Chương 6: Giải thuật sắp xếp

1. Sắp xếp chọn (Selection Sort)
2. Sắp xếp chèn (Insert Sort)
3. Sắp xếp nổi bọt (Bubble Sort)
4. Sắp xếp nhanh (Quick Sort)
5. Sắp xếp vun đống (Heap Sort)
6. Sắp xếp trộn (Merge Sort)

1

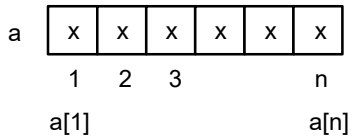
Khi tìm hiểu giải thuật cần tìm hiểu

1) Ý tưởng và ví dụ mình họa ý tưởng
2) Giả mã
3) O

2

Bài toán sắp xếp

- Cho dãy khóa là các số nguyên có n phần tử lưu trữ trong mảng một chiều. Sắp xếp dãy khóa tăng dần từ trái sang phải.



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

Xác định O

i	Số lần so sánh <
1	n – 1
2	n – 2
n-1	1

Tổng = 1 + 2 + … n-1 < 1 + 2 + … n =
(n*(n+1))/2 = ½*(n^2 + n)

Coi O(½*(n^2 + n))
Áp dụng quy tắc bỏ hằng số: O(n^2 + n)
Áp dụng quy tắc cộng: 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 insertSort(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, chèn vào sau j}
       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 sủi bọt (Bubble Sort)

3.1. Phương pháp

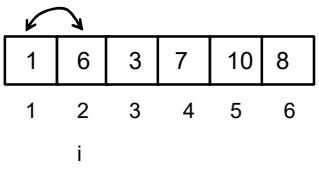
- Giả sử cần sắp xếp tăng dần dãy khoá $a_1, a_2,..., a_n$. Ý tưởng thuật toán như sau:
  - So sánh các **cặp khóa liền kề** gối nhau từ **phải qua trái**, nếu khóa đứng sau nhỏ hơn khóa đứng trước thì đổi chỗ khóa đứng sau với khóa đứng trước. Kết quả lần thứ nhất, khóa nhỏ nhất của dãy được đẩy lên vị trí 1 (gọi là khá được sắp).
  - Tiếp tục so sánh và đổi chỗ các cặp khóa liền kề của dãy chưa sắp, lần thứ 2 ta được khóa nhỏ nhất của dãy chưa sắp được đưa về vị trí 2.
  - Cứ tiếp tục làm tương tự như trên cho đến khi dãy chưa sắp 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

Minh họa giải thuật sắp xếp sủi bọt



Thủ tục sắp xếp sủi bọt

```
Procedure   bubbleSort(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)

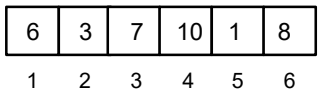
4.1. Phương pháp

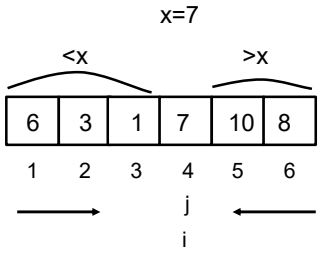
- Sắp xếp nhanh (quick sort) còn được sắp xếp phân đoạn (partition sort).
- Ý tưởng thuật toán:
  - Chọn ngẫu nhiên một phần tử x làm tiêu chí phân đoạn.
  - Duyệt từ trái sang phải cho tới khi gặp phần tử a_i>=x
  - Sau đó duyệt từ phải sang trái cho tới khi gặp phần tử aj<=x
  - Đổi chỗ a_i và a_j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

4.1. Phương pháp (*tiếp*)

- Kết quả dãy khóa được chia thành 2 đoạn: bên trái là các phần tử < x, bên phải là các phần tử > x.
- Áp dụng cách tương tự với đoạn bên trái và đoạn bên phải cho tới khi đoạn con chỉ còn 1 phần tử thì dừng.

Minh họa giải thuật sắp xếp nhanh





Thủ tục sắp xếp nhanh

```
Procedure  quickSort(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  quickSort(L,j-1);  { Thực hiện trên nửa <x }
    6) Call  quickSort(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:

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

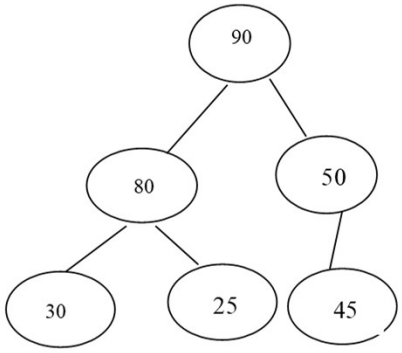
- Như vậy, với n khá lớn Quick sort có hiệu lực hơn 3 thuật giải trên.

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 tương đối hoàn chỉnh mà các nút lá ở mức h phải nằm phía bên trái.
  - Khoá ở nút cha bao giờ cũng lớn hơn khoá ở nút con.

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.

