数据结构作业第十章——排序

庄震丰 22920182204393

Dec. 12^{nd}, 2019

10-32

荷兰国旗问题：

设有一个仅由红、白、蓝三种颜色的条块组成的序列，请编写一个时间复杂度为 $O(n)$d 的算法，使得这些条块按红、白、蓝的顺序排列，即排成荷兰国旗的图案。

算法分析

先统计不同颜色的条纹数目，按照数目进行分块排序，先整体排序，把第 k2 个位置作为轴，将小于蓝色的放到左边，再以 k1 作为轴，把小于白色的放到左边，则满足红、白、蓝顺序。一共只进行了两趟排序。

时间复杂度 O(n), 空间复杂度 O(n).

code

```
include<bits/stdc++.h>
#define maxn 1000
using namespace std;
int n;
int a[maxn];
void flagsort(int l,int r,int index)
{
    int i=l,j=r;
    do
    {
        while(a[i]<index) i++;
        while(a[j]>=index) j--;
        if (i<=j)
            {
                swap(a[i],a[j]);
                i++;
                j--;
            }
    }while(i<j);
}
void init()
{
    cin>>n;
    int k1=0,k2=0;
    for (int i=1;i<=n;i++)
    {
        cin>>a[i];
        k1+=(a[i]==0);
        k2+=(a[i]<=1);
    }
    flagsort(1,n,2);
    flagsort(1,k2,1);
}
int main()
{
```

```
        init();
        for (int i=1;i<=n;i++)
            cout<<a[i]<<" ";
    }
```

10-34

已知 $(k_1,k_2,...,k_p)$, 是堆，则可以写出一个时间复杂度为 $O(logn)$ 的算法，将 $(k_1,k_2,k_3,...,k_{p+1})$ 调整为堆，试编写一个从 p=1 起，逐个插入建堆的算法，讨论建堆的复杂度。

算法分析

每当一个节点进入时，执行 insert 操作，加到叶子节点，再和其父亲比较，若为大根堆，且当前叶子节点大于其父亲节点，则交换，否则就保持。若为小根堆，当前叶子节点小于其父亲节点，则交换，否则保持。直到插入完所有节点。

单个插入时间复杂度 O(logn), 总时间复杂度 O(nlogn), 空间复杂度 O(n).

code

```
#include<bits/stdc++.h>
#define maxn 1000
using namespace std;
int n;
int a[maxn];
void Insert(int num,int pos)
{
    a[pos]=num;
    while(a[pos]<a[pos/2]&&pos>=1)
    {
        swap(a[pos],a[pos/2]);
        pos=pos/2;
    }
}
void init()
{
    cin>>n;
    int k;
    for (int i=1;i<=n;i++)
    {
        cin>>k;
        Insert(k,i);
    }
}
int main()
{
    init();
    for (int i=1;i<=n;i++)
        cout<<a[i]<<" ";
    return 0;
}
```

10-38

2-路归并排序的另一种策略，先对排序序列扫描一遍，找出划分为若干个最大有序序列，将这些序列作为初始归并段，写出一个算法在链表结构上实现这个策略。

算法分析

对于给定的列表序列，建立一个循环队列，所有的有序子序列的头节点先进队，之后每次取出两个队尾的头结点指针，将两个子序列分别进行重构，再将其中一个较小的子序列头指针 push 进队，直到最后只有一个头节点指针。

时间复杂度 O(logn), 空间复杂度 O(n).

