CHƯƠNG 6
GIẢI THUẬT SẮP XẾP

GV. Ngô Công Thắng
Bộ môn Công nghệ phần mềm
Khoa Công nghệ thông tin
Website: dse.vnua.edu.vn/ncthang
Email: ncthang@vnua.edu.vn

Chương 6: Giải thuật sắp xếp

1. Sắp xếp chọn (Selection Sort)

1.1. Phương pháp

- Giả sử cần sắp xếp tăng dần một dãy khoá a_1, a_2,..., a_n.
- Ý tưởng của thuật toán như sau:
  - Chọn phần tử có khoá nhỏ nhất .
  - Đổi chỗ nó với phần tử a_1.
  - Sau đó lặp lại thao tác trên với n-1 phần tử còn lại, rồi lại lặp lại như trên với n-2 phần tử còn lại,..., cho tới khi chỉ còn 1 phần tử.

1.1. Phương pháp (*tiếp*)

- Ví dụ:

Cho dãy khoá ban đầu là: 6, 10, 1, 8, 9

với n=5.

i=1 1, 10, 6, 8, 9

i=2 1, 6, 10, 8, 9

i=3 1, 6, 8, 10, 9

i=4 1, 6, 8, 9, 10

1.1. Phương pháp (*tiếp*)

```
Procedure   Selection_sort(a,n);
      For i:= 1 to n-1 Do
       Begin
            {Tìm phần tử nhỏ nhất ở vị trí k }
            k:=i;
            For  j:=i+1 To  n  Do
                 If  a[j] < a[k]  then  k:=j
            {Đổi chỗ phần tử nhỏ nhất k cho phần tử i}
            If k ≠ i then a[k]↔a[i];
       End
Return
```


2.2. Đánh giá giải thuật

- Với giải thuật trình bày ở trên thì phép toán tích cực là phép so sánh (a[j]<a[k]).
- Gọi C là số lượng phép so sánh, C được tính như sau: Ở lượt thứ i (i=1, 2,... , n-1), để tìm khoá nhỏ nhất cần n-i phép so sánh. Số lượng phép so sánh này không phụ thuộc vào tình trạng ban đầu của dãy khoá. Do đó ta có:

- Vậy, độ phức tạp tính toán là $O(n^2)$

2. Sắp xếp chèn (Insert Sort)

2.1. Phương pháp

- Phương pháp này được những người chơi bài hay dùng.
- Giả sử cần sắp xếp tăng dần dãy khoá a_1, a_2,..., a_n. Ý tưởng thuật toán như sau:
  - Các phần tử được chia thành dãy đích: a_1,..., a_{i-1} (kết quả) và dãy nguồn a_i,..., a_n.
  - Bắt đầu với i=2, ở mỗi bước phần tử thứ i của dãy nguồn được lấy ra và chèn vào vị trí thích hợp trong dãy đích sao cho dãy đích vẫn tăng dần. Sau đó i tăng lên 1 và lặp lại.

2.1. Phương pháp

- Ví dụ: Cho dãy khoá 6, 10, 1, 7, 4 với n=5 (dãy số có 5 phần tử).

	Dãy đích	Dãy nguồn
	6	10, 1, 7, 4
i=2	6, 10	1, 7, 4
i=3	1, 6, 10	7, 4
i=4	1, 6, 7, 10	4
i=5	1, 4, 6, 7, 10	

Thủ tục chèn

```
Procedure Insert_sort(a,n)
      1) a[0]:=-∞
      2) For i:=2 to n Do
         Begin
               tg:=a[i];  j:=i-1;
               While  tg<a[j] Do
                 Begin
                       a[j+1]:=a[j];  j:=j-1;
                 End;
               a[j+1]:=tg;  {đưa tg vào đúng vi trí}
         End;
Return
```


2.2. Đánh giá thuật toán

- Phép toán tích cực trong thuật toán này là phép so sánh (tg<a[j]). Số phép toán so sánh C được tính như sau:
  - Trường hợp thuận lợi nhất là dãy khoá a_1, a_2,..., a_n đã được sắp, như vậy mỗi lần chỉ cần 1 phép so sánh. Do vậy

$$C_{min} = \sum_{i=2}^{n} 1 = n-1$$

2.2. Đánh giá thuật toán

- Trường hợp xấu nhất nếu dãy khoá sắp theo thứ tự ngược với thứ tự sắp xếp thì ở lượt i cần có: C= (i-1) phép so sánh. Do vậy

$$C_{max} = \sum_{i=2}^{n} (i-1) = \frac{n(n-1)}{2}$$

- Trường hợp trung bình: Giả sử mọi giá trị khoá đều xuất hiện đồng khả năng thì trung bình phép so sánh ở lượt thứ i là $C_i = i/2$, do đó số phép so sánh trung bình của giải thuật này là:

$$C_{tb} = \sum_{i=2}^{n} \frac{i}{2} = \frac{n^2+n-2}{4}$$

- $O(n^2)$

3. Sắp xếp nổi bọt (Bubble Sort)

3.1. Phương pháp

- Giả sử cần sắp xếp tăng dần dãy khoá a_1, a_2,..., a_n. Ý tưởng thuật toán như sau:
  - Đổi chỗ các phần tử liền kề nhau theo thứ tự tăng dần, lần thứ nhất số nhỏ nhất của dãy được đẩy lên vị trí đầu tiên (gọi là phần tử được sắp).
  - Tiếp tục đổi chỗ các phần tử liền kề của dãy chưa sắp, lần thứ 2 ta được số nhỏ nhất của dãy chưa sắp được đưa lên đầu dãy chưa sắp.
  - Cứ tiếp tục làm tương tự như trên cho đến khi dãy chỉ còn 1 phần tử.

3.1. Phương pháp (*tiếp*)

- Ví dụ: Cho dãy khoá ban đầu là: 6, 3, 7, 10, 1, 8 với n=6.

	6, 3, 7, 10, 1, 8
i=1	1, 6, 3, 7, 10, 8
i=2	1, 3, 6, 7, 8, 10
i=3	1, 3, 6, 7, 8, 10
i=4	1, 3, 6, 7, 8, 10
i=5	1, 3, 6, 7, 8, 10

Thủ tục sắp xếp nổi bọt

```
Procedure   Bubble_sort(a,n)
    For  i:= 1  to  n-1  Do
     For  j:= n downto i+1  Do
          If  a[j]<a[j-1]  then
               a[j] <-> a[j-1];
Return
```


3.2. Đánh giá thuật toán

- Giải thuật này tương tự như giải thuật sắp xếp bằng cách chọn trực tiếp (mục 1), do đó có:

$$C_{\max} = C_{\min} = C_{tb} = \sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2}$$

- Nhận xét: Với 3 phương pháp sắp xếp trên, nếu n vừa và nhỏ thì phương pháp chèn trực tiếp (insert sort) tỏ ra tốt hơn, nếu với n lớn thì cả 3 phương pháp đều có cấp $O(n^2)$, đây là một chi phí thời gian khá cao.

4. Sắp xếp nhanh (Quick Sort)

4.1. Phương pháp

- Sắp xếp nhanh (quick sort) còn được sắp xếp phân đoạn (partition sort).
- Ý tưởng thuật toán:
  - Chọn ngẫu nhiên một phần tử x.
  - Duyệt từ bên trái mảng cho tới khi có một phần tử a_i>=x
  - Sau đó duyệt từ bên phải mảng cho tới khi có một phần tử aj=<x
  - Đổi chỗ a_i và a_j
  - Tiếp tục duyệt và đổi chỗ cho tới khi 2 phía gặp nhau.

4.1. Phương pháp (*tiếp*)

- Kết quả mảng được chia thành 2 phần: bên trái là các phần tử < x, bên phải là các phần tử > x.

Thủ tục sắp xếp nhanh

```
Procedure  Q_sort(L,R);
    1) If  L>=R then return;
    2) i:=L; j:=R ; k:=(L+R) div 2;
    3) x:=a[k];
    4) Repeat
            While a[i] <x  Do  i:=i+1;
            While a[j] >x  Do  j:=j-1;
            If  i<j  then  a[i] ↔ a[j]
       Until  i=j
    5) Call  Q_sort(L,j-1);  { Thực hiện trên nửa <x }
    6) Call  Q_sort(j+1,R);  { Thực hiện trên nửa >x }
Return
```


4.2. Đánh giá

- Người ta đã chứng minh được thời gian trung bình thực hiện giải thuật là:

$$T_{tb} = O(n\log_2 n)$$

- Như vậy, với n khá lớn Quick sort có hiệu lực hơn 3 thuật giải trên.

5. Sắp xếp vun đống (Heap Sort)

5.1. Phương pháp

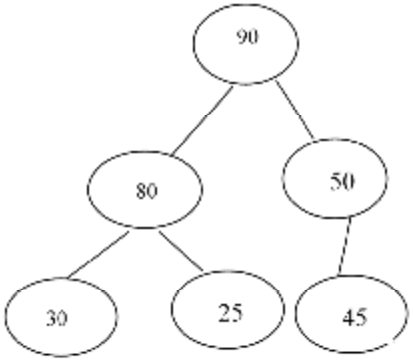
- Một cây nhị phân có chiều cao h được gọi là đống khi:
  - Là cây nhị phân hoàn chỉnh mà các nút lá ở mức h-1 phải nằm phía bên trái.
  - Khoá ở nút cha bao giờ cũng lớn hơn khoá ở nút con.

