BUỔI 12, 13 – FULL HOUSE

I. Một vài bài tập

11.9 Số lớn nhất và nhỏ nhất

```c
int main(){
    int n;
    scanf("%d",&n);
    int a[n+1];
    for(int i=0;i<n;i++) scanf("%d",&a[i]);
    int max=a[0],min=a[0],va=0,vi=0;
    for(int i=0;i<n;i++) {
        if(a[i]>=max){
            va=i;
            max=a[i];
        }
        else if(a[i]<=min){
            vi=i;
            min=a[i];
        }
    }
    printf("%d %d\n%d %d",max,va+1,min,vi+1);
}
```

11.1 Tìm số lớn nhất và số lớn thứ 2 trong mảng

```c
int main (){
    int n;
    scanf("%d",&n);
    int a[n];
    for (int i = 0; i < n; ++i){
        scanf("%d",&a[i]);
    }
    int max1 = a[0], max2 = a[1];
    for (int i = 1; i < n; ++i){
        if (a[i] > max1){
            max2 = max1;
            max1 = a[i];
        }
        else if (a[i] > max2) max2 = a[i];
    }
    printf("%d\n%d",max1, max2);
    return 0;
}
```

12.1 Tìm 3 số lớn nhất

```c
int main(){
    int n;
    scanf("%d",&n);
    int a[n+1];
    for(int i=0;i<n;++i) scanf("%d",&a[i]);
    int max1=a[0], max2=a[1], max3=a[2];
    for(int i=0;i<n;++i){
        if(a[i]>max1){
            max3=max2;
            max2=max1;
            max1=a[i];
        }else if(a[i]>max2&&i!=0){
            max3=max2;
            max2=a[i];
        }else if(a[i]>max3&&i!=0&&i!=1){
            max3=a[i];
        }
    }
    printf("%d %d %d",max1, max2,max3);
}
```

12.2 Liệt kê phần tử có ít nhất 2 phần tử lớn hơn

```c
//bai 12.2 c2
#include <stdio.h>

int main (){
    int n;
    scanf("%d",&n);
    int a[n+1];
    for (int i = 0; i < n; ++i){
        scanf("%d",&a[i]);
    }
    for (int i = 0; i < n; ++i){
        int dem=0;
        for(int j=0;j<n;++j){
            if(a[i]<a[j]) dem++;
        }
        if(dem>=2) printf("%d ",a[i]);
    }
    return 0;
}
```

```c
//12.2 c1
#include <stdio.h>

int main (){
    int n;
    scanf("%d",&n);
    int a[n+1];
    for (int i = 0; i < n; ++i){
        scanf("%d",&a[i]);
    }
    int max1 = a[0], max2 = a[1];
    for (int i = 1; i < n; ++i){
        if (a[i] > max1){
            max2 = max1;
            max1 = a[i];
        }
        else if (a[i] > max2) max2 = a[i];
    }
    for(int i=0;i<n;++i){
        if(a[i]<max2) printf("%d ",a[i]);
    }
    return 0;
}
```

11.14 Phần tử có số lần xuất hiện nhiều nhất

```c
//11.14 c1
#include <stdio.h>

int main(){
    int n;
    scanf("%d",&n);
    int a[n+1];
    for(int i=0;i<n;++i){
        scanf("%d",&a[i]);
    }
    int sl1=0,sl2,vt=0;
    for(int i=0;i<n;++i){
        sl2=0;
        for(int j=0;j<n;++j){
            if(a[j]==a[i]) sl2++;
        }
        if(sl1<sl2) sl1=sl2, vt=i;
    }
    printf("%d %d",a[vt],sl1);
}
```

```c
//11.14 c2
#include <stdio.h>
#include <string.h>

int main(){
    int n;
    scanf("%d",&n);
    int a[n+1];
    int b[100001];
    memset(b,0,sizeof(b));
    for(int i=0;i<n;++i){
        scanf("%d",&a[i]);
    }
    int max=a[0],sl=0;
    for(int i=0;i<n;++i){
        b[a[i]]++;
        if(sl<b[a[i]]) {
            sl=b[a[i]];
            max=a[i];
        }
    }
    printf("%d %d",max,sl);
}
```

12.10 Liệt kê phần tử có ít nhất một phần tử liền kề trái dấu nó

```c
//12.10
#include <stdio.h>

int main (){
    int n;
    scanf("%d",&n);
    int a[n+2];
    for(int i=1;i<=n;++i)
        scanf("%d",&a[i]);
    a[0]=0, a[n+1]=0;

    for(int i=1;i<=n;++i){
        if(a[i]*a[i-1]<0||a[i]*a[i+1]<0)
            printf("%d ",a[i]);
    }
    return 0;
}
```

