【数据集生成方法】

首先规定数据集长度，然后通过文件指针写入到 data.txt 数据集文件中。关闭文件，并以读取方式再次打开，每次打开都会清除上次的数据集。通过 for 函数循环来生成一个自然数集，以测试算法的最坏情况时间复杂度。

具体代码如下：

```
cout<<"How long is the data?      ";
int len,i;/*用 len 来表示数据集长度*/
ofstream data_in("data.txt",ios::trunc);/*每次删除并重建数据集*/
cin>>len;
for(i=1;i<=len;i++)
      data_in<<i<<" ";
data_in.close();/*创建数据集完成，保存并退出*/
ifstream data_out("data.txt");/*读取方式重新打开*/
i=0;

int A[len];
while(!data_out.eof())
{
      data_out>>A[i];
      i++;
}/*已经将数据集录入*/
```

【测试手段（时间评估方法）】

对于算法 1：算法 1 运行很慢，故而测量一次运行时间即可

对于算法 2：算法 2 运行很快，故而要测量算法而运行 10000 次的时间

测试时间用以下方法

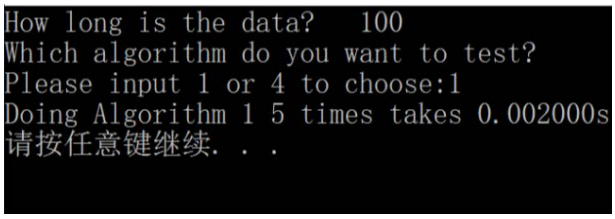
len的长度分别设置为100，500，1000，2000，4000，6000，8000，10000：

```
#include <time.h>
clock_t start, stop; /* clock_t is a built-in type for processor time (ticks) */
double duration; /* records the run time (seconds) of a function */
int main ( )
{/* clock() returns the amount of processor time (ticks) that has elapsed
        since the program began running */
start = clock(); /* records the ticks at the beginning of the function call */
function(); /* run your function here */
stop = clock(); /* records the ticks at the end of the function call */
duration = ((double)(stop - start))/CLK_TCK;
     /* CLK_TCK is a built-in constant = ticks per second */
     … …
     return 1;
     }
```

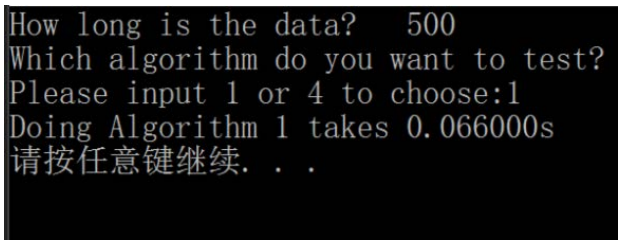
【测试结果】（用时间数据截图）

	N	100	500	1000	2000	4000	6000	8000	10000
Algorithm 1	Iterations (*K*)	5	1	1	1	1	1	1	1
	Total Time (sec)	0.002	0.066	0.475	3.519	28.868	102.292	243.992	468.885
	Duration (sec)	0.0004	0.066	0.475	3.519	28.868	102.292	243.992	468.885
Algorithm 4	Iterations (*K*)	10000	10000	10000	10000	10000	10000	10000	10000
	Total Time (sec)	0.003	0.017	0.031	0.058	0.115	0.169	0.216	0.274
	Duration (sec)	0.0000003	0.0000017	0.0000031	0.0000058	0.0000115	0.0000169	0.0000216	0.0000274

