

大数据计算基础课程报告

题 目 ：超图数据结构的实现以及基于CMC算法的顶点覆盖近似解

专 业 数据科学与大数据技术

学 号 L170300901

姓 名 卢兑玧

课 程 大数据计算基础

日 期 2021-01-14

# 摘 要

关键词： 超图、顶点覆盖、分支界限、CMC算法、受限顶点覆盖

本次大作业的题目是超图数据结构的实现以及基于CMC算法的顶点覆盖近似解。本次大作业基本思路如下：许多情况下，简单的二元关系已经不足以表达数据中的关系，如同一位教授可能参与多篇论文，而一篇论文也会有多个作者, 在简单图中，容易丢失同一篇文章的多个作者。但是对于超图来说，可以描述更为复杂的关系信息。超图（Hyper-Graph），是一种广义的图，其特点是一条超边可以连接多个点。首先笔者实现了超图的数据结构，通过集合来表示每一条边，另外再通过一个特殊的集合表示顶点，并且实现了边和顶点的增加、删除、转换操作。另外笔者使用spark框架和分支界限法实现了求解超图的顶点覆盖的问题。并参考论文中的CMC算法实现了求解受限顶点覆盖的近似解。由于此问题链接中提到的数据集无法被其项目组之外的人员接触，因此笔者使用了自己生成的数据集。

**目 录**

[摘 要 I](#_Toc27338278)

[第1章 绪 论 4](#_Toc27338279)

[1.1 研究问题的背景 4](#_Toc27338280)

[1.2 研究问题的挑战 4](#_Toc27338281)

[1.3 当前研究工作的不足之处 4](#_Toc27338282)

[1.4 本文的工作要解决的问题以及方法 4](#_Toc27338283)

[1.5 本文的贡献 4](#_Toc27338284)

[1.6 章节安排 4](#_Toc27338285)

[第2章系统/方法框架 4](#_Toc27338286)

[2.1 系统框架 4](#_Toc27338287)

[2.2 各部分简介 4](#_Toc27338288)

[第3章技术一 4](#_Toc27338289)

[3.1 4](#_Toc27338290)

[第4章技术二 4](#_Toc27338291)

[第5章...... 4](#_Toc27338292)

[第5章实验 5](#_Toc27338293)

[5.1 实验设计 5](#_Toc27338294)

[5.2 对比实验 5](#_Toc27338295)

[5.3 实验结果 5](#_Toc27338296)

[5.4 实验受参数的影响 5](#_Toc27338297)

[第6章相关工作 5](#_Toc27338298)

[第7章结论 5](#_Toc27338299)

# 第1章 绪 论

## 1.1 研究问题的背景

我们经常用图来表示我们的研究对象。但是，在许多现实问题中，我们的研究对象之间的关系更为复杂。将复杂的关系压缩为二元的关系将不可避免地导致信息丢失。因此，许多人考虑使用超图来表示这种复杂关系。

在数学中，超图是图的一般化，其中边可以连接任意数量的顶点。而在普通图中，一条边只能连接两个顶点。形式上，超图可以表示为 H =（X，E），其中X是一组称为节点或顶点的元素，而E是X的一组非空子集，称为超边或边。顶点集的大小称为超图的顺序，边集的大小称为超图的大小。

超图可以看作是一种特殊的图，它由一组节点和一组超边定义，其中超边连接两个或多个节点。 超图是用于对复杂的现实世界数据建模的有用工具，这些数据包括对象之间的高阶关系。 他们已经在多种应用中进行了研究，包括计算机视觉，VLSI CAD，生物信息学和社交网络分析。比如，科学家们已经提出了基于超图的图像检索和标记系统，其中利用超图来编码难以在普通图中表示的附加信息。

## 1.2 研究问题的挑战

由于这个问题是一个NP类问题，运算的时间复杂度取决于输入数据的数据规模，在搜寻算法方面遇到了很大的障碍，很多的算法只能适用于规模较小的数据集。另外，在实现受限顶点覆盖的搜索算法方面，参考了论文中的方法，但是在研究算法方面仍然存在有一些疑问。

