STUDY WITH ME CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

THUẬT TOÁN SẮP XẾP

1. Quick Sort

1.1. Khái niệm

Thuật toán Quick Sort là một thuật toán sắp xếp, còn được gọi là sắp xếp kiểu phân chia (Part Sort). Là một thuật toán hiệu quả dựa trên việc phân chia mảng dữ liệu thành các nhóm phần tử nhỏ hơn.

Có lẽ đây là thuật toán được nghiên cứu và sử dụng rộng rãi nhất trong các thuật toán sắp xếp.Thuật toán Quick sort cũng là thuật toán đệ quy. Tuy nhiên ngược lại với Mergesort gọi đệ quy rồi mới xử lý, Quick sort xử lý xong mới gọi đệ quy.

1.2. Thuật toán

Ý tưởng của thuật toán này dựa trên phương pháp chia để trị, nghĩa là chia dãy cần sắp xếp thành 2 phần, sau đó thực hiện việc sắp xếp cho mỗi phần độc lập nhau.Để làm việc này thì ta cần phải làm các bước sau:

Bước 1:

Chọn ngẫu nhiên một phần tử nào đó của dãy làm phần tử khóa (pivot). Kĩ thuật chọn phần tử khóa rất quan trọng vì nếu không may bạn có thể bị rơi vào vòng lặp vô hạn đối với các trường hợp đặc biệt. Tốt nhất là chọn phần tử ở vị trí trung tâm của dãy. Chọn phần tử đứng đầu hoặc đứng cuối làm phần tử khóa. Chọn phần tử đứng giữa danh sách làm phần tử khóa.Chọn phần tử trung gian trong 3 phần tử đứng đầu, đứng giữa và đứng cuối làm phần tử khóa. Chọn phần tử ngẫu nhiên làm phần tử khóa.

Bước 2:

Xếp các phần tử nhỏ hơn phần tử chốt ở phía trước phần tử khóa.

Bước 3:

Xếp các phần tử lớn hơn phần tử chốt ở phía sau phần tử khóa. Để có được sự phân loại này thì ở 2 bước trên, các phần tử sẽ được so sánh với khóa và hoán đổi vị trí cho nhau hoặc cho khóa nếu nó lớn hơn khóa mà lại nằm trước khóa, hoặc nhỏ hơn mà lại nằm sau khóa.Áp dụng kĩ thuật như trên cho mỗi đoạn đó và tiếp tục làm vậy cho đến khi mỗi đoạn chỉ còn 2 phần tử. Khi đó toàn bộ dãy đã được sắp xếp.

Quick sort là một thuật toán dễ cài đặt, hiệu quả trong hầu hết các trường hợp và tiêu tốn ít tài nguyên hơn so với các thuật toán khác. Độ phức tạp trung bình của giải thuật là O(NlogN).

Cài đặt thuật toán:

```
Bắt đầu hàm partitionFunc(left, right, pivot)
   leftPointer = left -1
   rightPointer = right

   while True thực hiện
      while A[++leftPointer] < pivot thực hiện
         //không làm điều gì
      kết thúc while

      while rightPointer > 0 && A[--rightPointer] > pivot thực hiện
         //không làm điều gì
      kết thúc while

      if leftPointer >= rightPointer
         break
      else
         Tráo đổi leftPointer,rightPointer
      kết thúc if

   kết thúc while

   Tráo đổi leftPointer,right
   return leftPointer

Kết thúc hàm
```

2. Interchange Sort

2.1.Khái niệm

Interchange sort hay còn gọi là thuật toán sắp xếp đổi chỗ trực tiếp. Ý tưởng của thuật toán này là bắt cặp tất cả các phần tử trong dãy cần sắp xếp và đổi chỗ hai phần tử trong cặp nếu chúng không thoả mãn điều kiện về thứ tự.

2.2.Ý tưởng và các bước thực hiện

Ý tưởng:

Như đã biết, để sắp xếp một dãy số, ta có thể xét các nghịch thế có trong dãy và triệt tiêu dần chúng đi. Ý tưởng chính của giải thuật là 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 kế tiếp theo trong dãy.

Các bước thực hiện:

- Bước 1:

  i=0;/ Bắt đầu từ đầu dãy

- Bước 2:

  j = i+1; // Tìm tất cả các phần tử a[j]<a[i], với j>i

- Bước 3:

  Trong khi j < N thực hiện

  Nếu a[j]<a[i] //xét cặp a[i], a[j]

  Swap(a[i],a[j]);

  j=j+1;

- Bước 4:

  i = i+1;

  Nếu i<n-1: Lặp lại Bước 2

  Ngược lại: Dừng

Ví dụ:

- Sắp xếp mảng A[8,4,1,6,5]

- Cài đặt thuật toán

Hướng dẫn cài đặt thuật toán:

```cpp
#include <iostream>
using namespace std;
void Interchange_Sort(int a[], int n){
    int i,j;
    for(i=0; i<n-1; i++){
        for(j=i+1; j<n; j++){
            if(a[i]>a[j]){
                swap(a[i],a[j]);//hoán vị a[i] và a[j]
            }
        }
    }
}

void main()
{
    int a[5] = {8, 4, 1, 6, 5};
    Interchange_Sort(a, 5);
    cout<<"Mang sau khi sap xep:"<<endl;
    for(int i=0;i<5;i++){
        cout<<a[i]<<" ";
    }
    system("pause");
}
```

Kết quả

```
Mang sau khi sap xep:
1 4 5 6 8
```

Đánh giá giải thuật

Số phép so sánh: Ở mỗi lượt thứ i, luôn có (n-i) lần so sánh a[i]>a[j]. Số lần so sánh là:

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

Số phép hoán vị tuỳ thuộc vào kết quả so sánh.

