Báo cáo đồ án

Cấu trúc dữ liệu và giải thuật

1.Selection Sort :

Ý tưởng : selection sort được gọi là thuật toán chọn vì nó sẽ cho thuật toán chạy trong mảng rồi chọn ra số nhỏ nhất trong mảng swap nó đến vị trí đầu tiên sau đó bắt đầu chạy từ vị trí thứ hai để tìm ra phần tử nhỏ nhất trong mảng và thực hiện như trên chạy như thế đến khi chỉ còn 1 phần tử thì kết thúc.

Thuật toán:

* Đặt vị trí i=0;
* Tìm phần tử nhỏ nhất trong mảng từ i->n-1
* Đổi chỗ cho phần tử nhỏ nhất và a[i]
* Nếu i<n-1 thì lặp lại bước trên. Ngược lại thì dừng

Đánh giá thuật toán :

Độ phức tạp của thuật toán : O(n2)

Trường hợp tốt nhất: số lần so sánh n(n-1)/2

Trường hợp xấu nhất: số lần so sánh 3n(n-1)/2

Ưu điểm:

* Thuật toán đơn giản , dễ hiểu , dễ thực hiện.
* Số lần sử dụng hàn swap ít (không quá O(n))

Nhược điểm:

* Nếu số lượng phần tử lớn thuật toán sẽ chạy vô cùng chậm
* Không nhận biết được mảng đã sắp xếp

2.Insertion Sort :

Ý tưởng: insertion sort còn được gọi là sắp xếp chèn. Xét danh sách gồm n phần tử chạy từ 1->n .Với n=1 phần tử đã được sắp xếp . Giả sử trong danh sách n-1 phần tử đã được sắp xếp và phần tử thứ n sẽ tìm ra vị trí thích hợp trong dãy

1->n-1. Vị trí mà tại đó nó sẽ đứng trước phần tử lớn hơn và sau phần tử nhỏ hơn hoặc bằng.

Thuật toán:

Bước 1: cho i chạy từ 1 đến n

Bước 2: đặt key=a[i] và j=i-1;

Bước 3: khi j>0 và a[j]>key

Bước 4: gán a[j+1]=a[j] và j=j-1;

Bước 5: quay lại bước 3 nếu điều kiện đúng , nếu không xuống bước 6

Bước 6: a[j+1]=key;

Bước 7: tiếp tục vòng lặp cho đến hết.

Đánh giá thuật toán:

Độ phức tạp thuật toán :O(n2)

Trường hợp tốt nhất : n-1 phép so sánh , 0 swap.

Trường hợp trung bình : n2/4 phép so sánh, n2/4 swap.

Trường hợp xấu nhất: n2/2 phép so sánh, n2/2 swap.

Ưu điểm :

* Thuật toán đơn giản .

Nhược điểm:

* Thích hợp với mảng đã sắp xếp 1 phần
* Không thích hợp để chạy những mảng lớn

3.BinaryInsertion Sort :

Ý tưởng: binaryinsertion sort còn gọi là thuật toán chèn nhị phân đây là thuật toán cải tiến của thuật toán sắp xếp. Như thuật toán chèn thuật toán chèn nhị phân cũng tìm vị trí để chèn trong phần đã sắp xếp nhưng bằng phương pháp tìm kiếm nhị phân.

Thuật toán:

Bước 1:gán key vào phần tử đầu tiên

Bước 2: tìm vị trí thích hợp để chèn key

Bước 3: sau khi tìm được vị trí dời mảng về sau

Bước 4: chèn key vào vị trí đó

Bước 5: dừng khi duyệt hết mảng

Đánh giá thuật toán:

Độ phức tạp thuật toán : O(n2)

Giảm số lượng thuật toán so sánh rất nhiều so với insertion sort

Ưu điểm:

* Số thuật toán so sánh phải sử dụng khá ít ( O(log n) trong trường hợp xấu nhất)

Nhược điểm:

* Khá phức tạp
* Sử dụng hai hàm : 1. Hàm sắp xếp 2. Hàm tìm kiếm nhị phân

4.Interchange Sort :

Ý tưởng :xuất phát từ đầu dãy, tìm tất cả nghịch thế chứa phần tử này triệt tiêu chúng bằng cách đổi chỗ phần tử này với phần tử tương ứng trong cặp nghịch thế. Lặp lại xử lý trên với các phần tử tiếp theo trong dãy.

Thuật toán:

Bước 1: i=0;

Bước 2:j=i+1;

Bước 3: Trong khi j<n thực hiện :

Nếu a[j]>a[i] thì đổi chỗ a[i],a[j]

J=i+1;

Bước 4: i=i+1;

Nếu (i<n-1): lặp lại bước 2

Ngược lại: dừng

Đánh giá thuật toán : O(n2)

Trường hợp tốt nhất : số lần so sánh n(n-1)/2 , số lần swap bằng 0.

Trường hợp xấu nhất: số lần so sánh n(n-1)/2 , số lần swap n(n-1)/2

Ưu điểm :

* Dễ thực hiện dễ hiểu ,thuật toán đơn giản.

