NP完全问题实际应用题目

某街道给封控在家的居民们发来了大礼包，大礼包里包含 N 种不同的瓜果蔬菜。居民们在探究如何更有效地把这些原材料组合做出更美味的菜肴。假定给出大小为$N \times N$的矩阵 a，矩阵中每个位置的元素 a[i][j]∈ $[0,1]$代表第 i 种蔬菜和第 j 种蔬菜的排斥程度，a[i][j]越小证明这两种蔬菜越适配，越大越排斥。假设 N=3 时，给定对称矩阵如下：

	1	2	3
1	0.0	0.6	0.2
2	0.6	0.0	0.1
3	0.2	0.1	0.0

例如，对角线上的元素全都是 0，证明第 i 种蔬菜和自己完美搭配，但是第 1 种蔬菜和第 2 种蔬菜的匹配程度不够好。假设居民选择某种蔬菜集合S，定义该集合的总排斥度为$T = \sum_{i \in S} \sum_{j>i, j \in S} a[i][j]$。在上述给定匹配矩阵中，选取全部三种蔬菜$S = \{1,2,3\}$时的总排斥度$T_S = 0.6 + 0.2 + 0.1 = 0.9$。

问题 1： 给定蔬菜总数N与对应大小为$N \times N$的排斥矩阵 a、及希望搭配的蔬菜总类个数M，即$|S| = M$，其中$|S|$代表集合S的元素个数。求解M种蔬菜可以组成的最小总排斥度T_S、以及对应的蔬菜集合S，并给出对应的时间、空间计算复杂度。M=1/2/3 时解如何？M∈ $[1, N]$时解如何？

问题 2： 给定蔬菜总数N与对应大小为$N \times N$的排斥矩阵 a、及希望得到的最小总排斥度上限L，求解$\max|S|$，$s.t. T_S \le L$。

问题 1、2 是否可以在多项式时间内求解，如果可以，请给出一个多项式时间的可行解；如果不可以，则给出一个近似的最优解，并给出对应的时间、空间计算复杂度。

Bonus：(1) 对于问题 1、2，尝试给出空间换时间的优化解法或对复杂度进行常数优化；(2) 尝试分析取得最优解的概率。

问题一

1.理论证明其NP完全性

- 证明此问题属于NP

  构造这样的非确定图灵机：

  1. 预设一个最小排斥度t=m^2(m组蔬菜总排斥度必小于这个值)
  2. 非确定的选择矩阵a中的由m种蔬菜组成的子矩阵
  3. 求这个子矩阵之间的总适配度，需计算C_m^2组，O(m^2)的时间复杂度
  4. 检查总适配度是否小于最小适配度
  5. 若是，则接受并更新最小适配度；否则，则拒绝

- 证明某个NP完全问题可归约到此问题

  1. 原问题可抽象为这样一个问题：在边带权完全图N中寻找m个结点的完全子图，使其边权值和最小
  2. 将团问题多项式时间归约到问题1：
     - 将团问题中的G转化为这样的完全图G’：顶点不变，若在G中两点无边，则该两点的边权值设为1；若在G中两点有边，则该两点的边权值设为0。（扫描所有边，最多O(n^2)的时间复杂度，n为顶点数，在多项式时间内完成）
     - 若要求解团问题，则是在G‘中寻找k个结点的完全子图，使其边权值和最小，若最小值为0，则有解；反之则无解。

  故此问题是NP完全问题，目前还没有在多项式时间内求解的方法

2.具体实例求解

1.m=1时

此时，选任何一种蔬菜都行，因为自己和自己并不排斥，总排斥度均为0。

2. m=2时

采用两层嵌套循环遍历，寻找边之间的最小值。

需要说明的是，这样求出来的最小值只是其中一种搭配，当搭配不唯一时，需要解决任何输出所有的搭配。

将搭配的两种蔬菜做成两个vector，当遍历到的适配度小于当前最小值，需要更新最小值，并清空错误的搭配vector；当遍历到的适配度小于等于当前最小值时，把这两个蔬菜编号记到vector中。因为每次两个vector都会加，所以两个vector序号相同的蔬菜两两即是一个搭配。

```
代码详见1_1.cpp
```

