**深 圳 大 学 实 验 报 告**

**课程名称： 算法设计与分析**

**实验项目名称： 动态规划—流水线问题**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 陆玉武**

**报告人： 刘俊楠 学号： 2017303010 班级： 01**

**实验时间： 2020/05/07**

**实验报告提交时间： 2020/05/09**

**教务处制**

### 一、实验目的：

* + 1. 掌握动态规划算法设计思想。
    2. 掌握流水线问题的动态规划解法。

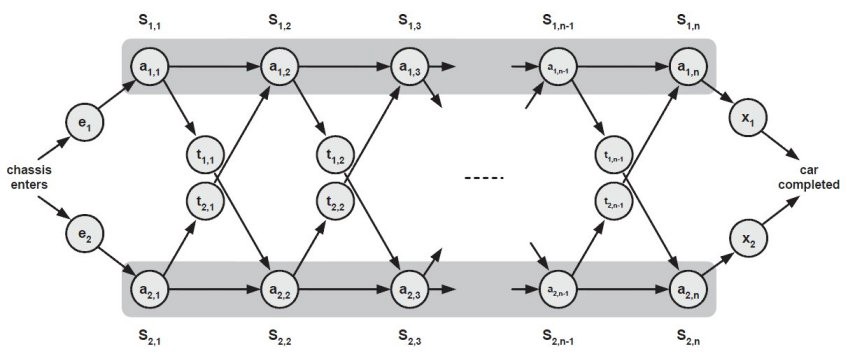
### 二、内容：

1、汽车厂有两条流水线，每条流水线有n个处理环节（station）: S1,1，…，S1,n 和 S2,1，…，S2,n，其中下标的第一个字母表示流水线编号（流水线1和流水线2）。其中S1, j 和 S2, j 完成相同的功能，但是花费的时间不同，分别是a1, j , a2, j 。两条流水线的输入时间分别为e1 和 e2, 输出时间是x1 和 x2。

每个安装步骤完成后，有两个选择：

1）停在同一条安装线上，没有转移代价；

2）转到另一条安装线上，转移代价： Si,j 的代价是ti,j , j = 1，…，n - 1



问题: 如何选择安装线1和安装线2的节点组合，从而最小化安装一台车的总时间？

1. **实验过程**

3.1 对于暴力法求解最小化安装一台车的总时间：

1、需要列出所有可能解，若其共有n个点，两条线，则一共为2n的可能。

2、比较出最短时间，并输出其路线。

3、与动态规划法得出结果作比较。

对于动态规划法求解最小化安装一台车的总时间：

1. 利用自底向上的方法，若要计算n个点，则得计算n-1个点的最短时间，如此递归。
2. 输出最短时间与最短时间所用路线。

**暴力法算法思路：**

在输入了n,e1,e2,factory[line][pointnum][0or1]后，我们需要列举出所有零件可能经过的路线。此时我们可分类成路线中有i个路线1（在代码中路线一为0），并借助next\_permutation（）的库函数来对每种情况进行排序，可得到2n个情况，再对其所用时间进行相加与比较，得出最短时间并输出其路线。

伪代码实现：

//记录当前解的标号

add\_num=0;

//记录当前解所用时间

Timee[pow(2,n)]

//记录每条路线的路线图

Stolee[pow(2,n)][n]

//暴力法

Violence(\*result,n,e1,e2,\*\*stolee)

for i=0~n

for j=0~i

result[j] = 0;

for k=i~n

result[k] = 1;

rank\_all(result, n, i, e1, e2,stolee)

//计算某种情况所有排列

rank\_all(\* result, n, num\_zero, e1, e2, \*\*stolee):

do

for i=0~n

stolee[add\_num][i] = A[i];

calculate(result, n, e1, e2);

add\_num++;

while (next\_permutation(A, A + n))

//计算总和

calculate(\* result, n, e1, e2):

//初始化

timee[add\_num] = 0

for i=0~n

//考虑本来在这轨道 以及变换到该轨道 四种情况计算

if i == 0 && result[i] == 0：

timee[add\_num] = e1

continue

if i == 0 && result[i] == 1：

timee[add\_num] = e2

continue

//如果没变轨道且在一轨道

if result[i] == 0 && result[i - 1] == 0 ：

timee[add\_num] += factory[0][i][0]

//如果没变轨道且在二轨道

else if result[i] == 1 && result[i - 1] == 1：

timee[add\_num] += factory[0][i][0]