Code

```
#include <stdio.h>
void Algo_10_38(LinkList L)
{
    int k, len;
    int i, m, n;
    int seg[101];                       //号单元计数0
    LinkList SR[101], TR[101];
    LinkList p, r;

    for(len=0,p=L->next; p; p=p->next)      //初始化链表指针数组
        SR[++len] = p;

    seg[0] = seg[1] = 1;

    for(k=2; k<=len; k++)                   //用最大有序子列生成初始归并段
    {
        if(SR[k]->data<SR[k-1]->data)
        {
            seg[0]++;
            seg[seg[0]] = k;
        }
    }

    MSort_10_38(SR, TR, seg, len, 1, seg[0]);  //对各初始归并段的指针排序

    for(k=1,r=L; k<=len; k++)                   //根据排好的指针顺序重排记录
    {
        r->next = TR[k];
        r = r->next;
    }

    r->next = NULL;
}

void MSort_10_38(LinkList SR[], LinkList TR[], int seg[], int len, int s, int t)
{
    int m, ks, ke;
    LinkList R[len+1];

    if(s==t)
    {
        ks = seg[s];
        ke = ++s<=seg[0] ? (seg[s]-1) : len;
        for( ; ks<=ke; ks++)
            TR[ks] = SR[ks];
    }
```

```
47          else
48          {
49              m = (s+t)/2;
50              MSort_10_38(SR, R, seg, len, s, m);
51              MSort_10_38(SR, R, seg, len, m+1, t);
52              Merge_10_38(R, TR, seg, len, s, m, t);
53          }
54      }
55
56      void Merge_10_38(LinkList SR[], LinkList TR[], int seg[], int len, int s, int m, int t)
57      {
58          int k;
59          int is, ie, js, je;
60
61          is = seg[s];
62          ie = seg[m+1]-1;
63          js = seg[m+1];
64          je = ++t<=seg[0] ? (seg[t]-1) : len;
65
66          k = is;
67
68          while(is<=ie&&js<=je)
69          {
70              if(SR[is]->data<=SR[js]->data)
71                  TR[k++] = SR[is++];
72              else
73                  TR[k++] = SR[js++];
74          }
75
76          while(is<=ie)
77              TR[k++] = SR[is++];
78
79          while(js<=je)
80              TR[k++] = SR[js++];
81      }
82
83      void PrintKey(LElemType_L e)
84      {
85          printf("%d ", e);
86      }
87
88      int main()
89      {
90          FILE *fp;
91          LinkList L;
92          int num;
93
94          printf("创建并输出一个任意序列...\n");
95          fp = fopen("Data/Algo_10_37.txt", "r");
96          Scanf(fp, "%d", &num);
97          CreateList_TL(fp, &L, num);
98          ListTraverse_L(L, PrintKey);
```

```
99          printf("\n\n");
100
101         printf("将关键字按递增顺序排列...\n");
102         Algo_10_38(L);
103         ListTraverse_L(L, PrintKey);
104         printf("\n\n");
105
106         return 0;
107     }
```

10-42

序列中值序列值的是，如果将此序列排序后，它的第 $\frac{n}{2}$ 个记录，试写出一个中值记录的算法。

算法分析

利用快速排序的思想，从序列中取一个数 mid, 然后把序列分成小于等于 mid 和大于等于 mid 的两部分，由两个部分的元素个数和 k 的大小关系可以确定这个数是在哪个部分。对部分序列的探查可以递归处理。采用分治的思想。时间复杂度 $O(n)$, 空间复杂度 $O(n)$

code

```
1   #include<bits/stdc++.h>
2   #define maxn 1000
3   using namespace std;
4   int n;
5   int a[maxn];
6   void flagsort(int l,int r,int rank)
7   {
8       int i=l,j=r;
9       int index=a[(l+r)/2];
10      do
11      {
12          while(a[i]<index) i++;
13          while(a[j]>index) j--;
14          if (i<=j)
15              {
16                  swap(a[i],a[j]);
17                  i++;
18                  j--;
19              }
20      }while(i<j);
21      if(l<=j && rank<=j-l+1) flagsort(l,j,rank);
22      if(i<=r && rank>=i-l+1) flagsort(i,r,rank-(i-l));
23  }
24  void init()
25  {
26      cin>>n;
27      for (int i=1;i<=n;i++)
28          cin>>a[i];
29      int k=(n+1)/2;
30      flagsort(1,n,k);
31      cout<<a[k];
32  }
33  int main()
```

```
34      {
35          init();
36          return 0;
37      }
```

10-43

已知序列 a[1..n] 中的关键字各不相同，可按照如下所述实现计数排序，另设一个数组 c[1...n]，对每一个记录 a[i], 统计序列中关键字比它小的记录个数存于 c[i], 则 c[i]=0 的记录为关键字最小的记录，然后依 c[i] 值的大小对 a 中记录进行重新排序，试编写上述算法。

算法分析

假设输入线性表 L 的长度为 n,$L = L_1, L_2, ..., L_n$：线性表的元素属于有限偏序集 S，|S|=k 且 k=O(n)，S=S1,S2,..Sk；则计数排序可以描述如下：1、扫描整个集合 S，对每一个 Si S，找到在线性表 L 中小于等于 Si 的元素的个数 T(Si)；2、扫描整个线性表 L，对 L 中的每一个元素 Li，将 Li 放在输出线性表的第 T(Li) 个位置上，并将 T(Li) 减 1。时间复杂度 $O(n)$, 空间复杂度 $O(n)$。

code

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 100000;
const int k = 1000;
int a[MAXN], c[MAXN], ranked[MAXN];

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        ++c[a[i]];
    }
    for (int i = 1; i < k; ++i)
        c[i] += c[i-1];
    for (int i = n-1; i >= 0; --i)
        ranked[--c[a[i]]] = a[i];
    for (int i = 0; i < n; ++i)
        cout << ranked[i] << endl;
    return 0;
}
```