复杂度事后分析报告

学生姓名：黄昌盛　　学　号：6109118110　专业班级：计科软件 183 班
实验类型：■ 验证 □ 综合 □ 设计 □ 创新　实验日期：2021.3.17　实验成绩：

一、题目描述和要求

货郎担问题是这样一个问题：给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路。它是组合优化中的一个 NP 困难问题，在运筹学和理论计算机科学中非常重要。

以教材 p6 算法 1.4 的货郎担问题为例，进行事后复杂度分析，最后写一个报告。

二、报告步骤

1、运行时间提取

利用 C++的 clock()函数，在进行调用解决货郎担问题的函数之前之后，分别记录算法的开始时间和结束时间，然后用结束时间减去开始时间乘 1000 与是 1 秒内 CPU 运行的周期数 CLOCKS_PER_SEC 做除法得到以毫秒为单位的运行时间。

```
clock_t start = clock(), end;
salesman_problem(n, mincost, path, c); // 计算最小花费函数
end = clock();
// 输出运行时间
cout << "n" << " = " << n << "   "
<< int(end - start) * 1000 / CLOCKS_PER_SEC << "ms" << endl;
```

2、随机数据生成

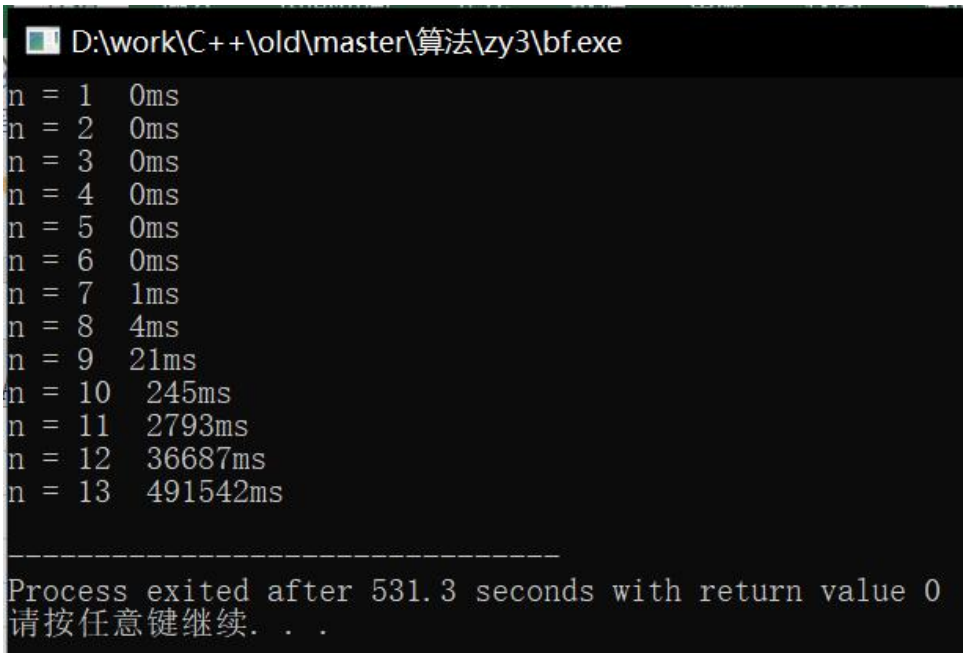
利用系统当前时间做种子，使用 C++11 中新加入的特性 mt19937 产生伪机数,获得费用矩阵的随机数据。

```
unsigned seed = std::chrono::high_resolution_clock::now()
                    .time_since_epoch().count();
```

```
3. std::mt19937 mt(seed);
4. // 随机数据生成，存入费用矩阵
5. for (int i = 0; i < n; i++)
6.     for (int j = 0; j < n; j++)
7.         c[i][j] = 100 + mt() % 100;
```

3、自动输出批量的不同规模及其运行时间数据

输出规模 n，和对应运行时间 t，运行结果如下：



4、输出数据放入 Excel

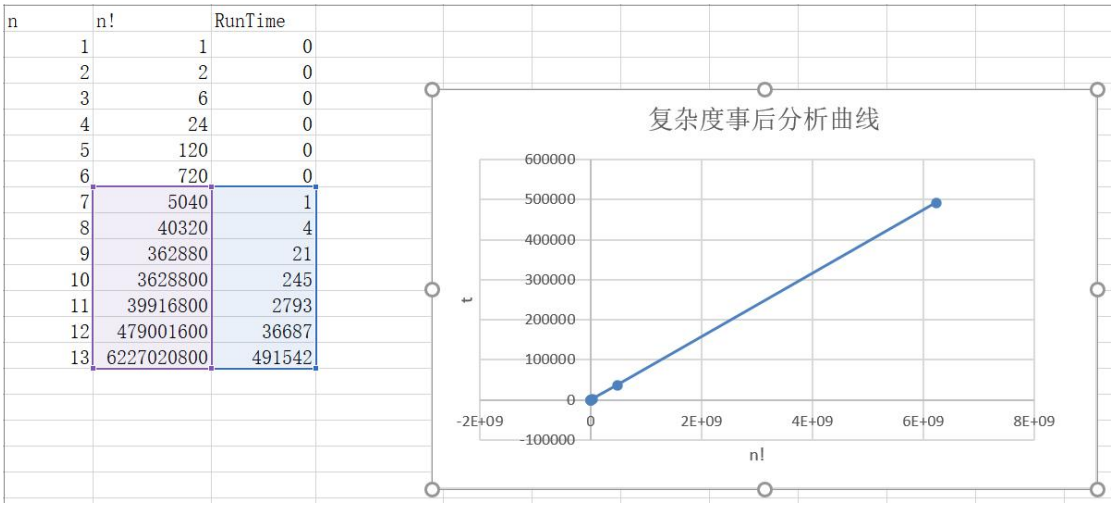
	A	B
1	n	RunTime
2	1	0
3	2	0
4	3	0
5	4	0
6	5	0
7	6	0
8	7	1
9	8	4
10	9	21
11	10	245
12	11	2793
13	12	36687
14	13	491542

