基数排序

- 不通过待排序数据元素之间的比较
- 根据关键字本身的性质进行排序

  分配排序，桶排序，基数排序
- 例1：名字按照字典序排序
- 例2：1—m之间的整数排序

基数排序

- 例3：一年的全国高考考生人数为500万，分数100--900，没有小数，把这500万考生按照分数排序。
  - 可能的分数为801个，创建801个“桶”，每个桶对应一个分值
  - 依次查看每个学生的成绩，将其放入对应其分值的桶
  - 从900分对应的那个桶开始依次收集，即完成排序
- 如果待排序的数据元素的取值是从0到2亿？
  - 那就要分配2亿个桶了，这“太浪费”的，所以桶排序有其局限性，适合元素值集合并不大的情况。

桶排序要求：关键字的取值范围已知

桶排序

- 桶排序的基本操作：
  - 分配关键字:根据每个数据元素关键字的值将其分配到相应的桶中
  - 每个“桶”内排序 数据元素均匀分布，则每个桶中数据元素个数均匀
  - 收集桶
- 不能象比较排序那样以统一的数据元素之间的“比较”次数衡量“工作量”
- 分析每一步骤地工作进行时间复杂度的分析

基数排序

- 一张牌有两个关键字组成：花色(♠<♥<♣<♦)+面值(2<3<4<...<A)。假如一张牌的大小首先被花色决定，同花色的牌由数字决定。
- 52张牌排序有两种算法：

(1)首先按照面值对所有牌进行排序，然后按照花色再次对所有牌进行排序。

(2)首先按照花色将所有牌分成4组。然后同组的牌(同花色)再按照面值进行排序。

基数排序—方法1

(1)首先按照面值对所有牌进行排序，然后按照花色再次对所有牌进行排序。牌的面值13个，准备13个空桶，分别对应13个面值

依次查看52张牌，根据其面值将其放入对应的桶中

基数排序—方法1

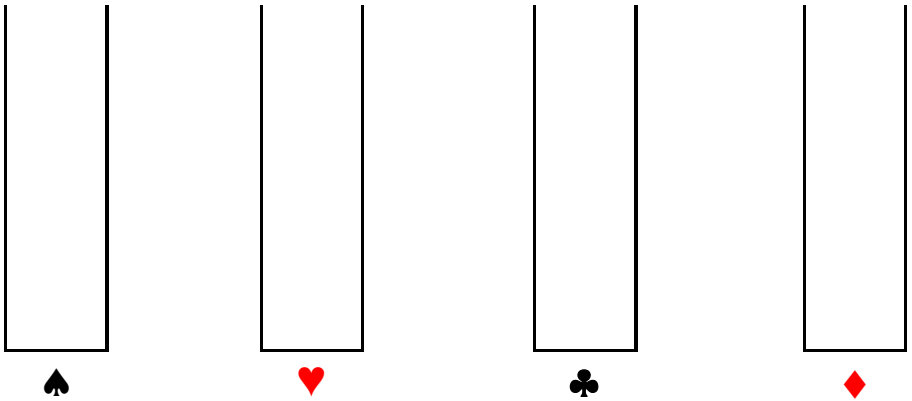
(1)首先按照面值对所有牌进行排序，然后按照花色再次对所有牌进行排序。牌的面值13个，准备13个空桶，分别对应13个面值