## 1.3 当前研究工作的不足之处

所实现的算法仍然受到数据集规模的限制，无法运行规模特别大的数据集。对于受限顶点覆盖的求解算法只是按照伪代码实现了算法，但是一些具体的细节还有待更深一步的理解。

## 1.4 本文的工作要解决的问题以及方法

除了最初的解决数据集的问题之外。本文主要的问题有两个在之前的小节中也提到过，由于这是一个NP类问题，在翻阅很多的资料之后，最终选择了使用分支界限法的思想来实现算法。受限顶点覆盖的求解则直接使用了论文中的方法。

## 1.6 章节安排

本章主要介绍了这篇论文的研究背景和基本的研究方向。第二章将介绍所用的系统和方法框架，接下来的两章将介绍具体的实现算法。第五章介绍具体的实验过程。第六章将要介绍相关的研究技术。

# 第2章系统/方法框架

## 2.1 系统框架

**Centos7：**

CentOS Linux发行版是一个稳定的，可预测的，可管理的和可复现的平台，源于Red Hat Enterprise Linux（RHEL）依照开放源代码（大部分是GPL开源协议）规定释出的源码所编译而成。

在VMware中安装centos7操作系统的虚拟机，用于接下来的spark的安装和配置。

**Spark：**

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark是UC Berkeley AMP lab (加州大学伯克利分校的AMP实验室)所开源的类Hadoop MapReduce的通用并行框架，Spark，拥有Hadoop MapReduce所具有的优点；但不同于MapReduce的是——Job中间输出结果可以保存在内存中，从而不再需要读写HDFS，因此Spark能更好地适用于数据挖掘与机器学习等需要迭代的MapReduce的算法。

Spark 是一种与 Hadoop 相似的开源集群计算环境，但是两者之间还存在一些不同之处，这些有用的不同之处使 Spark 在某些工作负载方面表现得更加优越，换句话说，Spark 启用了内存分布数据集，除了能够提供交互式查询外，它还可以优化迭代工作负载。

Spark 是在 Scala 语言中实现的，它将 Scala 用作其应用程序框架。与 Hadoop 不同，Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据集。

尽管创建 Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对 Hadoop 的补充，可以在 Hadoop 文件系统中并行运行。通过名为 Mesos 的第三方集群框架可以支持此行为。Spark 由加州大学伯克利分校 AMP 实验室 (Algorithms, Machines, and People Lab) 开发，可用来构建大型的、低延迟的数据分析应用程序。

## 2.2 各部分简介

各部分分别为生成数据，超图的增加，修改，删除操作以及顶点覆盖的求解和CMC算法的实现。还包括一个测试函数来测试求解结果的准确性。

# 第3章技术一

3.1用分支界限法和贪心思想计算超图的最小覆盖

求解一般图的最小覆盖的方法可以拓展到超图当中，这里我选择了分支界限法和贪心所结合的方法。用一个优先队列维护当前可行的节点，每个节点维护着该节点情况下还可以选择的顶点数目k、需要覆盖的剩余边数e、顶点的状态state、顶点的边数edge等信息，这些节点的排序遵循下面的贪心策略，节点的扩展遵循下面的分支定界策略。总体思路是：

①将原图数据构造成一个解空间树的节点，利用定界策略判断是否有解，如果无解直接退出，如果有可能有解则插入到优先队列中；

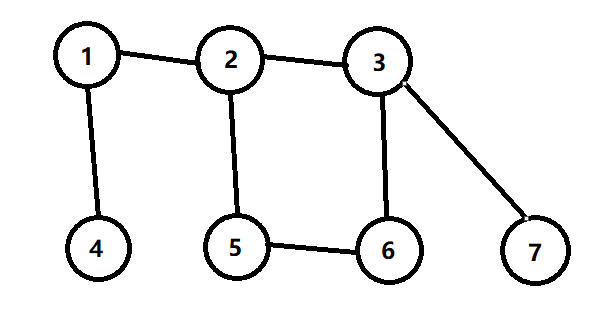
②若优先队列不为空，那么便从优先队列中取出第一个可行的节点，进入步骤③，如果优先队列为空则退出；

