**实验4 串模式匹配**

1. **实验要求**

文本串T的长度为n，对应的模式串P的长度为m，字符串均是随机生成的字符 (A-F,共6种不同字符)。(n,m)共取五组数据: (2^5,2), (2^8,3), (2^11,4), (2^14,5), (2^17,6)。

算法： Rabin-Karp算法； KMP算法； Boyer-Moore-Horspool算法。

1. **实验环境**

编译环境：Windows 10

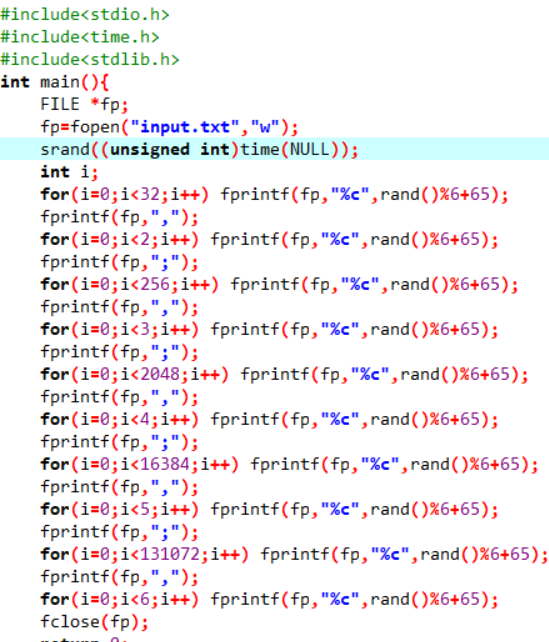
机器内存：8G

时钟主频：28GHz

1. **实验过程**

首先生成输入数据：

根据实验要求，采用srand生成的随机数生成随机字符串，共5组，文本串和模式串的长度分别为(2^5,2), (2^8,3), (2^11,4), (2^14,5), (2^17,6)。代码如下图：

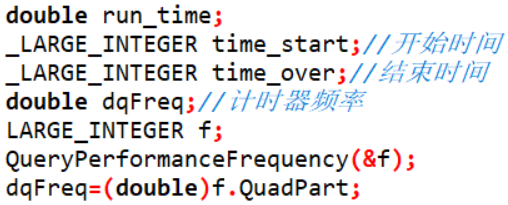


代码中使用rand()%6+65这个表达式是因为大写字母A的ASCII码值为65，F的ASCII码为70，这个表达式随机生成的范围就为65-70之间。每个字符串之间用标点符号逗号或分号分开。由代码可见将生成的结果写入了input/input.txt中。

