西安交通大学实验报告

课程名称： 算法设计与问题求解 实验名称： 动态规划算法-2

学 院： 机械工程学院 实验日期 2020 年 11 月 16 日

班 级： 机械97班 姓 名： 杨逢诜 学号： 2193712613

#### 一、实验内容和结果

* **题目1 求连续子数组的最大和问题**

一个有N个整数元素的一维数组A[0]、A[1]、…、A[N-1]，求其中**连续的**子数组和的最大值？不需要返回子数组的具体位置、数组中包含：正、负、零整数、子数组不能空。

例如：

int A[] = { 1, -1, 2, 3, -4, 4 };

符合条件的子数组为{1, -1, 2, 3}或{2,3}或{2, 3, -4, 4}，即答案为5；

#### 实验1 内容：

1. 实现动态规划、穷举法程序；
2. 对比两种方法处理长度为100000的数组时的运行时间差异；

长度为100000数组每一个元素用rand()%100-50随机产生。

【源程序】

实验1-1（动态规划-手动输入）

#include<stdio.h>

int DynamicProg(int n,int \*a) //函数声明：动态规划求解最大连续部分和

{

int summax[n],i,max=0; //声明变量：存放前n个数中最大连续部分和的数组，循环控制变量，最大连续部分和

summax[0]=a[0]; //初始化数组首元素：第一个数的最大连续部分和必是其自身或者0

for(i=1;i<n;i++) //计算序列逐渐增长时的最大连续部分和：

{

if(a[i]>0&&summax[i-1]>0) //若前一次运算结果与下一项相加能够获得更大的连续部分和，则将这个新的连续部分和存入数组中

{

summax[i]=summax[i-1]+a[i];

}

else summax[i]=a[i]; //若不能，连续求和序列中断，新元素本身则作为最大连续部分和

if(summax[i]>max) //若两种方法计算得到的最大连续部分和比已存储的最大连续部分和大，则新的最值将覆盖旧值

{

max=summax[i];

}

}

return max; //返回最大值

}

int main() //主函数开始

{

int n,num[1000],i,summax; //声明变量：序列长度、序列、循环控制变量、最大连续部分和

printf("请输入指定序列长度：\n"); //输出提示，按照提示依次输入序列长度、序列

scanf("%d",&n);

printf("请输入指定序列：\n");

for(i=0;i<n;i++)

{

scanf("%d",&num[i]);

}

summax=DynamicProg(n,num); //调用动态规划算法，计算最大连续部分和

printf("连续子序列和最大值为：%d",summax); //输出结论，主函数结束

return 0;

}

实验1-2（穷举，手动输入）

#include<stdio.h>

int main() //主函数开始

{

int n,num[1000],i,j,max,summax=0; //声明变量：序列长、序列数组、循环控制变量、连续部分和、最大连续部分和

printf("请输入指定序列长度：\n"); //输出提示，按照提示依次输入序列长和序列

scanf("%d",&n);

printf("请输入指定序列：\n");

for(i=0;i<n;i++)

{

scanf("%d",&num[i]);

}

for(i=0;i<n;i++) //从第一个元素作为起始，计算“从第i给元素起始的连续序列部分和”

{

max=0; //每次计算之前重置连续部分和为初始值0

for(j=i;j<n;j++) //计算从第i个元素起始的全部连续序列部分和（遍历，只经过不储存）

{

max=max+num[j]; //逐项相加

if(max>=summax) //每加一项，就将加和结果与目前已获得的最大连续部分和相比较，若较大则用新值覆盖旧值

{

summax=max;

}

}

} //输出计算得到的最大连续部分和，主函数结束

printf("连续子序列和最大值为：%d",summax);

return 0;

}

实验1-3（动态规划，自动生成）

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int DynamicProg(int n,int \*a) //函数声明：动态规划求解最大连续部分和

{

int summax[n],i,max=0; //声明变量：存放前n个数中最大连续部分和的数组，循环控制变量，最大连续部分和

summax[0]=a[0]; //初始化数组首元素：第一个数的最大连续部分和必是其自身或者0

for(i=1;i<n;i++) //计算序列逐渐增长时的最大连续部分和：

{

if(a[i]>0&&summax[i-1]>0) //若前一次运算结果与下一项相加能够获得更大的连续部分和，则将这个新的连续部分和存入数组中

{

summax[i]=summax[i-1]+a[i];

}

else summax[i]=a[i]; //若不能，连续求和序列中断，新元素本身则作为最大连续部分和

if(summax[i]>max) //若两种方法计算得到的最大连续部分和比已存储的最大连续部分和大，则新的最值将覆盖旧值

{

max=summax[i];

}

}

return max; //返回最大值

}

int main() //主函数开始

{

srand((unsigned int)time(NULL)); //生成种子，用于产生随机数

int n=10000,num[10000],i,summax; //定义变量，用于生成随机数列和计算最大连续部分和

for(i=0;i<n;i++) //生成随机序列

{

num[i]=rand()%100-50;

}

summax=DynamicProg(n,num); //调用动态规划算法，计算最大连续部分和

printf("连续子序列和最大值为：%d",summax); //输出计算结论，主函数结束

return 0;

