**8.11. THUẬT TOÁN SẮP XẾP TRỘN (MERGESORT)**

**8.11.1. Phép trộn 2 đường**

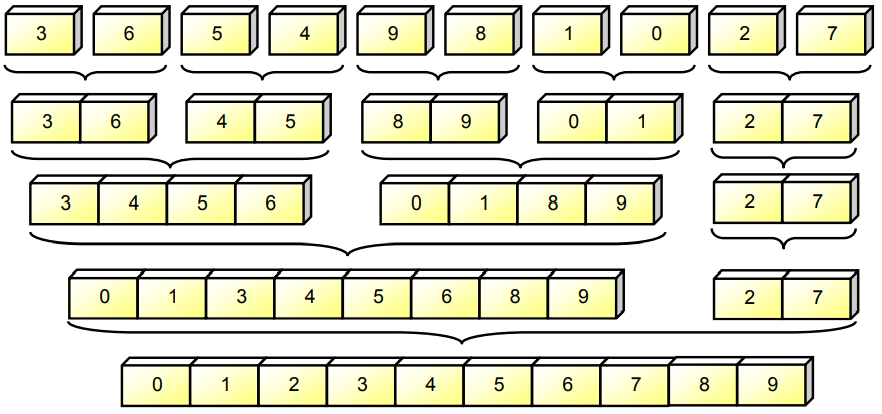
Phép trộn 2 đường là phép hợp nhất hai dãy khoá **đã sắp xếp** để ghép lại thành một dãy khoá có kích thước bằng tổng kích thước của hai dãy khoá ban đầu và dãy khoá tạo thành cũng có thứ tự sắp xếp. Nguyên tắc thực hiện của nó khá đơn giản: so sánh hai khoá đứng đầu hai dãy, chọn ra khoá nhỏ nhất và đưa nó vào miền sắp xếp (một dãy khoá phụ có kích thước bằng tổng kích thước hai dãy khoá ban đầu) ở vị trí thích hợp. Sau đó, khoá này bị loại ra khỏi dãy khoá chứa nó. Quá trình tiếp tục cho tới khi một trong hai dãy khoá đã cạn, khi đó chỉ cần chuyển toàn bộ dãy khoá còn lại ra miền sắp xếp là xong.

Ví dụ: Với hai dãy khoá: (1, 3, 10, 11) và (2, 4, 9)

|  |  |  |  |
| --- | --- | --- | --- |
| Dãy 1 | Dãy 2 | Khoá nhỏ nhất trong 2 dãy | Miền sắp xếp |
| (1, 3, 10, 11) | (2, 4, 9) | 1 | (1) |
| (3, 10, 11) | (2, 4, 9) | 2 | (1, 2) |
| (3, 10, 11) | (4, 9) | 3 | (1, 2, 3) |
| (10, 11) | (4, 9) | 4 | (1, 2, 3, 4) |
| (10, 11) | (9) | 9 | (1, 2, 3, 4, 9) |
| (10, 11) | **** | Dãy 2 là ****, đưa nốt dãy 1 vào miền sắp xếp | (1, 2, 3, 4, 9, 10, 11) |

**8.11.2. Sắp xếp bằng trộn 2 đường trực tiếp**

Ta có thể coi mỗi khoá trong dãy khoá k1, k2, …, kn là một mạch với độ dài 1, dĩ nhiên các mạch độ dài 1 có thể coi là đã được sắp. Nếu trộn hai mạch liên tiếp lại thành một mạch có độ dài 2, ta lại được dãy gồm các mạch đã được sắp. Cứ tiếp tục như vậy, số mạch trong dãy sẽ giảm dần sau mỗi lần trộn



**Thuật toán sắp xếp trộn**

Để tiến hành thuật toán sắp xếp trộn hai đường trực tiếp, ta viết các thủ tục:

Thủ tục Merge(var x, y: TArray; a, b, c: Integer); thủ tục này trộn mạch xa, xa+1, …, xb với mạch xb+1, xb+2 …, xc để được mạch ya, ya+1, …, yc.

Thủ tục MergeByLength(var x, y: TArray; len: Integer); thủ tục này trộn lần lượt các cặp mạch theo thứ tự:

Trộn mạch x1…xlen và xlen+1…x2len thành mạch y1…y2len.

Trộn mạch x2len+1…x3len và x3len+1 …x4len thành mạch y2len+1…y4len.

…

Lưu ý rằng đến cuối cùng ta có thể gặp hai trường hợp: Hoặc còn lại hai mạch mà mạch thứ hai có độ dài < len. Hoặc chỉ còn lại một mạch. Trường hợp thứ nhất ta phải quản lý chính xác các chỉ số để thực hiện phép trộn, còn trường hợp thứ hai thì không được quên thao tác đưa thẳng mạch duy nhất còn lại sang dãy y.

