### 时空矛盾

1. 以空间换时间的例子：

①桶排序

在对n个数进行桶排序时，首先创建m个桶，遍历n个数将所有的数放到这m个桶中，然后分别对桶内的数进行排序，最后依次输出m个桶内的n个数。 桶排序的时间复杂度是O(n+c),其中c=n\*(logn-logm)，空间复杂度O(n+m)。

②交换两个数

交换两个数的时候往往会引入第三个变量。如：要交换a,b。往往会：

int t=a;

a=b;

b=t;

1. 以时间换空间

①交换两个数

同样是交换两个数，我们也可以不引入中间变量，而进行多次运算得到，如：

b=a+b;

a=b-a;

b=b-a;

### 最大独立集问题

100年计算图的规模是n=50 n=50时，大概需要100年才能完成。

最大独立集的时间复杂度是O().

一年按365天算，一年有60\*60\*24\*365=31536000秒，计算机10亿次每秒，则计算机一百年工作3.1536\*次。

经过计算，当n=50时，需要2,814,749,767,106,560,000，即2.81474976710656\*秒，当n=51时，需要5.856931315393330048\*秒。

所以当计算机计算100年时，可以计算的规模是n=50.

当n=50时，需要的时间约为100年。

### 线性排序

O(1)是最低的空间复杂度，也就是耗空间与输入数据大小无关，无论输入数据增大多少倍，耗空间都不变。

首先定义一个下标范围比较大的数组a[k]（k是由自己进行设定，与输入数据n无关，所以空间复杂度是O(1) )。 对输入的n个数字，将对应数组下标的数组元素加一。最后，遍历一边所有这些数字就可得到0~k每个数字的个数，然后再顺序遍历数组，若a[x] = y，则输出y个x，依次将n个数输出，最后可得结果。第一次遍历是O(n)，第二次遍历是O(1)。时间复杂度为O(n)，而空间复杂度为O(1)

### 计数排序

定义一个长度为(b-a+1)的数组，那么a到b中的元素x所对应的数组下标为x-a，按照上述顺序将[a,b]范围内的数存入数组，每存一次对应数组的值加1。最后遍历数组，输出数组元素的下标值+a，元素的值是多少就输出多少次。 程序变化： 1、节省了统计元素的数组所占用的空间 2、要统计的数值转变为统计数组对应的下标值

### 改进插入排序

折半插入排序只减少了关键字间的比较次数，而没有进行优化移动次数，所以折半插入排序的时间复杂度还是O()。

伪代码如下图：