依次查看52张牌，根据其面值将其放入对应的桶中----第一次分配

从“2”号开始依次收集，将每个桶中的牌顺次倒出，52张牌现在的顺序是4张2，4张3，4张4，……，4张K,4张A----第一次收集

基数排序—方法1

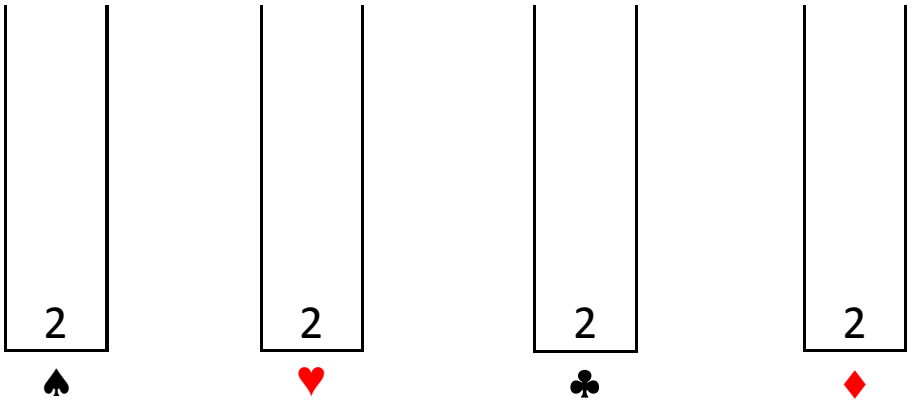
根据第一次收集的结果然后按照花色再次对所有牌进行排序。花色有4个，准备4个空桶，分别对应4个花色



依次查看第一次收集得到52张牌，根据花色将其放入对应的桶中----第二次分配

基数排序—方法1

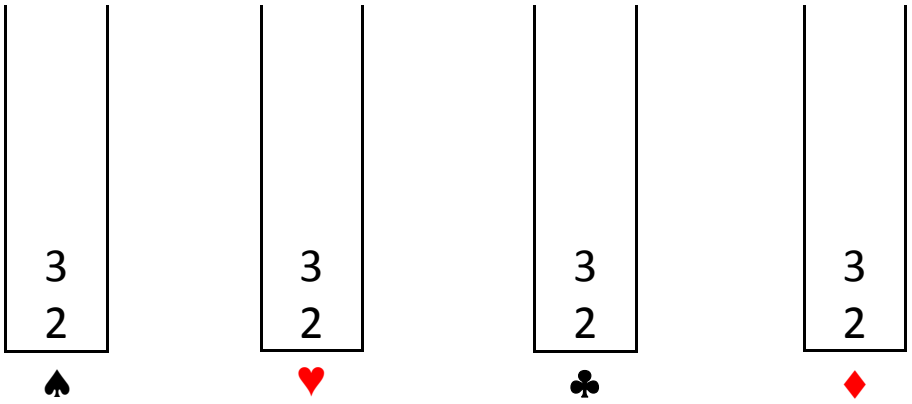
根据第一次收集的结果然后按照花色再次对所有牌进行排序。花色有4个，准备4个空桶，分别对应4个花色



