Trabalho 3
Árvore-B

SCC 0503 – Algoritmos e Estruturas de Dados II (Sistemas de Informação)
SCC 0603 – Algoritmos e Estruturas de Dados II (Engenharia da Computação)

Professora:Elaine Parros Machado de Sousa – parros@icmc.usp.br

Estagiários PAE:
Diego Silva – diegofsilva@icmc.usp.br
Ivone Penque Matsuno – ivone.matsuno@usp.br

Alunos:

Guilherme Prearo	/	8910409
Gustavo Nicolau Goncalves	/	9012945
Pedro Virgilio Basilio Jeronymo	/	8910559

1. Código Fonte

O código fonte do projeto foi dividido em 3 partes.

- btree.c
- regfile.c
- trabalho3.c

2. Solução

a. Estruturas do arquivo de indice

[RRN da raiz] [próximo rrn livre] [folha] [Quantidade de chaves na página] [$chave_1$][$offset_1$]..[$chave_n$][$offset_n$] [$filho_1$]..[$filho_{n+1}$]

b. Estrutura de dados utilizadas

Estrutura da àrvore-B

```
typedef struct {
        int raiz ;
        int nextRrn ;
        int ordem ;
        FILE *bTFile ;
} bTree;
```

Estrutura de uma página da Árvore-B:

```
typedef struct {
        int rrn ;
        int cntChave ;
        chave *chaves ;
        int *filhos ;
        char folha;
} pagina;
```

Estrutura de uma chave da Árvore-B:

```
typedef struct {
        int id;
        long offset;
} chave;
```

c. Estratégias de implementação das operações da árvore-b

i. Criação

Primeiramente verifica-se se há algum arquivo de Árvore-B atualizado existente, caso já exista armazena-se o RRN da página raiz. Caso contrário é necessário a criação de uma nova Árvore-B.

A criação da Árvore-B se dá pelo acesso sequencial do arquivo de dados, e a inserção de cada arquivo na Árvore-B.

ii. Inserção

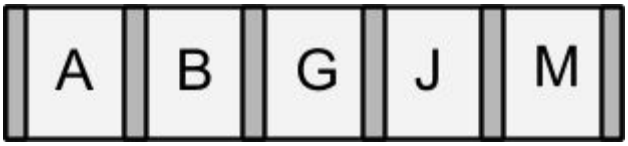
A inserção na Árvore-B é do tipo Botton-Up, o que significa que novas chaves são sempre inseridas nas folhas, e isso garante que as melhores chaves separadoras sejam escolhidas e que o balanceamento seja feito naturalmente.

Para a Inserção estudamos 4 casos: árvore vazia,overflow de nó raiz, inserção em nós folha com e sem overflow.

Árvore Vazia:

Ciração do nó raiz, após criado as chaves so inseridas e ordenadas até overflow.

EX: Inserção de B,A,M,G,J. Inserção ocorre normalmente ordenando os termos.

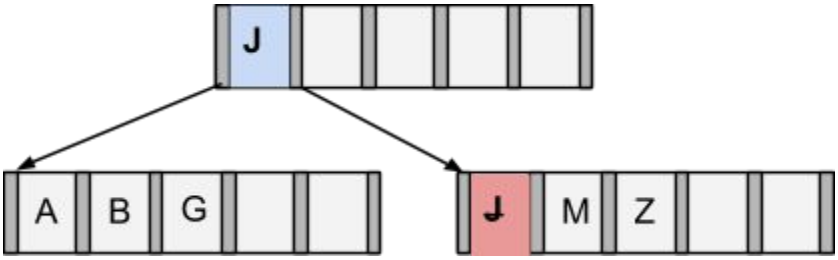


Overflow de nó Raiz:

Primeiramente fazemos o particionamento do nó em dois dividindo as chaves igualmente entre eles.

Exemplo inseção de Z.

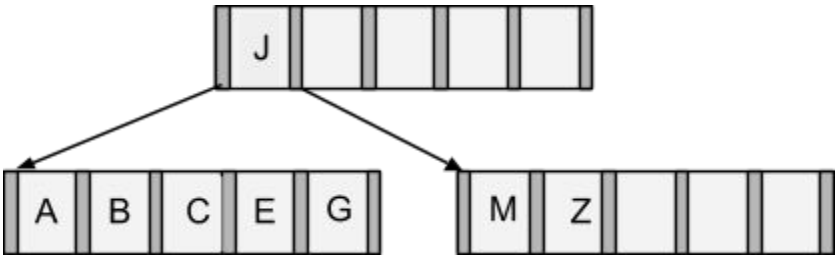
Próximo passo é a criação de um novo nó raiz e a promoção de uma chave para separadora, sendo ela a maior da página da direita.



Assim teremos a melhor chave separadora e também uma árvore balanceada.