//若果在一轨道而且变道

else if result[i] == 0 && result[i - 1] == 1：

timee[add\_num] += factory[1][i][0] + factory[0][i - 1][1]

else if result[i] == 1 && result[i - 1] == 0：

timee[add\_num] += factory[0][i][0] + factory[1][i - 1][0]

这样我们通过伪代码可以分析得到暴力法的时间复杂度为O（2n），因为共有2的n次方个解，并且要进行比对与输出，故复杂度十分十分大，为指数级。

**动态规划法算法思路（Johnson法则）：**

我们假设cost\*为最短时间，factory[line][pointnum][0]为线路line上的pointnum点所消耗时间，factory[line][pointnum][1]为线路line上pointnum点转到另一线路下一点所消耗的转换时间，则cost\* = min(cost[1][n]+x1 , cost[2][n]+x2)，那么根据题意我们知道：

cost[0][0] = e1+factory[0][0][0]

cost[1][0] = e2+factory[1][0][0]

现在我们再来考虑如何计算cost[line][i]，（这里line=0或1；i=2,3,4,...,n）。

我们得出：

cost[0][i]=min(cost[1][i-1]+factory[0][i][0], cost[1][i-1]+factory[0][i][0]+factory[1][i-1][1])

cost[1][i]=min(cost[1][i-1]+factory[1][i][0], cost[0][i-1]+factory[1][i][0]+factory[0][i-1][1])

**伪代码实现：**

cost[0][0] = factory[0][0][0] + e1;

cost[1][0] = factory[1][0][0] + e2;

record[0][0] = 0;//记录选择的流水线

record[1][0] = 1;

for i=1~n :

//若线路一的时间比转换到线路二时间少

if cost[0][i - 1] <= cost[1][i - 1] + factory[1][i - 1][1] :

//在线路一不动

cost[0][i] = cost[0][i - 1] + factory[0][i][0]

record[0][i] = 0

else

//转换到线路1

cost[0][i] = cost[1][i - 1] + factory[0][i][0] + factory[1][i - 1][1]

record[0][i] = 1

//若线路2时间比线路1时间少

if cost[1][i - 1] <= cost[0][i - 1] + factory[0][i - 1][1]:

//在线路2不动

cost[1][i] = cost[1][i - 1] + factory[1][i][0]

record[1][i] = 1

else

//转换到线路2

cost[1][i] = cost[0][i - 1] + factory[1][i][0] + factory[0][i - 1][1]

record[1][i] = 0

//显示最优线路

int line = cost[0][n - 1] <= cost[1][n - 1] ?0 : 1

for i=n-1~0:

输出所选点与流水线

line = record[line][i]

