# 想法

一种基于单基准元素的选择排序—双基准元素的考虑:如果对数组A进行选择分区时,选择两个元素p1,p2,将数组分为x<p1,p1<=x<p2,x>=p2三个部分,然后使用相同的方法继续对三个区进行选择.

# 实例

10 7 6 8 3 9 12 11 7 2 4 5,选取基准元素10 5,size=12

初始化i=0,j=11

**第一次**:i=first23(i,+)=0,j=first12(j,-)=11 判断i<j交换,2 7 6 8 3 9 12 11 7 10 4 5,i++,这时j所指为3类元素

接着,k=first2(k,-)=11,判断k>j,交换,2 7 6 8 3 9 12 11 7 5 4 10 k--,j--

**第二次**:i=first3(i,+)=1,j=first1(j,-)=4,判断i<j交换,2 3 6 8 7 9 12 11 7 10 4 5,i++,j所指为第三类元素

接着,k=first2(k,-)=10,判断k>j交换,2 3 6 8 4 9 12 11 7 10 7 5,k--,j—

**第三次**:i=first3(i,+)=3,j=first1(j,-)=1,判断i>j,于是j以及j左边都是1类元素,1类分类完毕,l’=j;进入2分区模式,分区元素以2类元素为标准

# 算法

数组a

partition3(l,r):

i=l,j=r,l=r

while true:

i=first3(I,+),j=first1(j,-)

if(i < j):

swap(a[i],a[j])

k=first2(k,-);

if(k > j)

swap(a[k],a[j])

k--,j--,i++

else

j--,i++//进入下一次循环

else

l’=j

r’=partition2(i,k,pvoit2)

break;

return l’,r’

注记:在寻找第二类元素的时候,可能先出现第一类元素,此时都不管

# 算法2

经过慎重的考虑,我重新构造了算法

对于数组a,初始化下标I,j指向首末,从i开始找到第一个 2-3 类元素,从j开始找到第一个1-2类元素, 有4种情况 3---1直接交换 2---1交换后无需对2类元素再查找 3—2 类似第二种情况 2----2 寻找第一个1/3类元素,与之交换即可

2—1 \* \* \* i - - - j \* \* \* \* 先寻找第一个3类元素,如果找到j+1,则3类元素设置界限

讨论一点,pos1和pos2设置的情况下对循环有什么影响?

pos1设置,pos2未设置,在某次循环中,要求i<j

pos1 <i,pos2=-10, j之前找到的都是2类元素,现在只剩下2—2 3—2两种类型

i=l,j=r;

iNeed=true,jNeed=true

while true://在每次循环时,总保证,i之前的元素是一类元素,j之后的元素是3类元素,I,i+1…j-1,j都是不确定的

//这个条件还可以写成while not set pos1 and pos2

if(i==j)//一个元素

判断元素分类,设置pos1,pos2

break

if(i>j)//一定有i=j+1,这是上一次设置的过程中溢出的,上一次只能是i++,j--.这种情况没有2类元素

设置pos1,pos2

break

i=first23(i) ;//iNeed=true时才进行查找,jNeed亦然

iNeed=true,jNeed=true //重置需求状态

if i=j+1//j以及j之前全部为1类元素

置pos1,pos2

break

j=first12(j);

if j==i-1//i以及i之后全为3类元素

置pos1,pos2

break

//I,j全部查找成功且i<=j

b1=(a[i]<p2),b2=(a[j]>=p1) //分别指示I,j是否为2类元素

if(b1 && b2):

iDot=first1(iDot),//给定范围查找,从左查找时,1类元素的边界提前存在一个变量中

if(iDot==j)//没有1类元素

pos1=i-1;//1类元素边界

i+1开始寻找3类元素,与就交换---\*///这是一个子过程

设置pos2

break

else

swap(iDot,i)

i++

jDot=first3(j)

if(jDot==i)

pos2=j+1//3类元素边界

从j开始向下设置一类元素//子过程 ---不一定需要子过程,只需要在pos1/2的设置中,如果pos1=-10表示未设置,那么重复这个过程就是可行的,所以无需子过程

设置pos1

break

else

swap(jDot,j)

j--

else

swap(I,j);

if(!b1 && !b2) :

i++,j--

else if(b1 && !b2):

i++,jNeed=false

else

j--,iNeed=false

return pos1,pos2

# java实现

# 性能分析和对比