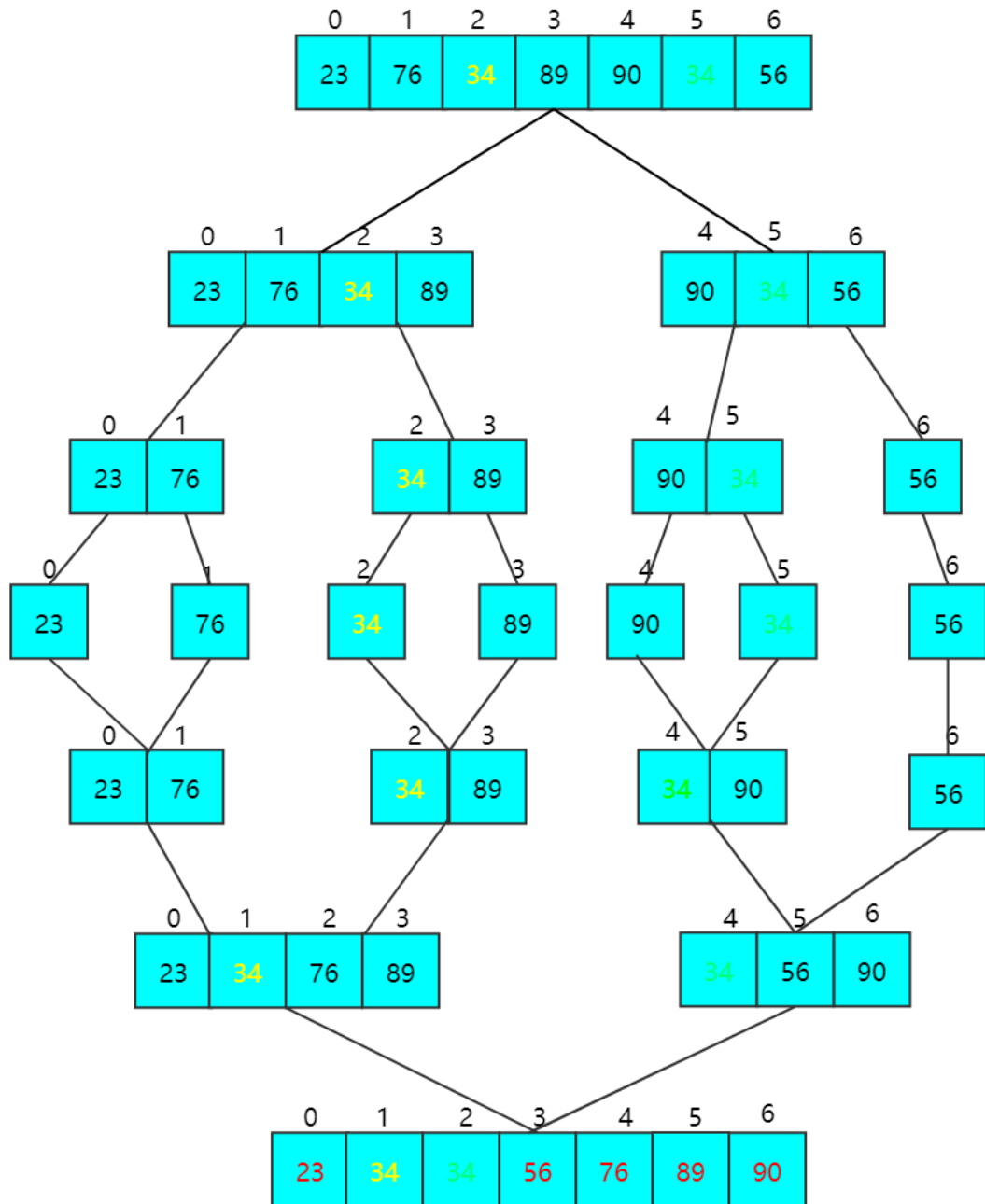


归并排序

排序思想

归并排序就是将两个或两个以上的有序表组合成一个新的有序表

从代码结构来看，归并排序类似树的后序遍历 —— （参考快速排序，类似树的先序遍历）



归并排序算法分析

归并排序的时间复杂度是 $O(n \log n)$ ，由于我采用的方式是下标元素的交换，所以没有用到辅助空间，因此空间复杂度是 $O(1)$ ，同时也是 **稳定排序**，由于它是一种分治思想，所以它的元素 **不是全局有序** 的。