Nhược điểm :

* Không nên sử dụng với số lượng phân tử lớn vì chạy sẽ rất lâu

5. shaker sort :

Ý tưởng: shaker sort là cải tiến của buble sort. Sau khi đưa phần tử nhỏ nhất về đầu dãy, thuật toán sẽ giúp bạn đưa phần tử lớn nhất về cuối dãy. Do đưa các phần tử về đúng vị trí ở cả hai phía nên shaker sort sẽ có thời gian chạy thuật toán ít hơn .

Thuật toán:

Bước 1: right=n-1, left =0;

Bước 2: nếu left < right

Bước 3: chạy từ left ->right thực hiện hoán đổi hai vị trí kế cho nhau nếu thỏa điều kiện

Bước 6: chạy từ right ->left thực hiện hoán đổi hai vị trí kế cho nhau nếu thỏa điều kiện

Bước 7: lặp lại cho đến đến khi hết dãy

Đánh giá thuật toán:

Trường hợp tốt nhất : độ phức tạp thuật toán : O(n)

Trường hợp trung bình : độ phức tạp thuật toán : O(n2)

Trường hợp xấu nhất: độ phức tạp thuật toán : O(n2)

Ưu điểm :

* Thuật toán nhận diện được mảng đã sắp xếp
* Thời gian thực hiện cải thiện hơn so với buble sort.
* Hoạt động tốt trong mảng gần như đã có thứ tự

Nhược điểm:

* Với mảng ngẫu nhiên với thứ tự đảo lộn thì thời gian sắp xếp so với buble sort là chênh lệch không nhiều.

6.shell sort :

Ý tưởng:shell sort là sự cả tiến của insertion sort. Giải thuật này tránh trường hợp phải trao đổi giữa hai phần tử xa nhau( phần tử nhỏ hơn bên phải xa so với phần tử lớn hơn bên trái) một khoảng n. sự xa nhau giữa hai phần tử gọi là interval- vị trí khoảng cách giữa 2 phần tử đang xét. Interval tính theo công thức: h=h\*3+1.

Giải thuật :

Bước 1: khởi tạo giá trị h

Bước 2: chia danh sách thành các danh sách con theo h

Bước 3: sử dụng insertion sort để sắp xếp các danh sách con

Bước 4: lặp lại liên tục đến khi danh sách hoàn thành.

Đánh giá thuật toán:

Độ phức tạp thuật toán: O(n2)

Ưu điểm :

* Là thuật toán hiệu quả nhất trong nhóm các thuật toán sắp xếp có độ phức tạp O(n2)
* Sẽ hiệu quả nếu dữ liệu đầu vào hầu như đã được sắp xếp

Nhược điểm:

* Hoạt động kém hiệu quả vì nó di chuyển các giá trị phần tử mỗi lần chỉ một vị trí

7.heap sort :

Ý tưởng: heap sort còn được gọi là giải thuật vun đống, nó có thể được coi bản cải tiến của selection sort khi chia phần tử làm 2 mảng con, 1 mảng đã sắp xếp mảng còn lại chưa sắp xếp. trong mảng chưa sắp xếp, phần tử lớn nhất sẽ được tách ra và đưa vào mảng sắp xếp.

Thuật toán:

Heap sort chia làm hai giai đoạn.

Giai đoạn 1: từ dãy dữ liệu input, sắp xếp chúng thành một heap. Heap này có thể là min-heap hoặc max-heap, ví dụ xài max-heap:

* Nút cha sẽ luôn lớn hơn con, gốc heap sẽ là phần tử lớn nhất.
* Heap được tạo thành phải là một cây nhị phân đầy đủ , tức ngoại trừ các nút lá, ở cùng một cấp các nhánh không được thiếu.

Giai đoạn 2: giai đoạn này lặp đi lặp lại cho đến khi mảng dữ liệu được hoàn tất:

* Đưa phần tử lớn nhất của heap được tạo vào mảng kết quả, mảng này chứa các phần tử đã được sắp xếp
* Sắp xếp lại heap sau khi loại bỏ nút gốc để tìm giá trị lớn tiếp theo
* Thực hiện thao thác trên gạch đầu dòng 1 giai đoạn 2 cho đến khi heap đều được đưa vào mảng.
* Mảng kết quả sẽ chứa các phần tử sắp xếp giảm dần

Đánh giá thuật toán:

Độ phức tạp thuật toán: O(nlogn)

Ưu điểm :

* Cải tiến so với selection sort dung cấu trúc dữ liệu heap thay vì tìm kiếm tuyến tính.
* Không khó thực hiện
* Giải thuật đảm bảo trong trường hợp xấu nhất
* Không cần dùng cấu trúc dữ liệu phụ trợ trong quá trình thực thi

Nhược điểm:

* Tốc độ chạy không chậm hơn quicksort .