【时间复杂度】

两层嵌套循环，循环里的更新、清除、添加操作都是常熟级的，故时间复杂度O(n^2)。

【空间复杂度】

需要用n*n矩阵存储蔬菜的适配度，还有两个vector用来存蔬菜搭配，最坏不过是把搭配都存进去，也是n*n，其他的都是常数级变量。故空间复杂度O(n^2)。

3. m=3时

思路同上，采用三层嵌套循环遍历，寻找三个蔬菜之间适配度和的最小值。将搭配的三种蔬菜做成三个vector，更新和清空操作同上。

```
代码详见1_2.cpp
```

【时间复杂度】

三层嵌套循环，循环里的更新、清除、添加操作都是常熟级的，故时间复杂度O(n^3)。

【空间复杂度】

需要用n*n矩阵存储蔬菜的适配度，还有三个vector用来存蔬菜搭配，最坏的情况是把所有搭配都存进去，是n^3，其他的都是常数级变量。故考虑所有搭配情况的空间复杂度是O(n^3)。但如果只是求最小适配度，则不需要考虑vector的空间，空间复杂度是O(n^2)。

3. 近似算法：贪心法

该算法的大致思路是，先找出两两之间的最小适配度，把这两种蔬菜放到蔬菜vector中，再寻找剩余每个蔬菜和这两种蔬菜的适配度和最小值。以此类推，即在未知点集合里找和已知点边权值和的最小值，再把这个点放到已知点集合中，这样一直循环，直到蔬菜vector数目达到给定值m。

显然这样得到的是局部最优解，不是整体最优解。

在具体的代码实现上，有一些操作和数据结构的选取需要进行说明。首先，对于某些循环操作，比如找到两两之间的适配度，可以在读入矩阵的这个循环中进行，减少遍历数，但要去掉自身0适配度的无效数据以及另外半矩阵的重复数据。然后就是未知点集合的数据结构采用set，因为点总是需要遍历，顺序就没那么重要了，同时每次要删除某编号的点，且编号唯一，故使用无序、不可重复的数据结构set。