接着编写实验算法，使用随机生成的数据进行测试，将实验结果和测量时间写入相应文件中。

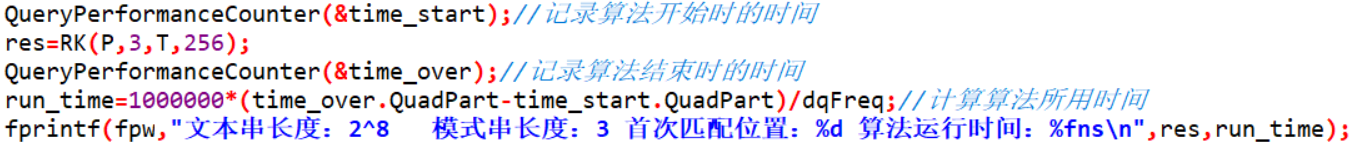
进行重复实验，统计结果，作图表分析。

1. **实验关键代码截图**



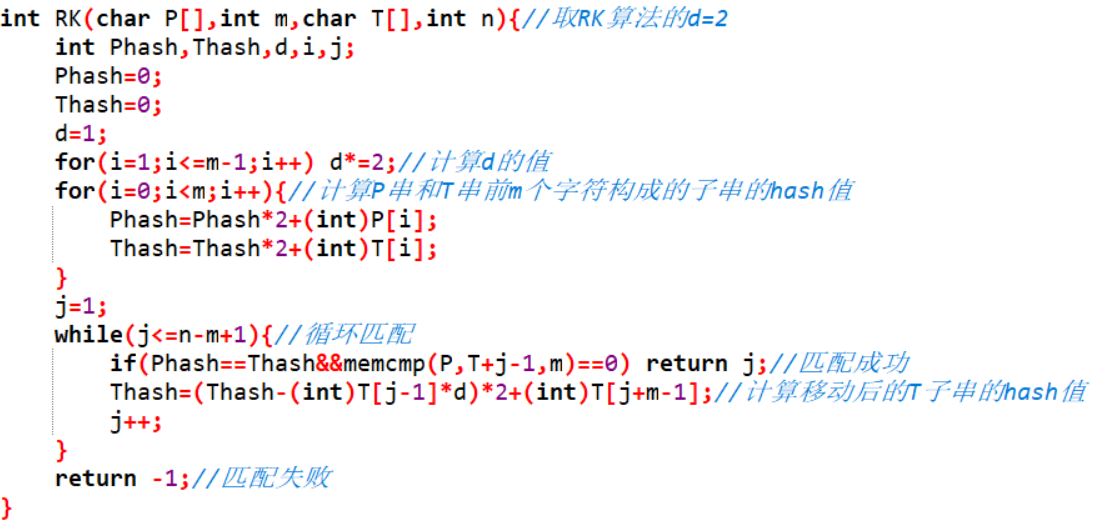
此部分用于算法的计时，可以精确到微妙。需要包括头文件windows.h。

用法示例：



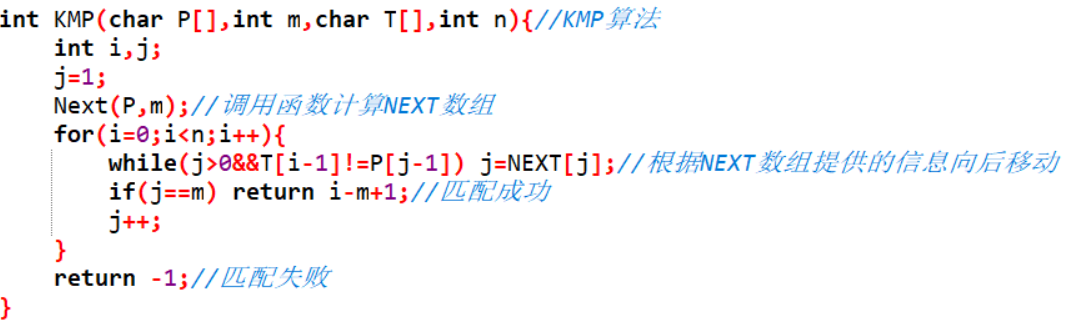
关键算法部分：

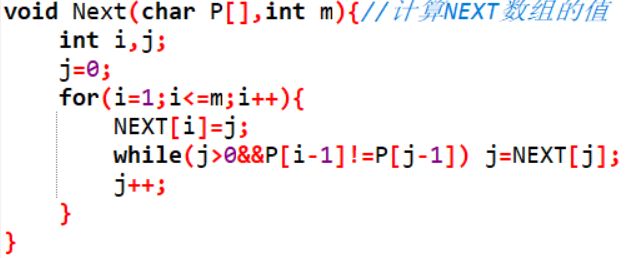
Rabin-Karp算法：