}

实验1-4（穷举，自动生成）

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int main() //主函数开始

{

srand((unsigned int)time(NULL)); //定义种子，用于生成随机序列

int n=10000,num[10000],i,j,max,summax=0;//声明变量，用于生成随机的整数序列和计算最大连续部分和

for(i=0;i<n;i++) //生成随机序列

{

num[i]=rand()%100-50;

}

for(i=0;i<n;i++) //从第一个元素作为起始，计算“从第i给元素起始的连续序列部分和”

{

max=0; //每次计算之前重置连续部分和为初始值0

for(j=i;j<n;j++) //计算从第i个元素起始的全部连续序列部分和（遍历，只经过不储存）

{

max=max+num[j]; //逐项相加

if(max>=summax) //每加一项，就将加和结果与目前已获得的最大连续部分和相比较，若较大则用新值覆盖旧值

{

summax=max;

}

}

} //输出计算得到的最大连续部分和，主函数结束

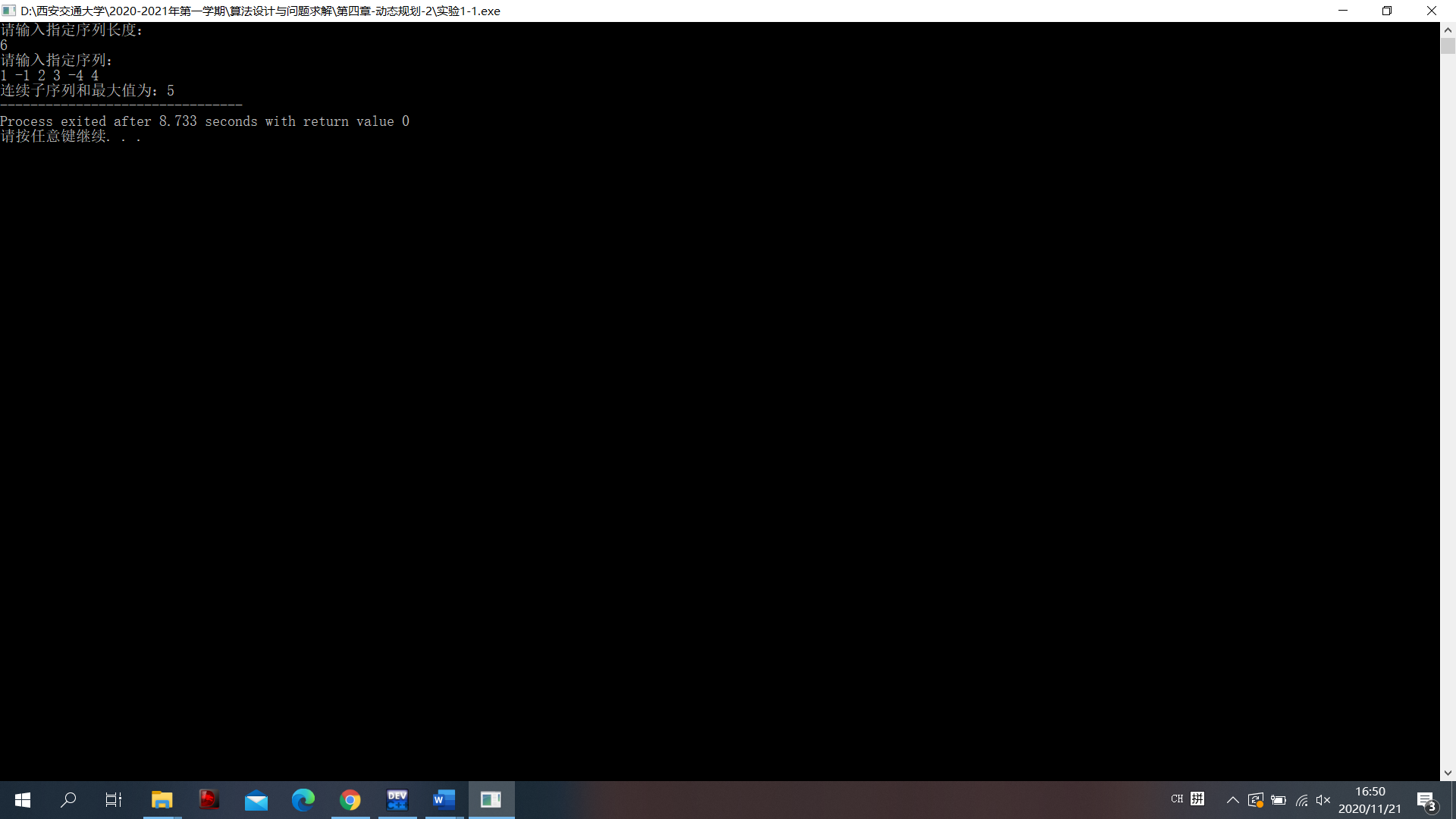
printf("连续子序列和最大值为：%d",summax);

return 0;