5. Sắp xếp vun đống (Heap Sort)

5.1. Phương pháp

- Thuật toán sắp xếp vun đống chia làm 2 giai đoạn.
- Giai đoạn 1: Tạo đống.

Ví dụ 1: Cây sau đây là một đống.



Khoá ở nút gốc của đống chính là khoá lớn nhất (khoá trội) so với các khoá trên cây.

- Giai đoạn 2:

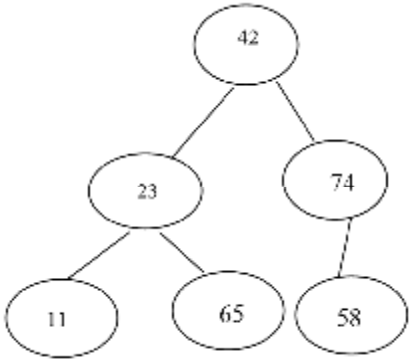
+ Đưa khoá trội về vị trí của nó bằng cách đổi chỗ cho khoá ở vị trí đó.

+ Vun lại đống với cây gồm các khoá còn lại (sau khi đã loại khoá trội)

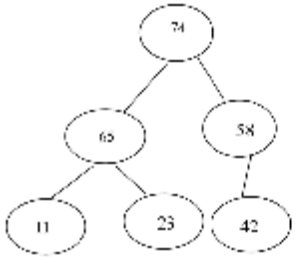
Quá trình trên lại được lặp lại.

Ví dụ: Có dãy khoá 42, 23, 74, 11, 65, 58

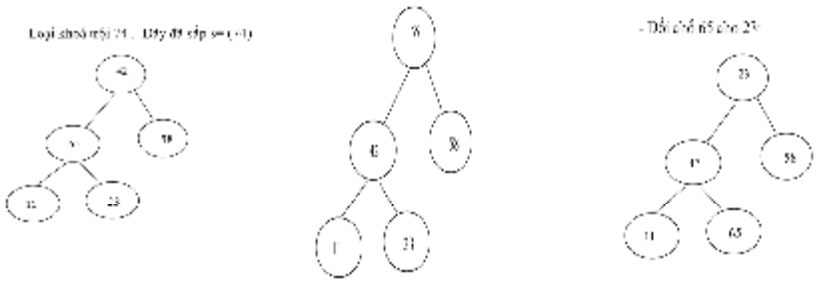
- Ta có cây hoàn chỉnh biểu diễn diễn dãy khoá:



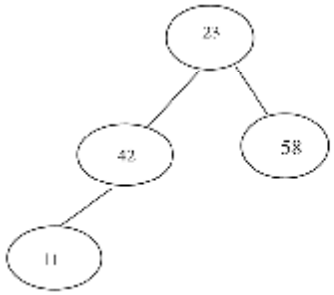
- Giai đoạn đầu: Vun đống ban đầu



- Giai đoạn 2: Đổi chỗ 74 cho 42



- Loại khoá trội 65 . Dãy đã sắp s= (65, 74)



- Lặp lại các bước tương tự cho các cây còn lại.

Cuối cùng ta thu được dãy đã sắp là s=(11, 23, 42, 58, 65, 74)

* Giải thuật vun đống:

- Một lá coi như cây con là một đống.

- Thuật toán tiến hành từ đáy lên: Chuyển đổi thành đống cho một cây con mà cây con trái và cây con phải của gốc đã là một đống.

Cây nhị phân hoàn chỉnh có n nút thì với chỉ số [n/2] trở lên có thể là nút cha: [n/2], [n/2]-1, . . . , 1.

a) Thủ tục vun đống:

Chỉnh lý cây nhị phân con hoàn chỉnh gốc i trên cây nhị phân có n nút để trở thành "đống" với điều kiện cây con trái và cây con phải có gốc là 2i và 2i+1 đã là đống.

```
Procedure  ADJUST(i,n)
1. { Khởi đầu }
Key:=K[i]; j:=2*i;
2. { Chọn con ứng với khoá lớn nhất trong 2 con của i }
While  j<=n  Do
  Begin
     If  (j<n) and (K[j]<K[j+1])  then  j:=j+1;
```


a) Thủ tục vun đống:

```
3. { So sánh khoá cha với khoá lớn nhất }
     If  Key > K[j]  then  Begin
           K[j/2]:=Key;
           Return;
       End;
     K[j/2]:=K[j]; j:=2*j;
  End; {Kết thúc while}
4. { Đưa Key vào vị trí của nó }
K[j/2]:=Key;
5. Return;
```

b) Thủ tục sắp xếp vun đống:

```
Procedure  Heap_Sort(K,n)
1. { Tạo đống ban đầu }
For  i:=[n/2]  Downto  1  Do
     Call  ADJUST(i,n)
2. { Sắp xếp }
For  i:= n-1  Downto  1 Do
 Begin
     tg:=K[1]; K[1]:=K[i+1];
     K[i+1]:=tg;
     Call  ADJUST(1,i);
 End;
3.  Return
```

5.2. Đánh giá

Thời gian thực hiện trung bình của giải thuật này là $O(n\log_2 n)$.