RK算法需要优秀的hash函数，此处采用PPT上的hash函数，并取d=2。算法中根据P和T数组下标起始位置为0进行了微调。

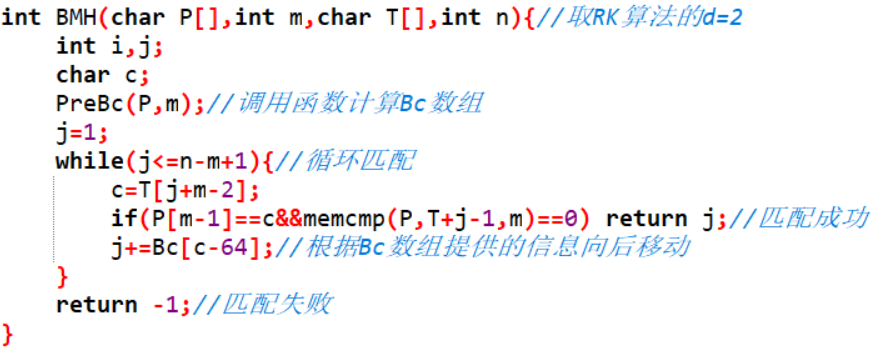
KMP算法：

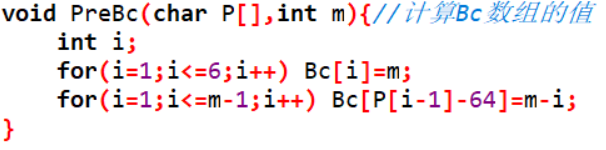




上图分别为KMP算法部分和预处理NEXT数组的函数Next()。其中NEXT数组下标是从1开始的，而P和T数组是从0开始的。

Boyer-Moore-Horspool算法：





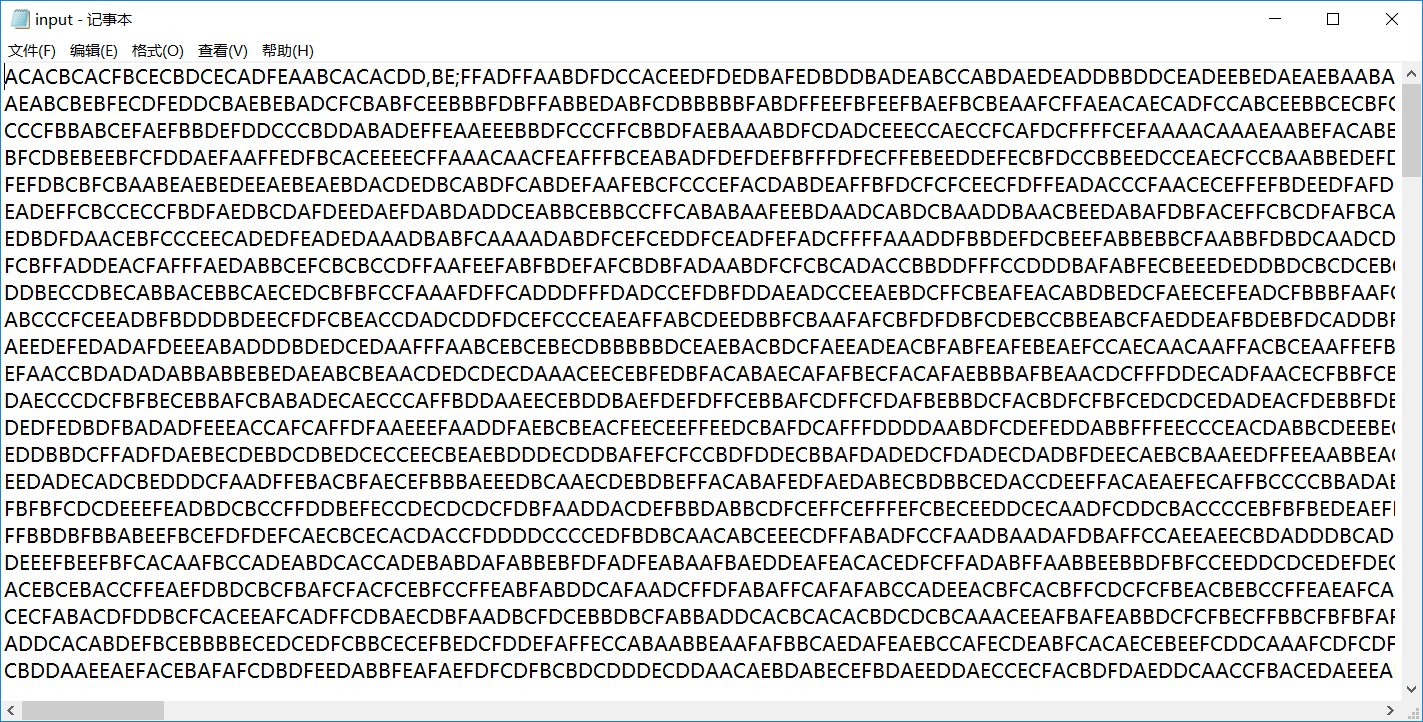
上图分别为Boyer-Moore-Horspool算法和预处理Bc数组的函数PreBc()。其中Bc数组下标是从1开始的。

