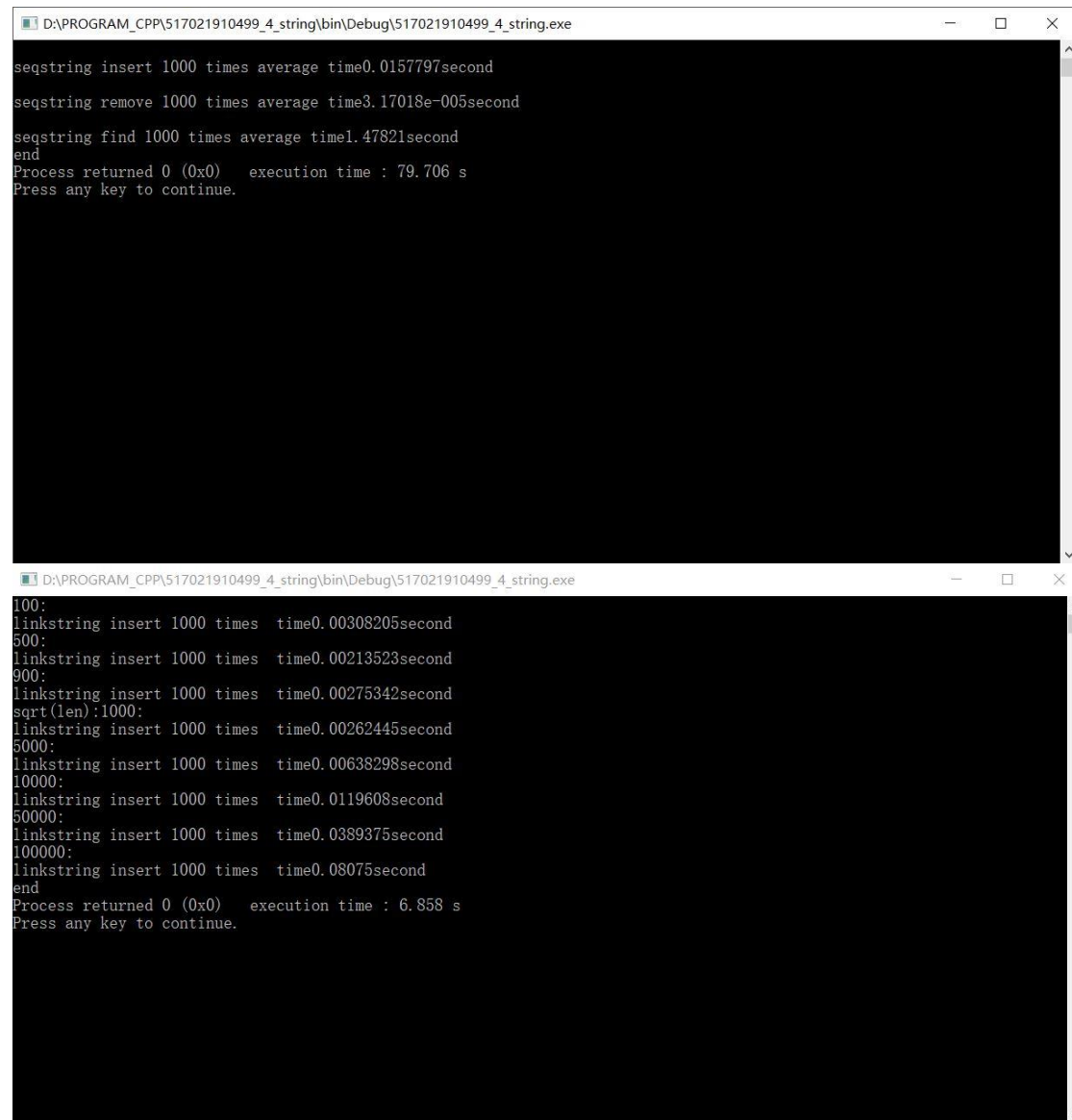


第四次实验报告

陈鸣阳 517021910499

本次测试除了 linkstring 的查找操作以外采用的方法是测试一段时间内该段代码运行的次数，再用总时间除以次数得出平均运行时间，故没有数据统计表格。

测试结果如下图所示：



```
D:\PROGRAM_CPP\517021910499_4_string\bin\Debug\517021910499_4_string.exe
seqstring insert 1000 times average time0.0157797second
seqstring remove 1000 times average time3.17018e-005second
seqstring find 1000 times average time1.47821second
end
Process returned 0 (0x0)   execution time : 79.706 s
Press any key to continue.

D:\PROGRAM_CPP\517021910499_4_string\bin\Debug\517021910499_4_string.exe
100:
linkstring insert 1000 times   time0.00308205second
500:
linkstring insert 1000 times   time0.00213523second
900:
linkstring insert 1000 times   time0.00275342second
sqrt(len):1000:
linkstring insert 1000 times   time0.00262445second
5000:
linkstring insert 1000 times   time0.00638298second
10000:
linkstring insert 1000 times   time0.0119608second
50000:
linkstring insert 1000 times   time0.0389375second
100000:
linkstring insert 1000 times   time0.08075second
end
Process returned 0 (0x0)   execution time : 6.858 s
Press any key to continue.
```

```
D:\PROGRAM_CPP\517021910499_4_string\bin\Debug\517021910499_4_string.exe
100:
linkstring remove 1000 times time2ms
500:
linkstring remove 1000 times time2ms
900:
linkstring remove 1000 times time3ms
sqrt(len):1000:
linkstring remove 1000 times time3ms
5000:
linkstring remove 1000 times time10ms
10000:
linkstring remove 1000 times time20ms
50000:
linkstring remove 1000 times time100ms
100000:
linkstring remove 1000 times time224ms
end
Process returned 0 (0x0) execution time : 1.825 s
Press any key to continue.
```

```
D:\PROGRAM_CPP\517021910499_4_string\bin\Debug\517021910499_4_string.exe
10:
linkstring find 1000 times time4.796second
30:
linkstring find 1000 times time4.003second
50:
linkstring find 1000 times time3.872second
100:
linkstring find 1000 times time3.716second
300:
linkstring find 1000 times time3.61second
1100:
linkstring find 1000 times time3.6second
1600:
linkstring find 1000 times time3.573second
2100:
linkstring find 1000 times time3.659second
end
Process returned 0 (0x0) execution time : 31.200 s
Press any key to continue.
```

基于数组字符串		
插入1000次	0.0157797s	
删除1000次	0.00317018ms	
查找1000次	1.47821s	
基于链表字符串		
结点容量	插入（秒）	删除（毫秒）
100	0.00308205	2
500	0.00213523	2
900	0.00275342	3
1000（长度开方）	0.00262445	3
5000	0.00638298	10
10000	0.0119608	20
50000	0.0389375	100
100000	0.08075	224

结点容量	10	30	50 (长度开方)	100	600	1100	1600	2100
1	5.105	4.324	4.233	4.214	3.943	3.904	3.902	3.926
2	5.32	4.701	4.228	4.078	3.922	3.84	3.88	3.851
3	5.631	4.451	4.181	3.96	3.868	3.836	3.861	3.863
4	5.651	4.438	4.083	4.023	3.891	3.855	3.879	3.895
5	5.622	4.436	4.111	3.972	3.836	3.871	3.821	3.817
平均值	5.4658	4.47	4.1672	4.0494	3.892	3.8612	3.8686	3.8704

结果分析：

本来想要作图，但是我觉得这次的数据关系较为明显，所以没有作图。

对于块状链表字符串，结点容量大的时候插入和删除操作所需的时间较长，而结点容量小的时候，查找所需的时间比结点容量大的时候要长，三项操作综合考虑，结点容量为字符串长度的开方时是效率较优的选择。

对于基于数组实现的字符串（与结点容量为字符串长度开方比较），插入操作所需的时间较长，但是删除操作所需的时间更短。

对于查找操作，我的电脑上字符串长度太大时会报内存错误，因此我将查找用的链表字符串取了较短的长度，当字符串长度增加时所需的时间应该会更长，所以基于数组的字符串在查找上应该要比基于链表的字符串快。