6. Sắp xếp kiểu hoà nhập (MERGE SORT)

6.1. Phép hoà nhập 2 đường

Thực hiện hợp nhất các bản ghi của 2 bảng đã được sắp xếp thành 1 bảng được sắp.

a) Phương pháp:

So sánh 2 khoá nhỏ nhất (hoặc lớn nhất của 2 bảng) để đưa vào miền sắp xếp.

Quá trình cứ tiếp tục cho tới khi 2 bảng đã cạn.

b) Giải thuật:

Bảng 1: $(x_b, ..., x_m)$

Bảng 2: $(x_{m+1}, ..., x_n)$

Bảng sắp: $(z_b, ..., z_n)$

Ví dụ: Bảng 1: (3, 5, 10, 16)

Bảng 2: (1, 4, 15)

Bảng sắp: (1, 3, 4, 5, 10, 15, 16)

* Thủ tục như sau:

```
Procedure  MERGE(X,b,m,n,Z);
1. i:=k:=b; j:=m+1;
2. While  (i<=m) and (j<=n)  Do
     Begin  If  x[i]<=x[j]  then
            Begin  Z[k]:=x[i];
                   i:=i+1;
            End
            Else  Begin  z[k]:=x[j];
                         j:=j+1;
                  End;
            k:=k+1;
End;
```


* Thủ tục như sau:

3. { Một trong 2 bảng con đã cạn }

If i>m then (zk, ..., zn):= (xj, ..., xn)

Else (zk, ..., zn):= (xi, ..., xm)

4. Return

6.2. Sắp xếp kiểu hòa nhập trực tiếp (Straight two way merge)

* Bảng con đã được sắp gọi là một mạch (run).
* Mỗi bản ghi coi như 1 mạch có độ dài (kích thước) là 1. Nếu hoà nhập 2 bảng như vậy ta được 1 mạch mới có độ dài =2. Hoà nhập 2 mạch có độ dài là 2 ta được một mạch có độ dài là 4, ...
* Thủ tục MPASS thực hiện một bước của sắp xếp hoà nhập. Nó hòa nhập từng cặp mạch kề cận nhau, có độ dài L, từ bảng X sang bảng Y, n là số lượng khoá (bản ghi) trong X.


```
Procedure MPASS(X,Y,n,L)
1. i:=1;
2. Hoà nhập cặp mạch có độ dài L }
 While  i<= n-(2L-1)  Do
    Begin  Call  MERGE(X,i,i+L-1,i+2L-1, Y)
           i:=i+2L;
    End;
3. { Hoà nhập cặp mạch còn lại cuối cùng với tổng độ dài <2L}
  If  i+L-1 <n  then  Call  MERGE(X,i,i+L-1,n,Y)
  Else  (y_i, ..., y_n):= (x_i, ..., x_n);
4. Return
```


Thủ tục MPASS trên sẽ được gọi tới trong thủ tục sắp xếp kiểu hoà nhập trình bày ở sau đây.

* Thủ tục sắp xếp kiểu hoà nhập trực tiếp:

Thủ tục này lưu trữ các mạch mới khi hoà nhập dùng biến phụ là Y(n). trong đó X và Y được dùng luân phiên. L là kích thước của mạch, sau mỗi lượt L được tăng gấp đôi.

```
Procedure  STRAIGHT_MSORT(X,n)
1. L:=1;
2. While  L<n  Do
      Begin
            Call  MPASS(X,Y,n,L);  L:=L+L;
            Call  MPASS(Y,X,n,L);  L:=L+L;
      End;
3. Return
```

Ví dụ: X= 9, 1, 7, 6, 4, 3, 8, 7

Bước 1:	1, 9, 6, 7, 3, 4, 7, 8	đưa vào Y	L=1
Bước 2:	1, 6, 7, 9, 3, 4, 7, 8	đưa vào X	L=2
Bước 3:	1, 3, 4, 6, 7, 7, 8, 9	đưa vào Y	L=4

6.3. Đánh giá

Thời gian thực hiện trung bình của giải thuật là:

$Ttb = O(n\log_2 n)$

* Nhận xét chung:

- Với n nhỏ có thể dùng các phương pháp: chọn trực tiếp, chèn trực tiếp, đổi chỗ trực tiếp.
- Với n lớn: Nếu dãy khoá không sắp dùng Quick sort, nếu dãy khoá có sắp dùng Heap sort.