③判断当前节点是否满足解的条件，如果满足便输出解退出，如果不满足便进入步骤④；

④检查当前节点是否可以扩展，不能扩展的话便进入②继续循环，如果能扩展的话则扩展，然后验证扩展到左右节点是否有解，将有解的扩展节点插入到优先队列中，然后进入②继续循环。

下面通过一个例子来分别介绍下分支定界和贪心这两个策略：

为了方便起见，以一个一般图的例子来说明：



1. 分支界限策略

首先，界的选择。在一个确定的无向图G中，每个顶点的边即确定了，那么对于该无向图中k个顶点能够覆盖的最多的边数e也就可以确定了。只要对顶点按照边的数目降序排列，然后选择前k个顶点，将它们的边数相加即能得到一个边数上界！因为这k个顶点相互之间可能有边存在也可能没有，所以这是个上界，而且有可能达到。

假设取k=3个点，那么有Up(e)=(3+3+2)=8 > 7 条边（7为图G的总边数），也就是说，如果从图G中取3个点，要覆盖8条边是有可能的。但是，如果取k=2个点，那么有Up(e)=(3+3)=6 < 7 条边，说明从图G中取2个点，是不可能覆盖G中的全部7条边的！基于这个上界，可以在分支树中扩展出来的节点进行验证，已知它还可以选择的顶点数目以及还需要覆盖的边的条数，加上顶点的状态（下面会分析说明）即可判断当前节点是否存在解。

其次，顶点的状态。该策略中顶点有三种状态，分别为已经选择了的状态S1，不选择的状态S2，可以选择的状态S3。其中，不选择的状态S2对应解空间树中的右节点，不选择该节点，然后设置该节点为不选择状态S2。这点很重要，因为有了这个状态，可以使得上界的判断更为精确，因为只能从剩余顶点集中选择那些状态S3的顶点，状态S1和S2都不行，那么上界便会更小，也就更加精确。

1. 贪心策略

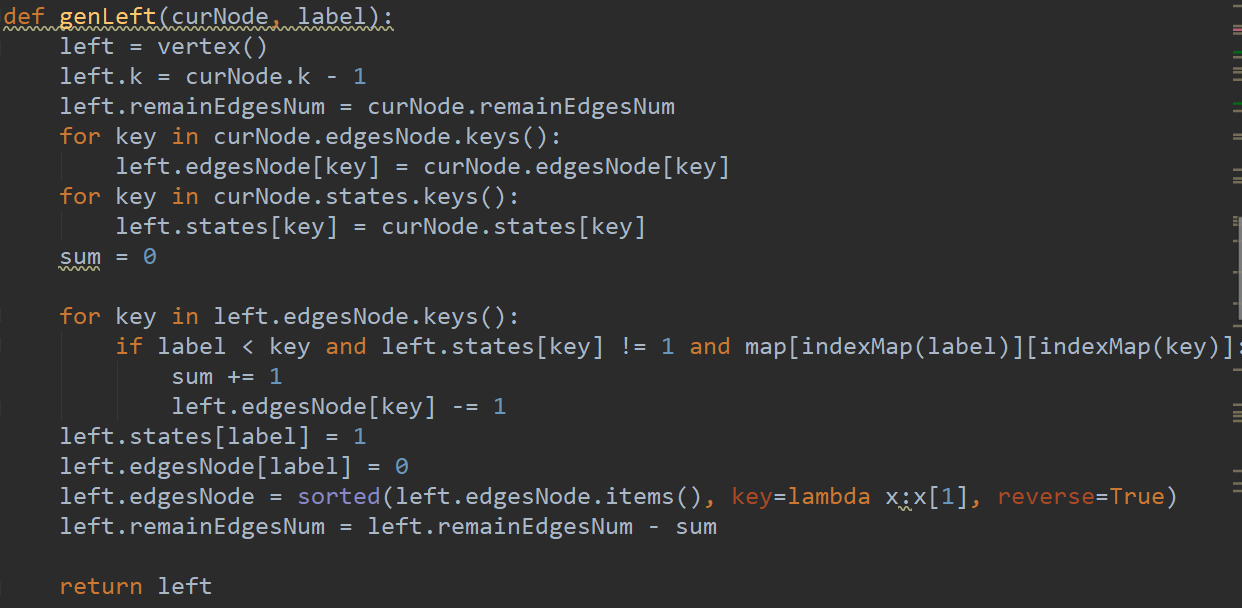
贪心的策略是指可行的结点都是按照还需要覆盖的剩余边数的降序排列，即，每次选择的节点都是可行节点中还需要覆盖的边数最小的那个节点，因为它最接近结果了。