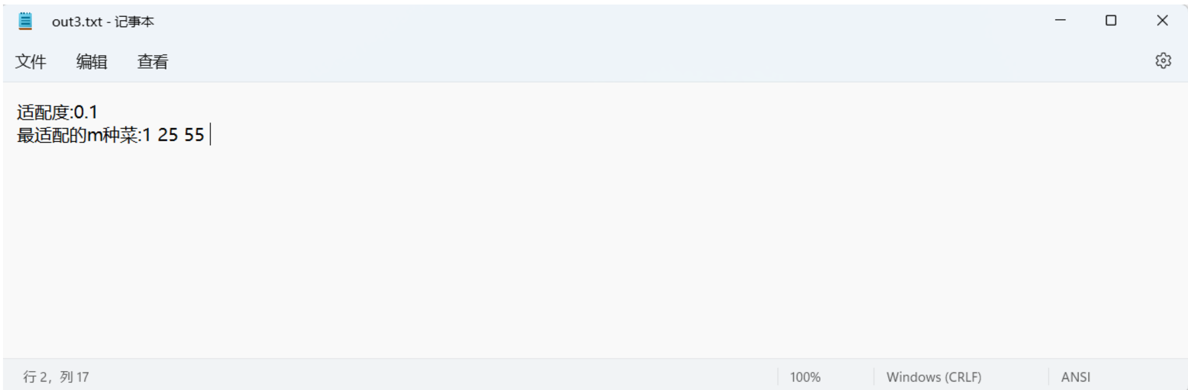
```
代码详见1_3_1.cpp
```

【问题】在测试实例时，发现了这样一个问题：如果多个结果适配度和相同，即能选取到的蔬菜并不唯一，如何选出较为好的那一个加入已知点集合中？

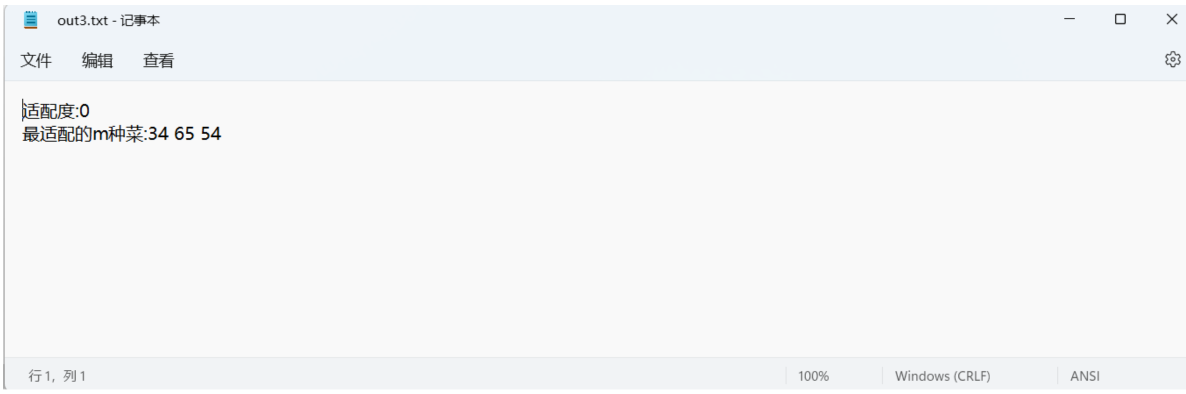
【改进】可以算出先各蔬菜与其他蔬菜适配度的平均值（求和也行，因为总数一样），在有多个适配度和相同的结果时，将求和最低的那个加到已知点集合中。因为这些点到已知点的权值相同，求和最低的那个说明它与后面点整体有更好的适配度，更有可能求到最优解。不过这时，找两两之间的适配度的循环就不能放到读入循环里了，因为适配度平均值数组还未求出，无法找出最优的那个，但可将求数组放到该循环中。在具体实例（见data.txt）中，未改进前求得的m是0.1；改进后求得的m是0，从一定程度上说明算法得到了优化。后续又用随机取样法，对比了改进前后的正确率，详见Bonus）。

```
代码详见1_3_2.cpp
```

改进前（1_3_1.cpp）



改进后（1_3_2.cpp）



【时间复杂度】

最开始是一个O(n)的set生成，然后是两个两层嵌套循环，读入和其他操作都是常数级的，故时间复杂度O(n^2)；之后有三层嵌套循环，第一层遍历添加m种蔬菜，第二层遍历未知点集合，第三层计算未知某点到已知点适配度的和，最坏情况时间复杂度O(n^3)。故时间复杂度O(n^3)。

【空间复杂度】

需要用n*n矩阵存储蔬菜的适配度，还有一个vector用来存已知点（蔬菜）和一个set来存未知点（蔬菜）（改进版还有一个适配度和数组），最坏的情况是把所有蔬菜都存进去，都是n的空间复杂度，其他的都是常数级变量。故空间复杂度O(n^2)。

【Bonus】

- 常数优化：

  【测试工具选取】因为编译器自带的控制台输入输出，在涉及多输入时不太好确定开始计时时间，故~~白嫖~~采用可上传题目和数据的oj，链接见附录

1. IO优化

   关闭同步，加快cin，cout的速度，达到scanf、printf的速度。

   添加函数：

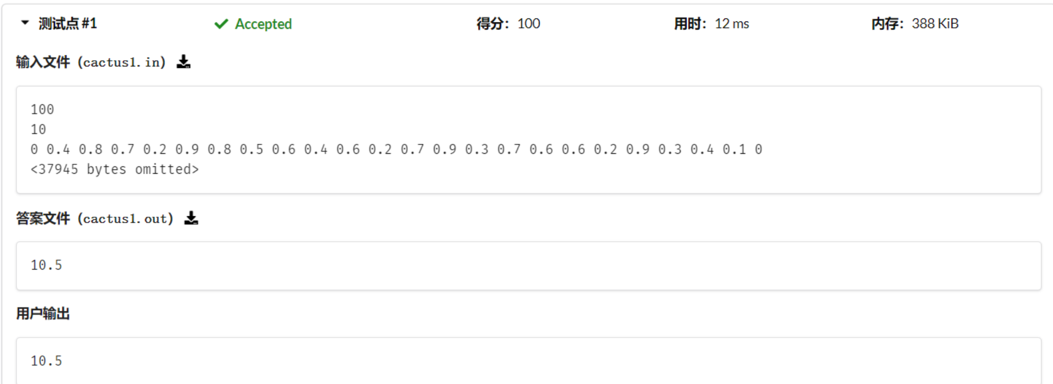
   ```
   std::ios::sync_with_stdio(false);
   ```

