基数排序

指院网络工程教研中心　陈卫卫

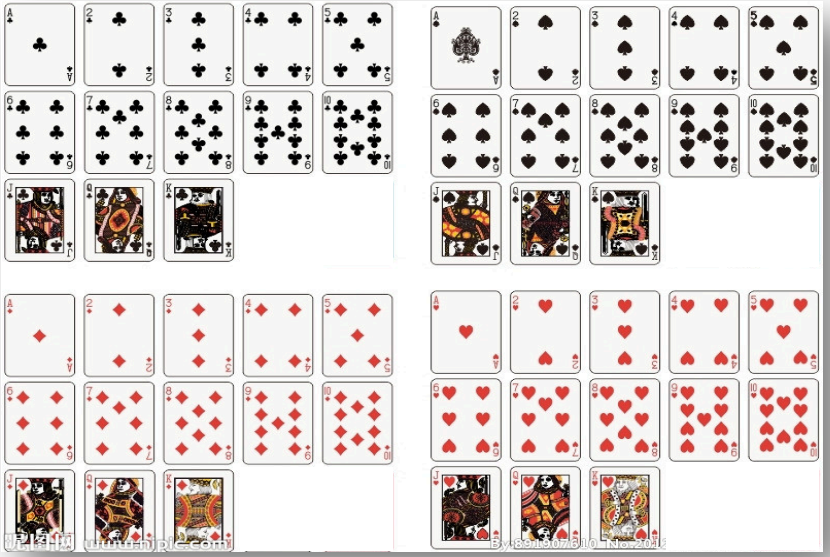
按个位分组：

0组	1组	2组	3组	4组	5组	6组	7组	8组	9组
450	671	522	203		315		007	478	219
	101	902					767		139

按个位收集：

450 671 101 522 902 203 315 007 767 478 219 139

如何整理扑克牌？



花色次序：♣ ♦ ♥ ♠

面值次序：2，3，…，10，J，Q，K，A

“有序”形式：

♣2♣3…♣A ♦2♦3…♦A ♥2♥3…♥A ♠2♠3…♠A

整理扑克牌的方法

要求整理成：

♣2♣3…♣A ♦2♦3…♦A ♥2♥3…♥A ♠2♠3…♠A

步骤 1）先按面值分成13堆：

2，3，4，…，10，J，Q，K，A

步骤2）收叠这些牌（2在最下，A在最上）

步骤3）再按花色把它们分成4堆：♣ ♦ ♥ ♠

步骤4）依次收叠在一起

——基数排序

基数排序

学习目标和要求

1. 准确复述基数排序的基本思想

2. 编写基数排序的算法

3. 知道算法的时间复杂性和稳定性

基数排序（radix sort）

又称桶排序，串排序，字典排序

基本原理

按照数据元素的各个分量值，反复将这些元素分组和收集，最后达到排序目的。

基数排序示例

[例] 671 203 450 219 522 478 315 007 139 101 902 767进行基数排序。

按个位分组：

0组	1组	2组	3组	4组	5组	6组	7组	8组	9组
450	671	522	203		315		007	478	219
	101	902					767		139

按个位收集：

450 671 101 522 902 203 315 007 767 478 219 139

基数排序示例

个位收集结果：450 671 101 522 902 203 315 007 767 478 219 139

按十位分组：

0组	1组	2组	3组	4组	5组	6组	7组	8组	9组
101	315	522	139		450	767	671		
902	219						478		
203									
007									

