ALGORITMI E STRUTTURE DATI

materiale per la prova pratica

master COPY COPISTERIA
050/8312126 388/9837745
MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

a cura di due studenti non troppo anonimi.

Copisteria MASTER COPY via G.Battista Niccolini, 21 56122 Pisa (PI) tel: 050 8312126

"If debugging is the process of removing bugs, then programming must be the process of putting them in."
(*Dijkstra)*

"Testing can only prove the presence of bugs, not their absence."
(Dijkstra)

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

STRUTTURE DATI

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

ALBERI BINARI DI RICERCA – BST (Binary Search Tree)

Gli alberi di ricerca sono strutture dai che supportano molte operazioni sugli insiemi dinamici, fra le quali `SEARCH, MINIMUM, MAXIMUM, PREDECESSOR, SUCCESSOR, INSERT e DELETE.`
Le operazioni di base su un albero di ricerca richiedono un tempo proporzionale all'altezza dell'abero (*h=log n*). Per un albero binario completo di n nodi, tali operazioni sono eseguite nel tempo *Θ(log n)* nel caso peggiore.
Un albero binario di ricerca è organizzato in un albero binario che può essere rappresentato da una struttura dati concatenata in cui ogni nodo è un oggetto. Oltre a una chiave (*key*) e ai dati satelliti, ogni nodo comprende gli attributi *`left, right e p`* che puntano ai nodi che corrispondono, rispettivamente, al figlio sinistro, al figlio destro e al padre del nodo. Se manca un figlio o un padre, i corrispondenti attributi contengono il valore `NULL`. Il nodo radice *(root)*, è l'unico nodo dell'abero il cui attributo padre è `NULL`.
Le chiavi in un BST sono sempre organizzate in modo da soddisfare la *proprietà fondamentale degli alberi binari di ricerca*: 'Sia `x` un nodo in un albero binario di ricerca. Se `y` è nodo del sottoalbero sinistro di `x`, allora `y.key <= x.key`. Se `y` è un nodo nel sottoalbero destro di `x`, allora `y.key >= x.key`.

```
// Struttura di un nodo composto da coppie intero-stringa
struct Node
{
        int key_;
        string string_;
        Node * left;
        Node * right;
        Node(int k, string s):
                key_(k) , string_(s) , left(NULL) , right(NULL) {}
};

// Classe per la creazione di BST
class BinTree{
        Node * root_;
public:
        BinTree() { root_ = NULL; }
        Node * getRoot() { return root_; }

// Funzioni per inserire in un albero già BILANCIATO
        void insert( int k , string s )
        {
                Node * node = new Node(k, s);
                Node * pre = NULL;
                Node * post = root_;
                while( post != NULL)
                {
                        pre = post;
                        if( k <= post->key_)
```

```
                        post = post->left;
                else
                        post = post->right;
        }
        if( pre == NULL )
                root_ = node;
        else if( k <= pre->key_ )
                pre->left = node;
        else
                pre->right = node;
}

// Codice per ottenere MAX e MIN di un BST
        Node * min()
        {
                Node * temp = root_;
                while( temp->left != NULL )
                        temp = temp->left;
                return temp;
        }
        Node * max()
        {
                Node * temp = root_;
                while( temp->right != NULL )
                        temp = temp->right;
                return temp;
        }

// Metodi di visita di un albero
        void inOrder( Node * tree )
        {
                if(tree!=NULL)
                {
                        inOrder(tree->left);
                        cout << tree->key_ << " " << tree->string_ << endl;
                        inOrder(tree->right);
                }
                cout << endl;
        }

        void postOrder( Node * tree )
        {
                if(tree!=NULL)
                {
                        postOrder(tree->left);
                        postOrder(tree->right);
                        cout << tree->key_ << " " << tree->string_ << endl;
                }
                cout << endl;
        }
```

Master COPY COPISTERIA 050/8312126 388/9837745