12.19 Đếm phần tử

```c
//12.19
#include <stdio.h>

int main (){
    int n;
    scanf("%d",&n);
    int a[n+1];
    for(int i=0;i<n;++i) scanf("%d",&a[i]);
    int d=1, max=a[0];
    for(int i=1;i<n;++i){
        if(a[i]>=max) {
            max=a[i];
            d++;
        }
    }
    printf("%d",d);
    return 0;
}
```

12.20 Đếm số lần xuất hiện

```c
//12.20
#include <stdio.h>
#include <string.h>

int main (){
    int n;
    scanf("%d",&n);
    int a[n+1];
    int b[100001];
    memset(b,0,sizeof(b));
    for(int i=0;i<n;++i) scanf("%d",&a[i]),b[a[i]]++;
    for(int i=0;i<n;++i){
        if(b[a[i]]!=0) {
            printf("%d xuat hien %d lan\n",a[i],b[a[i]]);
            b[a[i]]=0;
        }
    }
    return 0;
}
```

II. 4 thuật toán sắp xếp

1. Cặp nghịch thế

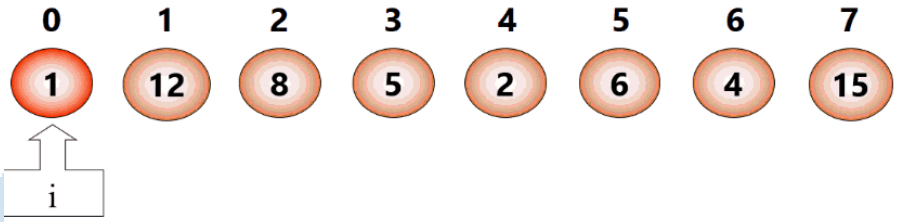
a. **Ví dụ:** Cho danh sách có n phần tử a0, a1, a2,…,an-1. Giả sử muốn sắp xếp danh sách a theo thứ tự tăng dần. Nếu i < j và ai > aj thì ai và aj là cặp nghịch thế.

b. **Ví dụ:** ta có mảng a: 85 4 12 36. A[0] < A[1] thì đây là cặp nghịch thế.

c. Nguyên tắt sắp xếp một danh sách ta cần triệt tiêu hết các cặp nghịch thế đó bằng cách hoán đổi vị trí của chúng.

2. Interchen Sort (sắp xếp đổi chỗ trực tiếp)

a. Ý tưởng:



Nếu a[i] > a[j] thì đổi chỗ a[i], a[j]

b. **Lý thuyết:** Xuất phát từ phần tử đầu danh sách, tìm tất các các cặp nghịch thế chứa phần tử này. Triệt tiêu chúng bằng cách đổi chỗ 2 phần tử trong cặp nghịch thế. Lặp lại xử lý trên với phần tử kế tiếp trong danh sách.

c. **Các bước thực hiện:**

Bước 1: i = 0

Bước 2: j=i+1;

Bước 3:

Khi j < n thì kiểm tra: nếu a[j] < a[i] thì hoán vị a[j] và a[i].

Gán j=j+1; rồi thực hiện lại bước 3.

Bước 4: i=i+1; nếu i < n-1 thì lặp lại bước 2, ngược lại -> Dừng.

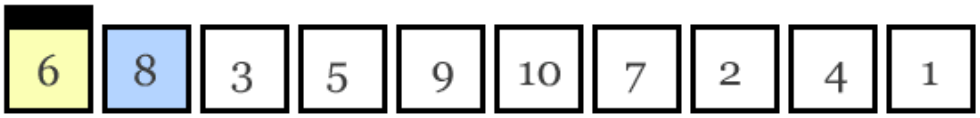
d. **Code:**

```
21 for(int i=0;i<n-1;++i){
22     for(int j=i+1;j<n;++j){
23         if(a[i]>a[j]){
24             int x=a[i];
25             a[i]=a[j];
26             a[j]=x;
27         }
28     }
29 }
```

3. Đánh giá: Độ phức tạp thuật toán là O(n^2)

4. Selection Sort (sắp xếp chọn)

a. Ý tưởng:



Yellow is smallest number found
Blue is current item
Green is sorted list

b. **Lý thuyết:** Chọn phần tử nhỏ nhất trong n phần tử ban đầu, đưa phần tử này về vị trí đúng là đầu tiên của dãy hiện hành. Sau đó không quan tâm đến nó nữa, xem dãy hiện hành chỉ còn n-1 phần tử của dãy ban đầu, bắt đầu từ vị trí thứ 2. Lặp lại quá trình trên cho dãy hiện hành đến khi dãy hiện hành chỉ còn một phần tử. Dãy ban đầu có n phần tử, vậy tóm tắt ý tưởng thuật toán là thực hiện n-1 lượt việc đưa phần tử nhỏ nhất trong dãy hiện hành về vị trí đúng ở đầu dãy.