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 ban đầu
- Từ dãy khóa ban đầu ánh xạ sang cây nhị phân
- Vun cây nhị phân từ dưới lên trên, từ cây con gốc [n/2] về cây gốc 1 để tạo đống ban đầu.
- Giai đoạn 2: Sắp xếp
- Đổi chỗ nút gốc 1 cho nút n, loại bỏ nút n, hiệu chỉnh lại (vun đống lại) cây gốc 1 với n-1 nút còn lại.
- Cứ tiếp tục như vậy cho tới khi cây chỉ còn 1 nút.

25

- Giai đoạn 2:

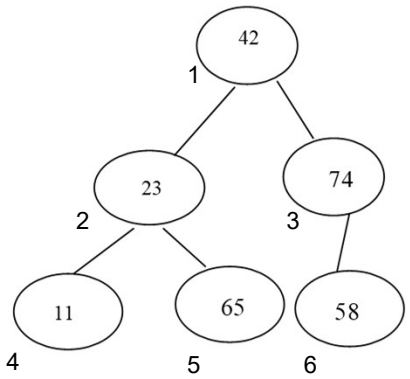
+ Đưa khoá trội về vị trí của nó bằng cách đổi chỗ cho khoá ở vị trí đó.

+ 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

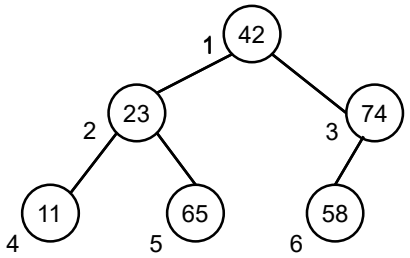
- Ta có cây hoàn chỉnh biểu diễn diễn dãy khoá:



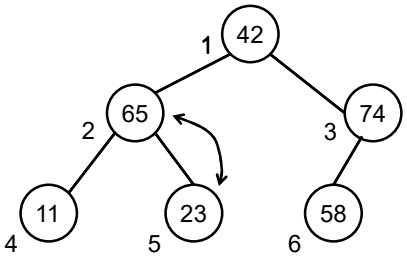
26

Vun đống ban đầu

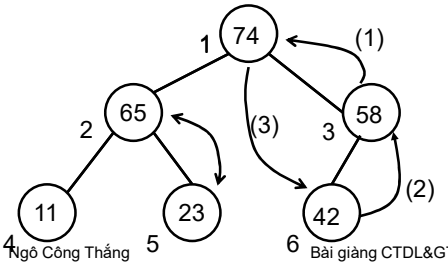
2) Vun đống cây gốc 3



3) Vun đống cây gốc 2



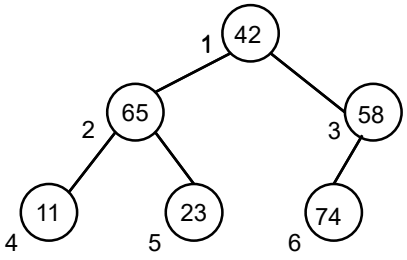
4) Vun đống cây gốc 1



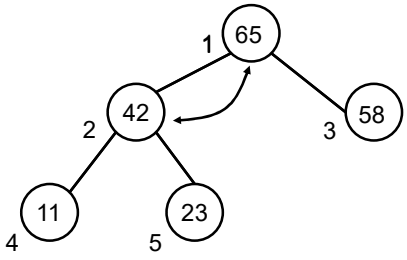
27

Sắp xếp

1) Đổi chỗ nút gốc 1 cho nút cuối cùng 6

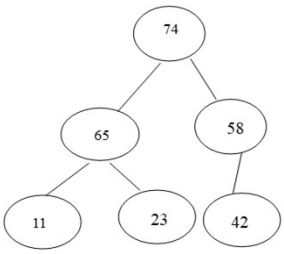


2) Loại bỏ nút 6, vun đống lại cây gốc 1

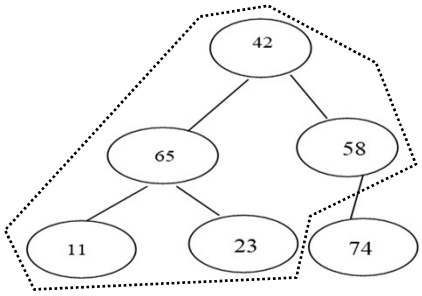


28

- Giai đoạn đầu: Vun đống ban đầu

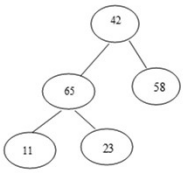


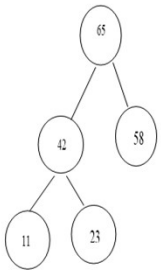
- Giai đoạn 2: Đổi chỗ 74 cho 42



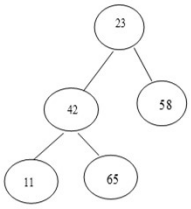
29

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

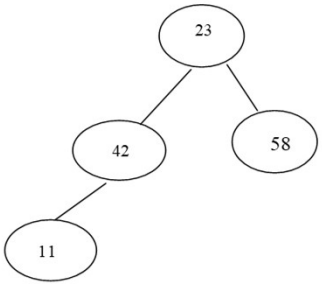




- Đổi chỗ 65 cho 23:



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



30

- 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.