1. 例子分析

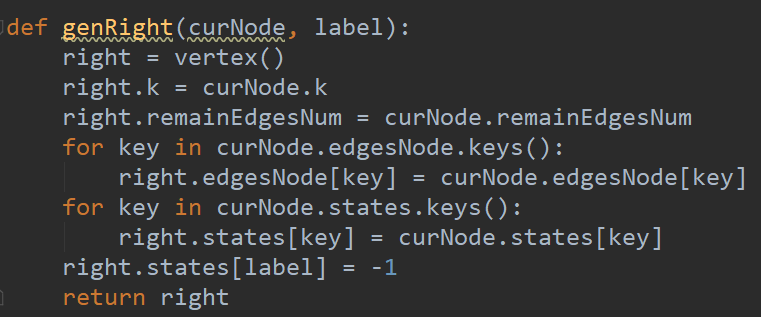
以图G为例，此时e=7（要覆盖的边数），取k=3，图G用邻接矩阵保存为全局数据，计算每个顶点的边数，然后降序排列。

步骤①判断是否可能有解，Up(e)=3+3+2=8>7，可能有解，那么将图G构造成一个解空间树的节点，它包含了还能选择的点数k=3，还需要覆盖的边数e=7，每个顶点的边数以及按边数大小的降序排列（上表），每个顶点的状态（初始时都是可选择的状态S3）。然后，将该节点插入到优先队列中，该优先队列是用最小堆实现的，按照前面的贪心策略对队列中的节点进行降序排列。

步骤②取出了优先队列中的根节点，很显然，还需要覆盖的边数为7，不为0，所以还不满足条件。接下来要检查是否能够进行扩展，从顶点集合中选择状态为可以选择的顶点中边数最多的点，该点存在为顶点2，接着进行扩展，扩展左节点时将还能选择的点数k-1=2，然后计算选择了该点之后删除了几条未覆盖的边，得到还需要覆盖的边数e=4，然后更新所有其他顶点的边数，并重新排序，最后将顶点2的状态设置为已经选择了；扩展右节点时，只要将顶点2的状态设置为不能选择，还能选择的点数k(=3)，还需要覆盖的边数e(=7)保持不变。扩展完了之后，同样判断左右节点是否可能有解，如果有解，将该节点插入到优先队列中。这里左右节点都有解，那么将左右节点都插入到优先队列中，因为左节点还需要覆盖的边数e=4小于右节点的e=7，所以根据贪心策略，左节点在右节点的前面。



算法然后继续进入步骤②，此时取出的是节点是刚才插入的左节点，很显然，还需要覆盖的边数为4，不为0，所以还不满足条件。接下来要检查是否能够进行扩展，从顶点集合中选择状态为可以选择的顶点中边数最多的点，该点存在为顶点3，接着进行扩展，扩展左节点时将还能选择的点数k-1=1，然后计算选择了该点之后删除了几条未覆盖的边，得到还需要覆盖的边数e=2，然后更新所有其他顶点的边数，并重新排序，最后将顶点3的状态设置为已经选择了；扩展右节点时，只要将顶点3的状态设置为不能选择，还能选择的点数k(=3)，还需要覆盖的边数e(=7)保持不变。