8.merge sort :

Ý tưởng: chia mảng lớn thành các mảng nhỏ hơn bằng cách tiếp tục chia đôi như vậy tới khi mảng con chỉ còn 1 phần tử. sau đó ta so sánh hai mảng con có cùng mảng cơ sở khi so sánh chúng sẽ vừa sắp xếp vừa ghép 2 mảng con thành mảng cơ sở . chúng ta tiến hành như thế cho đến khi xếp thành một mảng hoàn chỉnh.

Đánh giá thuật toán:

Độ phức tạp thuật toán: O(nlogn)

Ưu điểm:

* Sắp xếp nhanh hơn so với các thuật toán cơ bản (insertion sort , selection sort , interchange sort ), nhanh hơn quick sort trong một số trường hợp.

Nhược điểm:

* Thuật toán khó cài đặt
* Không nhận dạng được mảng đã sắp xếp.

9.quick sort :

Ý tưởng:là thuật toán chia để trị . quick sort chia một mảng lớn thành hai mảng con nhỏ hơn: mảng có phần tử nhỏ và mảng có phần tử lớn. sau đó quick sort có thể sort các mảng con bằng phương pháp đệ quy.

Thuật toán:

Bước 1: chọn một phần tử để so sánh, gọi là phần tử key, từ trong mảng đầu tiên

Bước 2: sau đó phân vùng và sort mảng con của sau phân vùng làm sao cho các phần tử lớn hơn phần tử key nằm sau và các phần tử bé hơn phần tử key nằm trước. đây được gọi là quá trình phân vùng.

Bước 3: cuối cùng là đệ quy sử dụng các bước trên cho các mảng với phần tử bé hơn và phân tách với các phần tử lớn hơn sau khi phân vùng.

Đánh giá thuật toán:

Trường hợp trung bình: độ phức tạp thuật toán: O(nlogn)

Trường hợp xấu nhất: độ phức tạp thuật toán: O(n2)

Ưu điểm:

* Tốc độ thuật toán khá nhanh

Nhược điểm:

* Tốc độ thuật toán dựa trên cách chọn phần tử so sánh key

10. radix sort :

Ý tưởng : Giả sử mỗi phần tử ai trong dãy a0, a1, …, an-1 là một số nguyên có tối đa m chữ số. Phân loại các phần tử này lần lượt theo các chữ số hàng đơn vị, hàng chục, hàng trăm, …,

Thuật toán:

Bước 1 : k cho biết chữ số dùng để phân loại hiện hành  
k = 0; k = 0: hàng đơn vị; k = 1: hàng chục; …  
Bước 2 : Tạo các lô chứa các loại phần tử khác nhau  
Khởi tạo 10 lô B0, B1, …, B9 rỗng;  
Bước 3 : For i = 1 .. n do  
Đặt ai vào lô Bt với t: chữ số thứ k của ai;  
Bước 4 : Nối B0, B1, …, B9 lại (theo đúng trình tự) thành a.  
Bước 5 : k = k+1;Nếu k < m thì trở lại bước 2. Ngược lại: Dừng

Đánh giá thuật toán:

Độ phức tạp thuật toán :O(n)

Ưu điểm:

* Thích hợp để sắp xếp dãy có rất nhiều phần tử

Nhược điểm:

* Thuật toán khá khó hiểu và phức tạp.

Tình trạng dữ liệu sorted:

Tình trạng dữ liệu reversed :

Tình trạng dữ liệu random:

Tình trạng dữ liệu nearlysorted:

Nhận xét: các thuật toán đều có những điểm mạnh và điểm yếu cụ thể tùy vào mục đích sử dụng mà sử dụng các thuật toán một cách hiệu quả nhất . ví dụ như muốn tìm kiếm những thuật toán dễ hiểu dễ cài đặt với số lượng số không lớn lắm có thể sử dụng slection sort và interchange sort . nếu muốn tìm thuật toán có thể nhận biết mảng đã sắp xếp thì là shaker sort . hay như binaryinsertion sort có cấu trúc gần như insertion sort nhưng la tìm kiếm nhị phân nên giảm được rất nhiều thuật toán so với insertion sort .hay shell sort là thuật toán chạy hiệu quả nhất trong các thuật toán có độ phức tạp O(n2) , cũng vì thế nên khá phức tạp và nó sẽ đạt hiệu quả hơn trong mảng gần như đã được sắp xếp. heap sort , merge sort , quick sort ,radix sort đều là những thuật toán có độ phức tạp thuật toán thấp chạy khá nhanh . nhưng đều có nhược điểm nhất định như heap sort chạy chậm hơn quick sort nhưng giải thuật đảm bảo trường hợp xấu nhất. merge sort chạy khá nhanh nhưng không nhân diện được mảng đã sắp xếp. quick sort tốc độ thuật toán lại pahri phụ thuộc vafo quá trình chọn key.radix sort thích hợp để sắp xếp nhiều phần tử nhưng thuật toán lại khó hiểu, khó cài đặt.

Nhìn vào đồ thị ở trên ta luôn nhận thấy đối với số phân tử càng lớn thì sự chênh lệch của interchange sort và selection sort càng nhiều hơn hẳn so với các sort còn lại. sự chênh lệch giữa các sort còn lại khá ngắn khó nhận ra