代码实现

```

1  class Solution{
2  public:
3      int counter = 0;
4      int ans[7] = {23,76,34, 89, 90,34,56};
5      void mergeSort(int low, int high) {
6          if(low>=high) return;
7          int mid = (low + high) / 2;
8          mergeSort(low,mid);
9          mergeSort(mid+1,high);
10         mergeTwoList(low,mid,high);
11         return;
12     }
13 private:
14     void mergeTwoList(int low, int mid, int high) {
15         int i = low;
16         int j = mid+1;
17         int tmp[10] = {0};
18         int flag = low;
19         while(i<=mid&&j<=high) {
20             if(ans[i]<ans[j]) {
21                 tmp[flag++] = ans[i++];
22             } else {
23                 tmp[flag++] = ans[j++];
24             }
25         }
26         while(i<=mid) tmp[flag++] = ans[i++];
27         while(j<=high) tmp[flag++] = ans[j++];
28         for(int k = low; k <= high; k++) {
29             ans[k]=tmp[k];
30         }
31
32         return;
33     }
34 };

```

加工后执行的结果

第0轮:	23	76	34	89	90	34	56
第1轮:	23	76	34	89	90	34	56
第2轮:	23	34	76	89	90	34	56
第3轮:	23	34	76	89	34	90	56
第4轮:	23	34	76	89	34	56	90
第5轮:	23	34	34	56	76	89	90