扩展完了之后，同样判断左右节点是否可能有解，如果有解，将该节点插入到优先队列中。这里左右节点都不可能有解，那么直接进入步骤②继续循环。

算法按照上面的方式不断进行，最后满足条件的分支的过程是：①不选择顶点2；②选择顶点3；③选择顶点1；④选择顶点5。

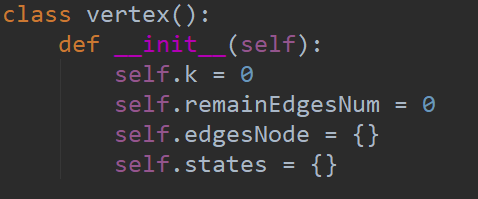
最后得到的满足条件的解是选择顶点1,3,5。

将该算法运用于超图中同样合适，本质上是一样的。

下面再来说一下优先队列。

这次大作业中我采取的是二叉树堆的方式来实现优先队列，根据剩余覆盖边数的大小来进行比较排序。二叉树堆是一棵完全二叉树，并且对于每一个节点（根节点除外），它的父节点小于或等于它，这样最小元素就会在堆顶，我们就很容易找到最小元素。

优先队列的节点里面包含可用顶点数k，剩余覆盖边数，每个顶点连接的边数，每个顶点的状态：



优先队列实现的函数有插入、获取并弹出堆顶元素、是否为空和打印优先队列：

进行插入操作时，需要保持堆的性质，即节点的值要大于等于它的父节点；进行获取并弹出堆顶元素操作时，需要找到一个元素来替代堆顶的位置，以保持堆的性质。

## 

# 第4章 CMC（Cheap Max Coverage）算法

## 4.1 算法简介：

CMC(cheap max coverage)算法基于贪心算法的部分最大覆盖启发式算法，该算法会选择具有最高边际权重的集，即那些覆盖最多未遍历元素的集。为了减少算法的开销，我们预测最佳解决方案的权重之和的值，将其称为B，然后尝试找到权重之和约为B的解决方案。我们从一个小的B值开始（等于k个最权值最小的集合的权重），这可能不足以覆盖所需数量的元素，如果有必要，我们尝试使用更大的B。为了避免权重较大的集合，我们根据权重将集合划分为多个级别，并设置高级别集合数量的阈值。。输入参数是元素集合T，集合集合C（每个集合都与某个成本相关联），最大解决方案大小k，覆盖范围阈值sˆ和参数b，该参数b确定当前预算不增加时成本预算将增加多少。相当于分支界限法中的预测代价。产生具有足够覆盖范围的集合集合。经过多次循环可以得到最终的近似解。

## 4.1 代价分析：

要计算解决方案的总成本，对于给定的B值，恰好存在（2 \* ciel（log2 k）+1）个级别Hi，并且从任何级别i中选择的集合的总成本最多为2B。代价最大的集合的成本不超过B/2i-1，我们从中最多选择2个i集合。从等级最低的集合中开始选择，我们最多选择k个集合，每个集合最多可以花费B/k。因此，从等级最低的集合中选择的集合的总成本最多为B。因此，对于给定的B值，总成本最多为（2 \* ciel（log2 k）+1）B。此外，由于该算法尝试以1 + b的定值参数增加B的不同值，它可以预测该参数内的最佳成本。如果存在使用例如l个集合覆盖例如R个元素的最优算法，则可以保证贪婪的最大覆盖启发式算法使用正好覆盖l个（1-1 / e）R个元素，因此可以保证所有的边都被覆盖。在我们的算法中，我们在每个级别调用此启发式搜索方法。具体来说，假设我们已经预测了B的最佳值，并考虑了使用该B值或B的（1 + b）因子之内的CMC算法的迭代。显然，最优解可以选择最多2 i在不超过总成本B的情况下，从Hi级别选取了几个集合，并且最多可以从最后一层选择k套。这恰好是允许CMC从每个级别选择的最大集数。假设一个最佳解决方案从覆盖Hi的每个Hi中选择一定数量的集合，涵盖Ri未被发现的元素。从贪婪最大覆盖启发式的近似保证中可以得出，CMC将从每个Hi中选择至少覆盖（1-1/e）Ri个未覆盖元素的集合，这意味着（1-1/e）ˆs | T |个元素将全面覆盖。因此CMC算法总成本最多为（1 + b）∗（2 log k +1）∗ C至少覆盖（1 − 1/e）\*ˆs \*| T |个元素。