按十位收集：

101 902 203 007 315 219 522 139 450 767 671 478

基数排序示例

十位收集结果：101 902 203 007 315 219 522 139 450 767 671 478

按百位分组：

0组	1组	2组	3组	4组	5组	6组	7组	8组	9组
007	101 139	203 219	315	450 478	522	671	767		902

按百位收集：

007 101 139 203 219 315 450 478 522 671 767 902

结果

字符串排序示例

- 例如，将下面几个英文单词排成字典次序。

  for end else goto file int new case char and

尾部加空格符凑成4个“字母”：

for□ end□ else goto file int□ new□ case□ char and□

字符串排序示例

- 例如，将下面几个英文单词排成字典次序。

for end else goto file int new case char and

第一遍，按第4个字母分组（没列出空组）为：

□组：for□ end□ int□ new□ and□

e组： else file case

o组： goto

r组： char

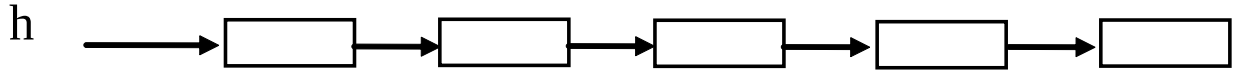
第一遍收集结果：

for□ end□ int□ new□ and□ else file case goto char

算法的实现

1。数据结构

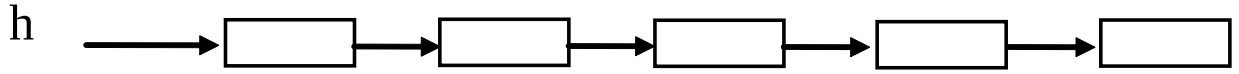
1）数据采用链式存储，总队列h



算法的实现

1。数据结构

1）数据采用链式存储，总队列h



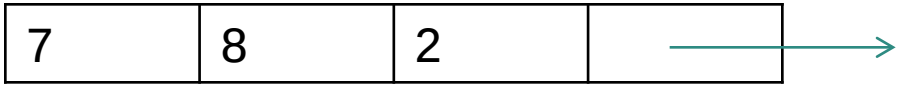
2）结点值域为数组a[k]，每个分量a[j]满足

$0 \le a[j]$的序号值$\le m-1$（j=0，2，…，k−1）

结点结构：

a[0]	a[1]	…	a[k-1]	next

整数782的存储结构为：

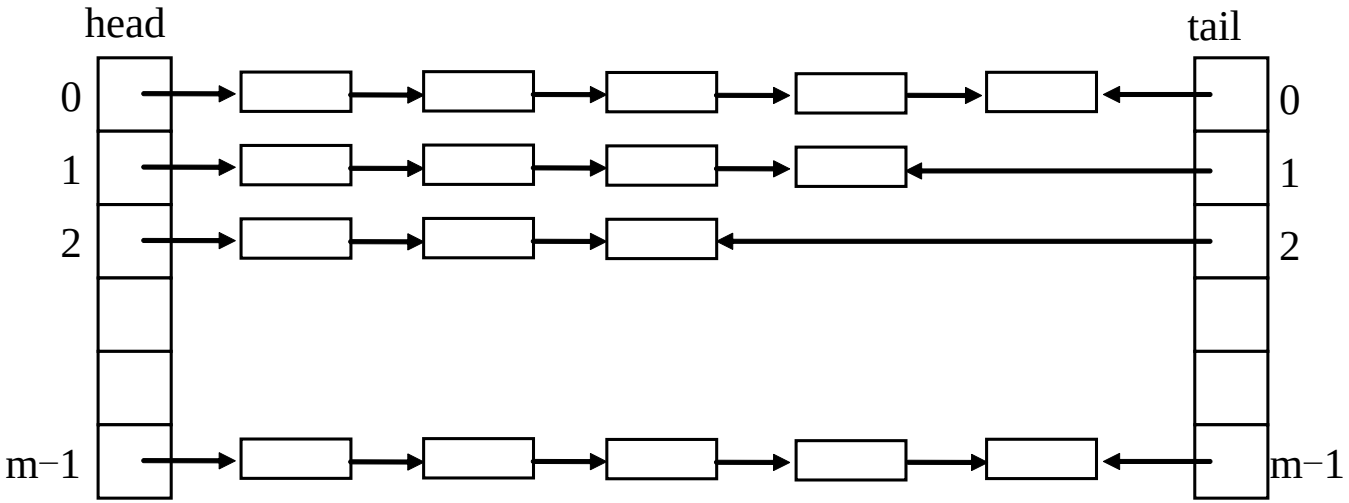


算法的实现

1。数据结构

m个分队列用于分组 q[0]，q[1]，q[2]，…，q[m−1]

struct {ptr head,tail;} q[m];



基数排序算法文字叙述形式

步骤1）置分量下标j等于k−1

步骤2）[分组]

①将各分队列q[0]，q[1]，q[2]，…，q[m−1]置空

②依次从总队列中取下元素a，设其第j个分量a[j]的序号值为t，将该元素放在分队列q[t]的尾部

反复如此，直到把总队列中元素取完

步骤3）[收集]