依次查看第一次收集得到52张牌，根据花色将其放入对应的桶中----第二次分配

基数排序—方法1

根据第一次收集的结果然后按照花色再次对所有牌进行排序。花色有4个，准备4个空桶，分别对应4个花色



依次查看第一次收集得到52张牌，根据花色将其放入对应的桶中----第二次分配

收集桶的方式：最先进入一个桶的“2”最先倒出----实现时考虑用队列模拟桶

基数排序—方法1

根据第一次收集的结果然后按照花色再次对所有牌进行排序。花色有4个，准备4个空桶，分别对应4个花色

扑克牌的排序是根据其2个值决定的：花色和面值，且花色的优先级高于面值

♠	♥	♣	♦
A	A	A	A
K	K	K	K
.	.	.	.
4	4	4	4
3	3	3	3
2	2	2	2

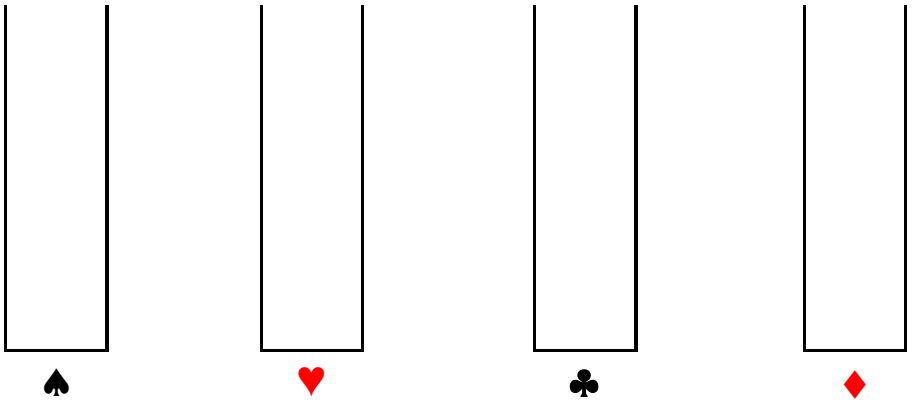
排序过程：先根据面值，后根据花色。二次分配二次收集。没有进行2张扑克牌的比较

依次查看第一次收集得到52张牌，根据花色将其放入对应的桶中----第二次分配

从“♠”号桶开始依次收集，将每个桶桶口封死，桶底打开，顺次倒出每一个桶中的牌，52张牌就排好序了：13张♠从2到A，13张♥从2到A，13张♣从2到A，13张♦从2到A ----第二次收集

