TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐỒ ÁN I
LẬP TRÌNH

1 Các thuật toán sắp xếp

1.1 Bài toán sắp xếp

Sắp xếp là quá trình bố trí lại các phần tử của một tập đối tượng nào đó theo một thứ tự nhất định. Chẳng hạn như thứ tự tăng dần (hay giảm dần) đối với một dãy số, thứ tự từ điển đối với các từ v.v... Yêu cầu về sắp xếp thường xuyên xuất hiện trong các ứng dụng Tin học với các mục đích khác nhau: sắp xếp dữ liệu trong máy tính để tìm kiếm cho thuận lợi, sắp xếp các kết quả xử lý để in ra trên bảng biểu v.v...

Nói chung dữ liệu có thể xuất hiện dưới nhiều dạng khác nhau. Một tập các đối tượng cần được sắp xếp là tập các bản ghi (records), mỗi bản ghi bao gồm một số trường (fields) khác nhau. Nhưng không phải toàn bộ mà chỉ là một trương nào đó (hay một vài trường nào đó được chú ý tới thôi. Trường như vậy chúng ta gọi là là khóa (textsfkey). Swasp xếp sẽ được tiến hành dựa vào giá trị của khóa này.

Khi sắp xếp, các bản ghi trong bảng sẽ được đặt vào các vị trí sao cho các giá trị khóa tương ứng của chúng có đúng thứ tự đã ấn định. thì kích thước của toàn bản ghi có thể rất lớn, nên nếu việc sắp xếp thực hiện trực tiếp trên các bản ghi sẽ đòi hỏi sự chuyển đổi vị trí của các bản ghi, kéo theo việc thường xuyên phải di chuyển, copy những vùng nhớ lớn, gây ra những tổn phí thời gian khá nhiều. Thường người ta khắc phục tình trạng này bằng cách xây dựng một bảng khoá: Mỗi bản ghi trong bảng ban đầu sẽ tương ứng với một bản ghi trong bảng khoá. Bảng khoá cũng gồm các bản ghi nhưng mỗi bản ghi chỉ gồm có hai trường:

- Trường thứ nhất chứa khóa.
- Trường thứ hai chứa liên kết tới một bản ghi trong bảng ban đầu, tức là một thông tin đủ để biết bản ghi tương ứng với nó trong bảng ban đầu là bản ghi nào.

Có thể coi *khóa như là đại diện cho các bản ghi* và để đơn giản, ta chỉ nói tới giá trị khóa mà thôi. Các thao tác trong kĩ thuật sắp xếp lẽ ra là tác động lên toàn bản ghi giờ đây chỉ làm việc trên khóa.

1.2 Một số quy ước

Mục này trình bày các cài đặt một số giải thuật sắp xếp phổ biến. Để thuận tiện hơn trong việc theo dõi chương trình sau này, ta đưa vào một số quy ước như sau:

- Dãy khóa cần được sắp xếp được lưu trong mảng `arr` gồm các phần tử từ $0...len-1$, trong đó, `len` là số phần tử của mảng.
- hàm `swap(a, b)` có tác dụng đổi chỗ hai phần tử `a` và `b`
- Kí hiệu arr[i...j] được hiểu là các phần tử từ arr[i] đến arr[j] trong mảng arr.

1.3 Thuật toán sắp xếp nổi bọt

Trong thuật toán săp xếp nổi bọt, các dãy khóa sẽ được duyệt từ cuối dãy lên đàu dãy (từ `arr[len-1]` về `arr[0]`), nếu gặp hai khóa kề nhau bị ngược thứ tự thì đổi chỗ của chúng cho nhau, sau lần duyệt như vậy, khóa nhỏ nhất sẽ trở về vị trí đầu tiên, quá trình lại tiếp tục với các khóa từ dãy `arr[1]` tới `arr[len-1]`:

Cài đặt thuật toán trong C++ như sau:

```
void bublesort()
{
  int i,j;
  for (i = 1; i < len; i++)
    for (j = len-1; j >= i; j--)
      //Neu arr[j] va arr[j-1] nguoc thu tu
      if (arr[j] < arr[j-1])
    {
      //Doi cho ve dung vi tri
      swap(arr[j], arr[j-1]);
      s++;
    }
}
```