# 第5章实验

## 5.1 实验设计

实验分为以下几步：

1. 生成实验数据
2. 对超图进行边和顶点的增加、修改以及删除操作
3. 使用分支界限法求超图的最小顶点覆盖，并验证。
4. 使用CMC算法求解超图的部分最大覆盖。

## 5.2 对比实验

本实验验证算法的正确性就可以了，所以并未设置对比实验。

## 5.3 实验结果

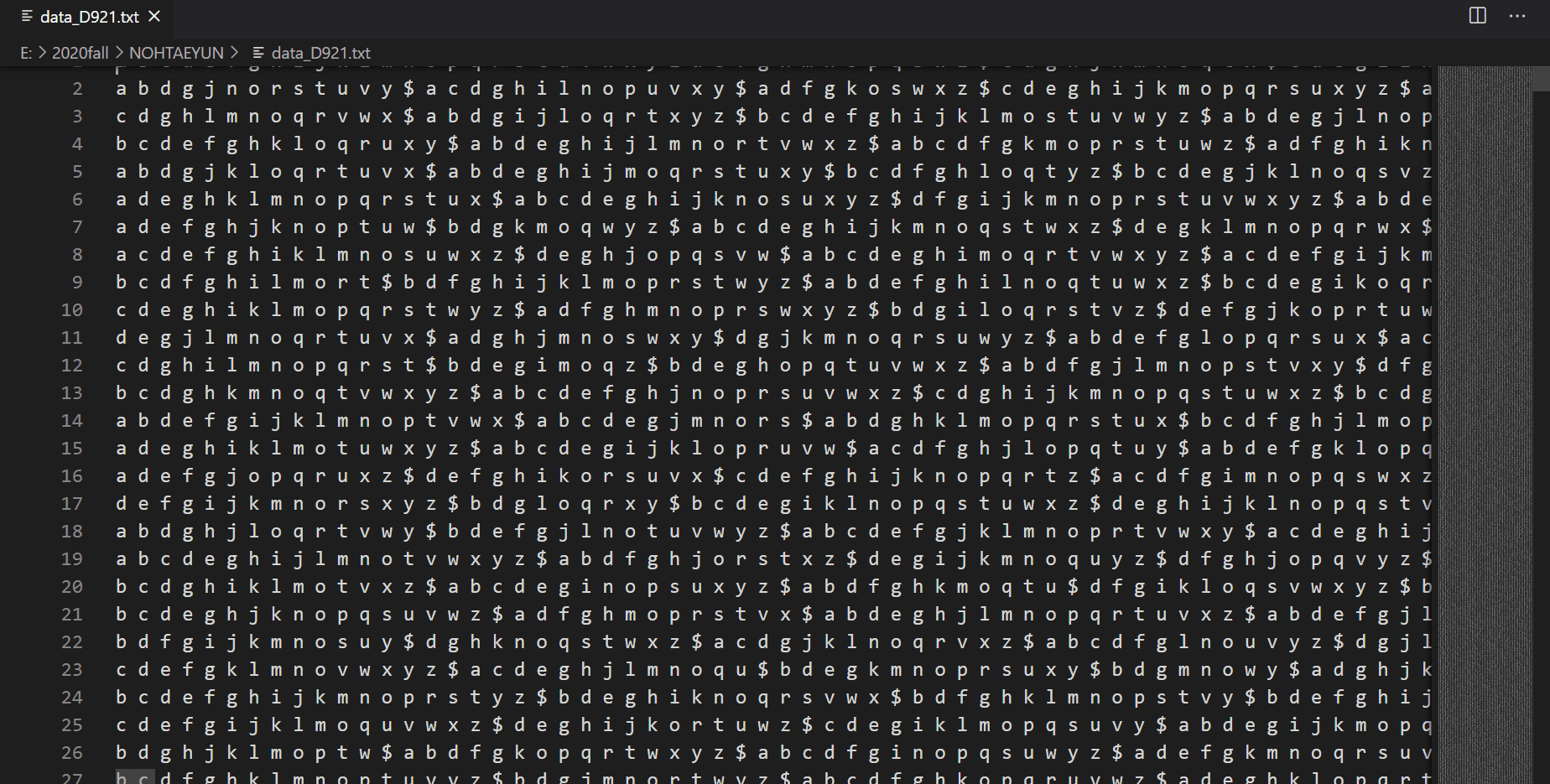
实验设计部分说到实验分为两步：

1. 生成实验数据
2. 运行代码得到实验结果

下面展示实验结果：

（1）

首先用代码生成实验数据：用a至z共计26个小写字母代表顶点，并随机生成顶点集合的子集代表超边。



可以看到一段字母之后会出现$符号，表示一个超边与超边的分隔。

用代码处理数据可以得到每个边的编号以及每个顶点连接的边数排序，由此可以应用分支界限法的代码求取超图的最小覆盖。

（3）

运行代码第二个问题文件的main函数，得到数据集的最小覆盖如下，例如：



由此可见k值小于9时则找不到最小覆盖，当k大于等于9时能找到最小覆盖。

（4）

使用CMC算法可以求解出数据的部分最大覆盖，例如：



