1. 代码 1-1 和 1-2 的测试

```
int max2(int array[],int size,int d)
{
    int max=0,i;
    for(i=0;i<size;++i)
          if(array[i]>max) max=array[i];
    return max*d;
}
```

代码 1-1

```
int max1(int array[],int size,int d)
{
    int max=0,i;
    for(i=0;i<size;++i)array[i]*=d;
    for(i=0;i<size;++i)
          if(array[i]>max) max=array[i];
    return max;
}
```

代码 1-2

运用随机数生成语句生成大小分别为 1000,10000,100000 的随机数组，分别测试代码 1-1 和 1-2，并运用高精度时控函数 QueryPerformanceFrequency（），QueryPerformanceCounter（）来得到程序运行时间，总共运行 10 次，并求取平均值，数据结果如下表：

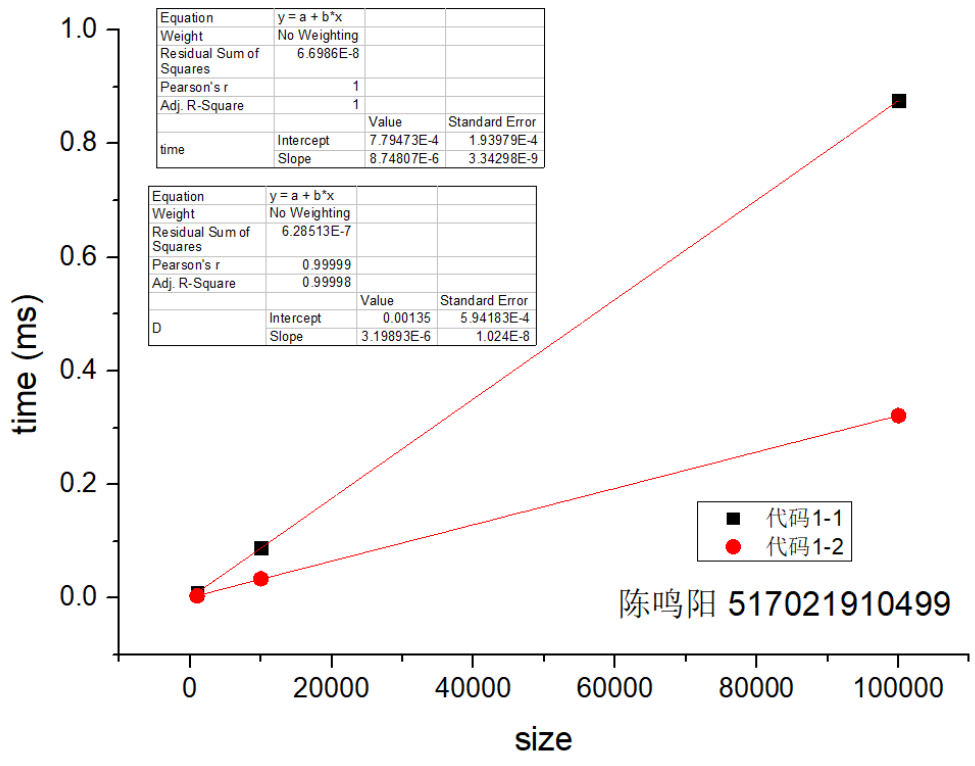
代码1-1（时间单位ms）

size	1	2	3	4	5	6	7	8	9	10	平均值
1000	0.00574359	0.0110769	0.0110769	0.00902564	0.0110769	0.0114872	0.0110769	0.00697436	0.00861539	0.00738462	0.00935384
10000	0.0562052	0.107487	0.11159	0.0754872	0.108308	0.107487	0.107897	0.0644103	0.0635898	0.0820513	0.08845128
100000	0.542359	0.798359	1.09456	0.84718	1.08103	1.20739	1.07856	0.641641	0.825846	0.638769	0.8755694

代码1-2（时间单位ms）

size	1	2	3	4	5	6	7	8	9	10	平均值
1000	0.0028718	0.00328205	0.00574359	0.00451282	0.00574359	0.00328205	0.00574359	0.0028718	0.00328205	0.0028718	0.004020514
10000	0.0266667	0.0299487	0.0512821	0.0299487	0.0299487	0.0299487	0.0504616	0.030359	0.030359	0.030359	0.03392822
100000	0.261744	0.317539	0.310564	0.307282	0.298667	0.299487	0.51039	0.300308	0.306872	0.299077	0.321193