Có thể ước lượng trong từng trường hợp như sau:

Trường hợp	Số lần so sánh	Số lần hoán vị
Tốt nhất	n(n-1)/2	0
Xấu nhất	n(n-1)/2	n(n-1)/2

3.Merge sort

3.1.Khái niệm

Thuật toán trộn (Merge sort) là một kỹ thuật sắp xếp dựa trên kỹ thuật chia và chinh phục. Với độ phức tạp thời gian trong trường hợp xấu nhất là O (n log n), nó là một trong những thuật toán được đánh giá cao nhất.

Sắp xếp hợp nhất trước tiên chia mảng thành các nửa bằng nhau và sau đó kết hợp chúng theo cách đã sắp xếp.

Các thuật toán trong Merge sort sẽ tìm cách tách mảng M thành các mảng con theo các đường chạy (run) rồi sau đó tiến hành nhập các mảng này lại theo từng cặp đường chạy để tạo thành các đường chạy mới có chiều dài lớn hơn đường chạy cũ. Sau một lần tách/nhập thì cuối cùng mảng M chỉ còn lại một đường chạy, lúc đó thì các phần tử trên mảng M sẽ trở nên có thứ tự.

Các thuật toán sắp xếp bằng phương pháp trộn (Merge sort) bao gồm:

- Thuật toán sắp xếp trộn thẳng hay trộn trực tiếp (straight merge sort),
- Thuật toán sắp xếp trộn tự nhiên (natural merge sort).

Đường chạy (Run):

Dãy M[I], M[I+1],…,M[J] (I <= J: 1 <= I, I <= N) là một đường chạy nếu nó có thứ tự.

Chiều dài đường chạy (Run’s Length):

Số phần tử của một đường chạy còn được gọi là chiều dài đường chạy.

Như vậy:

- Mỗi phần tử của dãy là một đường chạy có chiều dài bằng 1.
- Mỗi dãy có thể bao gồm nhiều đường chạy.

Trộn các đường chạy:

Khi ta trộn các đường chạy lại với nhau sẽ cho ra một đường chạy mới có chiều dài bằng tổng chiều dài các đường chạy ban đầu.

3.2. Cách hoạt động

Cho một mảng như sau:

Chúng ta biết rằng sắp xếp hợp nhất chia lặp đi lặp lại toàn bộ mảng thành các nửa bằng nhau trừ khi đạt được các giá trị nguyên tử. Ở đây chúng ta thấy rằng một mảng 8 mục được chia thành hai mảng có kích thước 4.

Điều này không thay đổi trình tự xuất hiện của các mục dữ liệu gốc. Bây giờ chúng ta chia hai mảng này thành một nửa.

Ta tiếp tục chia các mảng này và ta đạt được giá trị nguyên tử không thể chia được nữa.

Bây giờ, kết hợp chúng theo cách giống hệt như khi chúng được chia nhỏ. Lưu ý các mã màu được cung cấp cho các danh sách này.

Trước tiên, ta so sánh phần tử cho từng danh sách và sau đó kết hợp chúng thành một danh sách khác theo cách được sắp xếp. Chúng ta thấy rằng 14 và 33 ở các vị trí đã được sắp xếp. Ta so sánh 27 và 10 và trong danh sách mục tiêu có 2 giá trị, đặt 10 đầu tiên, tiếp theo là 27. Ta thay đổi thứ tự của 19 và 35 trong khi 42 và 44 được đặt tuần tự.

Trong lần lặp lại tiếp theo của giai đoạn kết hợp, chúng ta so sánh danh sách hai giá trị dữ liệu và hợp nhất chúng thành danh sách các giá trị dữ liệu được tìm thấy, đặt tất cả theo thứ tự đã sắp xếp

Sau lần hợp nhất cuối cùng, danh sách sẽ trở thành như thế này.

3.3. Thuật toán

Sắp xếp hợp nhất tiếp tục chia danh sách thành các nửa bằng nhau cho đến khi không thể chia được nữa. Theo định nghĩa, nếu nó chỉ là một phần tử trong danh sách, nó sẽ được sắp xếp. Sau đó, sắp xếp hợp nhất kết hợp các danh sách được sắp xếp nhỏ hơn giữ cho danh sách mới cũng được sắp xếp.

Bước 1: nếu nó chỉ là một phần tử trong danh sách thì nó đã được sắp xếp, hãy quay lại.

Bước 2: chia một cách đệ quy danh sách thành hai nửa cho đến khi không thể chia được nữa.

Bước 3: hợp nhất các danh sách nhỏ hơn thành danh sách mới theo thứ tự được sắp xếp.