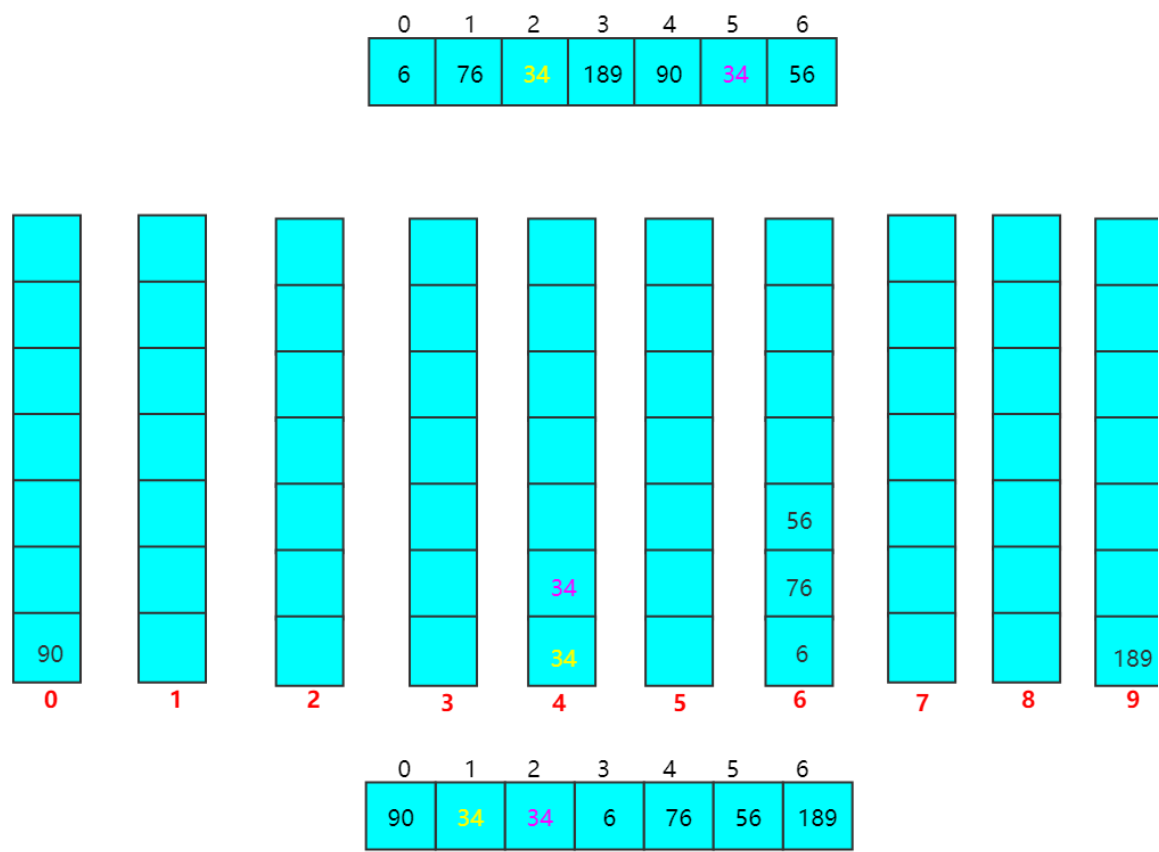
基数排序

排序思想

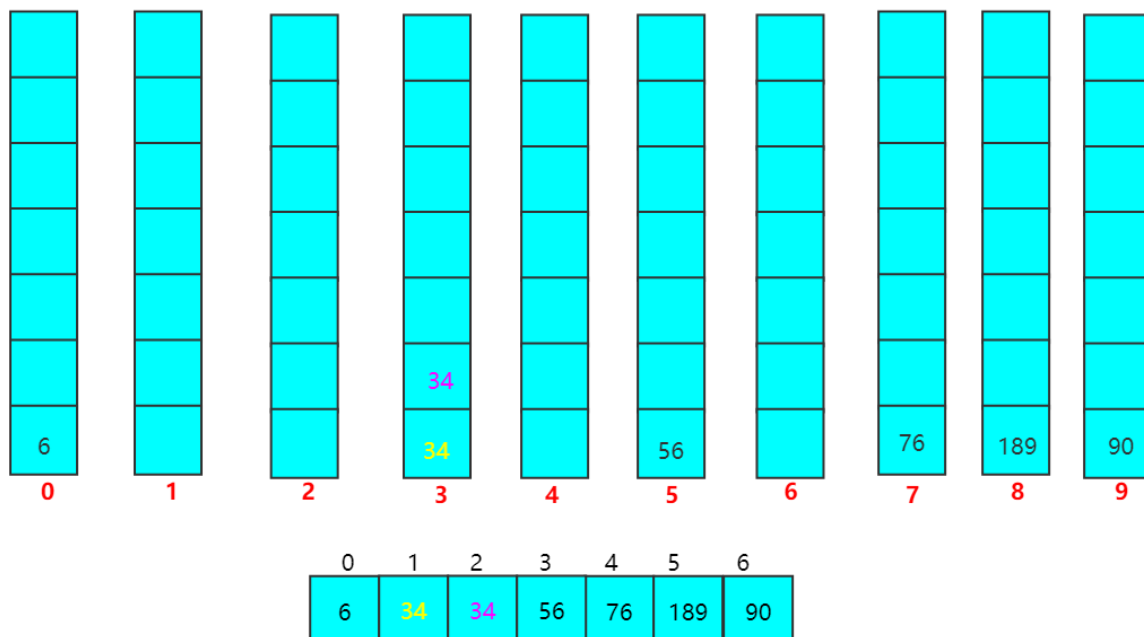
基数排序不基于比较进行排序，而是采用多关键字的排序思想。也就是说基数排序实际上是关于关键字各位的大小排序的。

通过示例解释一下

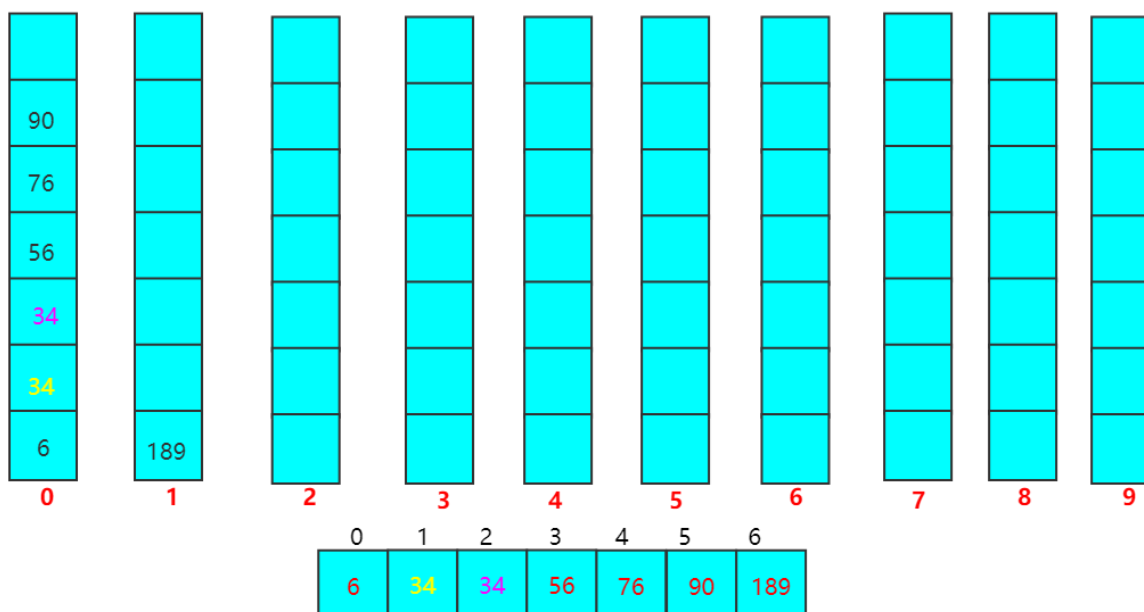
以下 `nums[0,6]` 的个位数依次为 `{6, 6, 4, 9, 0, 4, 6}`，按这个依据从小到大排序，
`{90, 34, 34, 6, 76, 56, 189}`



以上 `nums[0,6]` 的十位数依次为 `{9, 3, 3, 0, 7, 5, 8}`，按这个依据从小到大排序，`{6, 34, 34, 56, 76, 189, 90}`



以上 `nums[0,6]` 的百位数依次为 `{0, 0, 0, 0, 0, 1, 0}`，按这个依据从小到大排序，`{6, 34, 34, 56, 76, 90, 189}`