1. **实验结果、分析（结合相关数据图表分析）**

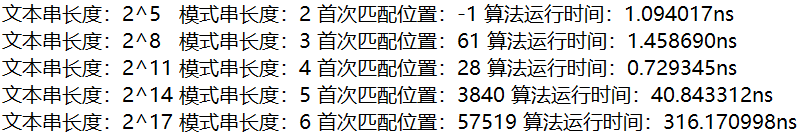
由于算法运行时间会受匹配成功与否的影响，故应进行多次实验取平均时间。本次实验中随机生成5次输入数据，对每种算法的每种规模的数据对应的测量时间取平均值。

5次输入数据和运行结果：

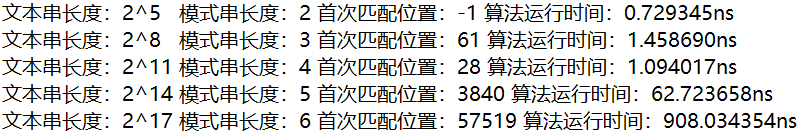
Input1：



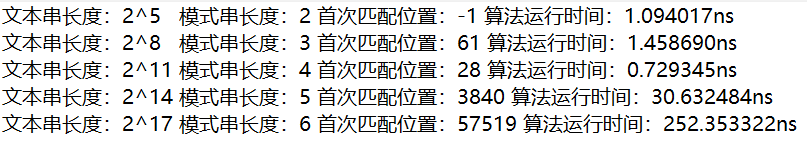