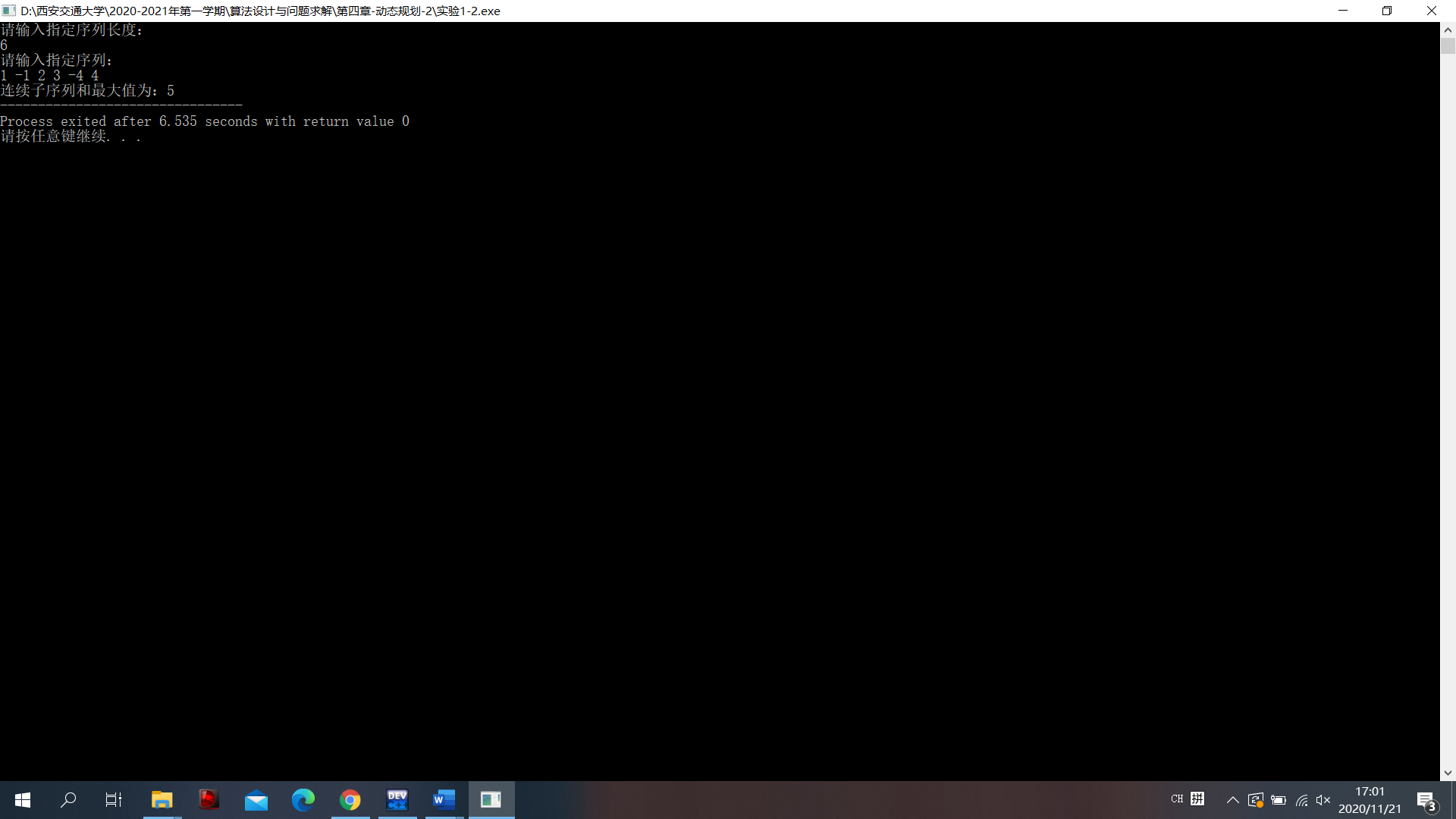
}

【运行结果】(截图，选中运行窗口，按下Alt + Print Screen组合键，在下方按Ctrl+V即可)