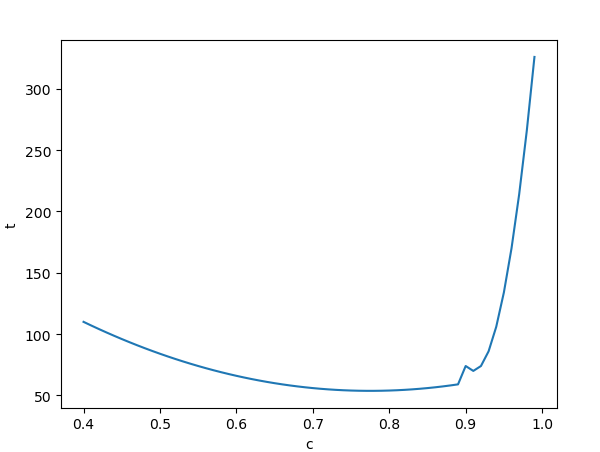
## 5.4 实验受参数的影响

第一个问题只是最基本的一些操作。

本实验中第二个问题唯一的参数是k值，一旦找到了一个可行的k值进行顶点覆盖，则比这个k值大的k值都可以成立，所以找到最小的k值就可以了。

在第三个问题中，C的选取会影响算法的效果，也就是说覆盖多大比例的超边，在同一个算法下运行的时间会有差别，主要是因为如果C过大，会导致个别边的计算影响总体的效率，如果C过小，在拥有众多的sunflower的前提下，也会导致效率减小。

在本实验中算法运行的效率随着C的大小变化如下：



# 第6章结论

本文实现了超图的数据结构以及对超图的基本的顶点和超边的增加、修改和删除操作。并且用分支界限法实现了对超图的顶点覆盖的求解。由于超图有的时候存在有有些超边与其它的超边几乎没有交集，因此在求解顶点覆盖的时候回拖延太长的时间，因此本文有使用了CMC算法来进行求解，求出部分顶点覆盖，这样可以使得求解出的解集是大量的sunflower的解集，同时也探究了部分顶点覆盖说覆盖的超边的百分比c的选取对实验结果的影响。