Banco de dados de imagens utilizando árvore B

Reberth Kelvin Santos de Siqueira
Algoritmo e Estrutura de dados II – Turma: Integral
e-mail: reberthkss@outlook.com

Resumo. *A árvore B é uma estrutura de dados muito utilizada em banco de dados. Seu paradigma é diferente da árvore binária, ela cresce de baixo para cima, armazena os dados em formato de blocos(ou páginas) e aplica indices para navegação. Este trabalho busca aplicar os conceitos da árvore B utilizando imagens separadas por categoria e mostrar como podemos utilizar a árvore B para realizar buscas, remoções e listagens.*

Introdução

Árvores B existem desde 1971, seus autores, Rudolf Bayer e Edward Meyers McCreight projetaram tal estrutura de dados para ser especialmente utilizada em memória secundária (como HDs, SSDs, etc), derivada da árvore binária AVL, está árvore foi projetada para ser auto-balanceavel, isso se deve ao seu paradigma de crescimento de baixo para cima e aplicação de indexação em blocos (ou páginas) de dados.

Este trabalho busca aplicar parte do conceito da árvore B, a unica diferença é que os dados não são armazenados em disco e sim na memória interna. Apesar disso, foi utilizado um repositorio para buscar os dados a serem inseridos.

Com o objetivo de avaliar todas as operações da árvore B foram desenvolvidos algoritmos de inserção, exclusão, busca e listagem. Todos algoritmos foram implementados em linguagem C utilizando a IDE CLion da desenvolvedora de Software JetBrains.

Procedimento Experimental

Para este projeto foi utilizado a linguagem C como ferramenta de programação e a IDE CLion da empresa Jetbrains.

A etapa inicial do projeto foi coletar 100 imagens do Google Imagens e de acervo pessoal separadas por categorias, sendo que para cada categoria são 5 imagens (foto1, foto2, foto3, foto4 e foto5), assim totalizando vinte categorias de fotos.

O programa foi desenvolvido para armazenar, listar, buscar e excluir os dados com base na categoria fornecida pelo usuário.

Com o objetivo de armazenar números inteiros invez de strings, foi projetado uma função que convertia uma string em um dado número n que é definido pela soma dos códigos ASC III de cada caractere.

A estrutura de dado

A estrutura desenvolvida se baseia em uma estrutura de árvore B para números inteiros, BBinary e seu ponteiro PBBinary.

```
typedef struct BBinary {
int elements, order, * values, isLeaf;
struct BBinary ** pointers;
} * PBBinary;
```

Destaca-se a implementação de um ponteiro duplo para a estrutura para a variavel **pointers**. Tanto essa variavel quanto a variavel **values** são utilizadas pelo programa para criar dinamicamente vetores. Para variavel pointers cria-se um vetor de ponteiros para a estrutura BBinary e para a variavel **values** cria-se um vetor de inteiros para armazenar as chaves de identificação.

A lógica do programa

O programa completo possui uma sessão de menu e manipulação de erros. Apesar disso, aqui será tratada apenas a lógica para implementação da árvore B.

Para inserir todos os dados referente as categorias, o programa busca os dados no arquivo nomeado como data.txt, este arquivo fornece um dado por linha, ao finalizar a leitura de cada linha o programa realiza a conversão da string utilizando a função **getStringAsInt**:

```
int getStringAsInt(char *string, int size) {
  int sum = 0;
  for (int i = 0 ; i < size ; i++) {
    sum += (int) string[i];
```

```
    }
    return sum;
}
```

Após obter o valor da string como inteiro, o programa inicia o processo de armazenamento na árvore B invocando a função insertInBBinary:

```
void insertInBBinary(PBBinary *root, int value)
{
    int order;
    PBBinary node = *root, newNode = NULL;
    order = node->order;
    if (node->elements == (2*order) - 1) {
        newNode = getBBinaryNode(order);
        newNode->isLeaf = false;
        newNode->pointers[0] = node;
        splitPage(newNode, 0, node);
        insertNonFull(newNode, value);
        *root = newNode;
    } else {
        insertNonFull(*root, value);
    }
}
```

Este processo repete-se até que o programa tenha finalizado a leitura de todos os dados. Destaca-se a utilização de um ponteiro duplo para a váriavel **root,** pois somente assim é possível subir o nível da árvore após a divisão.