对此表数据用 origin 进行线性拟合得下图：

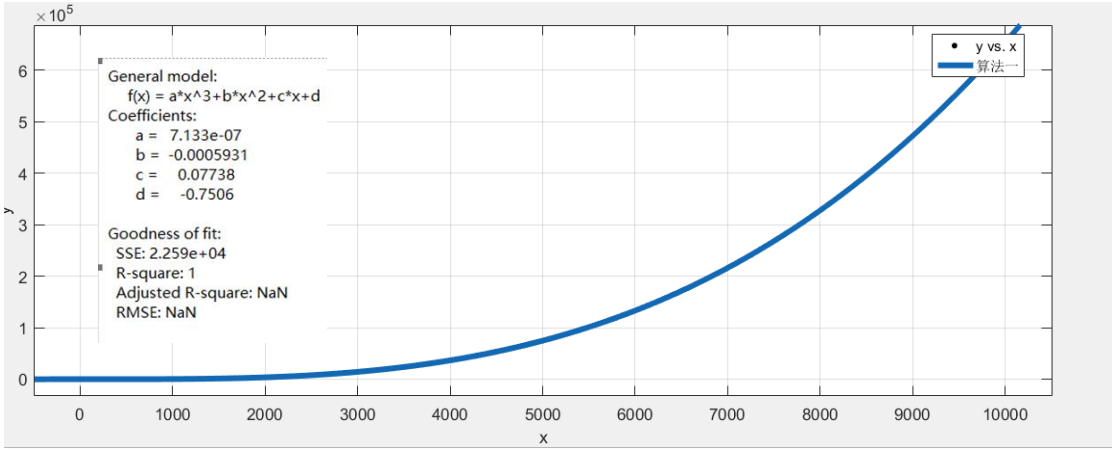


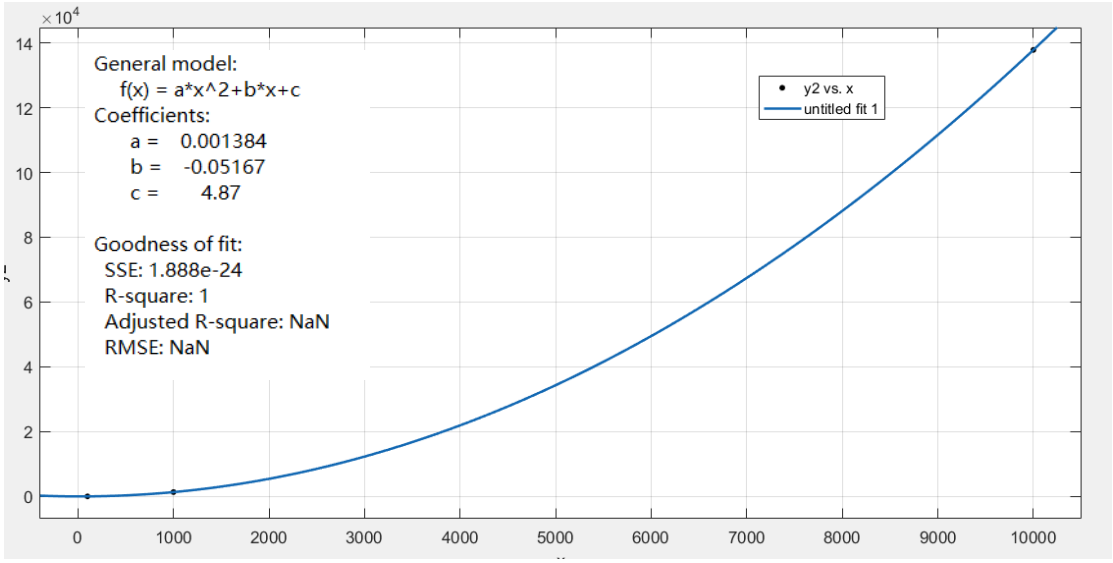
由上图可知，代码 1-1 和代码 1-2 运行时间关于数组大小的拟合曲线斜率大致为 2:1 的关系。因此在数组大小较小的情况下代码 1-2 的优势并不十分明显，但当数组大小很大时，代码 1-2 将比代码 1-1 节省大量时间，具有更大优越性。通过理论分析，代码 1-1 在最坏情况下总运算量是 8n+4,代码 1-2 最坏情况下总运算量 n 前的系数为 4，与实际结果一致。