c. **Các bước thực hiện:**

- Bước 1: i=1.
- Bước 2: Tìm phần tử a[min] nhỏ nhất trong dãy hiện hành từ a[i] đến a[n].
- Bước 3: Hoán vị a[min] và a[i]
- Bước 4: Nếu i<=n-1 thì i=i+1; Lặp lại bước 2.
- Ngược lại: Dừng. n-1 phần tử đã nằm đúng vị trí.

d. **Ví dụ:**

a[] = 54 12 65 14 -8

// Tìm phần tử nhỏ nhất trong a[0...4]

// Đổi chỗ nó với phần tử đầu tiên của a[0...4]

[-8] 12 65 14 54

// Tìm phần tử nhỏ nhất trong a[1...4]

// Đổi chỗ nó với phần tử đầu tiên của a[1...4]

-8 [12] 65 14 54

// Tìm phần tử nhỏ nhất trong a[2...4]

// Đổi chỗ nó với phần tử đầu tiên của a[2...4]

-8 12 [14] 65 54

// Tìm phần tử nhỏ nhất trong a[3...4]

// Đổi chỗ nó với phần tử đầu tiên của a[3...4]

-8 12 14 [54] 65

e. **Code:**

```
3  //selection sort
4  //sap xep chon
5  void selection(int a[], int n){
6      for(int i=0;i<n-1;++i){
7          int min = i;
8          for(int j=i+1;j<n;++j){
9              if(a[j]<a[min]) min =j;
10         }
11         int x = a[i];
12         a[i] = a[min];
13         a[min] = x;
14     }
15 }
```

f. Đánh giá:

Độ phức tạp thuật toán:

- Trường hợp tốt: O(n^2)
- Trung bình: O(n^2)
- Trường hợp xấu: O(n^2)

5. Buble Sort (Thuật toán sắp xếp nổi bọt)

a. Ý tưởng:

b. **Lý thuyết:** Thuật toán sắp xếp nổi bọt thực hiện sắp xếp dãy số bằng cách lặp lại công việc đổi chỗ 2 số liên tiếp nhau nếu chúng đứng sai thứ tự (số sau bé hơn số trước với trường hợp sắp xếp tăng dần) cho đến khi dãy số được sắp xếp.

c. **Ví dụ:**

Giả sử chúng ta cần sắp xếp dãy số **[7 5 -8 1 9]** này tăng dần.

Lần lặp đầu tiên:

(7 5 -8 1 9) –> (5 7 -8 1 9), Ở đây, thuật toán sẽ so sánh hai phần tử đầu tiên, và đổi chỗ cho nhau do 7 > 5.

(5 7 -8 1 9) –> (5 -8 7 1 9), Đổi chỗ do 7 > -8

(5 -8 7 1 9) –> (5 -8 1 7 9), Đổi chỗ do 7 > 1

(5 -8 1 7 9) –> (5 -8 1 7 9), Ở đây, hai phần tử đang xét đã đúng thứ tự (9 > 7), vậy ta không cần đổi chỗ.

Đã đưa ra được phần tử lớn nhất ra cuối cùng: [5 -8 1 7 **9**]

Lần lặp thứ 2:

(5 -8 1 7 **9**) –> (-8 5 1 7 **9**), Đổi chỗ do 5 > -8

(-8 5 1 7 **9**) –> (-8 1 5 7 **9**), Đổi chỗ do 5 > 1

(-8 1 5 7 **9**) –> (-8 1 5 7 **9**)

Đã đưa ra được phần tử lớn thứ 2 ra kế cuối: [-8 1 5 **7 9**]

Bây giờ, dãy số đã được sắp xếp, Nhưng thuật toán của chúng ta không nhận ra điều đó ngay được. Thuật toán sẽ cần thêm một lần lặp nữa để kết luận dãy đã sắp xếp khi và khi khi nó tìm được phần tử nhỏ nhất.

Lần lặp thứ 3:

(-8 1 5 7 9) –> (-8 1 5 7 9)

(-8 1 5 7 9) –> (-8 1 5 7 9)

Đã đưa ra được phần tử lớn thứ 3: [-8 1 5 7 9]

Lần lặp thứ 4:

(-8 1 5 7 9) –> (-8 1 5 7 9)

Đã tìm ra được phần tử nhỏ nhất: [-8 1 5 7 9]

d. Code

```
15  //Bubble sort
16  //Sap xep noi bot
17      for(int i=n-1;i>0;--i){
18          for(int j=0;j<i;++j){
19              if(a[j]>a[j+1]){
20                  int c=a[j];
21                  a[j]=a[j+1];
22                  a[j+1]=c;
23              }
24          }
25      }
```

e. Đánh giá

Độ phức tạp thuật toán:

