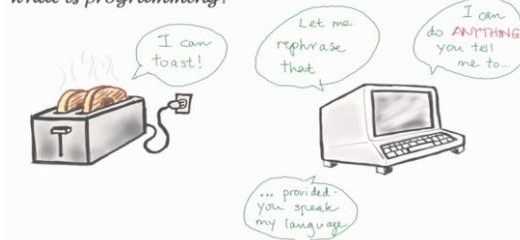


What is programming?



Tuần 16

BÀI TOÁN TÌM KIẾM – SẮP XẾP

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT

Mục đích

Giới thiệu 2 bài toán có ứng dụng rộng rãi là tìm kiếm và sắp xếp

Yêu cầu



- Hiểu được thuật toán tìm kiếm tuần tự và vận dụng được để viết chương trình.
- Hiểu được thuật toán tìm kiếm nhị phân và vận dụng được để viết chương trình.
- Hiểu được thuật toán tìm sắp xếp “nổi bọt” và vận dụng được để viết chương trình

Nội dung



1. Thuật toán tìm kiếm tuần tự
2. Thuật toán tìm kiếm nhị phân
3. Thuật toán sắp xếp “nổi bọt”

TÌM KIẾM TUẦN TỰ



- **Bài toán:**

Cho một mảng a có n phần tử và một giá trị X . Tìm xem trong mảng a có tồn tại một phần tử $a[i]$ nào đó mà $a[i]$ bằng X hay không?

- **Ý tưởng thuật toán:**

Xét các phần tử $a[i]$, tuần tự từ $a[0]$ đến $a[n-1]$, nếu tồn tại một i sao cho $a[i] == X$ thì kết luận tìm thấy, ngược lại không tìm thấy

TÌM KIẾM TUẦN TỰ



- **Tên hàm:** Search

- **Input:**

- Giá trị X cần tìm
- Mảng a các giá trị
- Số phần tử của mảng (n)

- **Output:** 1 nếu tìm thấy, 0 nếu ngược lại

int Search(DataType X, DataType a[], int n)

- **Mã giả:**

- Đặt i vào chỉ số đầu của mảng ($\text{int } i=0$)
- Lúc đầu, ta chưa tìm thấy ($\text{int Found}=0$)
- Trong khi i còn là chỉ số của mảng và chưa tìm thấy
Nếu $a[i] == X$ thì tìm thấy ($\text{Found}=1$); ngược lại tăng i lên 1 đơn vị
- Trả về Found

CHƯƠNG TRÌNH TÌM KIẾM TUẦN TỰ



```
#include <stdio.h>

typedef int DataType;

int Search(DataType X, DataType a[], int n){
    int i =0;
    int Found = 0;
    while (i<n) && (!Found)
        if (a[i] == X)
            Found = 1;
        else i++;
    return Found;
}

int main(){
    int X= 5, a[] = {4, 6, 15, 23, 5, 0, -3, 9};
    if Search(X,a,8) printf("Tim thay X trong mang a\n");
    else printf("Khong tim thay X trong mang a\n");
    return 0;
}
```

TÌM KIẾM NHỊ PHÂN



• Bài toán:

Cho một mảng a có n phần tử **đã có thứ tự** và một giá trị X. Tìm xem trong mảng a có tồn tại một phần tử a[i] nào đó mà a[i] bằng X hay không?

• Tên hàm: B_Search

• Input:

- Giá trị cần tìm
- Mảng a các giá trị đã có thứ tự tăng
- Số phần tử của mảng

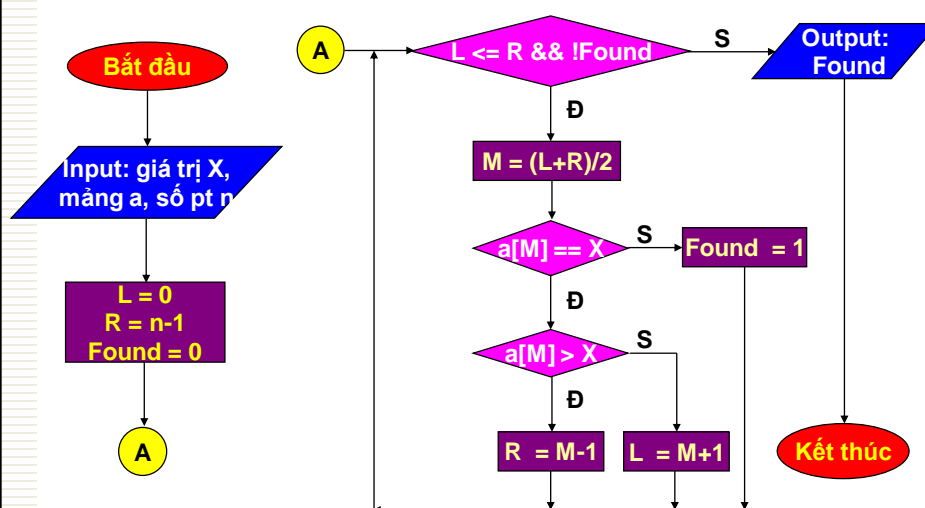
• Output: 1 nếu tìm thấy, 0 nếu ngược lại

int B_Search(DataType X, DataType a[], int n)