RK1：



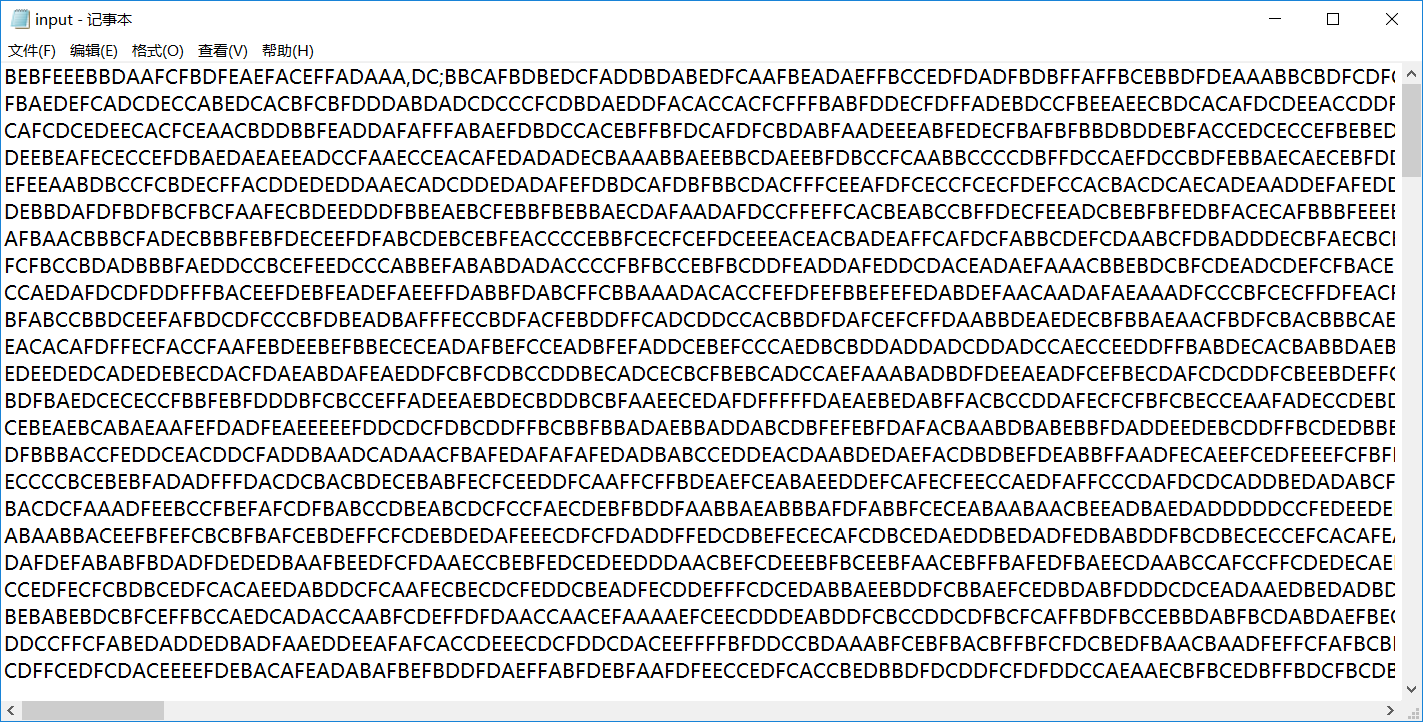
KMP1：



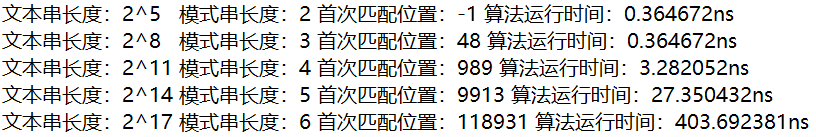
BMH1：



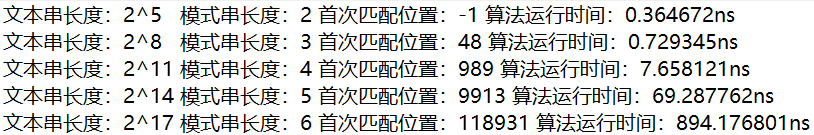
Input2：



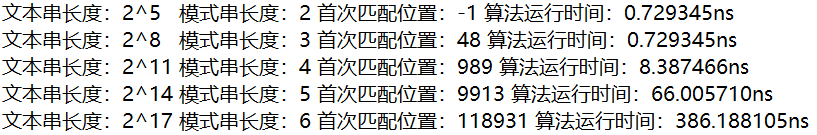
RK2：



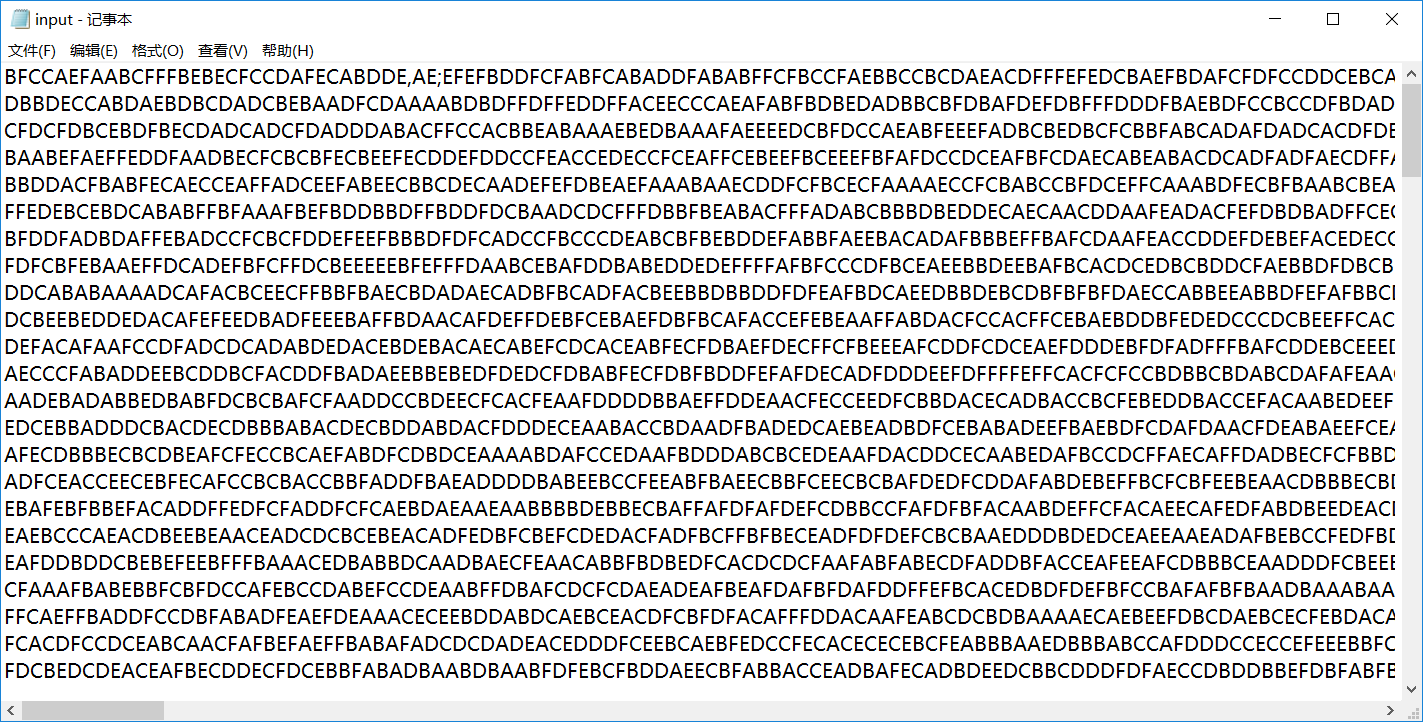
KMP2:



