ser entregue até o dia 14 de junho de 2023 pelo *AVA Moodle*. A entrega consistirá em um único arquivo .zip contendo todos os códigos juntamente com um breve relatório descrevendo o trabalho. Neste relatório você deve incluir uma breve introdução, decisões de implementação, funcionalidades não implementadas, problemas enfrentados na implementação e uma descrição do ambiente de desenvolvimento (sistema operacional, compilador, versão compilador, etc). O relatório deve ser entregue em um arquivo PDF.

O trabalho pode ser feito em grupos de até três alunos. Casos de plágio serão tratados com rigor, de forma que todos os envolvidos terão nota igual a **ZERO**. Caso você faça o trabalho em grupo, submeta apenas um trabalho e identifique os componentes do grupo no relatório e no código fonte.

O trabalho vale 10 pontos. Entregas com atrasos serão aceitas, porém será descontado 2 pontos por dia de atraso.

5 Atividade em Sala

Árvores B satisfazem as seguintes propriedades [Szwarcfiter and Markenzon, 2010]:

(a) Seja m o número de chaves em uma página P não folha. Então P tem $m + 1$ filhos. Consequentemente, cada página possui entre d e $2d$ chaves, exceto o nó raiz, que possui entre 1 e $2d$ chaves.

(b) Em cada página P, as chaves estão ordenadas: $s_1, \ldots, s_m$, $d \leq m \leq 2d$, exceto para a página raiz onde $1 \leq m \leq 2d$. E mais, P contém $m + 1$ ponteiros $p_0, p_1, \ldots, p_m$ para os filhos de P. Nas páginas correspondentes às folhas, esses ponteiros indicam `NULL`. A tripla (s_k, I_k, p_k), onde I_k indica a informação associada à chave s_k, ou omitindo-se I_k, o par (s_k, p_k) é chamado uma *entrada*, $k > 0$. Se $k = 0$, por sua vez, o ponteiro p_0 é definido como *entrada*.

(c) Seja uma página P com m chaves:

- para qualquer chave y, pertencente à página apontada por p_0, $y < s_1$;
- para qualquer chave y, pertencente à página apontada por p_k, $1 \leq k \leq m - 1$, $s_k < y < s_{k+1}$;
- para qualquer chave y, pertencente à página apontada por p_m, $y > s_m$;

Exercícios

1. Defina uma estrutura (*struct*) para representar uma *página* (nó) da árvore B. Neste exercício, considere que a chave da árvore seja valores inteiros. Também defina d como uma constante para representar o grau mínimo do nó.

2. Na função *main*, instancie um nó raiz para a árvore B, e construa manualmente a árvore representada em Figura 1.

3. Realize um percurso pré-ordem na árvore construída, imprimindo durante o percurso o conteúdo de cada nó. Atente-se que um nó pode ser composto por várias chaves.

Referências

[Szwarcfiter and Markenzon, 2010] Szwarcfiter, J. L. and Markenzon, L. (2010). *Estruturas de Dados e seus Algoritmos (3a. ediçao)*. Editora LTC.