   这个函数是一个“是否兼容 stdio”的开关，C++ 为了兼容 C，保证程序在使用C的IO的时候与C++的IO函数不发生混乱。因此，在使用时，应保证只有cin和cout。

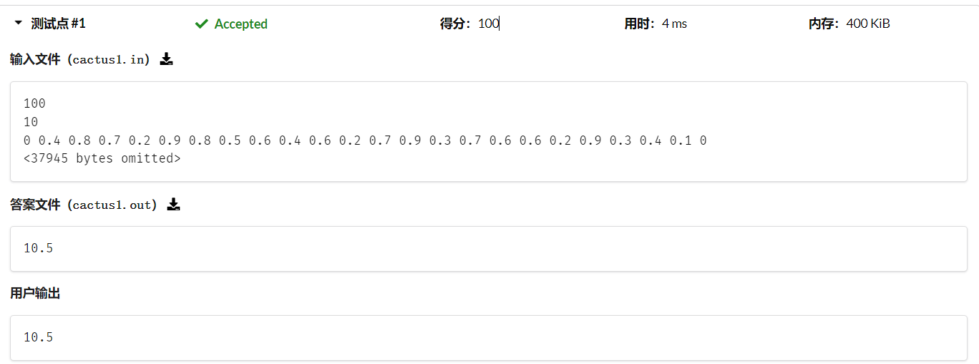
   ```
   代码详见1_4_1.cpp
   ```

   测试，以100种蔬菜中选10种为例，具体数据见cactus.zip。

   优化前：



   优化后：快了近三倍



2. register优化

   【原理】电脑CPU有一个高速缓存器（cache），速度非常快，但占用内存小，一个变量加上register后，这个变量的存放位置就会在高速缓存器里。由于cache命中率的考虑，一般用于频繁修改的变量（如循环中的变量）。但不是所有的循环都适合这么干，要局部性好的。一般最内层循环可以这样干，外层的反而会因为命中率低而使时间变慢。

   【操作】

如：

```
for(int i=1;i<=n;i++)
{
    float s=0;
    for(int j=1;j<=n;j++)
    {
        cin>>a[i][j];
        s+=a[i][j];
    }
    sum_v.push_back(s);
}
```

应变为：

```
for(int i=1;i<=n;i++)
{
    float s=0;
    for(register int j=1;j<=n;j++)
    {
        cin>>a[i][j];
        s+=a[i][j];
    }
    sum_v.push_back(s);
}
```

