





Bài thực hành số 1

Bài tập 1:

Viết chương trình minh họa các giải thuật tìm kiếm và sắp xếp trên mảng có kích thước n phần tử nguyên. Chương trình được mô tả với các yêu cầu như sau:

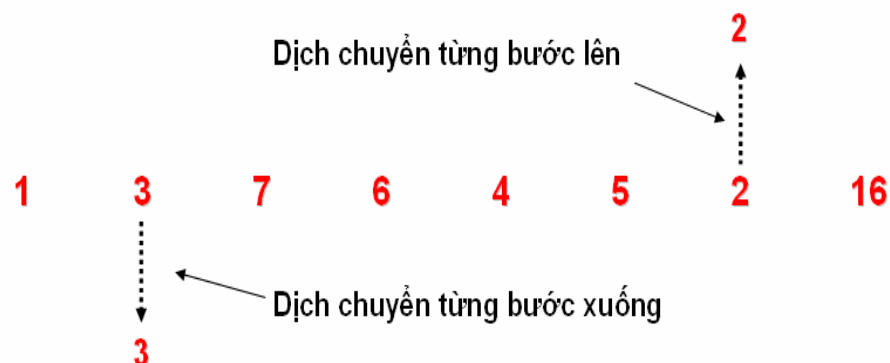
-  Cài đặt hàm tìm kiếm:
 - Tìm kiếm tuần tự (tuyến tính) cho mảng bất kỳ
 - Tìm kiếm nhị phân cho mảng dữ liệu được sắp tăng
-  Cài đặt các hàm sắp xếp (tăng) theo phương pháp:
 - Chọn trực tiếp
 - Chèn trực tiếp
 - Nổi bọt
 - Đổi chỗ trực tiếp
 - Shell Sort
 - Quick Sort
 - Radix Sort

Các hàm sắp xếp phải minh họa trực quan: tại mỗi bước hoán vị $a[i]$ và $a[j]$

Ví dụ: hoán vị $a[1] = 3$ và $a[6] = 2$

1 3 7 6 4 5 2 16

Bước 1: di chuyển 3 xuống dưới k dòng và 2 lên trên k dòng



Bước 2: di chuyển 3 qua vị trí cột của 2 và ngược lại

Dịch chuyển qua trái

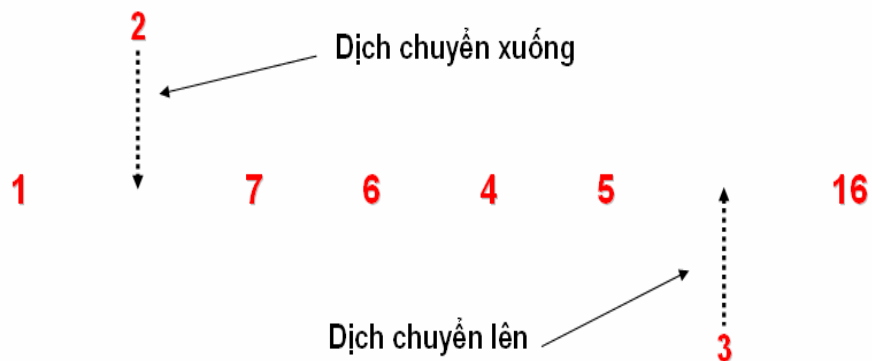
2 ← 2

1 7 6 4 5 16

3 3

Dịch chuyển qua phải

Bước 3: di chuyển 2 xuống và 3 lên đúng vị trí cuối cùng



Trường hợp sắp xếp theo kiểu chèn hơi khác, sinh viên tự tìm hiểu và cài đặt minh hoạ cho thuật toán sắp xếp chèn.

☛ **Lưu ý:** mỗi hàm sẽ có tham số vào là mảng cần xử lý và kích thước của mảng, không dùng biến toàn cục.

🗨 Trong hàm main xây dựng một menu chọn, cho phép nhập vào một số rồi thực hiện chức năng tương ứng, menu có mô tả như sau:

- 1. Khởi tạo mảng dữ liệu a, cho phép nhập vào n, sau đó chương trình phát sinh ngẫu nhiên các phần tử cho mảng a
- 2. Xem phần tử của mảng
- 3. Tìm kiếm tuần tự
- 4. Tìm kiếm nhị phân
- 5. Sắp xếp chọn

- 6. Sắp xếp chèn
- ...
- 11. Sắp xếp theo Radix Sort
- 12. Thoát

VD: khi user nhập 1 thì tạo mảng, nhập 2 xem các phần tử của mảng, còn 11 thực hiện sắp xếp theo Radix Sort. Khi user nhập 12 thì kết thúc chương trình!

● **Lưu ý:** khi chọn chức năng tìm kiếm nhị phân, thì phải kiểm tra xem mảng đã được sắp tăng chưa, SV tự cài đặt chức năng kiểm tra này.

Hướng dẫn

- Phân định nghĩa các hằng số dùng trong chương trình

```
#define MAX      50
#define COL      10
#define ROW      10
#define UER      4
#define WAIT     100
#define SPACING  8
```

- Hàm xuất mảng

```
void ShowArray(int a[], int n)
{
    for(int i=0; i < n; i++)
    {
        gotoxy(i*SPACING+COL, ROW);
        cprintf("%d", a[i]);
    }
}
```

- Hàm hoán vị minh họa di chuyển từng bước

```
void Swap(int a[], int n, int i, int j)
{
    int ai = a[i], aj=a[j];
    int xi = i*SPACING+COL, xj=j*SPACING+COL;
    int d=0;
    // di chuyển ai xuống và aj lên UER đơn vị
    while (d<UER)
    {
        gotoxy(xi, ROW+d); cprintf("  ");
        gotoxy(xj, ROW-d); cprintf("  ");
        d++;
        gotoxy(xi, ROW+d); cprintf("%d",ai);
        gotoxy(xj, ROW-d); cprintf("%d",aj);
        delay(WAIT);
    }
}
```

```
// bat dau dich chuyen
int xii = xi, xjj = xj;
while (xj > xii || xi < xjj)
{
    gotoxy(xi, ROW+d); cprintf("  ");
    gotoxy(xj, ROW-d); cprintf("  ");
    xj--; xi++;
    gotoxy(xi, ROW+d); cprintf("%d",ai);
    gotoxy(xj, ROW-d); cprintf("%d",aj);
    delay(WAIT);
}
// di chuyen ai len va aj xuống VER don vi
while (d>0)
{
    gotoxy(xi, ROW+d); cprintf("  ");
    gotoxy(xj, ROW-d); cprintf("  ");
    d--;
    gotoxy(xi, ROW+d); cprintf("%d",ai);
    gotoxy(xj, ROW-d); cprintf("%d",aj);
    delay(WAIT);
}
// hoan vi a[i] va a[j];
a[i] = aj;
a[j] = ai;
```

- Hàm bubble sort sử dụng Swap trực quan trên

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

- Minh hoạ 1 phần của hàm main()

```
clrscr();
textcolor(CYAN);
_setcursortype(_NOCURS);

ShowArray(a, n);
BubbleSort(a,n);

getch();
```

Phần còn lại sinh viên tự cài đặt!

Mọi thắc mắc email về: nguyenha.giang@yahoo.com