按分队列的序号由小到大，依次将分队列q[0]，q[1]，q[2]，…，q[m-1]首尾相接，组成新的总队列

步骤4）j=j−1，若j<0，排序结束；否则，转步骤2

基数排序算法（实现）

（1）类型定义

```
typedef  struct  node  //定义结点类型
 { char  a[k];      //值域和分量类型
   struct  node  *next; // 链域
 } *ptr;     // 指针类型名
```

a[0]	a[1]	...	a[k-1]	next

基数排序算法（实现）

```
void radix_sort(ptr &h,int k)
{ int i,j,y;    char x;    ptr p;
  struct {ptr head,tail;} q[m];
1. for(j=k−1;j>=0;j− −)
2.  { for(i=0;i<m;i++) q[i].head=NULL;
3.    while(h!=NULL)  //总队列不空
4.     {  x=h−>a[j];    //取出分量值
5.        y=value(x);     //将x换算成序号值
6.       if(q[y].head==NULL) q[y].head=h;
7.          else q[y].tail−>next=h;
8.       q[y].tail=h;
9.        h=h−>next; //取总队列下一个元素
     } //分组完毕
//收集
10.  i=0;
11.  while(q[i].head==NULL) i++;
12.  h=q[i].head;
13.  p=q[i].tail;
14.  for(i++;i<m;i++)
15.   if(q[i].head !=NULL)
        p−>next=q[i].head,p=q[i].tail;
16.  p−>next=NULL;
     }
   }
```

时间复杂性

一遍分组需要O(n)时间

```
void radix_sort(ptr &h,int k)
{ int i,j,y;    char x;    ptr p;
  struct {ptr head,tail;} q[m];
1. for(j=k-1;j>=0;j--)
2.  {  for(i=0;i<m;i++)  q[i].head=NULL;
3.    while(h!=NULL)  //总队列不空
4.      {   x=h->a[j];    //取出分量值
5.          y=value(x);     //将x换算成序号值
6.          if(q[y].head==NULL) q[y].head=h;
7.              else  q[y].tail->next=h;
8.          q[y].tail=h;
9.          h=h->next;  //取总队列下一个元素
      } //分组完毕
```

一遍收集用O(m)时间

```
//收集
10.  i=0;
11.  while(q[i].head==NULL)  i++;
12.  h=q[i].head;
13.  p=q[i].tail;
14.  for(i++;i<m;i++)
15.    if(q[i].head !=NULL)
           p->next=q[i].head,p=q[i].tail;
16.  p->next=NULL;
    }
  }
```

时间复杂性：T(n)=O(k(n+m))

空间复杂性

```
void radix_sort(ptr &h,int k)
{ int i,j,y;    char x;    ptr p;
  struct {ptr head,tail;} q[m];
1. for(j=k-1;j>=0;j--)
2.  { for(i=0;i<m;i++) q[i].head=NULL;
3.    while(h!=NULL)  //总队列不空
4.      {  x=h->a[j];    //取出分量值
5.         y=value(x);     //将x换算成序号值
6.        if(q[y].head==NULL) q[y].head=h;
7.           else q[y].tail->next=h;
8.        q[y].tail=h;
9.         h=h->next;  //取总队列下一个元素
      } //分组完毕
//收集
10.  i=0;
11.  while(q[i].head==NULL) i++;
12.  h=q[i].head;
13.  p=q[i].tail;
14.  for(i++;i<m;i++)
15.    if(q[i].head !=NULL)
          p->next=q[i].head,p=q[i].tail;
16.  p->next=NULL;
    }
```

空间复杂性：S(n)=O(n+m)

各种内部排序方法比较

排序方法	平均情况	最坏情况	辅助存储	稳定性
直接插入排序	$O(n^2)$	$O(n^2)$	$O(1)$	√
希尔排序	与增量序列有关：$O(n^{3/2})$,$O(n(\log n)^2)$			×
简单选择排序	$O(n^2)$	$O(n^2)$	$O(1)$	×
冒泡排序	$O(n^2)$	$O(n^2)$	$O(1)$	√
快速排序	$O(n\log n)$	$O(n^2)$	$O(\log n)$	×
堆排序	$O(n\log n)$	$O(n\log n)$	$O(1)$	×
归并排序	$O(n\log n)$	$O(n\log n)$	$O(n)$	√
基数排序	$O(k(n+m))$	$O(k(n+m))$	$O(n+m)$	√
	k:待排元素的位数，m为基数的个数			