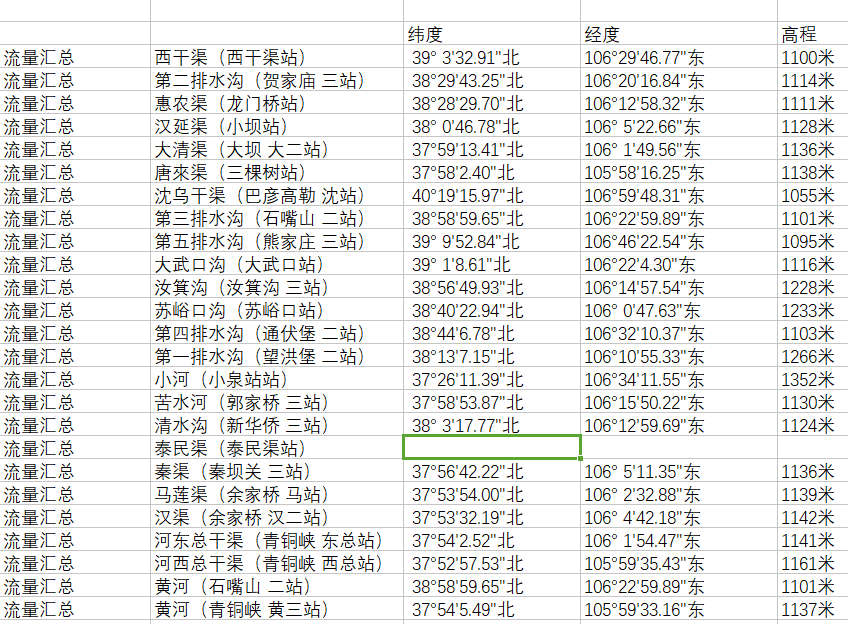
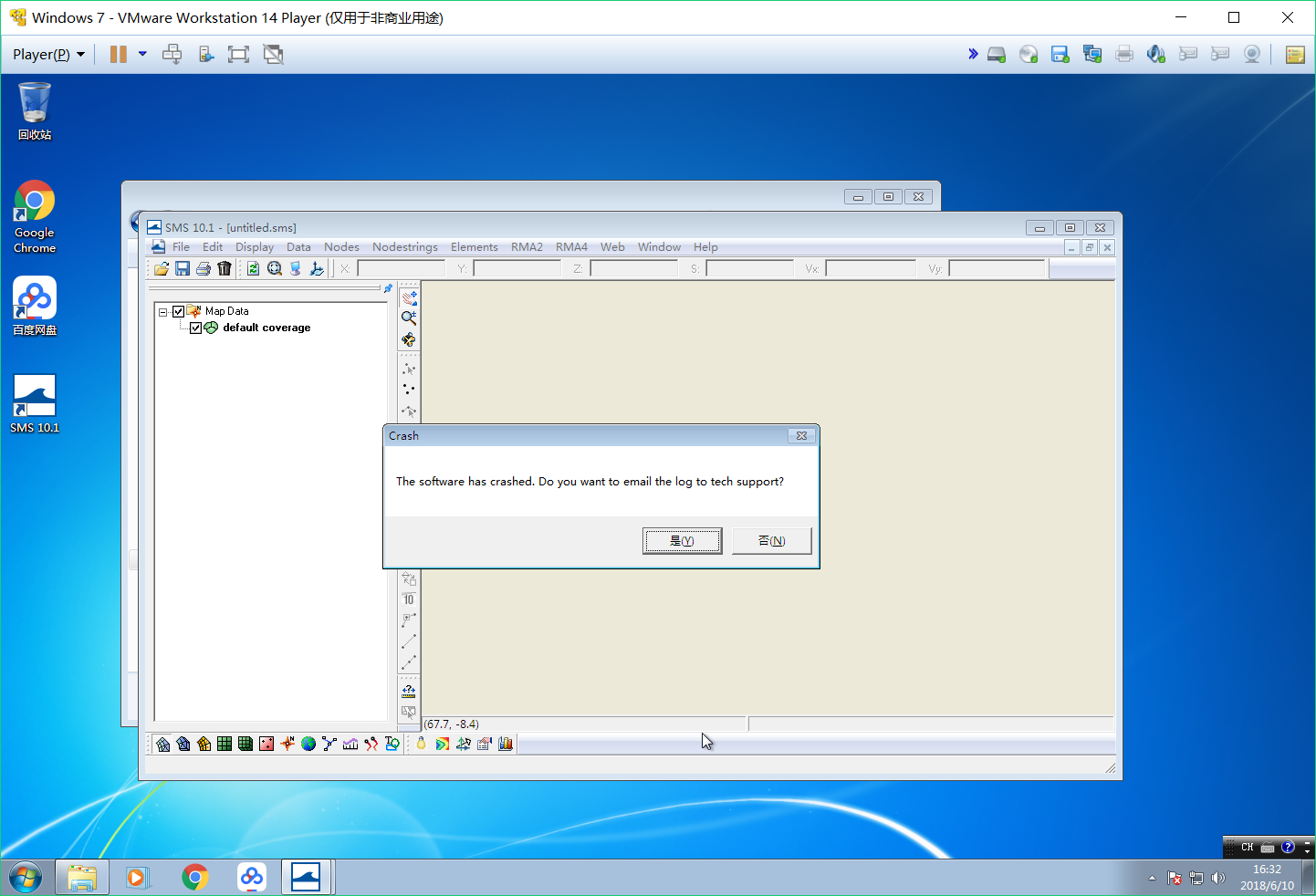
**周报**