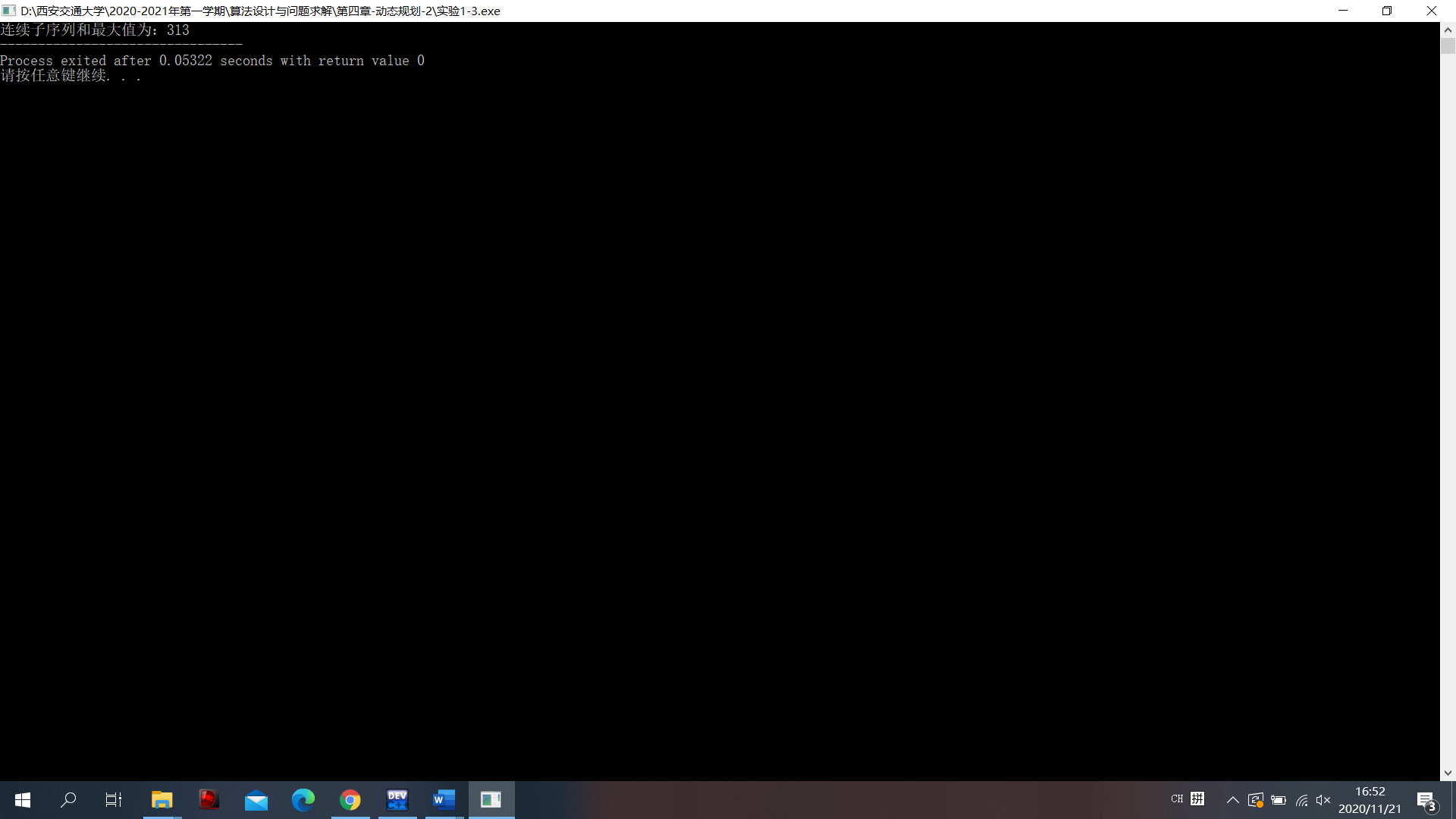
实验1-1（动态规划-手动输入）

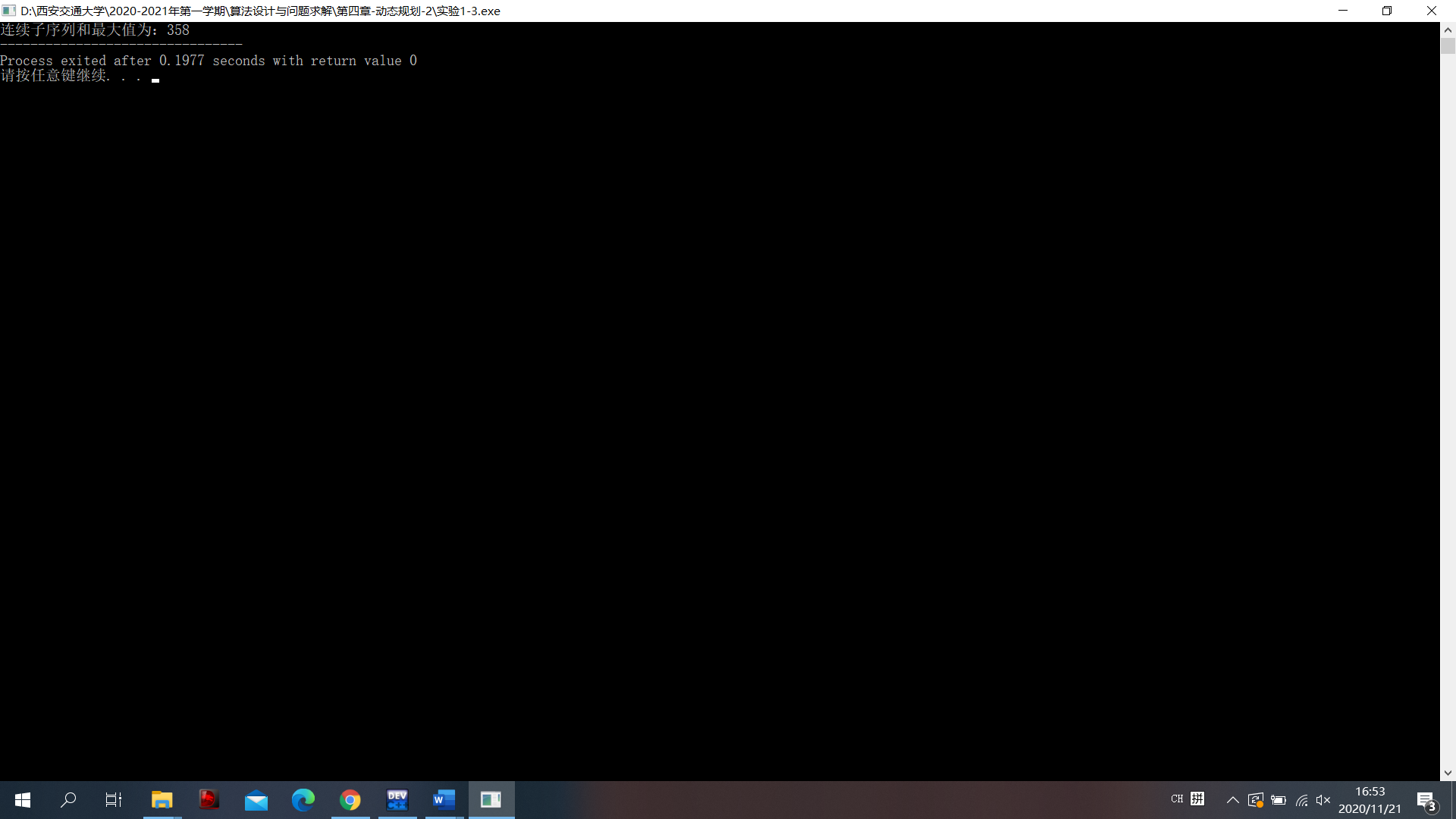


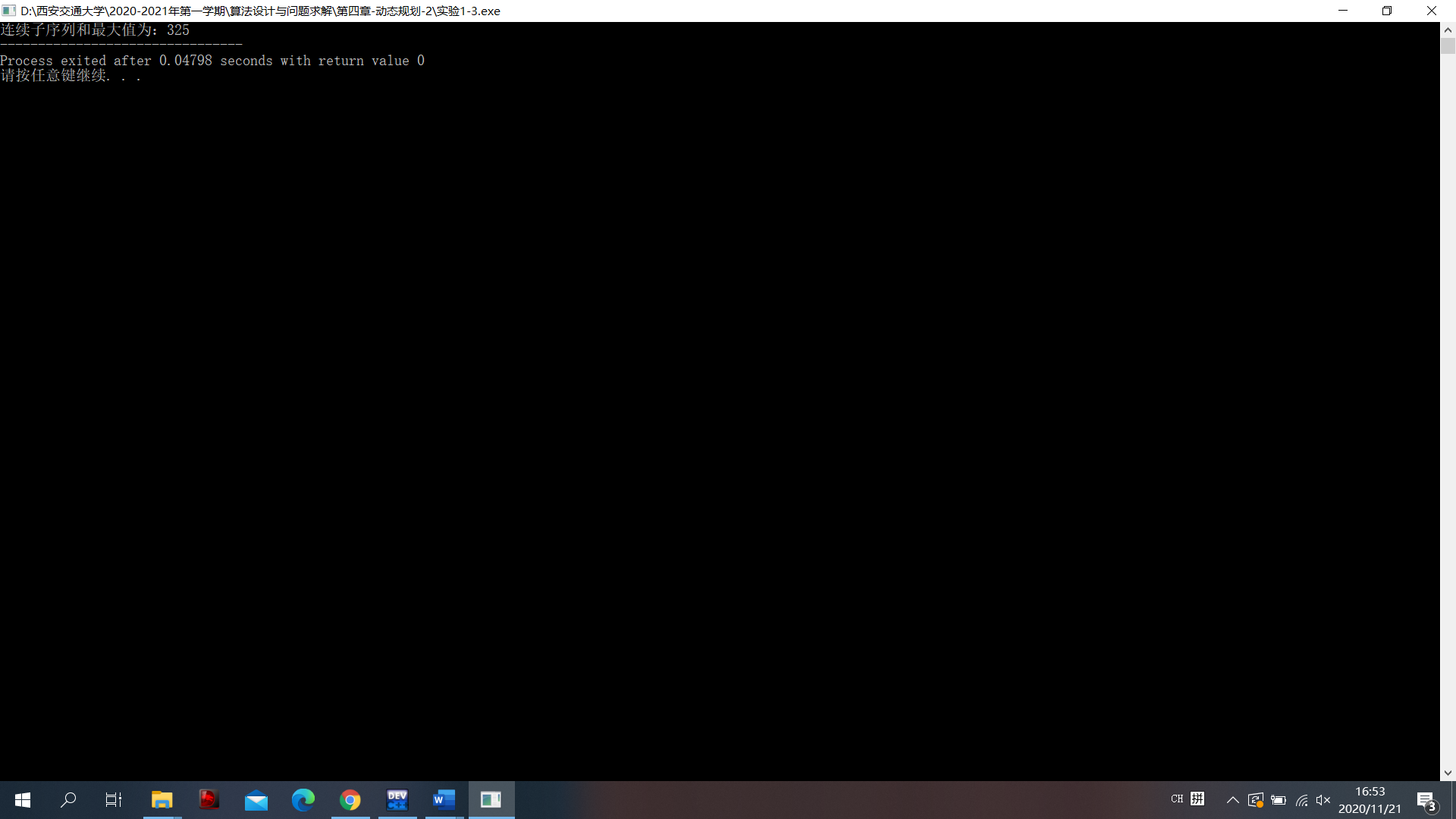
实验1-2（穷举，手动输入）



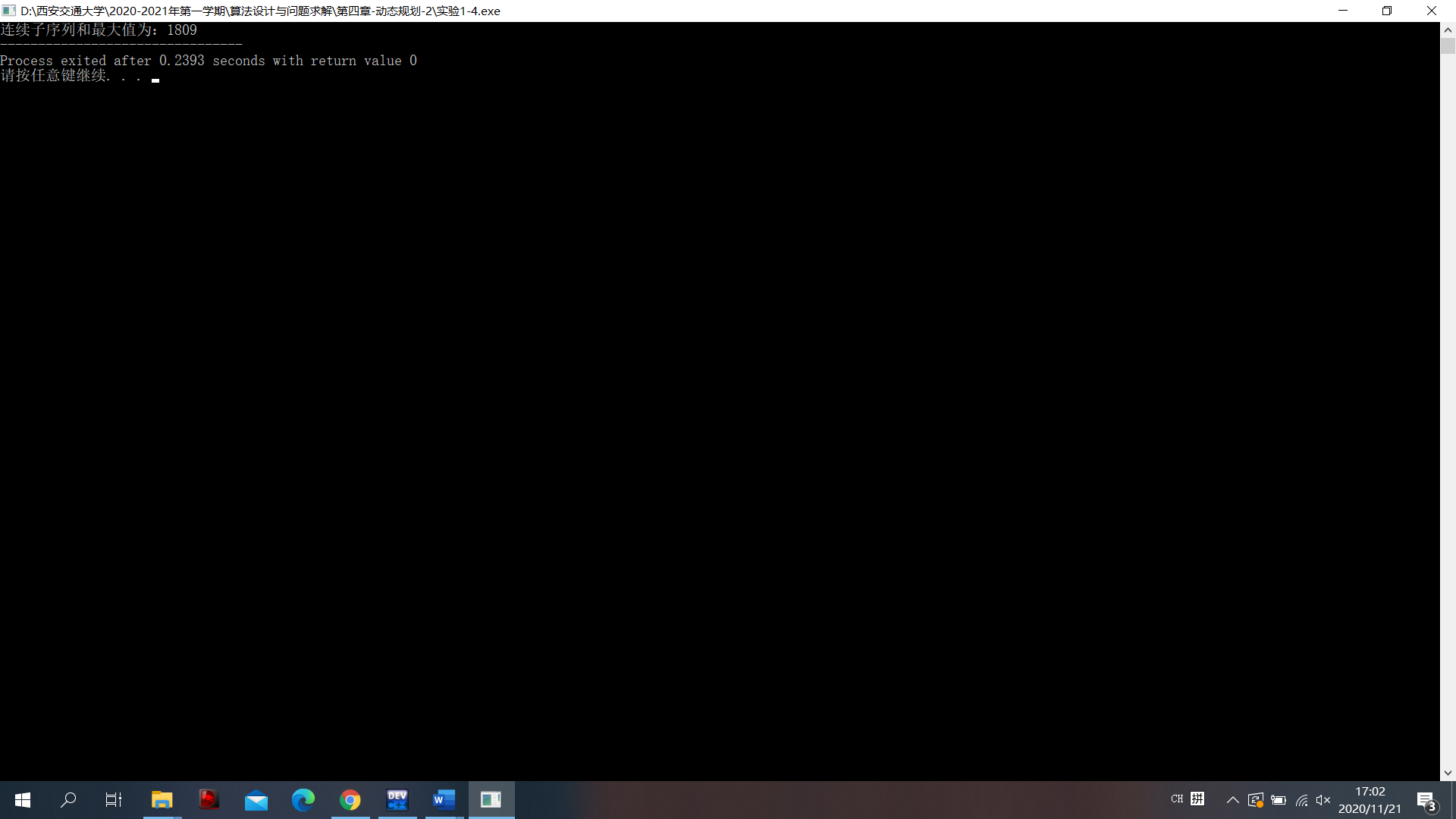
实验1-3（动态规划，自动生成）

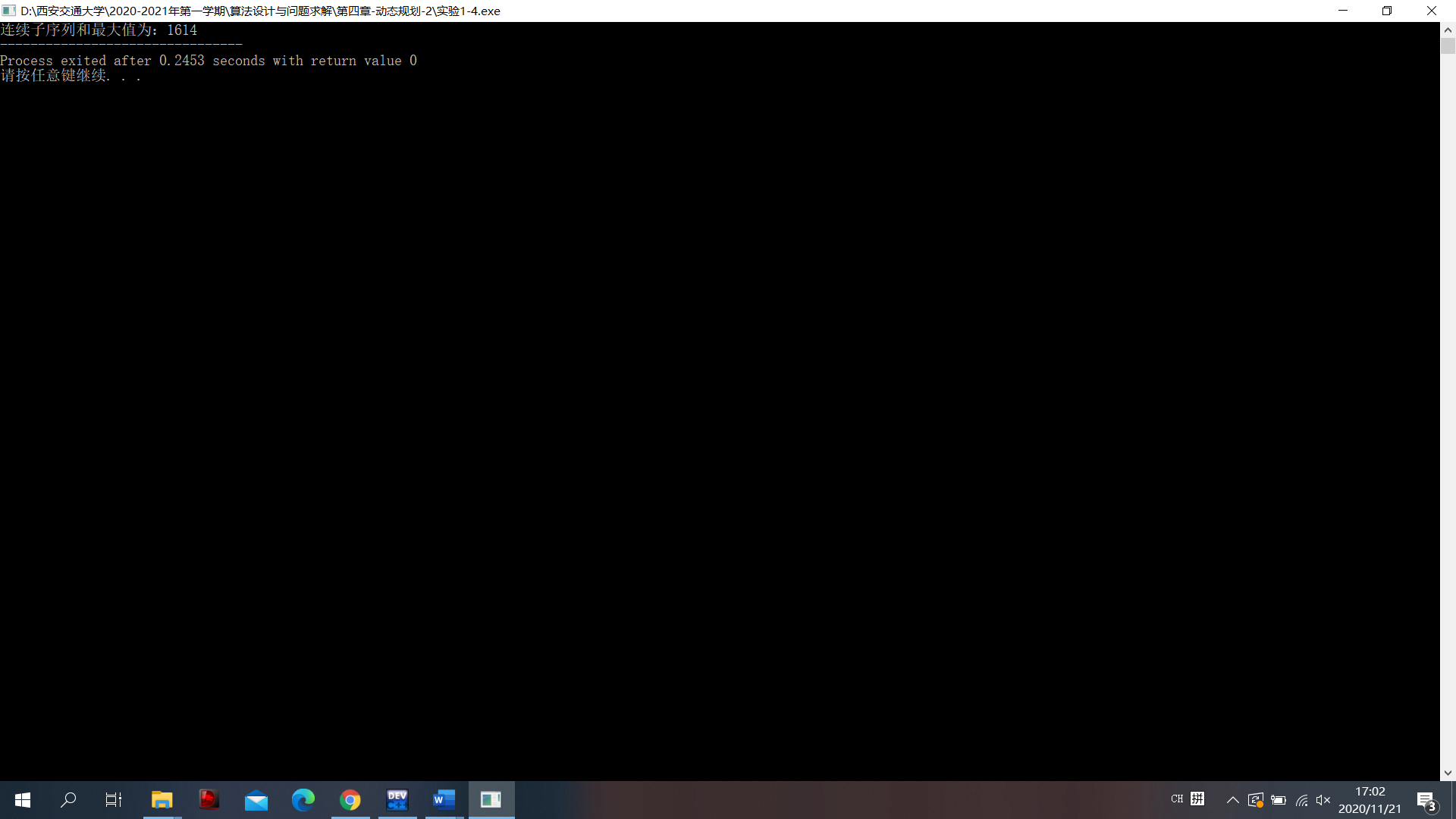


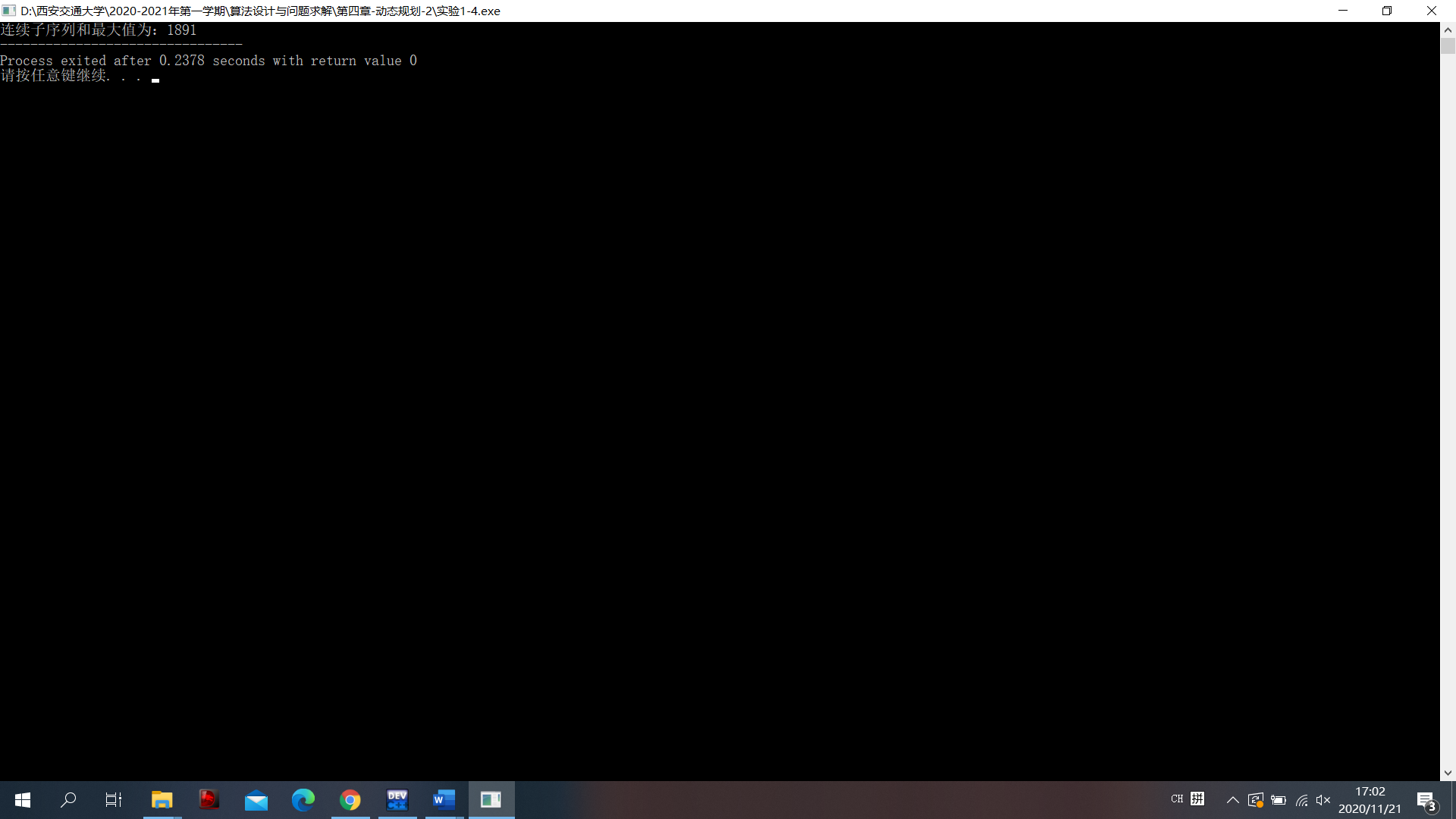




实验1-4（穷举，自动生成）







### 题目2 最长公共子序列问题

**子序列就是在给定的序列中删除若干元素后得到的序列。给定两个序列X和Y，当另一个序列Z既是X的子序列又是Y的子序列时，就称Z是序列X和序列Y的公共子序列。最长公共子序列，就是指元素个数最多的公共子序列。