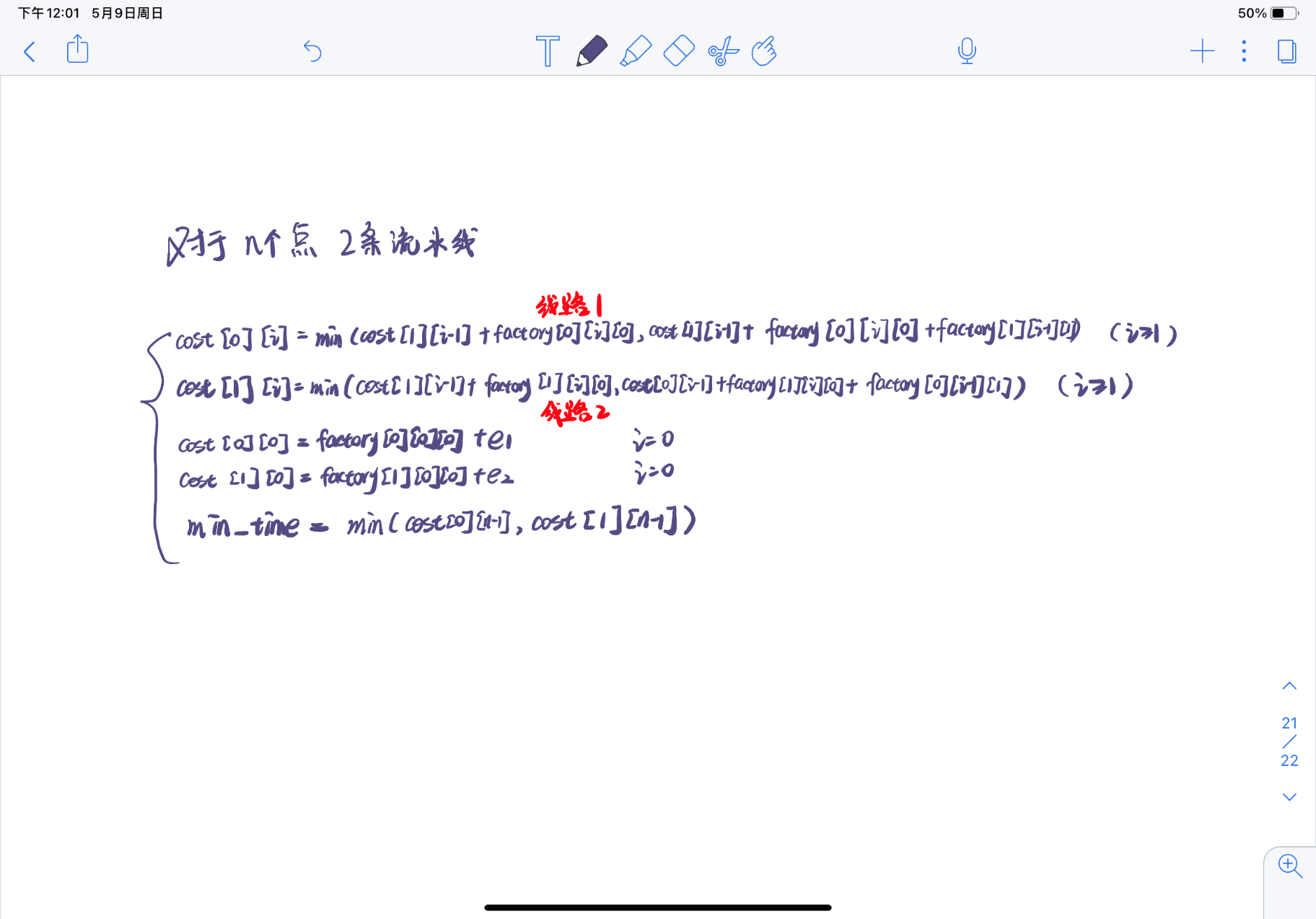
if cost[1][n - 1] > cost[0][n - 1]:

min\_time = cost[0][n - 1]

else

min\_time = cost[1][n - 1]

