程序说明

关于「质数」的要求

程序使用 Miller-Rabin 素性测试算法判断一个 BigInteger 对象是否为质数。既然程序使用的是 Miller-Rabin 算法，那就肯定有 M-R 算法的特点：

> Miller-Rabin 算法是一个 RP 算法。RP 是时间复杂度的一种，主要针对判定性问题。一个算法是 RP 算法表明它可以在多项式的时间里完成，对于答案为否定的情形能够准确做出判断，但同时它也有可能把对的判成错的（错误概率不能超过 1/2）。RP 算法是基于随机化的，因此多次运行该算法可以降低错误率。
>
> 算法有 $(\frac{1}{4})^K$ 的概率错误，本程序默认选取 K=4，故程序给出的素数中，有 0.00390625 的概率为合数。当然本程序中 K 的值是可以修改的，最大值为 111。当 K 的值取 111 的时候，错误率为：$1.484 * 10^{-67}$。

虽然目前有多项式的、确定的、无需其它条件的素性判断算法（AKS 算法）。但是该算法的复杂度为 O(n!)，时间复杂度过高，不适合用于判断大素数。

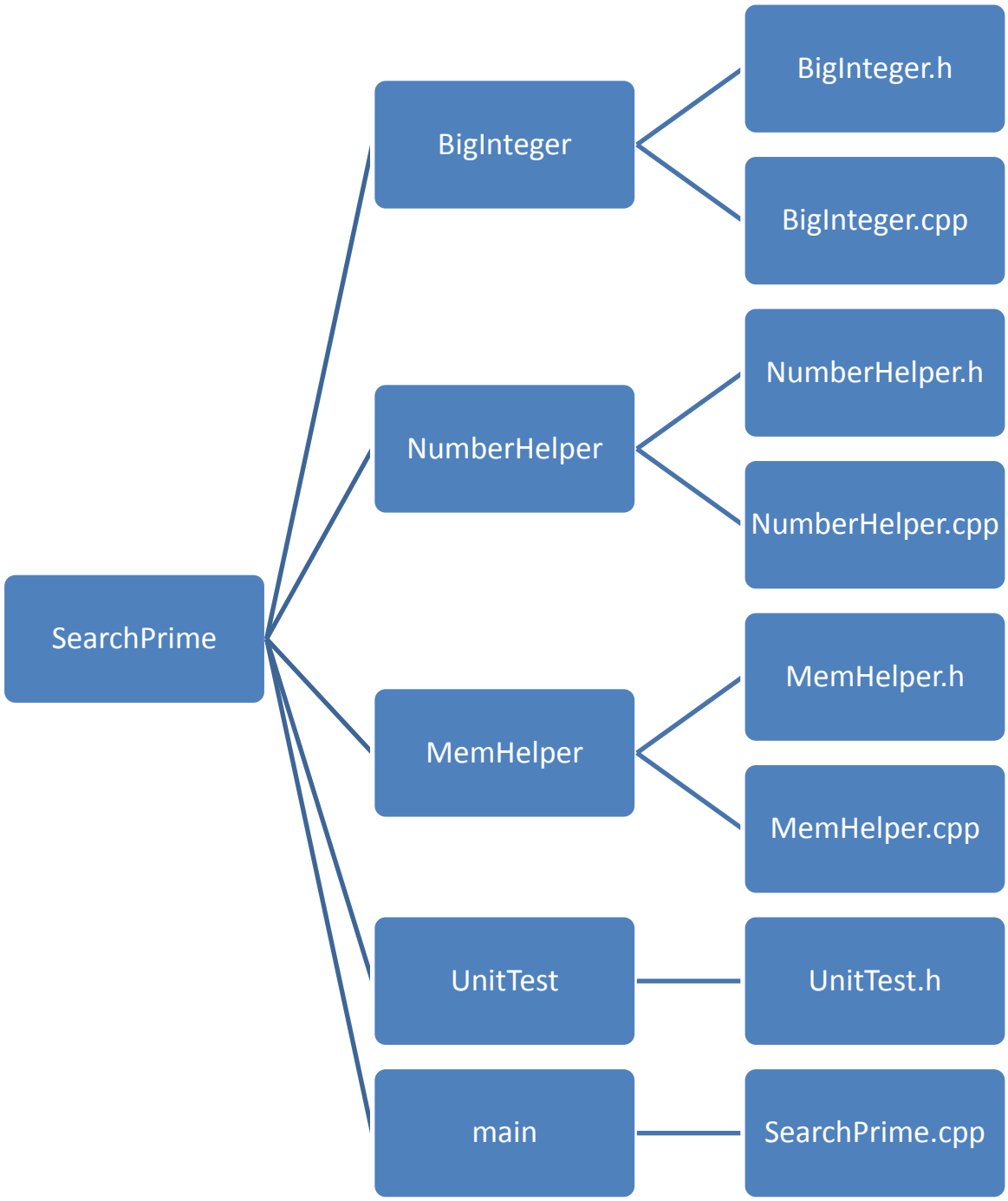
综上所述，选择 Miller-Rabin 素性测试算法最为合适。

关于「尽可能大」的要求

为了实现题目「编程计算一个尽可能大的素数」中「尽可能大」的要求，我用 C++ 实现了一个 BigInteger 类，最值为 $2^{2^{31}-1}$ 即 $2^{2147483647}$，最小的数为 $-2^{2147483647}$。虽然 C\C++中有扩展的大整数库/类，但是我想题目的作者本意应该不是直接使用这种别人做好的东西来完成，所以才自己造轮子，构造了一个 BigInteger 类。

程序实现

程序的结构图：



其核心为 BigInteger 的设计和实现。

1/13/2014

BigInteger 有以下几个难点：

1. 乘法
2. 除法
3. 字符串转为大整数
4. 大整数转为字符串

在实现上面的难点时，借鉴了 TAOCP 中的部分算法，Karatsuba 算法和 RSA 加密体系中的部分算法。