1.4 Thuật tóan sắp xếp chọn

Một trong những thuật toán sắp xếp đơn giản nhất là phương pháp sắp xếp chọn. Ý tưởng của thuật toán:

- Trong lần duyệt đầu tiên, duyệt dãy từ [0...len-1] tìm ra phần tử có giá trị nhỏ nhất đổi vị trí về đầu dãy.
- Trong lần duyệt thứ k duyệt dãy từ [k-1...len-1] tìm ra phần tử tử nhỏ nhất trong dãy và đổi về vị trí k-1.
- ...
- Trong lần duyệtthứ `len`-1 chọn trong hai khóa `arr[len-2]` và `arr[len-1]` đổi với vị trí `len-2`

Kết thúc quá trình duyệt thì dãy còn lại đã được sắp xếp.
Sau đây là cài đặt của thuật toán trên C++:

```
void selectionsort()
{
  int i,j,min;
  for (i=0;i<len-1;i++)
    {
      //Dat arr[i] lam phan tu be nhat
      min=i;
      //Tim phan tu be nhat trong day arr[i]-->arr[len-1]
      for (j=i+1;j<len;j++)
    if (arr[j]<arr[min])
```

```
      min=j;
      if (i!=min)
    //Doi cho phan tu be nhat ve vi tri
    //Day arr[0]-->arr[i] la day tang dan
    swap(arr[i],arr[min]);
    }
}
```

1.5 Thuật toán sắp xếp kiểu chèn

Xét dãy khóa arr[0...len-1], ta thấy chỉ gồm một khóa arr[0] chỉ gồm 1 khóa và có thể coi là đã sắp xếp rồi. Xét thêm arr[1], ta so sánh nó với arr[0], nếu thấy arr[1]<arr[0] thì ta chèn nó vào trước arr[1],... Một cách tổng quát, ta sẽ sắp xếp dãy arr[0...i-1] trong điều kiện dãy khóa đã được sắp xếp rồi chèn arr[i] vào dãy đó tại đúng vị trí để được dãy arr[0...i] đã được sắp xếp.

```
void insertionsort()
{
  int i,j,tmp;

  for (i = 1; i < len; i++)
    {
      tmp = arr[i];
      //Chen phan tu arr[i] vao day da co thu tu
      j = i - 1;
      //Tim tu ben phai sang phan tu be hon arr[i]
      while ((j >= 0) && (arr[j] > tmp))
    {
      //Don cac phan tu ve ben phai tao 1 cho chua cho arr[i]
      arr[j+1] = arr[j];
      j = j - 1;
    }
      //Chen phan tu vao dung vi tri,
      //Day arr[0]-->arr[i] la day tang dan
      arr[j+1] = tmp;
    }
}
```

1.6 Shell sort

Nhược điểm của thuật toán sắp xếp chèn thể hiện khi mà ta luôn phải chèn một khá vào vị trí gần đầu dãy. Trong trường hợp đó, người ta sử dụng phương pháp *Shell Sort*. Xét dãy khóa arr[0...len-1]. Với một số nguyên dương $0 <= h <= len - 1$, ta có thể chia dãy đó thành dãy con:

- Dãy con 1: arr[0], arr[h], arr[2h], ...
- Dãy con 2: arr[1], arr[h+1], arr[2h+1], ...
- Dãy con 3: arr[2], arr[h+2], arr[2h+2], ...

- ...
- Dãy con h-1: arr[h-1], arr[2h-1], arr[3h-1], ...

Những dãy con như vậy được gọi là dãy con sắp xếp theo độ dài h. Tư tưởng của thuật toán ShellSort là : Với một bước h, áp dụng thuật toán sắp xếp kiểu chèn từng dãy con độc lập để làm mịn dần dãy khóa chính. Rồi lại làm tương tự đối với bước $hdiv2$ ... cho tới khi $h = 1$ thì ta được dãy khóa đã sắp xếp. Đây chính là nguyên nhân ShellSort hiệu quả hơn thuật toán sắp xếp chèn: khóa nhỏ được nhanh chóng đưa về *gần* vị trí đúng của nó.