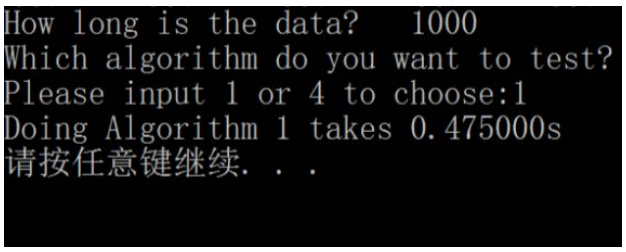
Algorithm 1，len=100， 5 times



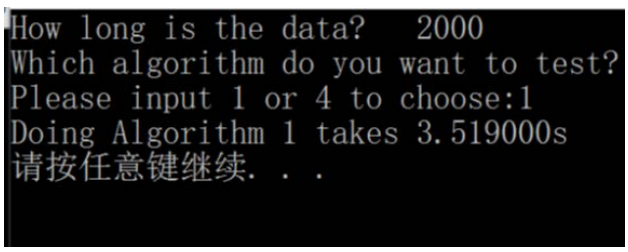
Algorithm 1，len=500， 1 times



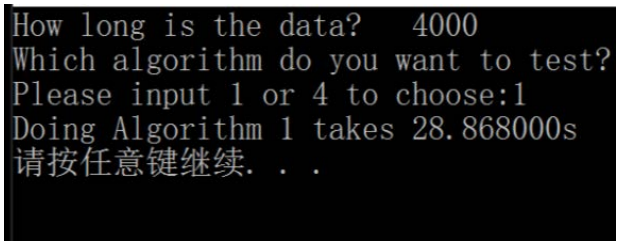
Algorithm 1，len=1000， 1 times



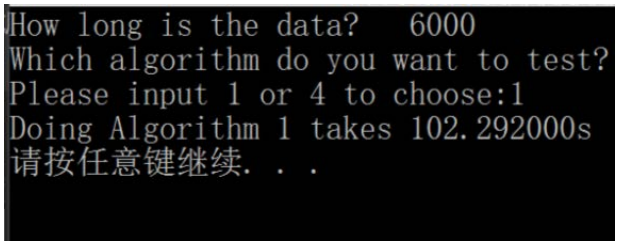
Algorithm 1，len=2000， 1 times



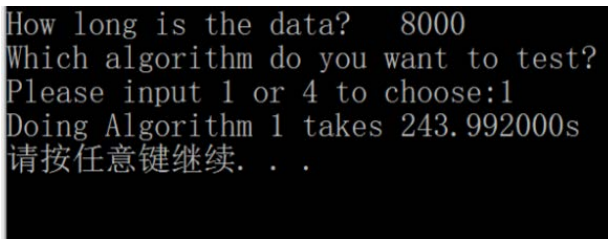
Algorithm 1，len=4000， 1 times



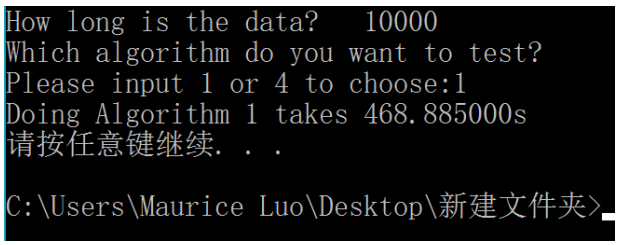
Algorithm 1，len=6000， 1 times



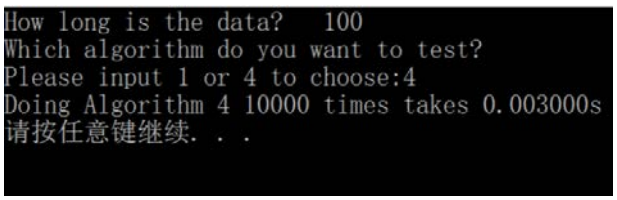
Algorithm 1，len=8000， 1 times



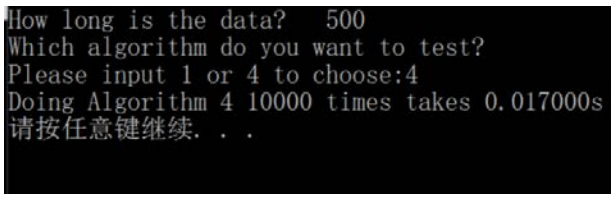
Algorithm 1，len=10000， 1 times



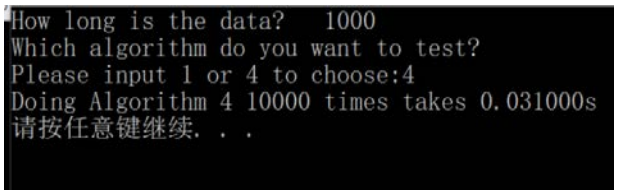
Algorithm 4，len=100，10000 times



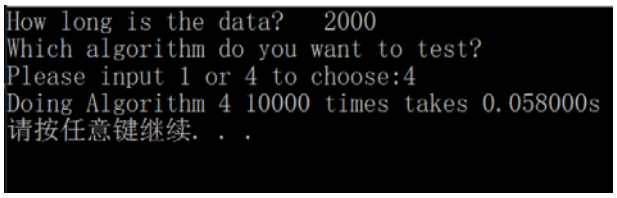
Algorithm 4，len=500，10000 times



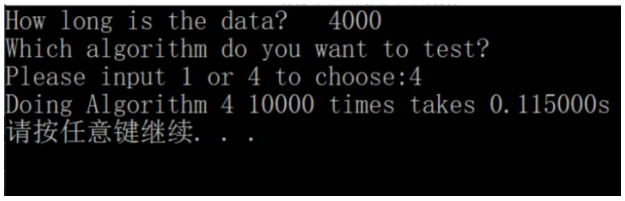
Algorithm 4，len=1000，10000 times



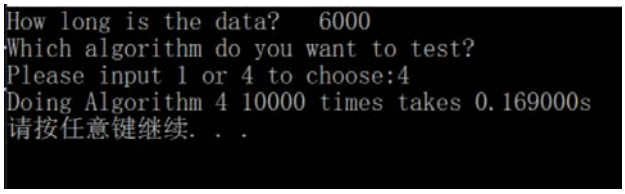
Algorithm 4，len=2000，10000 times



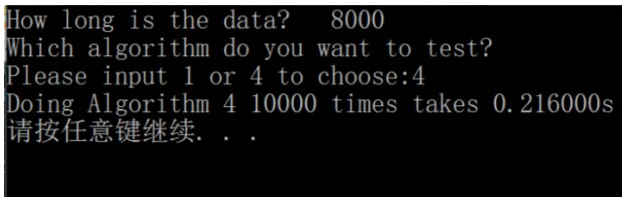
Algorithm 4，len=4000，10000 times



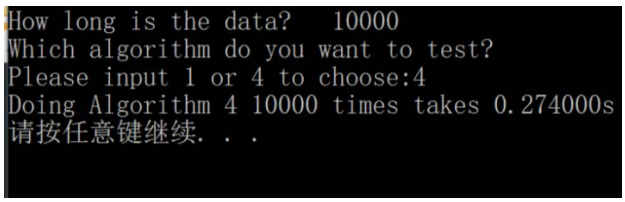
Algorithm 4，len=6000，10000 times



Algorithm 4，len=8000，10000 times



Algorithm 4，len=10000，10000 mes



【测试结论】

随着数据量的增大，拥有时间复杂度为 $O(N^3)$的算法 1 消耗时间急剧增加，而拥有时间复杂度为 O(N)的算法 4 消耗的时间基本呈线性趋势增长。故而当我们进行编程时，尤其是在面对大数据量的情况下，就要考虑到算法渐进时间复杂度带来的影响。通过数据可以看出算法 4 的效果远超算法 1，在实际应用中更加实用。

网工 14 班 罗暄澍 学号 :2015211527