* 动态规划方程如下图：



* 动态规划算法分析

对伪代码分析可以知道，动态规划算法的时间复杂度为O(n)，是与点数正相关，比起暴力法指数级的时间复杂度，对于此题两条流水线，时间复杂度仅为O（n），故十分快速。

**三．实验结果**

* 分别对n=5,10,15,20,25进行暴力法求解，并验证其正确性：

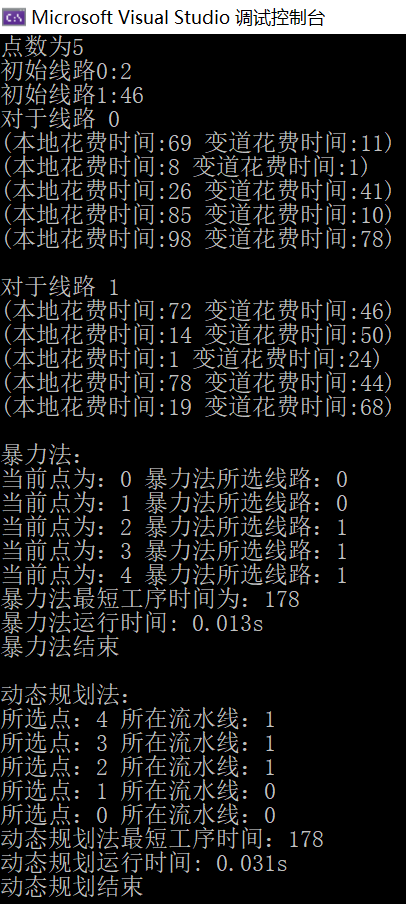


图1

由图1可知，在n=5时对0-4的点分别选择0/0/0/1/1线路是最节约时间的通过计算发现2+69+8+26+41+1+78+19=178，答案正确。

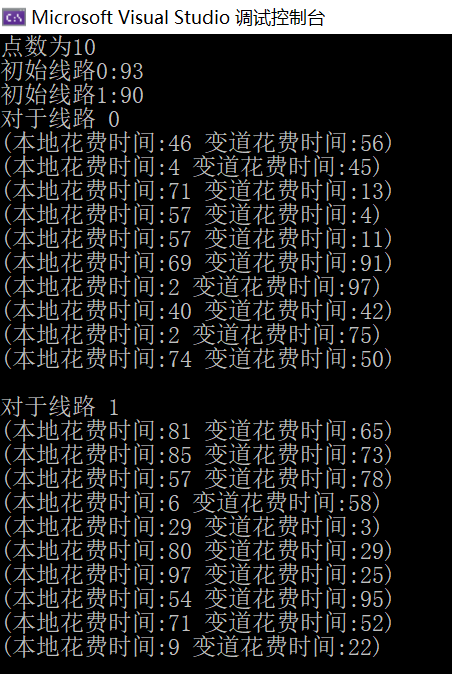
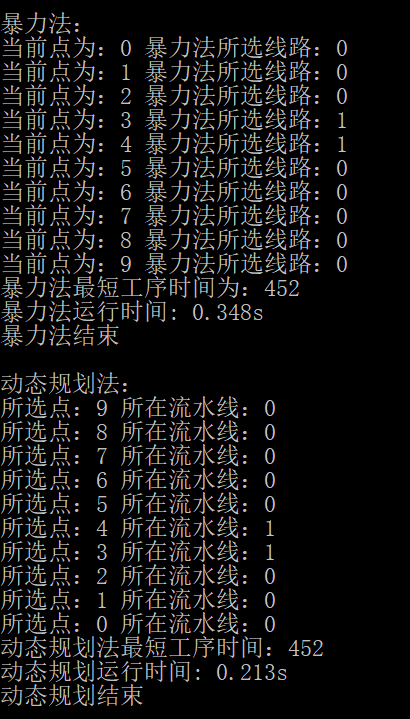


图2

由图2可知，在n=10时对0-9的点分别选择0/0/0/1/1/0/0/0/0/0线路是最节约时间的通过计算发现93+46+4+71+13+6+29+3+69+2+40+2+74=452，答案正确。

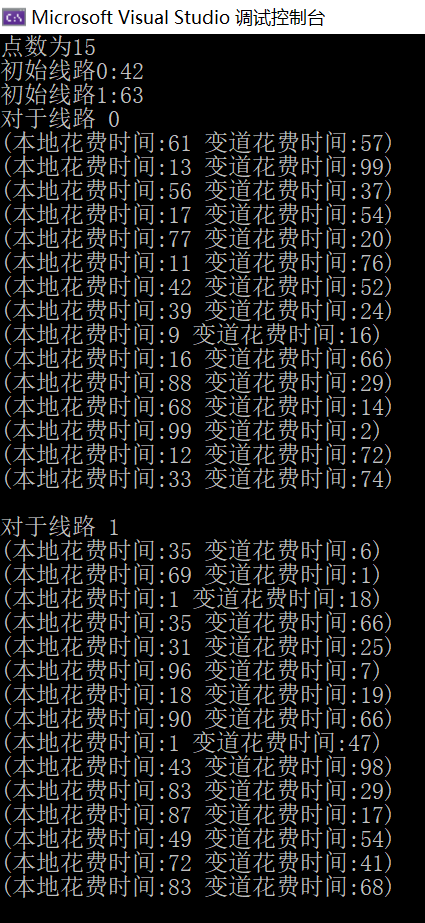
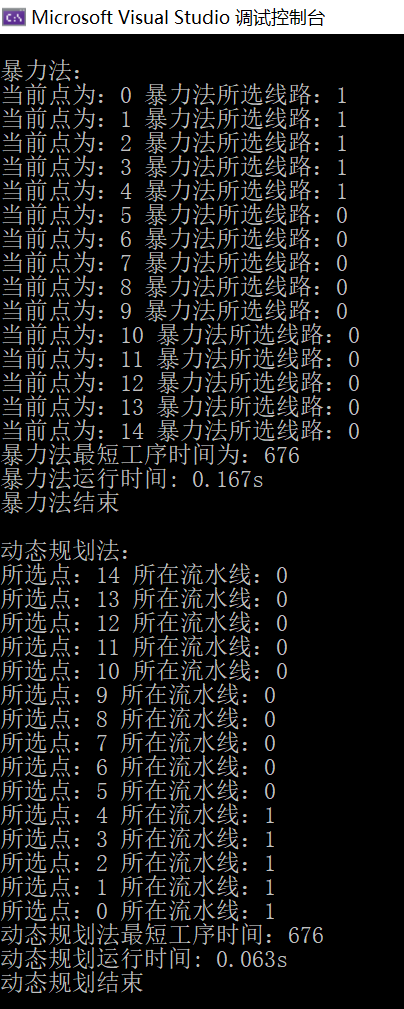
 

