HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

C Programming Basic

Searching – part 2

Binary Search Tree

- Objective
  - Manipulate with Binary Search Tree (BST)
  - Apply BST in a profiles management problem

- Binary Search Tree
  - Key of each node is greater than the keys of nodes of the left sub-tree and less than the keys of nodes of the right sub-tree

Binary Search Tree

- **Exercise** Given a BST initialized by NULL. Perform a sequence of operations on a BST including:
  - insert k: insert a key k into the BST (do not insert if the key k exists)
- **Input**
  - Each line contains command under the form: “insert k”
  - The input is terminated by a line containing #
- **Output**
  - Write the sequence of keys of nodes visited by the pre-order traversal (separated by a SPACE character)

Binary Search Tree

- **Example**

Input	Output
insert 20 insert 10 insert 26 insert 7 insert 15 insert 23 insert 30 insert 3 insert 8 #	20 10 7 3 8 15 26 23 30

Binary Search Tree

- **Exercise** Each node of a binary tree has a field key which is the key of the node. Build a binary tree and check if the tree is a binary search tree (BST), and compute the sum of keys of nodes of the given tree (keys of the nodes are distinct and in the range 1, 2, …, 10^5)
- **Input**
  - Line 1 contains MakeRoot u: make the root of the tree having id = u
  - Each subsequent line contains: AddLeft or AddRight commands with the format
    - AddLeft u v: create a node having id = u, add this node as a left-child of the node with id = v (if not exists)
    - AddRight u v: create a node having id = u, add this node as a right-child of the node with id = v (if not exists)
  - The last line contains * which marks the end of the input
- **Output**
  - Write two integers z and s in which s is the sum of keys of nodes of the tree and z = 1 if the tree is a BST and z = 0, otherwise

Binary Search Tree

- **Example**

Input	Output
MakeRoot 4 AddRight 5 4 AddLeft 3 4 AddRight 8 5 AddLeft 1 3 AddLeft 7 8 AddLeft 6 7 AddRight 2 1 AddRight 10 8 AddLeft 9 10 *	1 55

Profile management

- **Exercise** A profile of a student consists of following information which are strings
  - Name
  - Email
- Write a program running in an interactive mode with following instructions
  - Load <filename>: load data from 1 text file
  - Find <student_name>: return profile of the student given the name
  - Insert <student_name> <email>: insert a new profile into the list
  - Remove <student_name>: remove a profile from the lists
  - Store <filename>: store the list in a text file
  - Quit: terminate the program
- Requirement: use Binary Search Tree for storing profiles

Profile management

```
#include <stdio.h>
#define MAX_L 256
#define MAX 100000

typedef struct Node{
    char name[256];
    char email[256];
    struct Node* leftChild;
    struct Node* rightChild;
}Node;
Node* root;
```

Profile management

```
Node* makeNode(char* name, char* email){
    Node* p = (Node*)malloc(sizeof(Node));
    strcpy(p->name,name); strcpy(p->email,email);
    p->leftChild = NULL; p->rightChild = NULL;
    return p;
}
Node* insert(Node* r, char* name, char* email){
    if(r == NULL) return makeNode(name,email);
    int c = strcmp(r->name,name);
    if(c == 0){
        printf("Student %s exists, do not insert\n",name); return r;
    }else if(c < 0){
        r->rightChild = insert(r->rightChild,name,email); return r;
    }else{
        r->leftChild = insert(r->leftChild,name,email); return r;
    }
}
```

Profile management

```
Node* find(Node* r, char* name){
    if(r == NULL) return NULL;
    int c = strcmp(r->name,name);
    if(c == 0) return r;
    if(c < 0) return find(r->rightChild,name);
    return find(r->leftChild,name);
}
Node* findMin(Node* r){
    if(r == NULL) return NULL;
    Node* lmin = findMin(r->leftChild);
    if(lmin != NULL) return lmin;
    return r;
}
```

Profile management

```
Node* removeStudent(Node* r, char* name){
    if(r == NULL) return NULL;
    int c = strcmp(r->name,name);
    if(c > 0) r->leftChild = removeStudent(r->leftChild,name);
    else if(c < 0) r->rightChild = removeStudent(r->rightChild,name);
    else{
        if(r->leftChild != NULL && r->rightChild != NULL){
            Node* tmp = findMin(r->rightChild);
            strcpy(r->name,tmp->name); strcpy(r->email,tmp->email);
            r->rightChild = removeStudent(r->rightChild,tmp->name);
        }else{
            Node* tmp = r;
            if(r->leftChild == NULL) r = r->rightChild; else r = r->leftChild;
            free(tmp);
        }
    }
    return r;}
```

Profile management

```
void freeTree(Node* r){
    if(r == NULL) return;
    freeTree(r->leftChild);
    freeTree(r->rightChild);
    free(r);
}
void load(char* filename){
    FILE* f = fopen(filename,"r");
    if(f == NULL) printf("Load data -> file not found\n");
    root = NULL;
    while(!feof(f)){
        char name[256], email[256];
        fscanf(f,"%s%s",name, email);
        root = insert(root,name,email);
    }
    fclose(f);
}
```

Profile management

```
void inOrder(Node* r){
    if(r == NULL) return;
    inOrder(r->leftChild);
    printf("%s, %s\n",r->name,r->email);
    inOrder(r->rightChild);
}
void inOrderF(Node* r, FILE* f){
    if(r == NULL) return;
    inOrderF(r->leftChild,f);
    fprintf(f,"%s  %s\n",r->name,r->email);
    inOrderF(r->rightChild,f);
}

void printList(){
        inOrder(root);
        printf("\n");
}
```

Profile management

```
void processStore(){
  char filename[256];
  scanf("%s",filename);
  FILE* f = fopen(filename,"w");
  inOrderF(root,f);
  fclose(f);
}
void processInsert(){
    char name[256], email[256];
    scanf("%s%s",name,email);
    root = insert(root,name,email);
}
void processRemove(){
    char name[256];
    scanf("%s",name);
    root = removeStudent(root,name);
}
```

Profile management

```
void main(){
    while(1){
        printf("Enter command: ");
        char cmd[256];
        scanf("%s",cmd);
        if(strcmp(cmd,"Quit")==0) break;
        else if(strcmp(cmd,"Load")==0) processLoad();
        else if(strcmp(cmd,"Print")==0)  printList();
        else if(strcmp(cmd,"Find")==0) processFind();
        else if(strcmp(cmd,"Insert")==0) processInsert();
        else if(strcmp(cmd,"Remove")==0) processRemove();
        else if(strcmp(cmd,"Store")==0) processStore();
    }
    freeTree(root);
}
```

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

soict.hust.edu.vn/ fb.com/groups/soict