Nó folha(sem overflow):

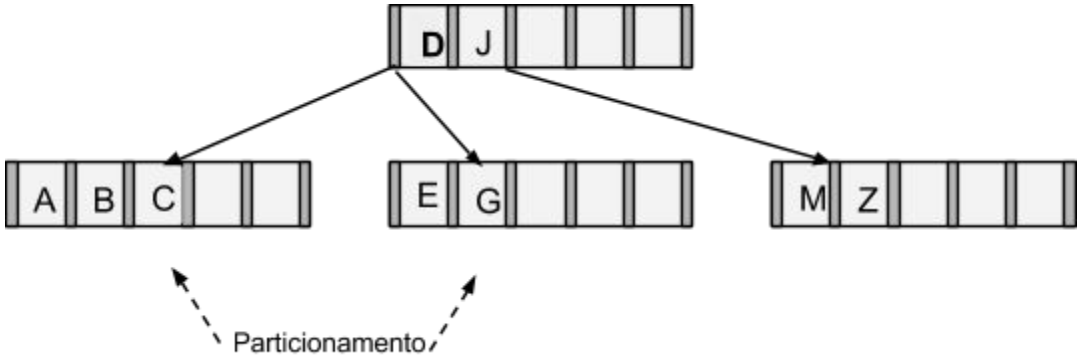
Primeiramente fazemos a busca para descobrir em qual página devemos inserir o elemento em questão. Caso haja espaço o elemento é inserido e a página ordenada.
Exemplo inserção de C e E:



Nó folha(com overflow):

Caso não haja mais espaço, faz-se a busca e ao achar a página a inserir, faz-se o particionamento em duas com distribuição igual de elementos para cada página, com o elemento a ser inserido. Promove o maior elemento da menor página, inserindo-o no nó pai, e propagando o overflow se necessário.

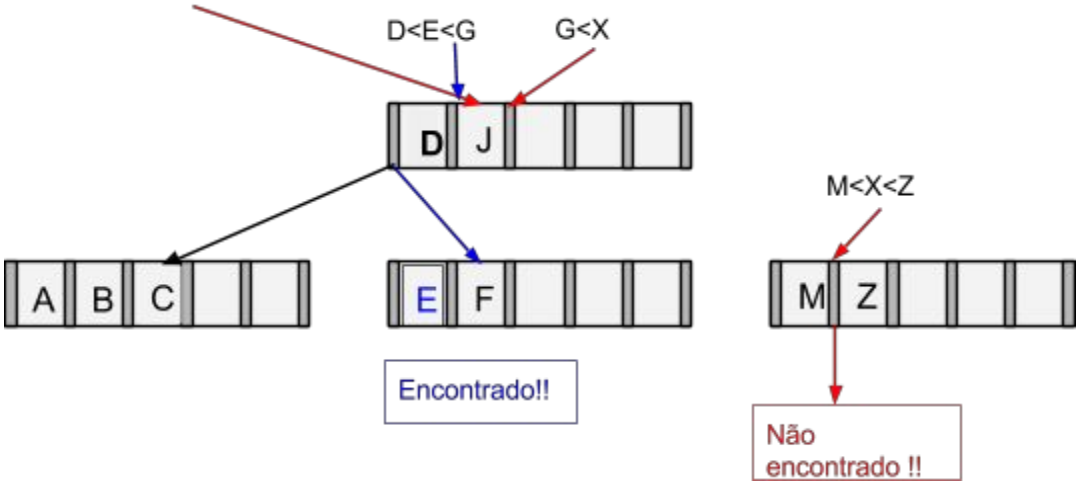
EX: Inserção de D:



iii. Busca

A busca se resume em, a partir da raiz, verificar se a chave está em uma determinada página, se não estiver verificar subárvore apropriada. E assim até encontrar a chave ou até chegar a uma folha.

EX: Busca por E e por X:



iv. Remoção

Para a remoção optamos por um método bem simples. Normalmente na remoção o item seria removido da Árvore-B, porém, a remoção da árvore tem um alto custo de processamento e acesso a disco. Então, nossa solução foi remover o registro apenas do arquivo de dados e futuramente poderia ser feita uma rotina que reconstŕoi a árvore eliminando os itens que já foram removidos do arquivo de dados. O método utilizado não afeta as outras partes do programa, pois mesmo que a busca na árvore retorne um dado de um registro já removido, na hora de recuperar esse registro no arquivo, a operação irá falhar.

d. Complexidade

Para a complexidade estudaremos dois casos, o pior e o melhor. Sendo o pior a árvore com ocupação mínima de 2 elementos por página (nesse caso de ordem 6), e o melhor com a ocupação máxima de 5 elementos por página.
Seja N o número de Chaves.

i. Tempo

Pior caso:

Busca = $\log_3(N)$
Inserção = $\log_3(N)$
Remoção = $\log_3(N)$

Melhor Caso:

Busca = $\log_6(N)$
Inserção = $\log_6(N)$
Remoção = $\log_6(N)$

ii. Espaço

O(N), ou seja, proporcional à N

3. Conclusão

Concluímos que a Árvore-B é um excelente método de indexação de arquivos, pois permite uma complexidade logarítmica para todas as operações necessárias, além de precisar de memória proporcional ao número de elementos armazenados.