Giải Thuật Radix Sort

❖ *Ý tưởng thuật toán*

○ Khác với các thuật toán trước, Radix sort là một thuật toán tiếp cận theo một hướng hoàn toàn khác. Nếu như trong các thuật toán khác, cơ sở để sắp xếp luôn là việc so sánh giá trị của 2 phần tử thì Radix sort lại dựa trên nguyên tắc phân loại thư của bưu điện. Vì lý do đó nó còn có tên là Postmans sort. Nó không hề quan tâm đến việc so sánh giá trị của phần tử và bản thân việc phân loại và trình tự phân loại sẽ tạo ra thứ tự cho các phần tử.

○ Ta biết rằng, để chuyển một khối lượng thư lớn đến tay người nhận ở nhiều địa phương khác nhau, bưu điện thường tổ chức một hệ thống phân loại thư phân cấp. Trước tiên, các thư đến cùng một tỉnh, thành phố sẽ được sắp chung vào một lô để gửi đến tỉnh thành tương ứng. Bưu điện các tỉnh thành này lại thực hiện công việc tương tự. Các thư đến cùng một quận, huyện sẽ được xếp vào chung một lô và gửi đến quận, huyện tương ứng. Cứ như vậy, các bức thư sẽ được trao đến tay người nhận một cách có hệ thông mà công việc sắp xếp thư không quá nặng nhọc.

○ Mô phỏng lại qui trình trên, để sắp xếp dãy a1, a2, …, an, giải thuật Radix Sort thực hiện như sau:

▪ Trước tiên, ta có thể giả sử mỗi phần tử ai trong dãy a1, a2, …, an là một số nguyên có tối đa m chữ số.

▪ Ta phân loại các phần tử lần lượt theo các chữ số hàng đơn vị, hàng chục, hàng trăm, . tương tự việc phân loại thư theo tỉnh thành, quận huyện, phường xã, ..

○ Các bước thực hiện thuật toán như sau:

▪ 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

❖Mã CODE

```
#include <stdio.h>
#define MAX 5
#define SHOWPASS

void print(int *a,int n)
{
    int i;
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
}

void RadixSort(int *a,int n)
{
        int i,b[MAX],m=0,exp=1;
        for(i=0;i<n;i++)
        {
            if(a[i]>m)
                m=a[i];
        }

        while(m/exp>0)
        {
            int bucket[10]={0};
            for(i=0;i<n;i++)
                bucket[a[i]/exp%10]++;
            for(i=1;i<10;i++)
                bucket[i]+=bucket[i-1];
            for(i=n-1;i>=0;i--)
                b[--bucket[a[i]/exp%10]]=a[i];
            for(i=0;i<n;i++)
                a[i]=b[i];
            exp*=10;
```

```
#ifdef SHOWPASS
                printf("\nPASS   : ");
                print(a,n);
#endif
        }
      }


int main()
{
        int arr[MAX];
        int i,n;

        printf("Enter total elements (n < %d) : ",MAX);
        scanf("%d",&n);

        printf("Enter %d Elements : ",n);
        for(i=0;i<n;i++)
                        scanf("%d",&arr[i]);


        printf("\nARRAY  : ");
        print(&arr[0],n);

        radixsort(&arr[0],n);

        printf("\nSORTED : ");
        print(&arr[0],n);
        printf("\n");

        return 0;
}
```

❖ *Ví dụ minh họa*

Cho dãy số a:

701 1725 999 9170 3252 4518 7009 1424 428 1239 8425 7013

Phân lô theo hàng đơn vị:

12	0701										

11	1725										
10	0999										
9	9170										
8	3252										
7	4518										
6	7009										
5	1424										
4	0428										
3	1239										0999
2	8425						1725			4518	7009
1	7013	9170	0701	3252	7013	1424	8425			0428	1239
CS	B	0	1	2	3	4	5	6	7	8	9

Phân lô theo hàng chục:

12	0999										
11	7009										
10	1239										
9	4518										
8	0428										
7	1725										
6	8425										
5	1424										
4	7013			0428							
3	3252			1725							
2	0701	7009	4518	8425							
1	9170	0701	7013	1424	1239		3252		9170		0999
CS	A	0	1	2	3	4	5	6	7	8	9

Phân lô theo hàng trăm:

12	0999										
11	9170										
10	3252										
9	1239										
8	0428										
7	1725										
6	8425										
5	1424										
4	4518										
3						0428					
2	7009	7013		3252		8425			1725		
1	0701	7009	9170	1239		1424	4518		0701		0999
CS	B	0	1	2	3	4	5	6	7	8	9

Phân lô theo hàng ngàn:

12	0999										
11	1725										
10	0701										
9	4518										
8	0428										
7	8425										
6	1424										
5	3252										
4	1239										
3	9170	0999	1725								
2	7013	0701	1424						7013		
1	7009	0428	1239		3252	4518			7009	8425	9

CS	B	0	1	2	3	4	5	6	7	8	9

Lấy các phần tử từ các lô B0, B1, ., B9 nối lại thành a:

12	9170										
11	8425										
10	7013										
9	7009										
8	4518										
7	3252										
6	1725										
5	1424										
4	1239										
3	0999										
2	0701										
1	0428										
CS	B	0	1	2	3	4	5	6	7	8	9

❖*Độ phức tạp*

- Với một dãy n số, mỗi số có tối đa m chữ số, thuật toán thực hiện m lần các thao tác phân lô và ghép lô.
- Trong thao tác phân lô, mỗi phần tử chỉ được xét đúng một lần, khi ghép cũng vậy.
- Như vậy, chi phí cho việc thực hiện thuật toán hiển nhiên là O(2mn) = O(n).

Sưu Tầm

Comments

You do not have permission to add comments.