Após a inserção o usuário está hábil à buscar a imagem desejada. Este processo é realizado pela função **searchByRotule** que requisita o nome da categoria que o usuário deseja buscar, realiza a busca e abre as imagens caso encontre todas as 5 imagens para a categoria desejada:

```
int searchByRotule(PBBinary root) {
    char rotule[10];
    printf("Insert the rotule for search: ");
    scanf("%s", rotule);
    printf("Searching for %s ...\n", rotule);
    for (int i = 1 ; i <= 5 ; i++) {
        int imgValue = getStringAsInt(rotule,
getSize(rotule))+48+i;
        char numberAsString[100] = {(char)
(48+i), '\0'};
        char rotuleCopy[100];
        strcpy(rotuleCopy, rotule);
        strcat(rotuleCopy, numberAsString);
        strcat(rotuleCopy, ".jpg");

        if (searchForFile(root, imgValue)) {
            system(rotuleCopy);
        } else {
            return 0;
        }
    }
    return 1;
}
```

Destaca-se aqui a manipulação necessária para converter a categoria selecionada para um número inteiro que corresponda corretamente ao que foi armazenado na árvore e a conversação direta de um número inteiro para string.

O usuário também pode inserir apenas uma imagem utilizando a função **insertInBBinary** descrita anteriormente.

Foi implementada também uma função que aplica a exclusão de uma dada categoria:

A função para listagem da árvore B foi implementada conforme a seguir;

```
void printfBBinary(PBBinary root) {
    if (root->isLeaf) {
        for (int i = 0 ; i < root->elements ; i++) {
            printf("%i ", root->values[i]);
        }
    } else {
        for (int i = 0; i <= root->elements ; i++) {
            printfBBinary(root->pointers[i]);
            if (root->elements > i) printf("%i ", root-
>values[i]);
        }
    }
}
```

A abertura das imagens

Inicialmente o método desejado para abertura das imagens era a utilização da função **ShellExecute** da biblioteca windows.h:

```
ShellExecute(NULL,
        "open",
        "C:\\path\\to\\jpg\\my.jpg",
        NULL,
        NULL,
        SW_SHOWDEFAULT);
```

Após identificar que este método precisaria do caminho completo do arquivo e para cada máquina o programa pode estar em caminhos diferentes optou-se por utilizar a função **system** como pode ser observado em um trecho da função **searchByRotule**:

```
system(rotuleCopy);
```

Resultados

Testes de desempenho foram aplicados nesta estrutura, porém sem nenhuma base de comparação com outros tipos de árvores. Os resultados seguem na tabela 1.

Operação	Complexidade	Categoria desejada	Tempo (s)
Busca	O(log n)	skate	3.428s
		grecia	2.902s
		mulher	2.768s
		naruto	4.871s
		nasa	3.227s
		trem	2.539s
		trilha	2.305s
		praia	4.028s
		bike	3.614s
		carro	2.522s
		buda	2.266s
		mesa	2.442s
		ouro	3.828s
		matrix	3.623s
		flor	2.670s

Tabela 1 – Tabela de resultados da busca por categoria

Discussão

Com esta pesquisa desenvolvida foi possível identificar o quão vantajoso pode ser a utilização de uma árvore com estrutura de dados de indexação. Além de ser auto-balanceada ela consegue fornecer análise de dados muito mais rápidamente que a árvore binária AVL ou comum.

Conclusão

A aplicação da árvore B foi realizada com sucesso, suas operações de busca, inserção e exclusão foram aplicadas corretamente e mostram-se superiores à implementação da árvore binária de balanceamento.

Referências:

Lorenzo. Open jpeg with win32 api c++, Stackoverflow, 2012-11-01, Disponível em: https://stackoverflow.com/questions/13179327/open-jpeg-with-win32-api-c, Acesso em: 01 de fevereiro de 2021.

Binks, OpenFileDialog Window in C, Março de 2011, Disponível em: https://cboard.cprogramming.com/c-programming/136029-openfiledialog-window-c.html, Acesso em: 01 de fevereiro de 2020.

Donatelo, Appending one string at the end of another - strcat, Codingame, Disponível em: https://www.codingame.com/playgrounds/14213/how-to-play-with-strings-in-c/string-concatenation, Acesso em: 01 de fevereiro de 2021.