Ý TƯỞNG THUẬT TOÁN TÌM KIẾM NP

- Lấy phần tử giữa mảng $a[M]$ so sánh với phần tử cần tìm.
- Nếu $a[M] == X$ thì tìm thấy
- Nếu $a[M] > X$ ta tìm X trong mảng con bên trái
- Nếu $a[M] < X$ ta tìm X trong mảng con bên phải
- Đối với mảng con bên trái và bên phải, cũng thực hiện tương tự
- Sau mỗi bước, phạm vi tìm kiếm giảm đi một nửa so với bước trước đó. Nếu hết phạm vi tìm kiếm thì không tìm thấy.

LƯU ĐỒ THUẬT TOÁN TÌM KIẾM NP



CHƯƠNG TRÌNH TÌM KIẾM NP



```
#include <stdio.h>
typedef int DataType;
int BSearch(DataType X, DataType a[], int n){
    int L = 0, R = n-1;
    int Found = 0;
    while (L<=R) && (!Found) {
        M = (L+R)/2;
        if (a[M] == X) Found = 1;
        else if (a[M] > X) R = M-1;
        else L = M+1;
    }
    return Found;
}
int main(){
    int X= 10, a[] = {4, 7, 10, 23, 25, 30, 35, 49};
    if BSearch(X,a,8) printf("Tim thay X trong mang a\n");
    else printf("Khong tim thay X trong mang a\n");
    return 0;
}
```

TKNP: VIẾT BẰNG KỸ THUẬT ĐỆ QUY



- Biến cục bộ L, R trở thành tham số
- Có thể return trực tiếp mà không cần dùng biến Found

HÀM ĐỆ QUY TKNP




```
int RBSearch(DataType X, DataType a[], int L, int R){
    if (L>R) return 0;
    int M = (L+R)/2;
    if (a[M] == X) return 1;
    if (a[M] > X) return RBSearch(X, a, L, M-1);
    return RBSearch(X, a, M+1, R);
}
```

Lời gọi ban đầu: RBSearch(X, a, 0, n-1)

SO SÁNH TÌM KIẾM TT VÀ TÌM KIẾM NP



- Tìm kiếm tuần tự: thực hiện n lần so sánh các phần tử của mảng a với giá trị cần tìm X.
- Tìm kiếm nhị phân: Chỉ cần $\log_2 n$ lần so sánh.
Thực vậy:
 - Khởi đầu: tìm kiếm trong phạm vi có n phần tử
 - Sau bước 1, phạm vi tìm kiếm còn n/2 phần tử
 - Sau bước 2, phạm vi tìm kiếm còn n/4 phần tử
 -
 - Sau bước i, phạm vi tìm kiếm còn $n/2^i$ phần tử
 - Kết thúc khi $n/2^i = 1$, tức là $i = \log_2 n$
- Nếu mảng chưa có thứ tự thì không thể TKNP




GHI NHỚ

Tìm kiếm nhị phân

NHANH HƠN

Tìm kiếm tuần tự

CT101 - Lập trình căn bản 15 Khoa CNTT&TT



BÀI TOÁN SẮP XẾP

- **Bài toán:** Cho một mảng a có n phần tử, mà giá trị của các phần tử thuộc một kiểu có thứ tự (kiểu số, kiểu ký tự hoặc chuỗi ký tự). Hãy sắp xếp các phần tử của mảng theo thứ tự từ nhỏ đến lớn.
- **Tầm quan trọng của bài toán sắp xếp:**
 - Có rất nhiều ứng dụng cần phải sắp xếp dữ liệu.
 - Ví dụ ta cần sắp xếp danh sách sinh viên theo điểm trung bình với thứ tự từ cao đến thấp.
 - Sắp xếp là một yêu cầu không thể thiếu trong khi thiết kế các phần mềm.
 - Do đó cần nghiên cứu các phương pháp sắp xếp.

