Organização e Recuperação da Informação

Árvores B+

DC – UFSCar
Jander Moreira

Conteúdo

- Agenda
  - Árvores B+
    - Característica
    - Inserção
    - Remoção
    - Pesquisa
    - Mapeamento para arquivo

Árvores B+

- Características
  - Árvore de busca multi-caminhos
  - Organização
    - Divisão de tipos de nós
      - Nós internos: guardam valores de chave como critério de descida na árvore
      - Nós folhas: guardam as chaves dos registros que efetivamente são válidos
    - Balanceamento
      - Todos os nós ficam no mesmo nível
      - Crescimento da árvore *bottom-up*
    - Árvore de ordem *k*
      - Possui máximo de *k* filhos
      - Possui máximo de *k* – 1 chaves no nó
      - Possui mínimo de *k*/2 filhos
      - Possui mínimo de (*k* – 1)/2 chaves no nó, exceto pela raíz
    - Não existem "ponteiros sem chaves" ou "chaves sem ponteiros" na árvore (sempre completa)

Árvores B+

- Inserção
  - Ocorre sempre no nível das folhas
  - Critério
    - Se houver espaço no nó folha
      - A chave é alocada nesse nó
    - Se não houver espaço no nó folha
      - Um novo nó é criado
      - As chaves são distribuídas equitativamente entre o nó original e o novo nó
      - Uma das chaves é escolhida para ser inserida **(promovida)** no nível superior (critério de separação entre os dois nós)
    - Nós internos acomodam as chaves promovidas
      - Havendo espaço, as chaves são inseridas
      - Se não há espaço, há nova divisão e uma nova promoção de chave ocorre

Árvores B+

```
insira(chave)
    se raiz não existe, então
        crie novo nó
        insira a chave no nó
        ajuste os ponteiros
        marque o nó como folha
        defina esse nó como raiz
    senão
        insiraRecursivo(raiz, chave)
        se houve divisão, então
            crie um novo nó
            marque o nó como interno
            coloque a chave promovida no nó
            faça o ponteiro esquerdo desta chave \
                   apontar para a raiz
            faça o ponteiro direito desta chave \
                   apontar para o novo nó do nível inferior
            defina esse novo nó como raiz
        fim-se
    fim-se
```

Árvores B+

```
insiraRecursivo(nóAtual, chave)
    se nóAtual é folha, então
        se existe espaço em nóAtual, então
            insere ordenadamente a chave no nóAtual
            registre que não houve divisão
        senão
            TrateDivisãoNóFolha()
        fim-se
    senão
        determine qual por qual ramo deverá haver a descida na árvore
        defina o nóAlvo como o nó vinculado a esse ramo
        insiraRecursivo(nóAlvo, chave)
        se houve divisão, então
            se há espaço em nóAtual, então
                insira ordenadamente a chave promovida no nóAtual
                faça o ponteiro à direta desta chave apontar para o \
                                nó criado no nível inferior
                registre que não houve divisão
            senão
                TrateDivisãoNóInterno()
            fim-se
        fim-se
    fim-se
```

Árvores B+

```
TrateDivisãoNóFolha()
   crie um novo nó e marque-o como folha
   distribua as chaves entre o nó original e o novo
   ajuste os ponteiros entre os nós
   escolha como chave promovida a maior chave do nó original
   registre que houve divisão
```

```
TrateDivisãoNóInterno()
   crie um novo nó e marque-o como interno
   distribuas as chaves entre o nó original e o novo nó
   ajuste os ponteiros para os nós inferiores
   escolha como chave promovida a maior chave do nó original
   remova a chave promovida do nó original
```

Árvores B+

- Remoção
  - Ocorre sempre nas folhas
  - Critério
    - A chave é removida do nó incondicionalmente
    - No caso do nó ficar com ocupação inferior a $(k - 1)/2$
      - Verificar a possibilidade de empréstimo
      - Realizar a fusão de nós
    - Sempre que houver fusão de nós
      - Remover a chave que era critério de separação dos nós
      - Verificar se a condição de ocupação mínima foi violada

Árvores B+

```
remova(chave)
    se raiz existe, então
        removaRecursivo(raiz, chave)
        se a raiz ficou vazia, então
            defina o nó do único ponteiro existente como raiz
            libere o nó raiz antigo
        fim-se
    fim-se
```

Árvores B+

```
removaRecursivo(nóAtual, chave)
    se nóAtual é folha, então
        remova a chave do nó
        se a ocupação do nó for inferior ao mínimo, então
            registre a sub-ocupação
        senão
            registre que não há sub-ocupação
        fim-se
    senão
        determine qual por qual ramo deverá haver a descida na árvore
        defina o nóAlvo como o nó vinculado a esse ramo
        removaRecursivo(nóAlvo, chave)
        se há sub-ocupação no nível inferior, então
            trateSubOcupação()
        fim-se
    fim-se
```

Árvores B+

```
trateSubOcupação()
   se é possível emprestar do nó à esquerda, então
      trateEmprestimoEsquerda()
   senão
      se é possível emprestar do nó à direita, então
         trateEmpréstimoDireita()
      senão
         trateFusaoDeNós()
         remova a chave que era critério de separação entre os nós
         se a ocupação do nó for inferior ao mínimo, então
            registre a sub-ocupação
         senão
            registre que não há sub-ocupação
         fim-se
      fim-se
   fim-se
```

Arvores B+

```
trateEmprestimoEsquerda
   se os nós filhos são folhas, então
      transfira a maior chave do nó à esquerda para nóAlvo
      corrija a chave critério de separação desses nós
      registre que não sub-ocupação
   senão
      transfira a chave que separa os nós para o nó da direita
      transfira a maior chave do nó da esquerda para nóAtual
      transfira o último ponteiro do nó da esquerda como \
                primeiro ponteiro do nó da direita
   fim-se
```

```
trateEmprestimoDireita
   se os nós filhos são folhas, então
      transfira a menor chave do nó à direita para nóAlvo
      corrija a chave critério de separação desses nós
      registre que não sub-ocupação
   senão
      transfira a chave que separa os nós para o nó da esquerda
      transfira a menor chave do nó da direita para nóAtual
      transfira o primeiro ponteiro do nó da direita como \
                último ponteiro do nó da direita
   fim-se
```

Árvores B+

```
trateFusãoDeNós()
   se existe um nó imediatamente à esquerda do nóAlvo, então
      defina chaveCritério como a que separa nóEsquerdo de nóAlvo
      façaFusão(nóEsquerdo, nóAlvo, chaveCritério)
   senão
      defina chaveCritério como a que separa nóAlvo de nóDireito
      façaFusão(nóAlvo, nóDireito, chaveCritério)
   fim-se
```

```
façaFusão(nóEsquerdo, nóDireito, chaveCritério)
   se os nós são folhas, então
      transfira todas as chaves do nóDireito para o nóEsquerdo
      ajuste os ponteiros do nó
   senão
      insira a chaveCritério no nóEsquerdo
      copie para o ponteiro à direita dela o primeiro \
                  ponteiro do nóEsquerdo
      transfira todas as chaves do nóDireito para o nóEsquerdo
   fim-se
   libere o nóDireito
```

Mapeamento para arquivo

- Mapeamento
  - Cada nó deve possuir o tamanho de um bloco de disco
    - Cada acesso a um nó significa um acesso a disco
    - A árvore tende a minimizar o número de nós envolvidos em cada processo
    - A ordem da árvore é dada por quantas chaves podem ser colocadas em um bloco

Leitura

- Leituras
  - Drozdek
  - Tanembaum
  - Garcia-Molina