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

C Programming Basic

Searching – part 1

Content

- Application of binary search on a sequence of numbers
- Management of a list of profiles

Application of binary search on a sequence

- **Exercise** Given a sequence of distinct integers a_1, a_2, …, a_n and an integer Q. Count number M of pairs (i, j) such that $1 \le i < j \le n$ and $a_i + a_j = Q$
  - Implement of a brute force algorithm
  - Implement of an improvement algorithm using binary search
  - Compare the performance of the two algorithms on random sequence of 10^2, 10^3, 10^5 and 10^6 elements

Application of binary search on a sequence

- **Input**
  - Line 1: contains two integers *n* and *Q* (1 <= n,Q <= 10^6)
  - Line 2: contains a_1, a_2, …, a_n
- **Ouput**
  - Write the value of *M*

Input	Output
5 8 4 6 5 3 2	2

Application of binary search on a sequence

```
void input(){
    scanf("%d%d",&n,&Q);
    for(int i = 1; i<= n; i++)
       scanf("%d",&a[i]);
}
void bruteForceSolve(){
    int cnt = 0;
    for(int i = 1; i < n; i++){
        for(int j = i+1; j <= n; j++)
            if(a[i] + a[j] == Q)
                cnt++;
    }
    printf("%d\n",cnt);
}
```

Application of binary search on a sequence

```
void swap(int i, int j){
    int tmp = a[i]; a[i] = a[j]; a[j] = tmp;
}
void heapify(int i, int n){
    int L = 2*i;
    int R = 2*i+1;
    int max = i;
    if(L <= n && a[L] > a[max]) max = L;
    if(R <= n && a[R] > a[max]) max = R;
    if(max != i){
        swap(i,max); heapify(max,n);
    }
}
```

Application of binary search on a sequence

```
void buildHeap(){
    for(int i = n/2; i >= 1; i--) heapify(i,n);
}
void heapSort(){
    buildHeap();
    for(int i = n; i > 1; i--){
        swap(1,i); heapify(1,i-1);
    }
}
```

Application of binary search on a sequence

```
int binarySearch(int L, int R, int Y){
    // return 1 if Y appears in the sequence a[L,...,R]
    if(L > R) return 0;
    if(L == R) if(a[L] == Y) return 1; else return 0;
    int m = (L+R)/2;
    if(a[m] == Y) return 1;
    if(a[m] > Y) return binarySearch(L,m-1,Y);
    return binarySearch(m+1,R,Y);
}
```

Application of binary search on a sequence

```
void binarySearchSolve(){
    heapSort();
    int cnt = 0;
    for(int i = 1; i < n; i++){
        int ok = binarySearch(i+1,n,Q-a[i]);
        cnt += ok;
    }
    printf("%d\n",cnt);
}
int main(){
    input();
    bruteForceSolve();
    binarySearchSolve();
}
```

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: Maintain sorted list and use Binary Search for searching

Profile management

- Data structure

```
#include <stdio.h>
#define MAX_L 256
#define MAX 100000
typedef struct Profile{
    char name[MAX_L];
    char email[MAX_L];

}Profile;
Profile students[MAX];
int n = 0;
```

Profile management

```
void insert(char* name, char* email){
    // maintain increasing order of name
    int i = n-1;
    while(i >= 0){
       int c = strcmp(students[i].name,name);
       if(c == 0){
            printf("Name %s exists, do not insert\n",name); return;
       }else if(c > 0){
            students[i+1] = students[i]; i--;
       }else break;
    }
    i++;
    strcpy(students[i].name,name);
    strcpy(students[i].email,email);
    n++;
}
```

Profile management

```
void removeStudent(int idx){
    for(int i = idx; i < n-1; i++) students[i] = students[i+1];
    n--;
}
void load(char* filename){
    FILE* f = fopen(filename,"r");
    if(f == NULL) printf("Load data -> file not found\n");
    n = 0;
    while(!feof(f)){
        char name[256], email[256];
        fscanf(f,"%s%s",name, email);
        insert(name,email);
    }
    fclose(f);
}
```

Profile management

```
void printList(){
    for(int i = 0; i < n; i++)
        printf("student[%d]: %s, %s\n",i,students[i].name, students[i].email);
}
int binarySearch(int L, int R,char* name){
    if(L > R) return -1;
    if(L == R){
        if(strcmp(students[L].name,name)==0) return L;  else return -1;
    }
    int m = (L+R)/2;
    int c = strcmp(students[m].name,name);
    if(c == 0) return m;
    if(c < 0) return binarySearch(m+1,R,name);
    return binarySearch(L,m-1,name);
}
```

Profile management

```
void processFind(){
    char name[256];
    scanf("%s",name);
    int idx = binarySearch(0,n-1,name);
    if(idx == -1){
        printf("Not found student %s\n",name);
    }else{
        printf("Found student %s, at position %d, email
%s\n",students[idx].name,idx,students[idx].email);
    }
}
void processLoad(){
  char filename[256];
  scanf("%s",filename);
  load(filename);
}
```

Profile management

```
void processStore(){
  char filename[256];
  scanf("%s",filename);
  FILE* f = fopen(filename,"w");
  for(int i = 0; i < n; i++){
        fprintf(f,"%s %s",students[i].name,students[i].email);
        if(i < n-1) fprintf(f,"\n");
  }
  fclose(f);
}
void processInsert(){
    char name[256], email[256];
    scanf("%s%s",name,email);
    insert(name,email);
}
```

Profile management

```
void processRemove(){
    char name[256];
    scanf("%s",name);
    int idx = binarySearch(0,n-1,name);
    if(idx == -1) printf("Not found %s\n",name);
    else{
        removeStudent(idx);
    }
}
```

Profile management

```
int 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();
    }
}
```

Additional exercises

- **Exercise** Given a table NxN in which each cell contains a bit 0 or 1. Given a dictionary which is a set D of words (each word is a binary sequence of length <= 50). A word of the table is a sequence bits in consecutive cells on some row of the table. Compute the number of words of the table that appear in the dictionary D.

Additional exercises

- **Input**
  - Line 1: contains a positive integer N ($1 \le N \le 100$)
  - Line $i + 1$ ($i = 1,\ldots, N$): contains the i^{th} of the table (including N bits 0 or 1)
  - Line N+2: contains a positive integer M ($1 \le M \le 10^5$)
  - Line $N + 2 + i$ ($i = 1,\ldots, M$): contains a word (binary sequence) of the dictionary D.

Input	Output
4 **1 0 0 1** **1 1 1 1** **1 0 1 1** **0 0 0 1** **3** **001** **011** **101**	4

- **Output**
  - Write number of words of the table that appear in D

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