- Trường hợp tốt: O(n)
- Trung bình: O(n^2)
- Trường hợp xấu: O(n^2)

6. Insertion sort (Thuật toán sắp xếp chèn)

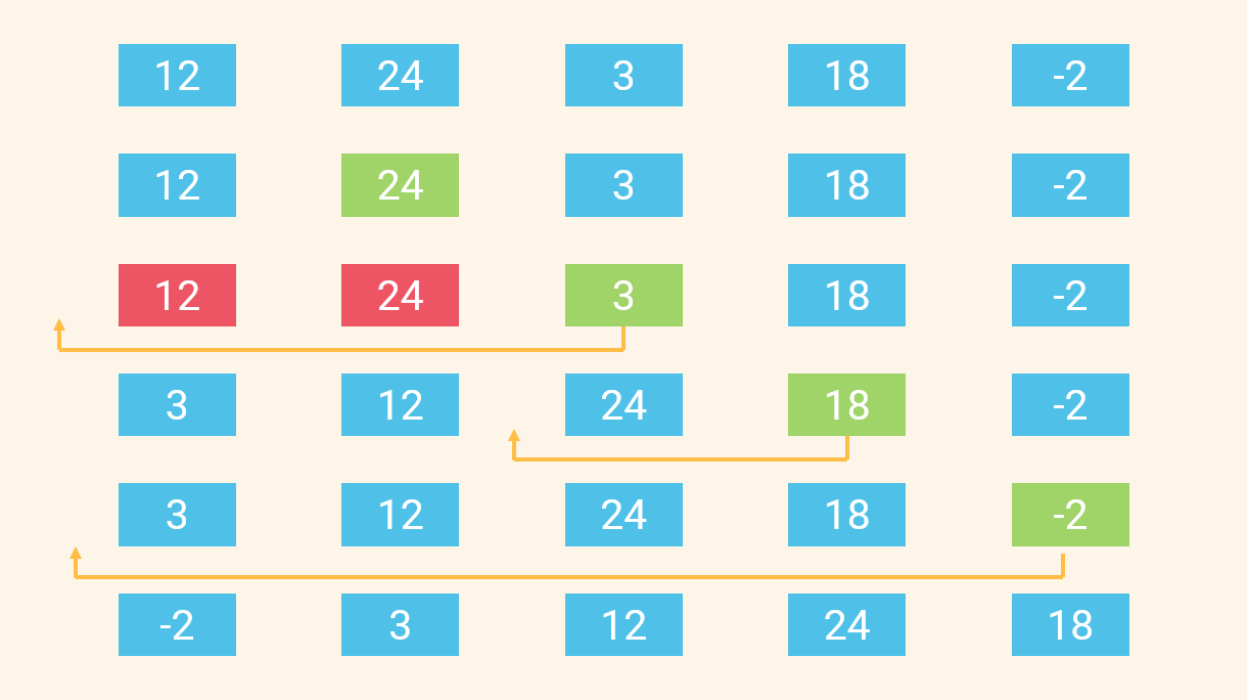
a. Ý tưởng:

b. **Lý thuyết:** Thuật toán sắp xếp chèn thực hiện sắp xếp dãy số theo cách duyệt từng phần tử và chèn từng phần tử đó vào đúng vị trí trong mảng con(dãy số từ đầu đến phần tử phía trước nó) đã sắp xếp sao cho dãy số trong mảng sắp đã xếp đó vẫn đảm bảo tính chất của một dãy số tăng dần.

c. **Các bước thực hiện:**

- Khởi tạo mảng với dãy con đã sắp xếp có k = 1 phần tử(phần tử đầu tiên, phần tử có chỉ số 0)
- Duyệt từng phần tử từ phần tử thứ 2, tại mỗi lần duyệt phần tử ở chỉ số i thì đặt phần tử đó vào một vị trí nào đó trong đoạn từ [0…i] sao cho dãy số từ [0…i] vẫn đảm bảo tính chất dãy số tăng dần. Sau mỗi lần duyệt, số phần tử đã được sắp xếp k trong mảng tăng thêm 1 phần tử.
- Lặp cho tới khi duyệt hết tất cả các phần tử của mảng.

d. Ví dụ:



e. Code

```
15  //Insertion Sort - Sap xep chen
16      for(int i=1;i<n;++i){
17          int j=i-1;
18          int tam=a[i];
19          while(tam<a[j]&&j>=0){
20              a[j+1]=a[j];
21              j--;
22          }
23          a[j+1]=tam;
24      }
```

f. Đánh giá:

Độ phức tạp thuật toán:

- Trường hợp tốt: O(n^2)
- Trung bình: O(n^2)
- Trường hợp xấu: O(n^2)