例如，若X={A,B,C,B,D,A,B}，Y={B,D,C,A,B,A}，则序列{B,C,B,A}是X和Y的最长公共子序列之一，长度为4。**

**实现动态规划程序，算出最长公共子序列的长度；**

**利用上面产生的矩阵，求出最长公共子序列的内容**。

【源程序】

#include<stdio.h>

#include<string.h>

int str[200][200]; //声明数组，用于储存最长公共子序列的全部计算结果

char reg[200]; //声明数组，用于储存最长公共子序列，别问我为什么把它放这，问就是扔里面有警告

char\* DynamicProg(char \*s1,char \*s2,int l1,int l2) //声明函数，动态规划法计算最长公共子序列

{

int i,j,n=0; //声明变量，控制循环或者计算最长公共子序列长度

//printf("1");

for(i=0;i<=l1;i++) //初始化运算初值：在不引入第二个序列之时，最长公共子序列必然是0

{

str[i][0]=0;

}

//printf("1");

for(j=0;j<=l2;j++)

{

str[0][j]=0;

}

//printf("1");

for(i=1;i<=l1;i++) //计算引入第一个序列前i个字母、第二个序列前j个字母时可以获得的最长序列

{

for(j=1;j<=l2;j++)

{

if(s1[i-1]==s2[j-1]) //若对应序列末尾的字母恰好相同，此时的最长序列长度比两个序列各短1个字母时的最长序列长度大1

{

str[i][j]=str[i-1][j-1]+1;

//printf("1");

}

else //若序列末尾字母不同，则最长公共子序列的长度则是“第一个序列删掉尾字母”和“第二个序列删掉尾字母”两种情形下的最长序列中的最大序列

{

if(str[i-1][j]>=str[i][j-1])

{