基数排序—方法2

(2)首先按照花色将所有牌分成4组。然后同组的牌(同花色)再按照面值进行排序。花色有4个，准备4个空桶，分别对应4个花色

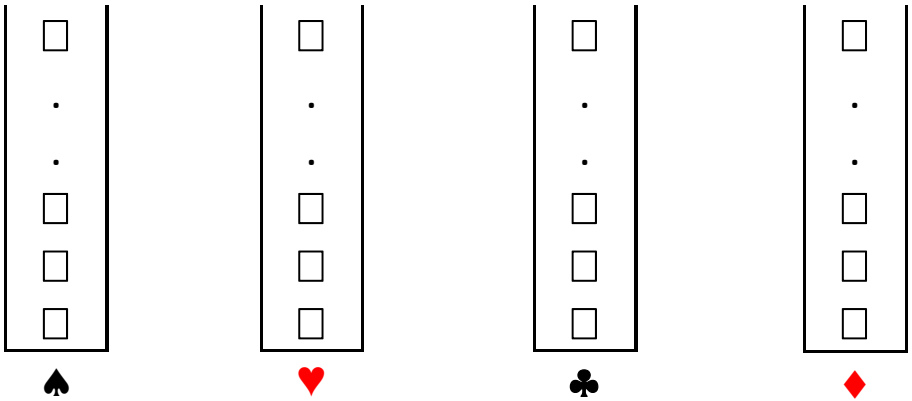


依次查看52张牌，根据花色将其放入对应的桶中----第一次分配

基数排序—方法2

(2)首先按照花色将所有牌分成4组。然后同组的牌(同花色)再按照面值进行排序。花色有4个，准备4个空桶，分别对应4个花色

花色相同的牌放入同一个桶中，每个桶中有13张牌



依次查看52张牌，根据花色将其放入对应的桶中----第一次分配

第一次收集？能否和前面的方法一样将所有桶依次收集在一起？

基数排序—方法2

若4个花色的桶依次一起收集，则第一次收集的52张牌为：13张♠，13张♥，13张♣，13张♦。牌的面值13个，准备13个空桶，分别对应13个面值

依次查看第一次收集得到的52张牌，根据其面值将其放入对应的桶中

基数排序—方法2

若4个花色的桶依次一起收集，则第一次收集的52张牌为：13张♠，13张♥，13张♣，13张♦。牌的面值13个，准备13个空桶，分别对应13个面值

依次查看第一次收集得到的52张牌，根据其面值将其放入对应的桶中

基数排序—方法2

若4个花色的桶依次一起收集，则第一次收集的52张牌为：13张♠，13张♥，13张♣，13张♦。牌的面值13个，准备13个空桶，分别对应13个面值

依次查看第一次收集得到的52张牌，根据其面值将其放入对应的桶中

基数排序—方法2

若4个花色的桶依次一起收集，则第一次收集的52张牌为：13张♠，13张♥，13张♣，13张♦。牌的面值13个，准备13个空桶，分别对应13个面值

依次查看第一次收集得到的52张牌，根据其面值将其放入对应的桶中

基数排序—方法2

若4个花色的桶依次一起收集，则第一次收集的52张牌为：13张♠，13张♥，13张♣，13张♦。牌的面值13个，准备13个空桶，分别对应13个面值

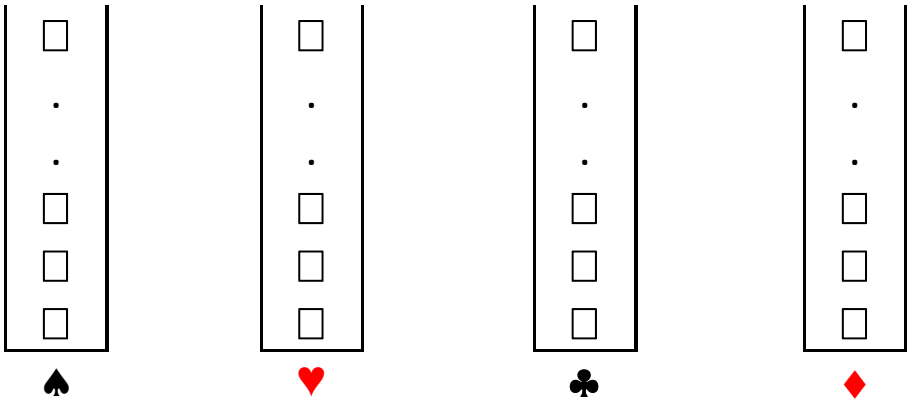
依次查看第一次收集得到的52张牌，根据其面值将其放入对应的桶中

第二次收集?　　一起收集得不到正确的排序结果。

基数排序----正确的方法一

(2)首先按照花色将所有牌分成4组。然后同组的牌(同花色)再按照面值进行排序。花色有4个，准备4个空桶，分别对应4个花色

花色相同的牌放入同一个桶中，每个桶中有13张牌



依次查看52张牌，根据花色将其放入对应的桶中----第一次分配

第一次收集?

不一起收集。先收集♠桶，对收集的结果按照面值进行桶排序；再收集♥桶，对收集的结果按照面值进行桶排序；……

基数排序

- 基数排序是一种借助“多关键字排序”的思想来实现“单关键字排序”的内部排序算法。
- n 个记录的序列 $\{R_1, R_2, \cdots, R_n\}$,对关键字 $(K_i^0, K_i^1, \cdots, K_i^{d-1})$ 有序是指：对于序列中任意两个记录 R_i 和 $R_j (1 \leqslant i < j \leqslant n)$ 都满足下列(词典)有序关系：

$$(K_i^0, K_i^1, \cdots, K_i^{d-1}) < (K_j^0, K_j^1, \cdots, K_j^{d-1})$$

K^0 被称为“最主”位关键字

K^{d-1} 被称为“最次”位关键字

基数排序----2种方法

- 最高位优先MSD法
- 最低位优先LSD法：先对K^0进行桶排序排序，再对K^1进行桶排序，...…，依次类推，直至最后对最次位关键字桶排序完成为止。

- 通常采用低位优先。
- 以下以低位优先为例介绍基数排序

基数排序

- 例 {369，367，167，239，237，138，230，139}低位优先基数排序：
  - 首先按其“个位数” 取值分配到“0, 1, …, 9” 10 个桶，之后按从 0 至 9 的顺序将它们“收集” 在一起；
  - 然后按其“十位数” 取值分配到“0, 1, …, 9” 10个桶，之后再按从 0 至 9 的顺序将它们“收集” 在一起；
  - 最后按其“百位数”重复一遍上述操作。