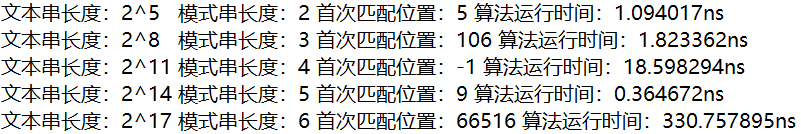
BMH2:



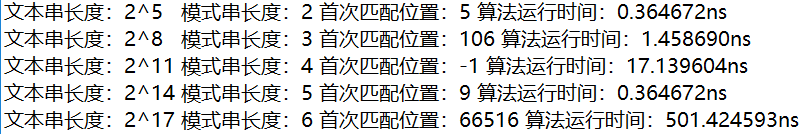
Input3：



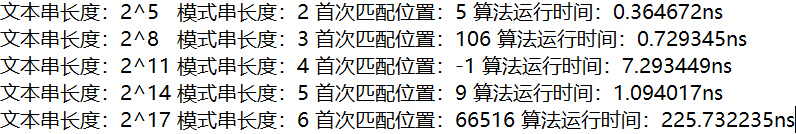
RK3：



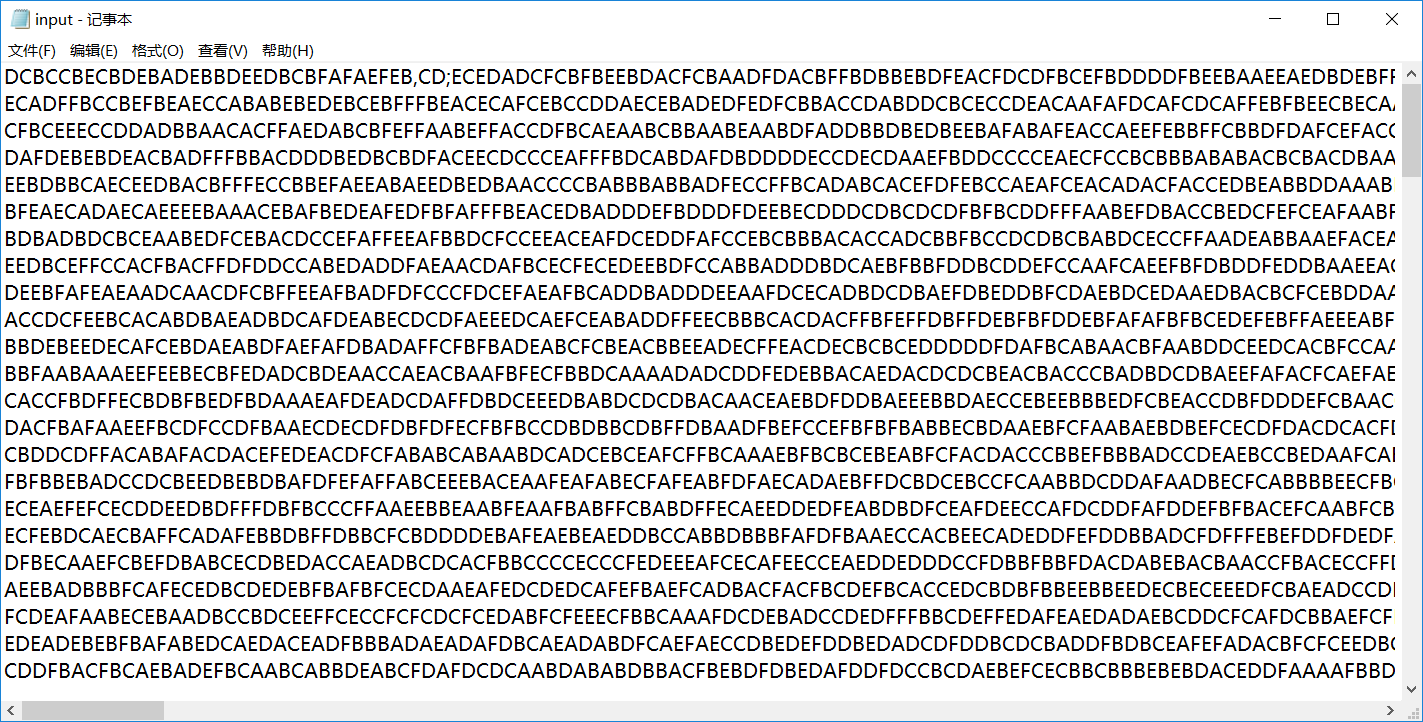
KMP3：