1.7 Thuật toán sắp xếp Quick Sort

Thuật toán Quick Sort được đề xuất bởi C.A.R.Hoare là một phương pháp sắp xếp tốt nhất, nghĩa là dù dãy khóa thuộc kiểu dữ liệu *có thứ tự nào*, Quick Sort cũng có thể sắp xếp được và chauw có một thuật toán sắp xếp tổng quát nào nhanh hơn Quick Sort về mặt tốc độ trung bình.

Ý tưởng chủ đọa của phương pháp có thể tóm tắt như sau:

Sắp xếp dãy khóa arr[0...len-1] thì có thể coi là sắp xếp từ chỉ số 0 tới chỉ số len-1 trong dãy khóa đó. Để sắp xếp một đoạn trong dãy khóa, nếu đoạn đó có ít hơn 1 khóa thì không cần phải làm gì cả, còn nếu đoạn đó có ít nhất 2 khóa thì ta chọn ngẫu nhiên khóa nào đó của đoạn làm "chốt" (*pivot*). Mọt khóa nhỏ hơn chốt đươc xếp vào vị trí đứng sau chốt. Sau phép hoán chuyển như vậy thì đoạn đang xét được chia làm hai đoạn khá rỗng mà mọi khóa trong đoạn đầu đều <= chốt và một khóa trong đoạn sau đều >= chốt. Và vấn đề trở thành sắp xếp hai đoạn mới tạo ra (có độ dài ngắn hoăn đoạn ban đầu) bằng phương pháp tương tự.

Để cài đặt thuật toán này trong C++, ta sử dụng hai hàm:

- Hàm `void partition(int L, int H` chọn một chốt pivot bất kì trong dãy từ arr[L...H], chuyển các phần tử trong dãy có giá trị bé hơn pivot về trước pivot và các phần tử có giá trị lớn pivot về sau pivot. Sau đó gọi đệ quy tiếp tục với hai dãy con nhỏ hơn.
- Hàm `void quicksort()` gọi hàm `partition(0, len-1)` thực hiện sắp xếp toàn bộ mảng. Thực ra ta có thể không cần tới hàm này.

Cài đặt `quicksort`:

```
1  void partition(int L, int H)
2  {
3      int i,j,pivot;
4      //Neu chi co duoi 1 phan tu thi khong phai lam di
5      if (L>=H)
6          return;
7      i=L;j=H;
8      srand(time(NULL));
9      //Chon chot ngau nhien, tranh cac truong hop dac biet
10     pivot=arr[L+rand()%(H-L+1)];
11
12     while (i<=j)
```

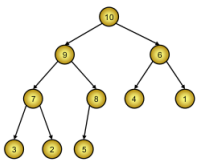
```
13      {
14          //Tim phan tu lon hon hoac bang pivot tu trai sang
15          while (arr[i]<pivot)
16              i++;
17          //Tim phan tu lon hon hoac bang pivot tu phai sang
18          while (arr[j]>pivot)
19              j--;
20
21          if (i<=j)
22          {
23              //Neu tim thay va hai phan tu khac nhau thi doi cho
24              if (i<j)
25                  swap(arr[i],arr[j]);
26              i++; j--;
27          }
28      }
29      //Tiep tuc voi day con trai
30      partition(L,j);
31      //Tiep tuc voi day con phai
32      partition(i,H);
33  }
34
35
36  void quickSort()
37  {
38      partition(0,len-1);
39  }
```

1.8 Thuật toán sắp xếp vung đống - Heap Sort

Heap Sort được đề xuất bởi J.W.J.Williams năm 1981, thuật toán không những đóng góp một phương pháp sắp xếp quan trọng để biểu diễn một cấu trúc dữ liệu quan trọng

1.8.1 Đống - Heap

Đống là một dạng *cây nhị phân hoàn chỉnh đặc biệt* mà giá trị tại mọi nút có độ ưu tiên cao hơn hay bằng giá trị lưu trong hai nút con của nó. Trong thuật toán sắp xếp kiểu vun đống, ta coi quan hệ "ưu tiên hơn hay bằng" là quan hệ "lớn hơn hay bằng": >=



Hình 1: Heap