习题17

* 1. 插入排序

1.1.1.算法描述

这是一种比较常规的排序方式，是一个稳定性的排序方式，可改进为希尔排序。

1.1.2.伪代码

insertSort()

//插入排序

//输入：数组a

//输出：排序后数组

for i from 1 to len

for j from i to 1

if (a[j]>=a[j-1]) break

交换a[j]和a[j-1]

1.1.3.算法实现（C语言）

void insertion\_sort(int a[],int len){

for(int i=1;i<len;i++){

for(int j=i;j>0&&a[j]<a[j-1];j--){

int tmp=a[j];

a[j]=a[j-1];

a[j-1]=tmp;

}

}

}

实际上我们可以对交换进行优化，不必每次进行右移动，优化后我们的C程序变为

void insertion\_sort\_update(int a[],int len){

for(int i=1;i<len;i++){

int j,tmp=a[i];

for(j=i;j>0&&tmp<a[j-1];j--){

a[j]=a[j-1];

}

a[j]=tmp;

}

}

* 1. **将插入排序优化为希尔排序**

1.2.1.算法描述

我们的希尔排序，每次将长度除以二进行分组，然后进行插入排序。这样我们的算法时间复杂度为O(n^1.3)，空间复杂度为O(1)。

1.2.2.伪代码

shell\_sort(int\* a,int len)

//希尔排序

//输入：数组a

//输出：排序后的数组

for gap from len>>2 to 1

gap/=2

进行插入排序算法

1.2.3.算法实现

void shell\_sort(int a[],int len){

for(int gap=len>>2;gap>0;gap/=2){

for(int i=gap;i<len;i++){

int j,tmp=a[i];

for(j=i;j>0&&tmp<a[j-gap];j-=gap){

a[j]=a[j-gap];

}

a[j]=tmp;

}

}

}

* 1. **试验小结**

关于插入排序和希尔排序，总体来讲比冒泡排序和选择排序好，插入排序的算法复杂度为O(n^2)。但是如果原序列基本有序，那么进行插入排序，依旧可以效率比冒泡和选择来的快。

关于冒泡排序的优化。我们建立一个标识符flag，如果一轮下来没有任何交换操作，那么默认进行return。表示序列已经整齐。

插入排序，我们可以进行2次优化，1是进行交换的优化，比较后直接后移，而不是进行每次交换。2是进行划分思想，划分为希尔，算法复杂度变为O(n^1.3)。