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

C Programming Basic

Searching – part 3

Hashing

- Objectives
  - Apply hashing techniques for solving problems related to storing and searching objects
  - Manipulation with basic hash functions over integers, strings
  - Profile management application

Hash functions

- Hash functions over integers: $h(k) = k \bmod m$
- Hash function over strings
  - A key s is a string: sequence of q characters: $s = s[1 \ldots q]$, the hash function can be defined as:

$$h(s) = (c[1]*256^{q-1} + c[2]*256^{q-2} + \ldots + c[q]*256^{0}) \bmod m$$

  in which $c[i]$ is the ASCII code (integer from 0 to 255) of the character $s[i]$

- **Exercise** Compute the hash code of n given strings
- Input
  - Line 1: n and m ($1 <= n, m <= 100000$)
  - Line i+1 (i = 1,2,..., n): contains a string (the length of each string is less than or equal to 200)
- Output
  - Each line contains the corresponding hash code of n given strings

Hash functions

- Example

Input	Output
4 1000 a ab abc abcd	97 930 179 924

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 hash table combining with binary search tree (as buckets) in a chaining fashion when resolving collision

Profile management

```
#include <stdio.h>
#define MAX_L 256
#define MAX 100000
#define M 100
typedef struct Node{
    char name[256];
    char email[256];
    struct Node* leftChild;
    struct Node* rightChild;
}Node;
Node* root[M];

int h(char* s){// hash function
    int rs = 0;    int n = strlen(s);
    for(int i = 0; i < n; i++)        rs = (rs*255 + s[i])%M;
    return rs;
}
```

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");
    for(int i = 0; i < M; i++) root[i] = NULL;
    while(!feof(f)){
        char name[256], email[256];
        fscanf(f,"%s%s",name, email);
        int idx = h(name);
        root[idx] = insert(root[idx],name,email);// insert to bucket (BST) idx
    }
    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(){
        for(int i = 0; i < M; i++)
            inOrder(root[i]);
        printf("\n");
}
```

Profile management

```
void processStore(){
  char filename[256];
  scanf("%s",filename);
  FILE* f = fopen(filename,"w");
  for(int i = 0; i < M; i++)
    inOrderF(root[i],f);
  fclose(f);
}
void processInsert(){
    char name[256], email[256];
    scanf("%s%s",name,email);
    int idx = h(name);
    root[idx] = insert(root[idx],name,email);
}
```

Profile management

```
void processRemove(){
    char name[256];
    scanf("%s",name);
    int idx = h(name);
    root[idx] = removeStudent(root[idx],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();
    }
    for(int i = 0; i < M; i++)
        freeTree(root[i]);
}
```

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