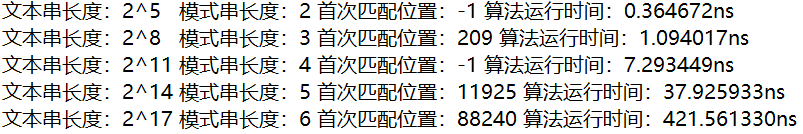
BMH3:



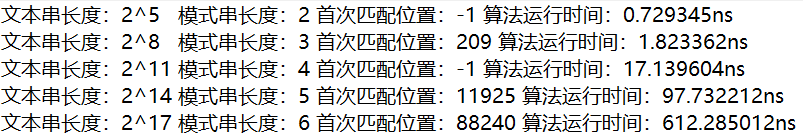
Input4：



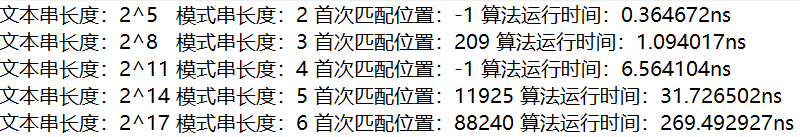
RK4:



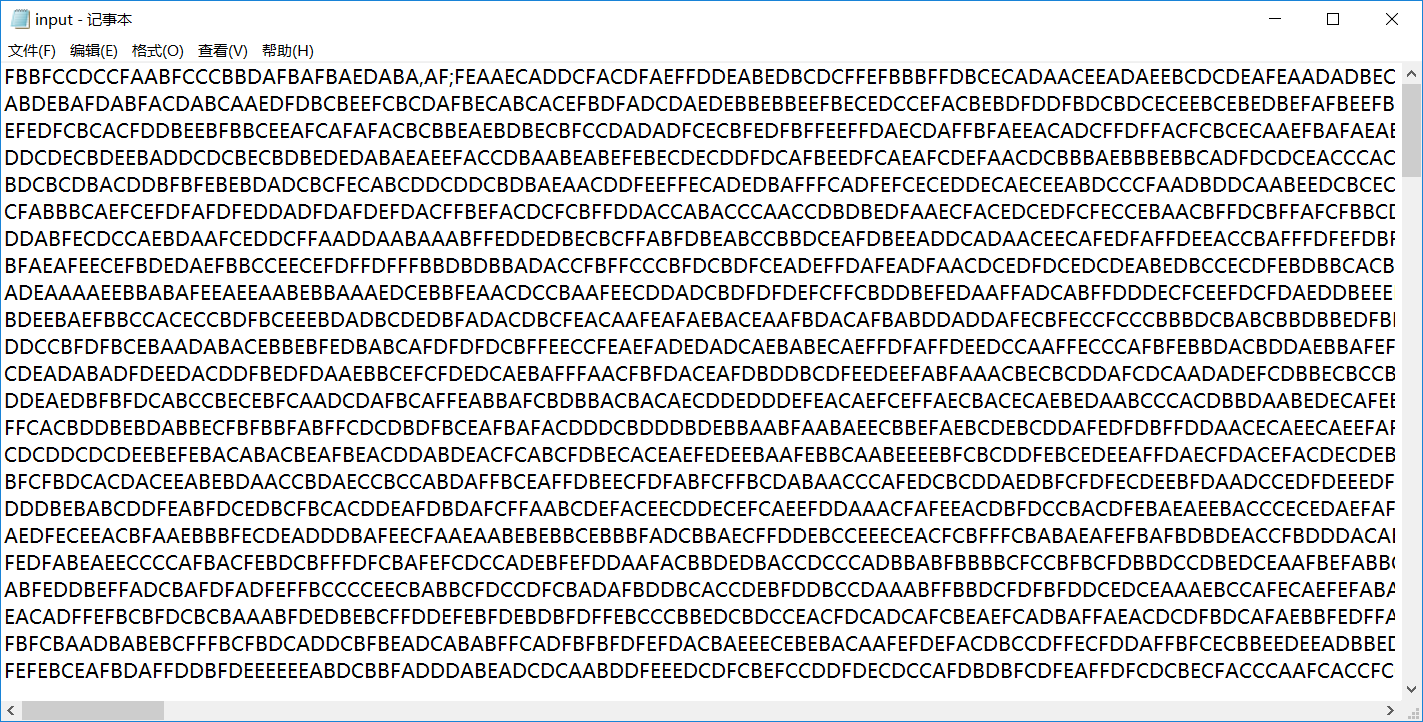
KMP4:



