UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
Campus Chapecó
Ciência da computação
2015.2
Estrutura de dados II

Árvore B+

Integrantes: Nicholas Brutti e
Jefferson Coppini

Chapecó
01/12/2015

Objetivo:

Dadas as técnicas de manipulação de dados, o grande objetivo é implementar uma estrutura capaz de armazenar dados de uma tabela(.csv). Para isso utilizamos uma árvore B+.
Foi utilizado o carregamento em massa para inserir os dados na árvore. Ou seja, foram armazenados, ordenados e após isso inseridos na estrutura.

Após o carregamento o usário terá a possibilidade de:

→ Remoção de Ids;
→ Visualização da estrtura;
→ Contrução de um Rank em ordem alfabético dos dados armazenados;

Estrutura da árvore:

```
typedef struct data { // dado de cada nodo
        int*id ;
        int quant_id;
        char *texto;
} data;


typedef struct nodo { // nodo da árvore
        struct nodo *filho;
        data *dado;
        struct nodo * pai;
        int num_data;
        struct nodo * ant;
        struct nodo * prox;
} nodo;

// Cria um novo nodo e retorna o ponteiro para o nodo criado
nodo * cria_nodo();

// Recebe como parâmetro um ponteiro para árvore
// Verifica e ordena a árvore
void verifica(nodo* arv);

// Recebe como parâmetro um ponteiro  para a árvore, um vetor de índice e o tamanho do vetor data
// Retorna um ponteiro para o ínicio da árvore
nodo* insere(nodo *arv, int I[],int tam_vetor);

// Recebe como parâmetro um ponteiro para a árvore e um interiro que representa o id a ser excluído
// Retorna um ponteiro para ínicio da árvore
void remove_id(nodo* arv, int id);
```

// Recebe como parâmetro um ponteiro para á arvore
// Imprime a árvore na tela
void print(nodo * arv);

// Recebe como parâmetro um ponteiro para árvore
// Gera um ranking em ordem alfabética
void rank(nodo * arv);

// Recebe como parâmetro um ponteiro para a árvore
// Limpa a árvore
void limpa_arvore(nodo * arv);

// Recebe como parâmetro um vetor de dados e um inteiro n
// Realiza a ordeção do vetor
void insertion_sort(data * a, int n);

// Recebe como parâmetro um ponteiro para uma string, um inteiro que representa o tamanho da substring e um inteiro com a posição a ser inserida no vetor
void split_lim(char *token, int numCar,int j);

// Recebe como parâmetro uma ponteiro para uma string, um inteiro representa o número do atributo e um inteiro que representa a posição a ser inserida.
void split(char *linha, int numCol, int numCar,int i);

// Recebe como parâmetro um ponteiro para uma string e um inteiro que representa a posição a ser inserida no vetor
void splitid(char *linha,int i);