CT101 - Lập trình căn bản 16 Khoa CNTT&TT

Ý TƯỞNG SẮP XẾP “NỔI BỌT”



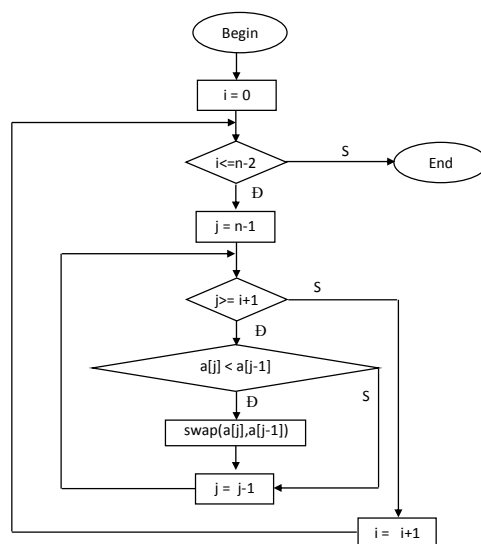
- Chúng ta tưởng tượng rằng mảng là một dãy các bọt bong bóng từ mặt nước đến đáy ao
- Qua quá trình sắp xếp, bong bóng nào “nhẹ” hơn sẽ được “nổi lên” trên.
- Chúng ta duyệt toàn mảng, từ dưới đáy ao lên trên. Nếu hai phần tử ở cạnh nhau mà phần tử “nhẹ hơn” nằm dưới thì phải cho nó “nổi lên”.

THUẬT TOÁN SẮP XẾP “NỔI BỌT”



- **Bước 1:** Xét các phần tử từ $a[n-1]$ đến $a[1]$, với mỗi phần tử $a[j]$, so sánh nó với phần tử $a[j-1]$ đứng ngay trước nó. Nếu $a[j] < a[j-1]$ thì hoán đổi $a[j]$ và $a[j-1]$ cho nhau.
- **Bước 2:** Xét các phần tử từ $a[n-1]$ đến $a[2]$, và làm tương tự như trên.
-
- **Bước i:** Xét các phần tử từ $a[n-1]$ đến $a[i]$, và làm tương tự như trên.
- Sau $n-1$ bước thì kết thúc.

LƯU ĐỒ THUẬT TOÁN SẮP XẾP "NỔI BỌT"



CT101 - Lập trình căn bản

19

Khoa CNTT&TT

CHƯƠNG TRÌNH SẮP XẾP "NỔI BỌT"

```
typedef int DataType;
```

```
void Swap (DataType *x,DataType *y)
{
    DataType Temp;
    Temp = *x;
    *x = *y;
    *y = Temp;
}
```

```
void BubbleSort(DataType a[],int n)
{
    int i,j;
    for(i= 0; i<= n-2; i++)
        for(j=n-1;j>=i+1; j--)
            if (a[j] < a[j-1])
                Swap (&a[j],&a[j-1]);
}
```

```
void in(DataType a[], int n)
{
    int i;
    for(i=0; i<n; i++)
        printf("%5d", a[i]);
}
```

```
int main(){
    DataType a[] = {4, 3, 2, 8,
                    5, 9, 10, -6};
    BubbleSort(a,8);
    printf("Ket qua sap
    xep:\n\n");
    in(a,8);
    return 0;
}
```

CT101 - Lập trình căn bản

20

Khoa CNTT&TT

Bài tập tổng kết



1. Tìm kiếm tuần tự khi DataType không so sánh được bằng phép toán ==, ví dụ DataType là một chuỗi ký tự hay cấu trúc
2. Hàm Locate: Output: Nếu tìm thấy trả về vị trí của phần tử tìm được đầu tiên, ngược lại trả về -1
3. Tìm giá trị trong một ma trận các giá trị.
4. Tìm một mẫu tin R có khóa K trong một mảng các cấu trúc
5. Tìm kiếm tam phân
6. Tìm kiếm nhị phân một mẫu tin R có khóa K trong một mảng các cấu trúc đã sắp thứ tự theo khóa

Bài tập tổng kết



7. Sắp xếp mảng n phần tử mà DataType là chuỗi ký tự
8. Cho mỗi sinh viên được biểu diễn bởi một cấu trúc bao gồm các trường MSSV (int), HoTen (char 25), Ten (char 10) và DiemTBTL (float).
 1. Viết hàm cho phép nhập vào n sinh viên (không phải nhập thông tin cho trường Ten)
 2. Viết hàm sắp xếp mảng các sinh viên theo thứ tự tăng của MSSV
 3. Viết hàm sắp xếp mảng các sinh viên theo thứ tự giảm của DiemTBTL
 4. Viết hàm tách tên của tất cả các SV ghi vào trong trường Ten.
 5. Viết hàm sắp xếp mảng các sinh viên theo thứ tự tăng của Ten