基数排序的实现

- 采用链表作存储结构，即链式基数排序，具体作法为：

1. 待排序记录以指针相链，构成一个链表；
2. “分配” 时，按当前“关键字位”所取值，将记录分配到不同的 “链队列”（桶）中，每个队列（桶）中记录的 “关键字位” 相同；
3. “收集”时，按当前关键字位取值从小到大将各队列（桶）首尾相链成一个链表；
4. 对每个关键字位均重复 2 和 3 两步。

p→369→367→167→239→237→138→230→139

进行第一次分配

f[0]→230←r[0]

f[7]→367→167→237←r[7]

f[8]→138←r[8]

f[9]→369→239→139←r[9]

进行第一次收集

p→230→367→167→237 →138 →369→239→139

p→230→367→167→237→138→369→239→139

进行第二次分配

f[3]→230→237→138→239→139←r[3]

f[6]→367→167→369←r[6]

进行第二次收集

p→230→237→138→239→139 →367→167→369

p→230→237→138→239→139→367→167→369

进行第三次分配

f[1]→138→139→167←r[1]

f[2]→230→237→239←r[2]

f[3]→367→369←r[3]

进行第三次收集之后便得到记录的有序序列

p→138→139→167→230→237→239→367→369

基数排序

- “分配”和“收集”的实际操作仅为修改链表中的指针和设置队列的头、尾指针；
- 为查找使用，该链表尚需应用算法Arrange 将它调整为有序表。
- 时间复杂度为O(d(n+2rd))。其中：
  - 分配为O(n)，收集为O(rd)(rd为“基”)，d为“分配-收集”的趟数
- 空间复杂度为O(rd)

0	1	2	3	4	5	6	7
369	367	167	239	237	138	230	139

0		1		2		3		4		5		6		7	
369		367		167		239		237		138		230		139	

0		1		2		3		4		5		6		7	
369	1	367		167		239		237		138		230		139	

0		1		2		3		4		5		6		7	
369	1	367	2	167		239		237		138		230			

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	

p

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=r[7]=-1; f[8]=r[8]=-1;

f[9]=r[9]=-1;

p→

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=r[7]=-1; f[8]=r[8]=-1;

f[9]=0;r[9]=0;

p ↘

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=r[7]=-1; f[8]=r[8]=-1;

f[9]=0;r[9]=0;

p ↘

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=r[7]=-1; f[8]=r[8]=-1;

f[9]=0;r[9]=0;

p ↘

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=1; f[8]=r[8]=-1;

f[9]=0;r[9]=0;

p↘

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=1; f[8]=r[8]=-1;

f[9]=0;r[9]=0;

p↘

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=2; f[8]=r[8]=-1;

f[9]=0;r[9]=0;

p↘

0		1		2		3		4		5		6		7	
369	1	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=2; f[8]=r[8]=-1;

f[9]=0;r[9]=0;

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=2; f[8]=r[8]=-1;

f[9]=0;r[9]=3;

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	3	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=2; f[8]=r[8]=-1;

f[9]=0;r[9]=3;

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=2; f[8]=r[8]=-1;

f[9]=0;r[9]=3;

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=r[8]=-1;

f[9]=0;r[9]=3;

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=r[8]=-1;

f[9]=0;r[9]=3;

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=3;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=3;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=3;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=3;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	4	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=3;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=3;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=3;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

f[0]→230←r[0]

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

f[7]→367→167→237←r[7]

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

f[8]→138←r[8]

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

f[9]→369→239→139←r[9]

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

进行第一次收集

p→230

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

进行第一次收集

p→230

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	7	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

进行第一次收集

p→230

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	1	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

进行第一次收集

p→230→367→167→237

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	1	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

进行第一次收集

p→230→367→167→237

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	6	230	1	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

进行第一次收集

p→230→367→167→237 →138

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

进行第一次收集

p→230→367→167→237→138

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第一次分配

f[0]=6; r[0]=6; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=1;r[7]=4; f[8]=5; r[8]=5;

f[9]=0;r[9]=7;