31

a) Thủ tục vun đống:

Hiệu chỉnh 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:=a[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 (a[j]<a[j+1])  then  j:=j+1;
```


32

a) Thủ tục vun đống:

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


b) Thủ tục sắp xếp vun đống:

```
Procedure  heapSort(a,n)
    1. { Tạo đống ban đầu }
    For  i:=n div 2  Downto  1  Do
            Call  adjust(i,n)
    2. { Sắp xếp }
    For  i:= n-1  Downto  1 Do
    Begin
            tg:=a[1]; a[1]:=a[i+1]; a[i+1]:=tg;
            Call  adjust(1,i);
    End;
Return
```

5.2. Đánh giá

$O(n\log_2{}^n)$.

6. Sắp xếp trộn (hoà nhập) (MERGE SORT)

6.1. Trộn 2 dãy đã sắp xếp

Trộn 2 dãy khóa đã sắp xếp tăng dần thành 1 dãy khóa sắp xếp tăng dần.

a) Ý tưởng:

So sánh 2 khoá nhỏ nhất của 2 dãy, khóa nào nhỏ hơn thì đưa sang dãy đích trước.

Quá trình cứ tiếp tục cho tới khi 1 trong 2 dãy đã hết. Dãy còn lại đưa nốt sang dãy đích.

Ví dụ: Dãy 1: (3, 5, 10, 16)

Dãy 2: (1, 4, 15)

Dãy đích: (1, 3, 4, 5, 10, 15, 16)

b) Giải thuật:

Dãy 1: (X_b, ..., X_m)

Dãy 2: (X_{m+1}, ..., X_n)

Dãy đích: (Z_b, ..., Z_n)

*** 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. {Đưa nốt dãy còn lại sang dãy đích}

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

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

Return

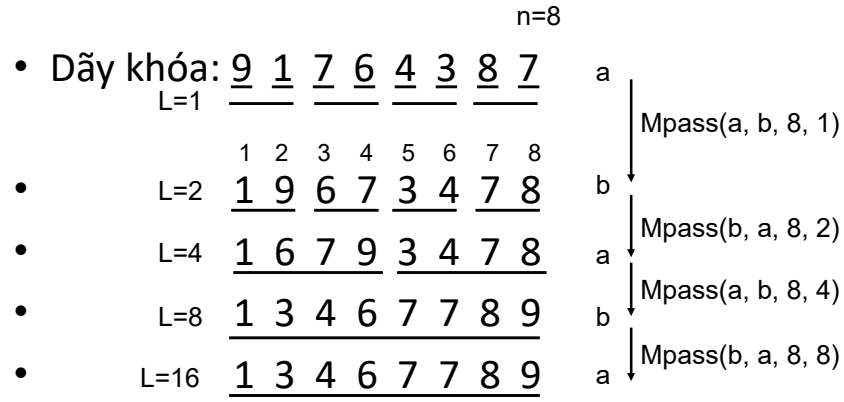
37

Ý tưởng của giải thuật sắp xếp trộn

- Đầu tiên, trộn các cặp dãy con liên tiếp có chiều dài bằng 1 thành các dãy con có chiều dài bằng 2.
- Tiếp theo, trộn các cặp dãy con liên tiếp có chiều dài bằng 2 thành các dãy con có chiều dài bằng 4…
- Cứ tiếp tiếp như vậy cho tới khi dãy con có chiều dài bằng n (số lượng khóa của dãy cần sắp xếp).

38

Ví dụ



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 trộn. Nó trộn từng cặp dãy con liền kề nhau, có độ dài L, từ mảng X sang mảng Y, n là số lượng khoá trong X.

```
Procedure  MPass(X,Y,n,L)
    1. i:=1;
    2. {Khi còn đủ hai dãy con liền kề có độ dài L }
     While  i + 2L-1 <= n Do
          Begin  Call  merge(X,i,i+L-1,i+2L-1, Y)
               i:=i+2L;
          End;
    3. {Từ vị trí i về cuối không còn đủ 2 dãy con
chiều dài L}
    If  i+L-1 <n  then  Call  merge(X,i,i+L-1,n,Y)
    Else  (y_i, ..., y_n):= (x_i, ..., x_n);
Return
```


41

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  mergeSort(a,n)
     1) L:=1;
     2) While  L<n  Do
          Begin
               Call  MPass(a,b,n,L);  L:=L+L;
               Call  MPass(b,a,n,L);  L:=L+L;
          End;
Return
```

Ví dụ: a= 9, 1, 7, 6, 4, 3, 8, 7

Bước 1: 1, 9, 6, 7, 3, 4, 7, 8 đưa vào b L=1

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

Bước 3: 1, 3, 4, 6, 7, 7, 8, 9 đưa vào b L=4

42

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.