Bài thực hành **"Các cấu trúc dữ liệu cơ bản"**

Thời lượng 4 tiết

I. Mục tiêu:
II. Nội dung ôn tập:
III. Luyện tập

I. Mục tiêu

Các cấu trúc dữ liệu cơ bản và cài đặt các thuật giải tìm kiếm trong.

II. Nội dung ôn tập

- Các cấu trúc dữ liệu cơ bản (ôn tập tại lớp)
- Thuật giải tìm kiếm: Tìm kiếm tuyến tính, tìm kiếm nhị phân

1. Thuật giải tìm kiếm tuyến tính

a. Phát biểu bài toán :

Tìm x có trong dãy a?

- Input : $a_0, a_2, \ldots a_{n-1}$, x

  int a[n], x;
- Output :
  - Nếu có, trả về chỉ số i đầu tiên để a[i] = x ;
  - Nếu không có, trả về -1

b. Mô tả thuật giải:

- Bước 1: Xuất phát từ phần tử đầu tiên của dãy: **i=0**
- Bước 2: So sánh a[i] với giá trị x, có 2 trường hợp:
  - a[i] = x: tìm thấy, dừng thuật giải
  - a[i] ≠ x: sang bước 3
- Bước 3: Xét phần tử kế tiếp trong mảng: **i = i+1**
  - nếu i>n-1: hết mảng, không tìm thấy, dừng thuật giải
  - ngược lại: quay lại bước 2.

b. Cài đặt:

- TH1: Không dùng lính canh

```
int LinearSearch (int a[], int n, int x)
{
        int i = 0;
        while ((i<n) && (a[i])!=x)
                i++;
        if (i==n)
                return -1;  // tìm hết mảng nhưng không có x
        return  i;  // tìm thấy x tại vị trí x
}
```

- TH2: Dùng lính canh

- Đặt thêm phần tử có giá trị x vào cuối mảng (luôn tìm thấy x trong mảng)
- Dựa vào vị trí tìm thấy x để kết luận.

```
int LinearSearch (int a[], int n, int x)
{
        int i = 0;
        a[n] = x; // đặt phần tử lính canh
        while (a[i]!=x)
```

```
        i++;
    if (i==n)
        return -1;  // tìm hết mảng nhưng không có x
    return i;  // tìm thấy x ở vị trí i
}
```

2. Thuật giải tìm kiếm nhị phân

(Chỉ sử dụng cho các dãy đã có thứ tự)

a. Phát biểu bài toán :

Tìm x có trong dãy tăng a?

- Input : $a_0, a_2, \ldots a_{n-1}$, x

  int a[n], x;
- Output :
  - Nếu có, trả về chỉ số i để a[i] = x ;
  - Nếu không có, trả về -1

b. Ý tưởng:

Giả sử dãy đã có thứ tự tăng : $i < j \Rightarrow a_i \leq a_j$

- Nếu $x > a_k$ thì x chỉ có thể xuất hiện trong đoạn $[a_{k+1}, a_{n-1}]$
- Nếu $x < a_k$ thì x chỉ có thể xuất hiện trong đoạn $[a_0, a_{k-1}]$

c. Mô tả thuật giải:

- Bước 1: left=0; right = n-1; // tìm trên tất cả các phần tử
- Bước 2: mid = (left+right)/2; // lấy mốc so sánh

  So sánh a[mid] với giá trị x, có 3 trường hợp:
  - a[mid] = x: tìm thấy, dừng thuật giải
  - a[mid] > x: righ = mid-1 // tìm tiếp trong dãy con $a_{left} .. a_{mid-1}$
  - a[mid] < x: left = mid+1 // tìm tiếp trong dãy con $a_{mid+1} .. a_{right}$
- Bước 3:
  - nếu left≤right: lặp lại bước 2 // còn phần tử chưa xét, tìm tiếp
  - ngược lại: dừng // đã xét hết mọi phần tử

d. Cài đặt:

```
int BinarySearch (int a[], int n, int x)
{
    int left = 0, right = n-1, mid;
    do
    {
        mid = (left+right)/2;
        if (x==a[mid])
            return mid; // tìm thấy x tại vị trí mid
        else
            if (x<a[mid])
                right = mid -1;
        else
            left = mid+1;
    }
    while (left<=right);
    return -1;  // tìm hết dãy mà không có x
}
```

II. Luyện tập:

Bài 1: (Bài toán Đếm) Viết chương trình nhập vào một mảng a gồm tối thiểu 10 số nguyên từ một file. Thực hiện các thao tác sau trên mảng a:

1. Dem : Đếm số lần xuất hiện của x trong mảng a.
2. Dem_Am: Đếm các số âm
3. Dem_Duong: Đếm các số dương.
4. Dem_Nt : Đếm các số nguyên tố.
5. Đếm số lượng các đường chạy.

Đường chạy: Dãy con có thứ tự dài nhất gồm những phần tử kế tiếp.

Bài 2: (Kiểm tra tính đúng sai) Viết chương trình nhập vào một mảng a gồm tối thiểu 10 số nguyên từ một file. Thực hiện kiểm tra các phát biểu sau trên mảng a:

1. a không chứa 0.
2. a có thứ tự tăng.
3. a chứa ít nhất 3 phần tử liên tiếp trùng nhau.
4. a chỉ chứa 2 giá trị.
5. a chỉ chứa các giá trị từ 0 đến n-1.
6. Nếu a có chưa phần tử 0 thì phải chứa phần tử có giá trị 1.
7. Giả thiết a, b cùng có n phần tử . Kiểm tra a, b có phải là hoán vị của nhau.

Bài 3: (Bài toán tính Max) Viết chương trình nhập vào một mảng a gồm tối thiểu 10 số nguyên từ một file. Thực hiện các thao tác sau trên mảng a:

1. Max: Tính max(a_0,..,a_{n-1}).
2. Cs_Max: Tìm chỉ max: Trả về chỉ số đầu tiên đạt max(a_1,..,a_n).
3. Cs_Am_Max: Tìm chỉ số (đầu tiên) của số âm lớn nhất, nếu có. Nếu không, trả về 0.
4. Kc_Max: Khoảng cách lớn nhất giữa số x và các phần tử trong a.
5. Kc_Nt_Max: Trả về chỉ số của số nguyên tố có khoảng cách lớn nhất đến 1 phần tử trong a. (nếu có);