进行第一次收集

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=r[3]=-1; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=6; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=6; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=r[6]=-1; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=6; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=1; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=6; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=1; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=6; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=2; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	1	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=6; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=2; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=4; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=2; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=4; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=2; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→368→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=5; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=2; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	4	239	7	237	5	138	0	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=5; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=2; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	0	239	7	237	5	138	0	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=5; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=0; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	0	239	7	237	5	138	0	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=5; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=0; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	0	239	7	237	5	138	3	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=3; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=0; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	0	239	7	237	5	138	3	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=7; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=0; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

p→230→367→167→237→138→369→239→139

p

0		1		2		3		4		5		6		7	
369	3	367	2	167	0	239	7	237	5	138	3	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=7; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=0; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

进行第二次收集

p ↘

0		1		2		3		4		5		6		7	
369	3	367	2	167	0	239	7	237	5	138	3	230	4	139	-1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=7; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=0; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

进行第二次收集

p→230→237→138→239→139

p ↘

0		1		2		3		4		5		6		7	
369	-1	367	2	167	0	239	7	237	5	138	3	230	4	139	1

进行第二次分配

f[0]=-1; r[0]=-1; f[1]=r[1]=-1; f[2]=r[2]=-1;

f[3]=6; r[3]=7; f[4]=r[4]=-1; f[5]=r[5]=-1;

f[6]=1; r[6]=0; f[7]=-1;r[7]=-1; f[8]=-1; r[8]=-1;

f[9]=-1;r[9]=-1;

进行第二次收集

p→230→237→138→239 →139→367→167→369

- #define MAX_NUM_OF_KEY 8
- #define *radix* 10
- #define MAX_SPACE 1000
- typedef struct {

```
 Keystype keys[MAX_NUM_OF_KEY];
  ……
 int next;
}SLCell;
```

- **typedef struct {**

  SLCell R[MAX_SPACE];

  int keynum;

  int recnum;

  }SLList;

- **typedef int ArrType[RADIX];**

- **void Distribute(SLCell &R,int i,ArrType &f,Arrtype &r,int head)**

```
{   for(j=0;j<radix;j++) f[j]=-1;
    for(p=head;p!=-1;p=R[p].next)
    {   j=ord(R[p].keys[i]);
        if (f[j]==-1)f[j]=p;
        else R[r[j]].next=p;
        r[j]=p;
    }
}
```

- void collect(SLCell &R, int i, ArrType f, ArrType r, int &*head*)

```
{   for(j=0; j<Radix &&f[j]==-1; j++);
    head=f[j];  t=r[j];
    while(j<Radix)
     {   for (++j; j<Radix-1 &&f[j]==-1; j++);
         if(f[j]!=-1)
          {   R[t].next=f[j];t=r[j];  }
     }
     R[t].next=-1;
}
```

- **void RadixSort(SLList &L)**

```
{   for(j=0; j<L.recnum-1; j++) L.R[j].next=j+1;
    L.R[L.recnum-1].next=-1;
    head=0;
    for (i=0; i<L.keynum ; i++);
    {  Distribute (L.R,i,f,r;head); Collect(L.R,i,f,r,head);}
}
```

各种排序方法的综合比较

- 时间复杂度--平均的时间性能
  - O($n$log$n$)--快速排序、堆排序和归并排序
  - O(n^2)--直接插入排序、起泡排序和简单选择排序
  - O(n)—基数排序
- 时间复杂度--当待排记录序列按关键字顺序有序时
  - 直接插入排序和起泡排序能达到O(n)的时间复杂度
  - 快速排序的时间性能蜕化为O(n^2)
- 简单选择排序、堆排序和归并排序的时间性能不随待排序数据序列中关键字的分布而改变。

各种排序方法的综合比较

- 空间性能--排序过程中所需的辅助空间大小
  - 所有的简单排序方法(包括：直接插入、起泡和简单选择) 和堆排序的空间复杂度为O(1)；
  - 快速排序为O(log*n*)，为递归程序执行过程中，栈所需的辅助空间
  - 归并排序所需辅助空间最多，其空间复杂度为 O(*n*);
  - 链式基数排序需附设队列首尾指针，则空间复杂度为 O(*rd*)。