1. 统计黄河流域降雨及流量信息，包括经纬度坐标及站点高程，以下是统计信息的部分结果：



各个站点的经纬度信息取自google map。

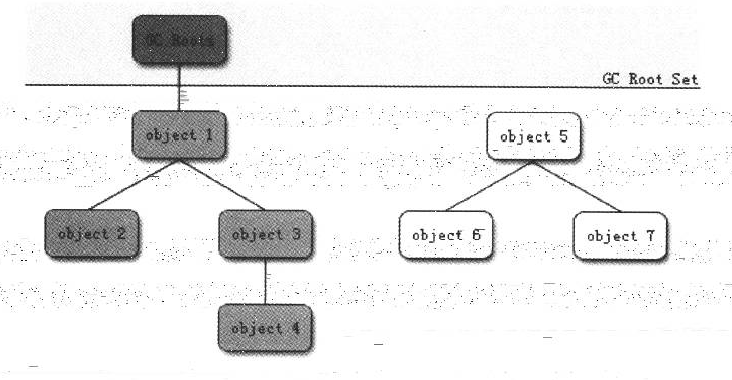
1. 学习使用SMS软件，但多次尝试之后，仍报错：



1. 深入理解JVM：

JVM的自动内存管理机制：主要分为运行时数据区域和非运行时数据区域；日常开发中需要关注或者优化的多数是运行时数据区域，该区域的分区划分为：程序计数器、Java虚拟机栈、本地方法栈、Java堆、方法区、运行时常量池、直接内存等。各个分区在不同的程序执行阶段被使用，常见的OOM主要包括：Java堆溢出、虚拟机栈和本地方法栈溢出、方法去与常量池溢出、本机直接内存溢出、其中直接内存是不在JVM设置的常规堆内存之中的，所以直接内存的溢出是较为难发现的内存溢出情况。

GC：回收对象内存的前提是判断内存中的对象是否可以被回收，常见的判断对象是“存活”还是“死亡”的算法有两个：引用计数算法和可达性算法。引用计数算法实现简单，判定效率也很高，但是它的一个主要的弊端就在于它很难解决对象之间相互循环引用的问题；而可达性算法是目前主流的JVM中广泛使用的对象判定算法，它的算法思想是通过一系列的称为“GC Roots”的对象作为起始点，从这些节点开始向下搜索，搜索所走过的路称为引用链，当一个对象到GC Roots没有任何引用链相连时（即不可达），则证明此对象是不可用的。如下图所示：



图中灰色填充的对象即表示可达，处于存活状态，而右侧的对象因为没有任何一种途径与GC Roots相连，所以是不可达的，这样的对象会在GC发生时被自动回收掉。

常见的垃圾回收算法主要有四种：标记-清除算法、复制算法、标记-整理算法、分代收集算法。第一个是最基础的收集算法，主要不足有两点：一是效率问题；另一个是空间问题；复制算法解决了效率问题，但是这种算法的代价是将内存缩小为原来的一半，未免有点太高了。

垃圾收集器：serial收集器、parnew收集器、g1收集器、cms收集器等，不同的收集器对应于不同的现实场景。