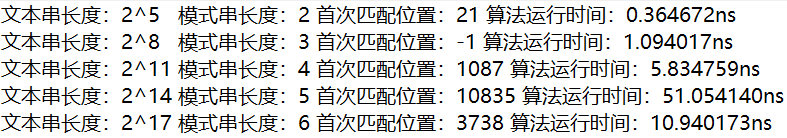
BMH4:



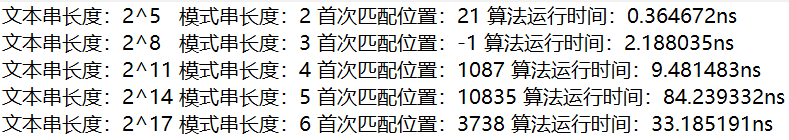
Input5:



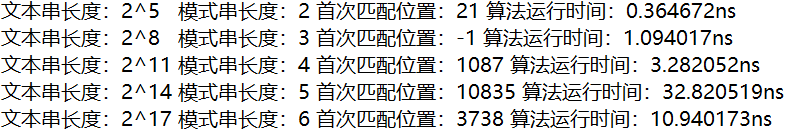
RK5:



KMP5:



BMH5:



平均时间：

RK：0.66ns 1.17ns 7.15ns 31.51ns 296.62ns

KMP：0.51ns 1.53ns 10.50ns 62.87ns 589.82ns

BMH：0.58ns 1.02ns 5.25ns 32.46ns 228.94ns

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rabin-Karp算法 | | | | |
| m | n | rt(ns) | lg(m\*n) | lg(rt) |
| **2** | **32** | **0.66** | **6．00** | **-0.60** |
| **3** | **256** | **1.17** | **9.58** | **0.23** |
| **4** | **2048** | **7.15** | **13.00** | **2.84** |
| **5** | **16384** | **31.51** | **16.32** | **4.98** |
| **6** | **131072** | **296.62** | **19.58** | **8.21** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KMP算法 | | | | |
| m | n | rt(ns) | lg(m+n) | lg(rt) |
| **2** | **32** | **0.51** | **5.09** | **-0.97** |
| **3** | **256** | **1.53** | **8.02** | **0.61** |
| **4** | **2048** | **10.50** | **11.00** | **3.39** |
| **5** | **16384** | **62.87** | **14.00** | **5.97** |
| **6** | **131072** | **589.82** | **17.00** | **9.20** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Boyer-Moore-Horspool算法 | | | | |
| m | n | rt(ns) | lg(m\*n) | lg(rt) |
| **2** | **32** | **0.58** | **6.00** | **-0.79** |
| **3** | **256** | **1.02** | **9.58** | **0.03** |
| **4** | **2048** | **5.25** | **13．00** | **2.39** |
| **5** | **16384** | **32.46** | **16.32** | **5.02** |
| **6** | **131072** | **228.94** | **19.58** | **7.84** |

使用Excel作图并分析：

Rabin-Karp算法:

KMP算法:

Boyer-Moore-Horspool算法:

由上面3张图可见第一个点处不太符合理论情况，可能与规模太小时算法时间受常数时间操作的影响较大有关。其余情况符合的较好。

1. **实验心得**

通过本次实验中的两个小例子，我加深了对串匹配算法的理解，提高了编程熟练度，进一步练习了绘图分析，巩固了将伪代码转化为真正可用代码的能力。