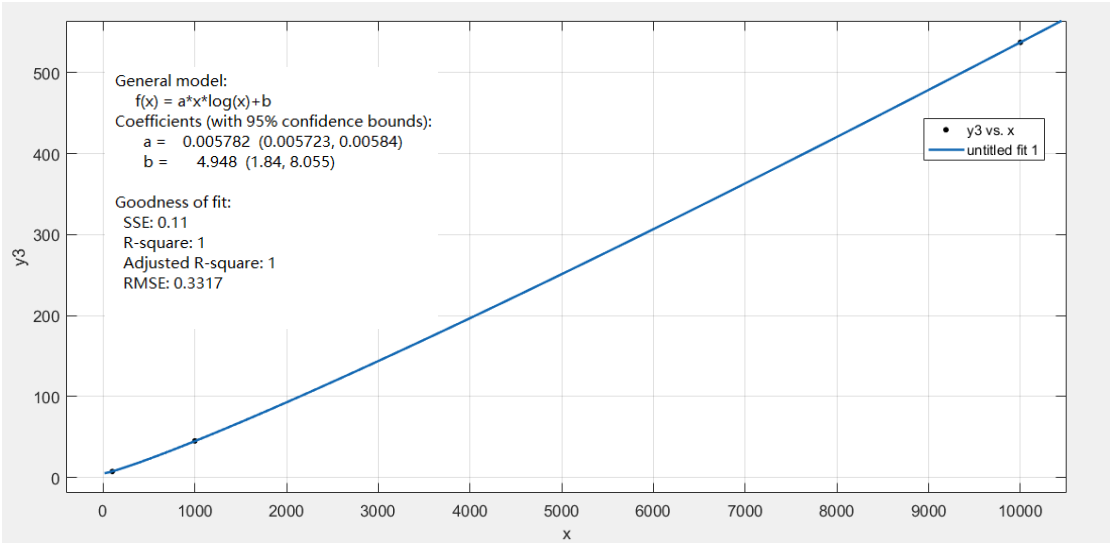
2. 求最大子序列和的四种算法

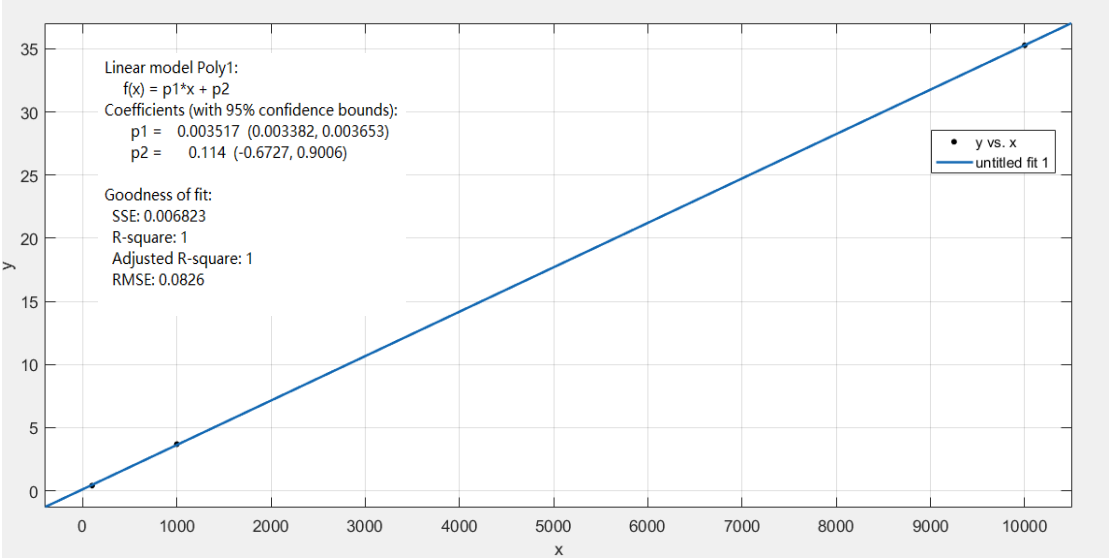
同第一题的方法，建立大小分别为 100,1000,10000 的随机数组对四种算法进行测试，数据如下表：

	算法一（单位ms）				算法二(单位微秒)			
size	1	2	3	平均	1	2	3	平均
100	0.0437744	0.045282	0.0485744	0.045876933	13.5385	13.5385	13.5385	13.5385
1000	46.5795	46.8159	46.2837	46.5597	1251.69	1379.28	1379.28	1336.75
10000	658914	723906	581592	654804	141952	135789	135789	137843.3333
	算法三(单位微秒)				算法四(单位微秒)			
	1	2	3	平均	1	2	3	平均
100	7.38462	7.79487	6.97436	7.384616667	0.410257	0.410257	0.410257	0.410257
1000	45.5385	45.5385	44.3077	45.12823333	3.69231	3.69231	3.69231	3.69231
10000	522.257	583.795	506.257	537.4363333	35.6923	34.4616	35.6923	35.28206667









由表格数据，利用 matlab 分别对算法一、算法二、算法三、算法四进行拟合得上示曲线，由曲线可知各算法的时间复杂度基本可认为分别符合 O(N^3),O(N^2),O(NlogN),O(N)。

注：为绘制三次曲线另对算法一进行了数组大小为 10 时的时间测试（100000 大小运行时间过长）。

3. 程序设计题

程序设计题 1 代码（**代码在最坏情况下时间复杂度为 n 的平方根**）

```
#include <iostream>
#include<cmath>
using namespace std;


bool judge(int n)
{
    int limit;
    if(n<=1) return false;
    if(n==2) return true;
    if((n>2)&&(n%2)==0) return false;
    limit=sqrt(n)+1;
    for(int i=3;i<=limit;i+=2)
          if ((n%i)==0) return false;
          return true;
}

int main()
{
    int m;
    cout<<"enter the number(>0)";
    cin>>m;
    if(judge(m)) cout<<"yes";
    else cout<<"no";
    return 0;
}
```

程序设计题 2 代码

```
#include <iostream>

using namespace std;
int n;
int main()
{
   cout<<"enter the number";
   cin>>n;

   if((n%2)==0) cout<<n/2;
   else cout<<n-n/2;

   return 0;
}
```

（各题运行情况截图见附件）