register加在外层循环上反而会变慢

相比于初始的12ms，速度也有明显提升

```
代码详见1_4_2.cpp
```

3. 自增使用++i

确实比i++要快一点

```
代码详见1_4_3.cpp
```

4. 缝合怪


```
代码详见1_4_4.cpp
```

- 概率分析

1. 在进行概率分析之前，首先我们要随机出一种情况，用暴力求解出正确的答案。

   这里面就有一个很棘手的问题，到底选多少种蔬菜也是随机的，即循环嵌套的层数不定。

   回到问题本身，我们要做的是从n种蔬菜中取m种，计算m种蔬菜之间的适配度。

   我们可以考虑一个n长的二进制串，那一位为1就代表那一种被选。

   这个二进制串恰恰可以用当前的方案数来表示，比如方案1（0001）代表选蔬菜1，方案7（0111）代表选蔬菜1，2，3，以此类推。

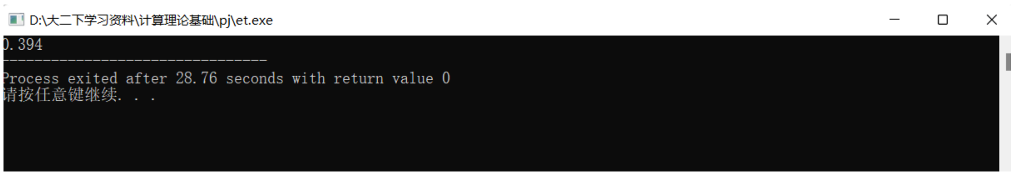
   方案数即1~2^n, m种蔬菜即这个二进制串有m个1。

   所以整体思路就是，生成1~2^n个数，判断每个数的二进制表示是否有m个1。若有，则将这m种蔬菜两两搭配，计算总适配度，与最小值对比，小则更新。

   ```
   代码详见1_5_1.cpp
   ```

2. 有了暴力算法之后，再将原有的近似算法生成结果与暴力算法的结果进行对比。循环并随机生成结果，将适配的次数与总次数相除，即得到适配率。

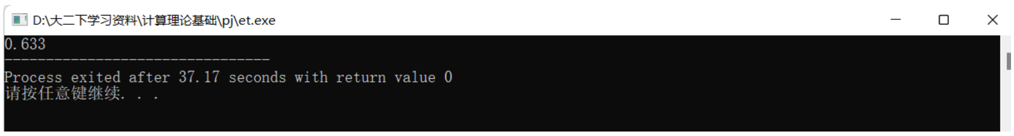
   - 在最开始计算的时候，发现适配的概率奇低，只有40%左右。

     将没有配上的结果相减后输出才想起来计组学的东西，double是用64位二进制数存储的，表示十进制数时会存在精度问题，所以，适配两结果时，相等的判定应该是两数相差在一定范围内，而不是简单的相等。

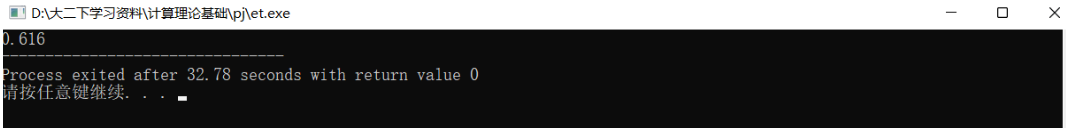
   - 需要注意的是，因为数最大能开64位（long long），所以用二进制表示方案数时，只能随机64以内的总蔬菜数。实际上呢，超过25的蔬菜数循环个1000次，时间就已经撑不住了。所以本实验测试的n限制在3~25。

   - 优化算法的适配度，经多次模拟后，预估在64%左右。

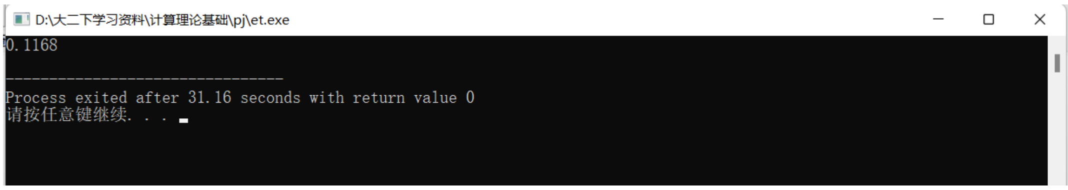
3. 副产品

   检验一下算法算法有优化，预估在62%左右。



能优化，但是只能优化一点点，不能优化多了。

来都来了，顺便看看到底总适配度，到底差多少



平均每次，总适配度要优化0.1左右

```
2,3代码详见1_5_2.cpp
```

问题二

1. 理论证明其NP完全性

- 证明此问题属于NP

  构造这样的非确定图灵机：

  1. 预设一个最大蔬菜数t=1(至少能搭配一种蔬菜)
  2. 非确定的选择一个数m，非确定的选择矩阵a中的由m种蔬菜组成的子矩阵
  3. 求这个子矩阵之间的总适配度，需计算C_m^2组，O(m^2)的时间复杂度
  4. 检查总适配度是否小于上限L
  5. 若是，则接受并更新最大蔬菜数t；否则，则拒绝

- 证明某个NP完全问题可归约到此问题

  1. 原问题可抽象为这样一个问题：在边带权完全图N中，给定一个值，寻找边权值小于等于该值的结点数最多的完全子图
  2. 将最大团问题多项式时间归约到问题1：
     - 将最大团问题中的G转化为这样的完全图G’：顶点不变，若在G中两点无边，则该两点的边权值设为1；若在G中两点有边，则该两点的边权值设为0。（扫描所有边，最多O(n^2)的时间复杂度，n为顶点数，在多项式时间内完成）
     - 若要求解最大团问题，则是在G‘中寻找边权值小于等于0（即是0）的结点数最多的完全子图

  简单说明一下最大团问题的NP完全性：(k)团问题归约到最大团问题，就是将k与求出的最大团点数m做比，若m大于等于k，则有k团；反之则无。

故此问题是NP完全问题，目前还没有在多项式时间内求解的方法

2.近似算法：贪心法

思路与问题一相似，只不过此时这时加边的循环没有给定的m，而是由当前还剩余的适配度与能添加点权值和的最小值来决定。如果前者小于后者，则循环结束；反之则相减、更新、继续循环。