图3

由图3可知，在n=15时对0-14的点分别选择1/1/1/1/1/0/0/0/0/0/0/0/0/0/0线路是最节约时间的通过计算发现63+35+69+1+35+31+25+11+42+39+9+16+88+68+99+12+33=676，答案正确。

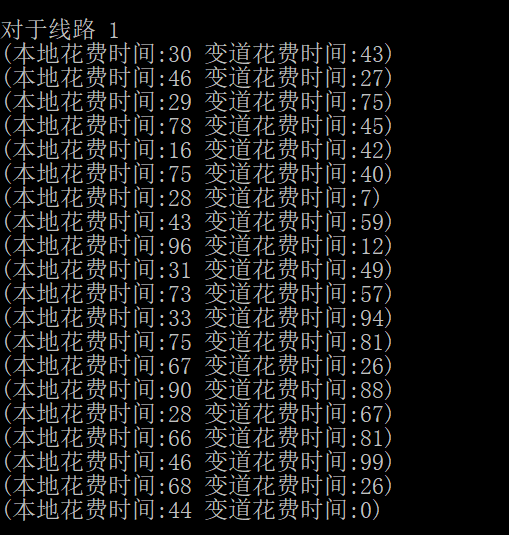
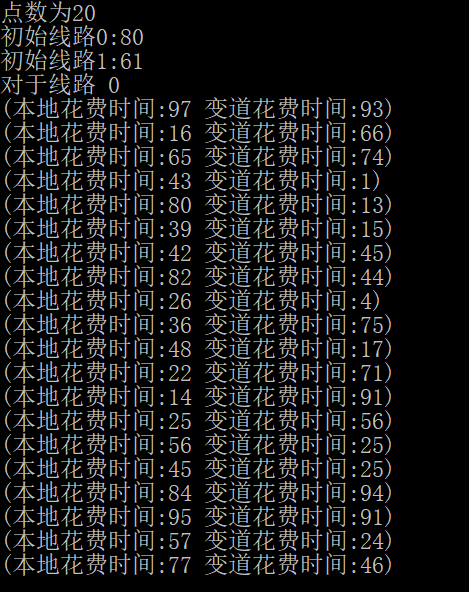


图4

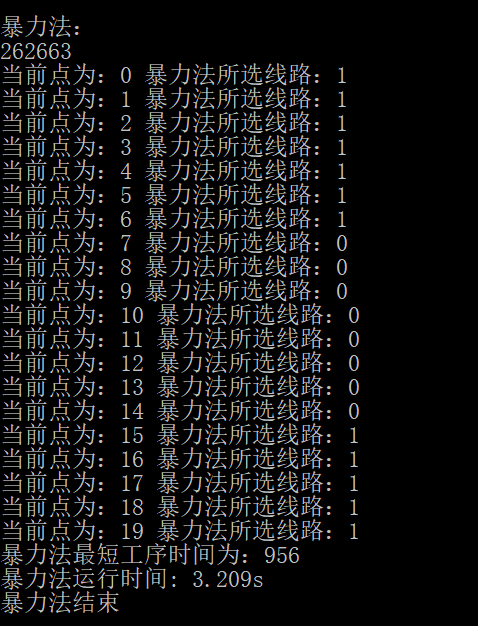


图5

由图4与图5可知，在n=20时对0-19的点分别选择1/1/1/1/1/0/0/0/0/0/0/0/0/0/0/1/1/1/1/1线路是最节约时间的通过计算发现61+30+46+29+78+16+75+28+7+82+26+36+48+22+14+25+56+25+28+66+46+68+44=956，答案正确。

