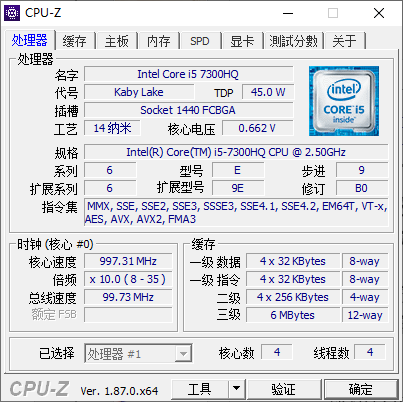
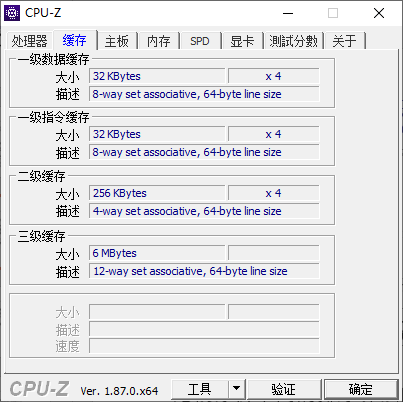
实验报告

一、实验内容：实现串匹配算法。其中，文本串T的长度为n，模式串P的长度为m，字符串中的字符均是随机生成的取自字符集[0-9A-Za-z]（共62个不同字符）的字符。(n, m)共有5组取值，分别为: (25，4), (28，8), (211,16), (214,32), (217,64)。具体字符串由助教给出。

算法： 朴素字符串匹配算法；Rabin-Karp算法；KMP算法；Boyer-Moore-Horspool算法。

二．实验环境

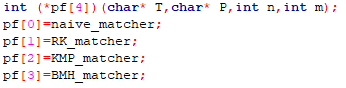
笔记本使用两年四个月



1. 实验方法和步骤

将main()单独写在一个lab.c里，四个串匹配算法写在matcher.c里，方便调试。

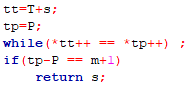
使用函数指针数组，分别指向四个排序算法，方便循环



手动在字符串尾部加上不同字符，这样比较时不用判断越界，减少比较次数



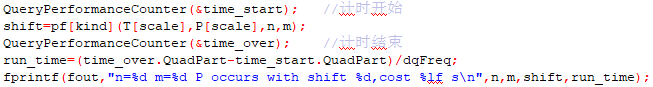
比如在naïve\_matcher中



在RK中，为了防止计算high越界



使用QueryPerformance计时



四个串匹配算法的实现参考pdf，只是将用到动态数组的部分用malloc实现，且没有使用memcmp，使各算法逐字符比较的方式相同，减少分析的系统误差。

1. 实验结果与分析

将算法在不同输入规模下的运行时间曲线图画出，与期望大致相符。

下图中只以n为横坐标而省略m， (n, m)共有5组取值，分别为: (2^5，4), (2^8，8), (2^11,16), (2^14,32), (2^17,64)

单看这几个数据，horspool确实是平均最快的算法，与老师上课分析一致，KMP其次，朴素的匹配算法与KMP在本例中表现差不多。而RK明显比其他几个算法要慢，几乎慢了一个数量级，说明计算p与t耗时相比比较还是太大了。

单看图片，算法的耗时都差不多与n成线性关系，这在后两个算法是合理的（预处理时间不长），但与naïve算法期望差距很大（O(m\*n)），与RK的期望也有差距。