str[i][j]=str[i-1][j];

}

else str[i][j]=str[i][j-1];

}

}

}

i=l1,j=l2; //倒推最长公共子序列本身，设定倒推起点和字符串终止符

n=str[i][j];

reg[n]='\0',n--;

while(str[i][j]>0) //在涉及的最长公共子序列长度已经等于0时，子序列的查找已经完成，终止查找

{

while(str[i][j]==str[i-1][j]) //向上寻找最长序列长度相等的情形，并在涉及的长度最小的符合条件的序列位置终止

{

i--;

}

while(str[i][j]==str[i][j-1]) //向左寻找最长序列长度相等的情形，并在涉及的长度最小的符合条件的序列位置终止

{

j--;

}

reg[n]=s1[i-1]; //该位置对应的字符为最长公共子序列中的元素，将其存入字符串数组中，从左上方元素继续进行搜索进程

i--,j--,n--;

}

return reg;

}

int main() //主函数开始

{

char str1[200],str2[200],\*str; //声明字符串变量，前两个用于储存输入的字符串，后一个用于存放运算结果

int ls1,ls2; //声明整型变量，用于存储字符串长度

printf("请输入两个字符串：\n"); //输出提示，按照提示输入两个字符串序列，存入内存中

scanf("%s",str1);

scanf("%s",str2);

ls1=strlen(str1); //计算两个序列的长度

ls2=strlen(str2);

//printf("1");

str=DynamicProg(str1,str2,ls1,ls2); //调用动态规划函数计算最长公共子序列，返回序列，按照题设输出序列及其长度，主函数结束

printf("最长子序列为%s,其长度为%d",str,strlen(str));

return 0;

}

【运行结果】