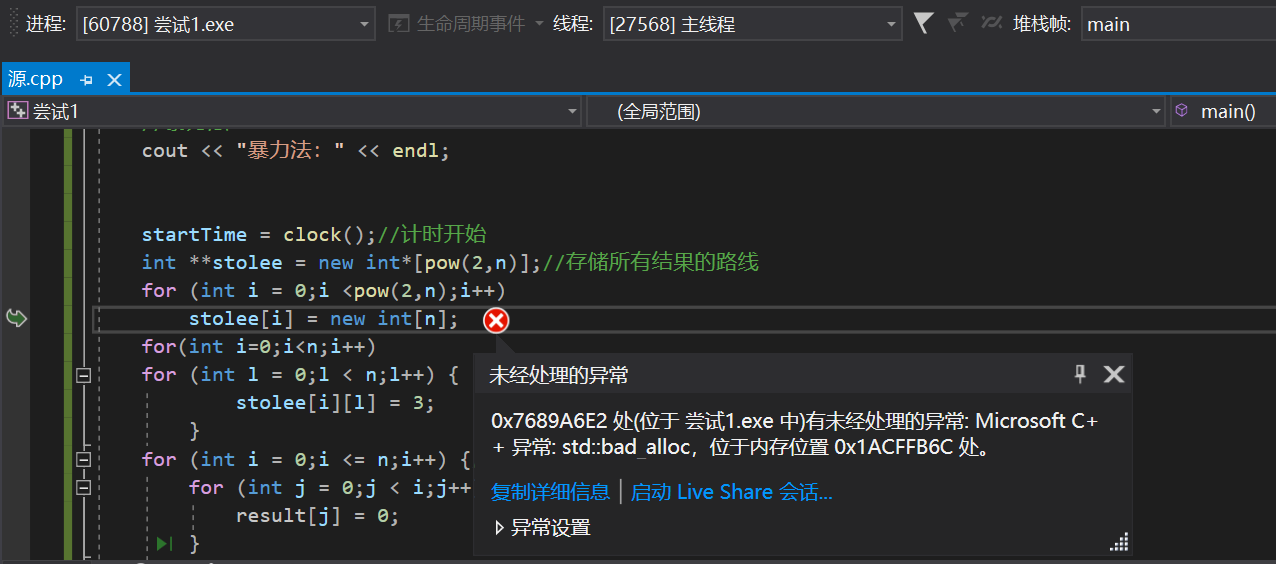


图6

由图6可知，在尝试对25点进行计算时，出现了堆栈溢出的错误，原因是new在生成225\*25的二维数组时堆栈溢出，故暴力法只能计算20个点。

* 对于动态规划算法，分别对n=100000-1000000进行测量时间，并验证正确性。

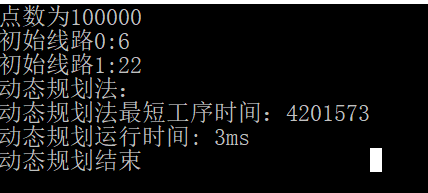


图7

对于n=100000，如图7所示，所用时间为3ms，计算得到最短工序时间为4201573.

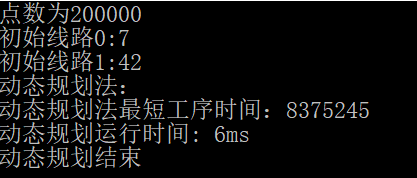


图8

对于n=200000，如图8所示，所用时间为6ms，计算得到最短工序时间为8375245.

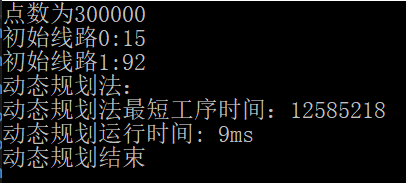


图9

对于n=300000，如图9所示，所用时间为9ms，计算得到最短工序时间为12585218.

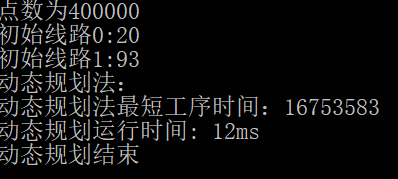


图10

对于n=400000，如图10所示，所用时间为12ms，计算得到最短工序时间为16753583.

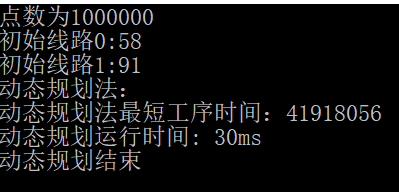


图11

对于n=1000000，如图11所示，所用时间为30ms，计算得到最短工序时间为41918056.

图12

如图12所示，测得理论值与动态规划法测的时间接近，在十万级数据前用时不超过30ms，也证明时间复杂度确实为O(n).

1. **心得体会**

流水线问题是动态规划中的典型问题。通过这个实验，我将对动态规划算法有了更深刻的理解，熟悉了流水线问题。掌握了通过分解和分析问题，寻找状态转移方程，从而使用动态规划更好地解决问题。同时对于流水线问题，纯暴力方法并不能很好地解决多个点的流水线的情况，所以就应该重新思考算法的选择是否合理，此时动态规划的Johnson法则。

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。