Cuối cùng là thủ tục MergeSort, thủ tục này cần một dãy khoá phụ t1, t2, …, tn. Trước hết ta gọi MergeByLength(k, t, 1) để trộn hai phần tử liên tiếp của k thành một mạch trong t, sau đó lại gọi MergeByLength(t, k, 2) để trộn hai mạch liên tiếp trong t thành một mạch trong k, rồi lại gọi MergeByLength(k, t, 4) để trộn hai mạch liên tiếp trong k thành một mạch trong t …Như vậy k và t được sử dụng với vai trò luân phiên: một dãy chứa các mạch và một dãy dùng để trộn các cặp mạch liên tiếp để được mạch lớn hơn.

procedure MergeSort;

var

t: TArray; {Dãy khoá phụ}

len: Integer;

Flag: Boolean; {Flag = True: trộn các mạch trong k vào t; Flag = False: trộn các mạch trong t vào k}

procedure Merge(var X, Y: TArray; a, b, c: Integer);{Trộn Xa…Xb và Xb+1…Xc}

var

i, j, p: Integer;

begin

{Chỉ số p chạy trong miền sắp xếp, i chạy theo mạch thứ nhất, j chạy theo mạch thứ hai}

p := a;

i := a;

j := b + 1;

while (i **** b) and (j **** c) then {Chừng nào cả hai mạch đều chưa xét hết}

begin

if Xi **** Xj then {So sánh hai phần tử nhỏ nhất trong hai mạch mà chưa bị đưa vào miền sắp xếp}

begin

Yp := Xi;

i := i + 1; {Đưa xi vào miền sắp xếp và cho i chạy}

end

else

begin

Yp := Xj;

j := j + 1; {Đưa xj vào miền sắp xếp và cho j chạy}

end;

p := p + 1;

end;

if i **** b then (Yp, Yp+1, …, Yc) := (Xi, Xi+1, …, Xb) {Mạch 2 hết trước, Đưa phần cuối của mạch 1 vào miến sắp xếp}

else (Yp, Yp+1, …, Yc) := (Xj, Xj+1, …, Xc); {Mạch 1 hết trước, Đưa phần cuối của mạch 2 vào miến sắp xếp}

end;

procedure MergeByLength(var X, Y: TArray; len: Integer);

begin

a := 1;

b := len;

c := 2 \* len;

while c **** n do {Trộn hai mạch xa…xb và xb+1…xc đều có độ dài len}

begin

Merge(X, Y, a, b, c);

a := a + 2 \* len;

b := b + 2 \* len;

c := c + 2 \* len; {Dịch các chỉ số a, b, c về sau 2.len vị trí}

end;

if b < n then Merge(X, Y, a, b, n) {Còn lại hai mạch mà mạch thứ hai có độ dài ngắn hơn len}

else

if a **** n then (Ya, Ya+1, …, Yn) := (Xa, Xa+1, …, Xn); {Còn lại một mạch thì đưa thẳng mạch đó sang miền y}

end;

begin {Thuật toán sắp xếp trộn}

Flag := True;

len := 1;

while len < n do

begin

if Flag then MergeByLength(k, t, len) else MergeByLength(t, k, len); len := len \* 2;

Flag := not Flag; {Đảo cờ để luân phiên vai trò của k và t}

end;

if not Flag then k := t; {Nếu kết quả cuối cùng đang nằm trong t thì sao chép kết quả vào k}

end;

Về độ phức tạp của thuật toán, ta thấy rằng trong thủ tục Merge, phép toán tích cực là thao tác đưa một khoá vào miền sắp xếp. Mỗi lần gọi thủ tục MergeByLength, tất cả các phần tử trong dãy khoá được chuyển hoàn toàn sang miền sắp xếp, nên độ phức tạp của thủ tục MergeByLength là O(n). Thủ tục MergeSort có vòng lặp thực hiện không quá log2n + 1 lời gọi MergeByLength bởi biến len sẽ được tăng theo cấp số nhân công bội 2. Từ đó suy ra **độ phức tạp của MergeSort là O(nlog2n)** bất chấp trạng thái dữ liệu vào.

Cùng là những thuật toán sắp xếp tổng quát với độ phức tạp trung bình như nhau, nhưng không giống như QuickSort hay HeapSort, MergeSort có tính **ổn định**. Nhược điểm của MergeSort là nó phải dùng thêm một vùng nhớ để chứa dãy khoá phụ có kích thước bằng dãy khoá ban đầu.

Người ta còn có thể lợi dụng được trạng thái dữ liệu vào để khiến MergeSort chạy nhanh hơn: ngay từ đầu, ta không coi mỗi phần tử của dãy khoá là một mạch mà coi những đoạn đã được sắp trong dãy khoá là một mạch. Bởi một dãy khoá bất kỳ có thể coi là gồm các mạch đã sắp xếp nằm liên tiếp nhau. Khi đó người ta gọi phương pháp này là phương pháp **trộn hai đường tự nhiên**.

Tổng quát hơn nữa, thay vì phép trộn hai mạch, người ta có thể sử dụng phép trộn k mạch, khi đó ta được thuật toán sắp xếp trộn k đường.