需要注意的是，结果至少得是1，因为自己和自己适配度为0。

```
代码详见2_1.cpp
```

【时间复杂度】

最开始是一个O(n)的set生成，然后是两个两层嵌套循环，读入和其他操作都是常数级的，故时间复杂度O(n^2)；之后是一个三层嵌套循环，第一层依次添加蔬菜，第二层遍历未知点集合，第三层计算未知某点到已知点适配度的和。最后计算总适配度，超过阈值就跳出循环。最坏情况时间复杂度O(n^3)。故时间复杂度O(n^3)。

【空间复杂度】

需要用n*n矩阵存储蔬菜的适配度，还有一个vector用来存已知点（蔬菜）和一个set来存未知点（蔬菜）（改进版还有一个适配度和数组），最坏的情况是把所有蔬菜都存进去，都是n的空间复杂度，其他的都是常数级变量。故空间复杂度O(n^2)。

【Bonus】

- 概率分析
    1. 我们依然需要设计暴力求解函数，大致框架与问题一相同，遍历1~n的最小适配度，找出不大于给定值的最大值即可。不过当然不是依次遍历，因为最小适配度随着点数的增加而增加，我们自然而然可以想到用二分法来找最小值。

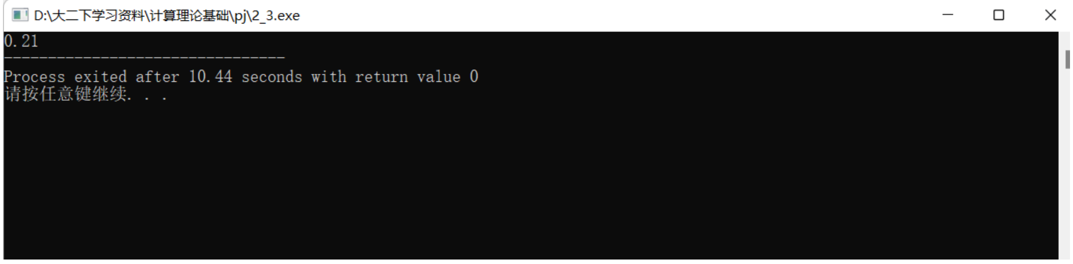
        ```
        代码详见2_2.cpp
        ```

    2. 计算最优解概率则与问题一相同了。

        ```
        代码详见2_3.cpp
        ```

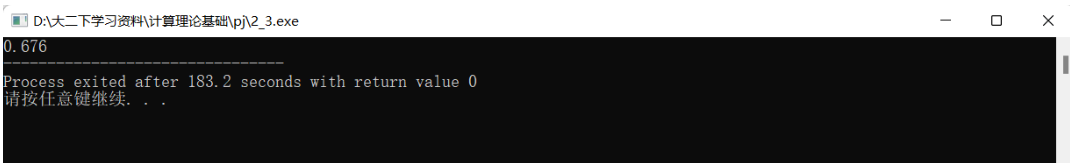
        【意外收获】

        计算概率是发现命中率特别拉跨，只有20%左右。



打表发现大部分的数据都比正确结果差一个值，那我不妨结果直接加一好了。

+1后：准确率68%左右。



其实还可以这样想，既然我+1了，那我不妨把算法退化一点，说不定+1更能命中。

但实际退化后，准确率还没有这个高。如果原算法不+1命中率能高点，说不定退化会有效果，奈何命中率太低了。

- 常数优化内容与问题一相同，不缀述

```
代码详见2_4.cpp
```

其他文件说明

matrix.cpp用于随机生成一个n*n的蔬菜矩阵，存在matrix.txt中

参考资料及工具

1. C++中set用法详解 ——CSDN博客
2. 测速oj
3. 算法竞赛C++常用技巧——输入输出优化（防止TLE） ——CSDN博客
4. 常数优化 ——博客园 (cnblogs.com)
5. 为什么说++i的效率比i++高？ —— 腾讯云
6. 从一个数组中找出 N 个数，其和为 M 的所有可能--最 nice 的解法 ——CSDN博客
7. C++ 实现随机小数的几种方法 ——CSDN博客 c++ 随机小数
8. 最大团问题 ——百度百科