位图排序

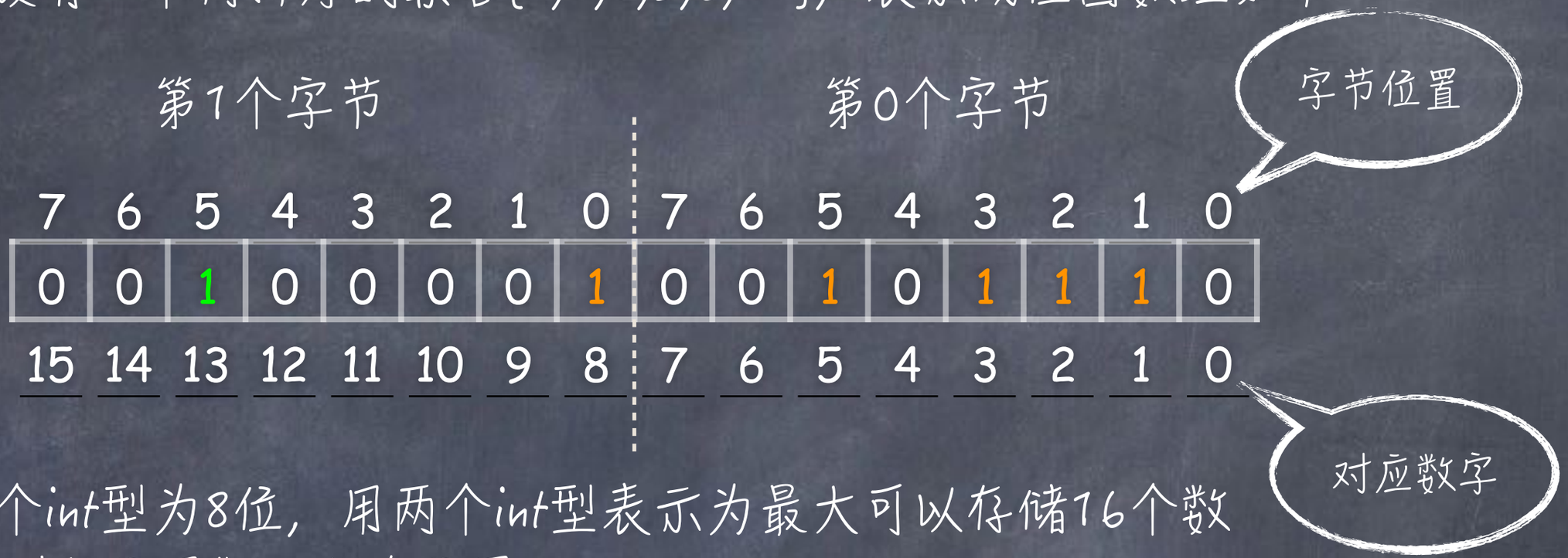
问题描述：假设有1kw个互不相同的数字，每个数字小于<10^7，那么，如何快速的进行排序，在内存空间为1M的情况下。

问题分析：1．输入：一个至多包含1千万个非负整数的文件
2．特征：①每个数都是小于10000000的非负整数；
②没有重复的数字；
③数据之间不存在关联关系。
3．约束：①最多1MB的内存空间可用；
②磁盘空间充足；
③运行时间最多几分钟，最好是线性时间。
4．输出：按升序排列的整数序列。

位图思想：1．根据待排序集合中最大的数，开辟一个位数组，用来表示待排序集合中的整数
2．待排序集合中的数字在位数组中的对应位置置1，其他的置0
3．遍历位图数组为1的数输出

难点实现：

示例：假设有一个待排序的集合{1,2,3,5,8,13}，表示成位图数组如下



说明：

1. 假设一个int型为8位，用两个int型表示为最大可以存储16个数
2. 以13举例在位图数组上的位置
    1) 13/8=1，表示13在第1个字节上，也可以表示为位运算13>>3
    2) 13%8=5，表示13在第5个位置上，也可以表示为位运算13&0b111
3. 设置该位置为1：
    1) 设置一个为00000001的新字节，将其左移5个位置：(1<<(13&0b111))
    2) 在第1个字节上“或”上新字节：bitmap[13>>3] |= (1<<(13&0b111))
    3) 00000001 | 00100000 = 00100001 为保证不影响其它位
4. 检查一个数在它的位置是否为1：
    1) 在第1个字节上“与”上新字节：bitmap[13>>3] & (1<<(13&0b111))

C代码实现：

```
#include <stdio.h>
#define MAX 10000000
#define MASK 0x1F //32的二进制表示
#define DIGITS 32 //设置为32位
#define SHIFT 5   //由于32=2^5

int bitmap[1+MAX/DIGITS];

void setbit(int n)    //将逻辑位置为n的二进制位置为1
{
    bitmap[n>>SHIFT] |= (1<<(n&MASK)); //n>>SHIFT右移5位相当于除以32求算字节位置，n&MASK相当于对32取余即求位位置，
}     //然后将1左移的结果与当前数组元素进行或操作，相当于将逻辑位置为n的二进制位置1.

void initbit(int n)
{
    bitmap[n>>SHIFT] &= ~(1<<(n&MASK)); //将逻辑位置为n的二进制位置0，原理同set操作
}

int test(int n)
{
    return bitmap[n>>SHIFT] & (1<<(n&MASK));  //测试逻辑位置为n的二进制位是否为1
}
```

C代码实现：

```
int main(int argc, char *argv[])
{
    int i,n;
    for(i=0;i<MAX;i++)
    {
        initbit(i);
    }
    while(scanf("%d",&n)!=EOF)
    {
        setbit(n);
    }
    for(i=0;i<MAX;i++)
    {
        if(test(i))
            printf("%d ",i);
    }
    return 0;
}
```

算法的时间复杂度：

1．时间复杂度：可以看到算法只需进行线性的扫描即可，故时间复杂度为O(n)

2．空间复杂度：由于可以采用“位”来表示一个数，故大大的降低了空间占用

位图排序的拓展：