基数排序算法分析

基数排序的时间复杂度是 $O(dn)$ ，也就是需要精心 d 趟分配，一趟分配和收集需要 $O(n)$ ，空间复杂度是 $O(10n)$ ，同时也是 **稳定排序**，但 **不是全局有序** 的。

代码实现

```
1 class Solution{
2 public:
3     void radixSort(vector<int> nums) {
4         vector<vector<int> > tmp;
5         vector<int> list;
```

```

6      int anchor = 1;
7      int counter = 0;
8      // lgchor + 1趟排序
9      while(anchor) {
10         tmp.clear();
11         // 需要辅助数列
12         for(int j = 0; j < 10; j++) {
13             list.clear();
14             list.push_back(j);
15             tmp.push_back(list);
16         }
17         int i = 0;
18         // 按关键字各个位数排序
19         while(i<7) {
20             int rest = nums[i] % (anchor * 10);
21             int radix = rest / anchor;
22             tmp[radix].push_back(nums[i++]);
23         }
24         // 放在一个列表中
25         nums.clear();
26         for(int j = 0; j < 10; j++) {
27             for(int k = 1; k < tmp[j].size(); k++) {
28                 nums.push_back(tmp[j][k]);
29             }
30             if(tmp[j].size()==7) anchor=0;
31         }
32         anchor *= 10;
33     }
34     return;
35 }
36 };

```

加工后执行的结果

```

第0轮: 90, 34, 34, 6, 76, 56, 189
第1轮: 6, 34, 34, 56, 76, 189, 90
第2轮: 6, 34, 34, 56, 76, 90, 189

```

归并排序测试代码

```

1  #include <stdio.h>
2  #include <vector>
3  using namespace std;
4
5  class Solution{
6  public:
7      int counter = 0;

```

```

8      int ans[7] = {23,76,34, 89, 90,34,56};
9      void mergeSort(int low, int high) {
10         if(low>=high) return;
11         int mid = (low + high) / 2;
12         mergeSort(low,mid);
13         mergeSort(mid+1,high);
14         mergeTwoList(low,mid,high);
15         printf("第%d轮: ", counter++);
16         for(int j = 0; j < 7; j++) {
17             printf(" %d ",ans[j]);
18             if(j!=6) printf(",");
19         }
20         printf("\n");
21         return;
22     }
23 private:
24     void mergeTwoList(int low, int mid, int high) {
25         int i = low;
26         int j = mid+1;
27         int tmp[10] = {0};
28         int flag = low;
29         while(i<=mid&&j<=high) {
30             if(ans[i]<ans[j]) {
31                 tmp[flag++] = ans[i++];
32             } else {
33                 tmp[flag++] = ans[j++];
34             }
35         }
36         while(i<=mid) tmp[flag++] = ans[i++];
37         while(j<=high) tmp[flag++] = ans[j++];
38         for(int k = low; k <= high; k++) {
39             ans[k]=tmp[k];
40         }
41
42         return;
43     }
44 };
45
46 int main() {
47     solution solution;
48     solution.mergeSort(0,6);
49     return 0;
50 }

```

基数排序测试代码

```

1  #include <stdio.h>
2  #include <vector>
3  using namespace std;
4
5  class Solution{

```

```

6 public:
7     void radixSort(vector<int> nums) {
8         vector<vector<int> > tmp;
9         vector<int> list;
10        int anchor = 1;
11        int counter = 0;
12        while(anchor) {
13            tmp.clear();
14            for(int j = 0; j < 10; j++) {
15                list.clear();
16                list.push_back(j);
17                tmp.push_back(list);
18            }
19            int i = 0;
20            // 第一轮排序
21            while(i<7) {
22                int rest = nums[i] % (anchor * 10);
23                int radix = rest / anchor;
24                tmp[radix].push_back(nums[i++]);
25            }
26            // 放在一个列表中
27            nums.clear();
28            for(int j = 0; j < 10; j++) {
29                for(int k = 1; k < tmp[j].size(); k++) {
30                    nums.push_back(tmp[j][k]);
31                }
32                if(tmp[j].size()==7) anchor=0;
33            }
34            anchor *= 10;
35            printf("第%d轮: ", counter++);
36            for(int j = 0; j < nums.size(); j++) {
37                printf("%d",nums[j]);
38                if(j!=nums.size()-1) printf(",");
39            }
40            printf("\n");
41        }
42        return;
43    }
44 };
45
46 int main() {
47     vector<int> v;
48     v.push_back(6);
49     v.push_back(76);
50     v.push_back(34);
51     v.push_back(189);
52     v.push_back(90);
53     v.push_back(34);
54     v.push_back(56);
55     solution solution;
56     solution.radixSort(v);
57     return 0;

```

