**KMP算法**

1. **实验思路**

根据实验要求，分别测试不同字符串长度、不同匹配模式长度下KMP算法与普通算法的性能差异，选取的字符串长度分别为1Kb, 64Kb, 256Kb, 512Kb, 1024Kb (1Mb)、匹配模式长度分别为1, 5, 10, 15, 20, 25。不失一般性，实验中所采用的字符集为{0, 1}，生成指定长度的随机字符串；匹配模式为指定长度的字符’1’串，以此模拟匹配中出现的前缀重复匹配现象。

1. **实验数据**

根据1中思路，重复测试100次取平均值得到的数据如下：

**表1 实验数据(时间单位：μs)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 匹配模式长度 | 1 | | 5 | | 10 | |
| 算法  字符串/Kb | KMP | Normal | KMP | Normal | KMP | Normal |
| 1 | 2.357 | 2.104 | 5.257 | 5.669 | 8.265 | 9.657 |
| 64 | 360.758 | 344.036 | 785.525 | 788.867 | 1236.54 | 1278.55 |
| 256 | 1462.14 | 1403.49 | 3209.3 | 3180.12 | 4987.1 | 5141.42 |
| 512 | 2940.51 | 2789.67 | 6399.46 | 6351.49 | 9902.15 | 10271.3 |
| 1024 | 5976.06 | 5621.66 | 12917.2 | 12756.1 | 19974.6 | 20569.3 |
|  | | | | | | |
| 匹配模式长度 | 15 | | 20 | | 25 | |
| 算法  字符串/Kb | KMP | Normal | KMP | Normal | KMP | Normal |
| 1 | 11.288 | 13.624 | 14.323 | 17.593 | 17.478 | 21.568 |
| 64 | 1687.40 | 1780.13 | 2133.38 | 2279.47 | 2574.67 | 2776.52 |
| 256 | 6794.95 | 7105.25 | 8547.51 | 9091.14 | 10286.8 | 11089.4 |
| 512 | 13449.0 | 14203.2 | 16990.0 | 18144.0 | 20481.0 | 22085.6 |
| 1024 | 27119.4 | 28433.6 | 34206.0 | 36332.3 | 41209.4 | 44223.3 |

不同字符串长度下，根据匹配模式长度绘制折线图：

**图1 字符串长度为1Kb时随匹配模式长度变化折线图**

**图2 字符串长度为64Kb时随匹配模式长度变化折线图**

**图3 字符串长度为256Kb时随匹配模式长度变化折线图**

**图4 字符串长度为512Kb时随匹配模式长度变化折线图**

**图5 字符串长度为1024Kb时随匹配模式长度变化折线图**

1. **结果分析**

观察以上数据，不难发现，当匹配模式长度较小时，普通算法略优于KMP算法；随着匹配模式长度的增长，KMP算法逐渐优于普通算法，二者性能的绝对差距不断扩大，相对差距基本保持不变(由出现前缀重复匹配的相对概率决定)。原因是：当匹配模式较长时，普通算法出现前缀重复匹配的情况增加，而KMP算法能够有效避免重复的前缀匹配，因此优于普通算法。实验结果与理论分析基本符合。