5、Excel 数据表格及曲线生成

使用 Excel 计算出 n 的阶乘，这里注意要排除时间为 0 的点，使用 n!和时间 t 绘制曲线：

n	n!	RunTime
1	1	0
2	2	0
3	6	0
4	24	0
5	120	0
6	720	0
7	5040	1
8	40320	4
9	362880	21
10	3628800	245
11	39916800	2793
12	479001600	36687
13	6227020800	491542

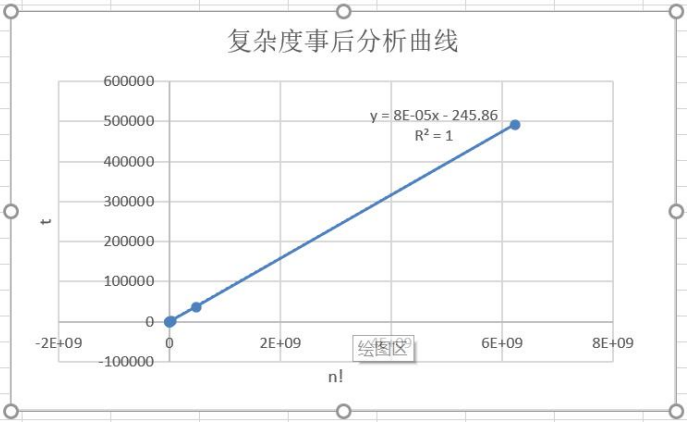
观察分析图像发现两者存在线性关系。



6、EXcel 趋势线的生成及复杂度的提取

采用 n!和运行时间 t 进行拟合

n	n!	RunTime
1	1	0
2	2	0
3	6	0
4	24	0
5	120	0
6	720	0
7	5040	1
8	40320	4
9	362880	21
10	3628800	245
11	39916800	2793
12	479001600	36687
13	6227020800	491542



由结果可知，t = 8E-5*n!，R^2 = 1,该算法的时间复杂度为 O(n!)

7、算法的分析

由程序运行结果可知,虽然穷举法很简单易懂,对个城市枚举一个的全排列,在其中找到最优解，这样确实能够找到最优解，但是复杂度达到(n-1)!，当城市数量很小时，运行时间是很小，但当城市数量大于 12 个时，我们就不能够在短时间内求解，n=13 时我的电脑跑了 8 分多钟，显然这不是一个好的方法。可以考虑其他如状压 dp 的优化算法等。

三、程序代码

```
#include <bits/stdc++.h>
using namespace std;
const int INF = 0x3f3f3f3f;
unsigned seed = std::chrono::high_resolution_clock::now()
                  .time_since_epoch().count();
std::mt19937 mt(seed);
int t, n, path[100];

void salesman_problem(int n, double &min, int t[], vector<vector<int>> c) {

    int p[n], i = 1;
    double cost;
    for (i = 0; i < n; i++) p[i] = i;

    min = INF;
    do {
        cost = 0;
        for (int i = 0; i < n - 1; i ++)
            cost += c[p[i]][p[i + 1]];
        cost += c[p[n - 1]][p[0]];
```

```
        if (cost < min) {
            for (i = 0; i < n; i++)
                t[i] = p[i];
            min = cost;

            // 打印当前最优解
//          cout << min << endl;
//          for (int i = 0; i < n; i++) {
//              cout << t[i] << "->";
//          }
//          cout << t[0] << endl;
        }
    } while (next_permutation(p, p + n));

}

int main() {
    while (n < 13) {
        n++;
        vector<vector<int> > c(n, vector<int> (n));
        double mincost = 0;
        // 随机数据生成，存入费用矩阵
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                c[i][j] = 100 + mt() % 100;

        clock_t start = clock(), end;
        salesman_problem(n, mincost, path, c); // 计算最小花费函数
        end = clock();
        // 输出运行时间
        cout << "n" << " = " << n << "   "
        << int(end - start) * 1000 / CLOCKS_PER_SEC << "ms" << endl;

        // 打印答案
//      cout << mincost << endl;
//      for (int i = 0; i < n; i++) {
//          cout << path[i] << "->";
//      }
//      cout << path[0] << endl;
    }
}
```