俗话说：时间像海绵里的水，挤一挤总会有的。从数据存储的角度看，存储空间也是如此。在大数据时代，十亿百亿级别的业务数据规模也不再鲜见，存储优化的脚步也从未停歇。

假设在某种业务场景下，数据表内有若干条记录，每条记录的内容是N个uin32（无符号4字节）整数（**N <= 1000**），每个整数描述了一个特定的业务字段，且每个字段都有自己特定的值域范围。

 一种存储优化的思路是：对单条记录进行编码压缩。已知业务字段的值域范围，则可以利用值域范围的最大值对应的bit位宽信息（例如字段值域空间最大值是7，则对应存储位宽是3个bit），实际用位宽对应的bit位数的数据空间代替原始类型长度（4字节）存储空间，即可完成数据存储目标。

基于以上思路，压缩存储的需求目标描述如下：

1. 将N个uint32整数，按照值域空间内最大值的bit位宽，压缩存储到M（M<=N）个uint32整数组成的数组中。

2. 基于数据读写原子性考虑，每个压缩前的整数，存储位置一定在压缩后数组内的某个元素内，不可以跨两个相邻的数组元素

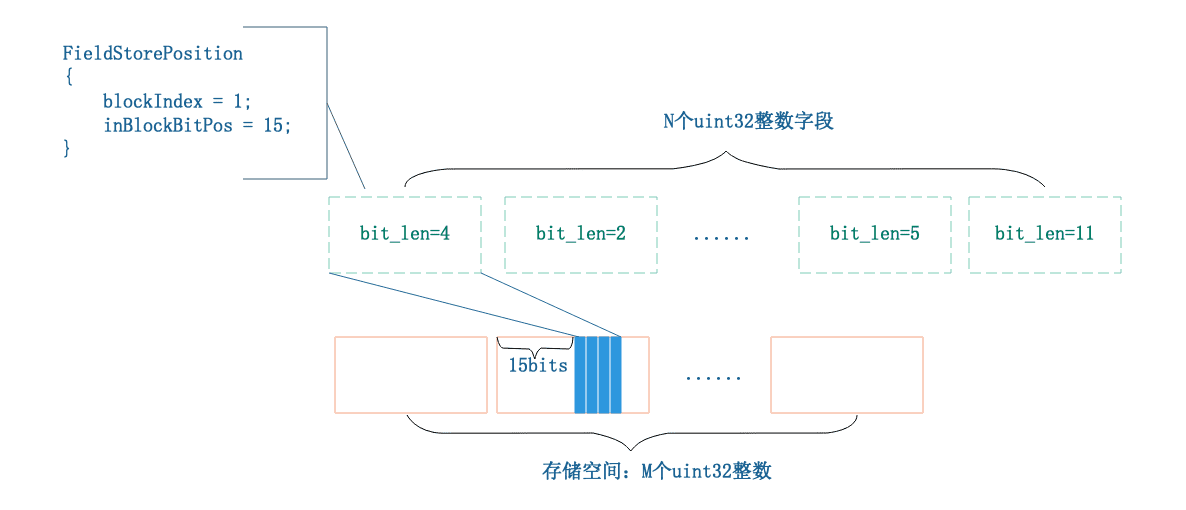
**编码存储示例：**

假设由6个字段组成的单条记录，每个字段的位宽信息依次为**{4, 2, 31, 15, 5, 11}**

编码后的位宽分布为**{31},  {15, 4, 11, 2},  {5}**

其中第一个字段**（bitLen=4）**的字段所在位置是：**block下标=1，block内bit下标=15**，

如下图所示：



在已知每个字段bit位宽信息的前提下，请编码实现

1. 设计算法给出对N个字段的最优（M值最小）压缩存储方案，压缩方案中需要给出每个字段在压缩数组中的存储位置信息（实际存储的整数下标，以及字段在目标整数内存储的bit起始位置）

2. 基于压缩存储方案描述信息（每个字段的存储位置信息），实现对每个字段的存取接口（set/get）

**编译器版本:** gcc 4.8.4

请使用标准输入输出(stdin，stdout) ；请把所有程序写在一个文件里，勿使用已禁用图形、文件、网络、系统相关的头文件和操作，如sys/stat.h , unistd.h , curl/curl.h , process.h

**时间限制:** 3S (C/C++以外的语言为: 5 S)   **内存限制:** 94M (C/C++以外的语言为: 606 M)

**输入:**

输入解析： 第一行：依次给出每个字段的bit位宽，以","分隔 第二行：依次给出要设置的字段信息（由字段下标和字段值组成，下标和字段值以":"分隔）。字段信息之间以";"分隔 第三行：编码压缩存储后的数组元素个数

**输出:**

输出解析： 单个bool值描述过程运行是否符合预期 PASS：通过 FAIL：不符合预期

**输入范例:**

输入范例： 第一行：3,7,9,11,31,2,5,1 第二行：0:4;1:9;2:11;3:15 第三行：3

**输出范例:**

PASS 或 FAIL

**消息同步时间端合并**

**题目描述**

现在很多聊天软件为了方便用户多设备查看历史聊天记录，都提供了消息同步功能，即可以帮用户将A设备上的聊天记录同步到B机器，但是可能B机器已经同步过部分聊天记录，如果再次同步，会造成流量，服务器资源浪费，为了解决这个问题客户端机器上存储了已经同步过的时间段集合（由于数据不可能一次性全部拉回来，就造成了本地同步区间段非连续），需要过滤掉这些区间，筛选出需要同步的时间段，向服务器拉取数据补齐，并将本次补齐的时间段和本地时间段Merge，并保障有序。下面针对同步后合并时间段逻辑进行实现。

**输入描述**

时间段为左闭右开区间，例如：[5-60)

输入1：本次同步的时间段

输入2：本地已经同步过的时间段（有序，无交叉）

**输出描述**

合并后的时间段（有序，无交叉）

**示例**

 输入1: [5-60)， [75-100)，[120-160)  输入2: [70-90) 输出：[5-60)， [70-100)，[120-160)

 输入1: [5-60)， [75-100)，[120-160) 输入2: [70-120)  输出：[5-60)， [70-160)

**编译器版本:** gcc 4.8.4

请使用标准输入输出(stdin，stdout) ；请把所有程序写在一个文件里，勿使用已禁用图形、文件、网络、系统相关的头文件和操作，如sys/stat.h , unistd.h , curl/curl.h , process.h

**时间限制:** 1S (C/C++以外的语言为: 3 S)   **内存限制:** 64M (C/C++以外的语言为: 576 M)

**输入:**

输入数据包含两段，;分割： 第一段：本地已同步的时间段,要求有序，前后不交叉，格式为1-100,200-300 第二段：本次同步过的时间段，格式为1-100

**输出:**

合并后的时间段，保障有序，无交叉

**输入范例:**

例如： 5-60, 75-100,120-160;70-90

**输出范例:**

例如： 5-60,70-100,120-160