2020 年大数据计算基础必修大作业题目

1. 【X=1.1】冷热表问题

在实际工程应用中，经常会遇到冷热表问题。即我们想知道在实际数据库中有哪些表是经常被查询或使用的。然后再针对这些表进行操作上的优化，以提升用户体验。一些不常用的表则有可能会被删除。请你设计一种算法在数据库中可以实现对冷热表问题的求解。

要求：1、需注意“冷表”被删除时有可能包含了其他表中的外键

2、请使用 HIVE 或者 MySQL 进行算法的实现（请阐述你的实现思路）

3、数据量不要小于 100 张表，30w 条元组

4、数据可以自行生成模拟情况，不做要求

2. 子图匹配【X=1.2】

图是一类重要的数据结构。针对图数据，子图匹配是一种基本的操作，其要求在一个大图 G 中找到与某个给定的图 g 同构的 G 的子图。子图匹配在社交网络，知识图谱等领域都有大量的应用。最近，有研究人员提出了分别针对静态图和动态图的分布式子图匹配算法，并声称可以达到理论最优（见这篇论文）。但是理论最优的算法实用价值不一定很高。请你进行实验测试其在真实数据集上的表现。对于静态图（不随时间变化的图），请你实现文中描述的一个针对静态图的子图匹配算法，并实现一个之前的大规模图数据上的子图匹配算法（比如这个），比较二者的优劣（包括但不限于性能上的）。

对于动态图，请你实现文中描述的针对动态图的子图匹配算法。并进行实验说明其在处理动态图方面的优越性。

请使用 Apache Spark 或者其他课上提到的大数据计算引擎实现上述三个算法。请选择足够大的数据集进行实验，如图数据集可以选择 LiveJournal。

3、【X=1.1】Paxos 算法实现

Paxos 算法是 1990 年由 Leslie Lamport 提出的一种基于消息传递且具有高度容错特性的共识算法。基于消息传递通信模型的分布式系统，不可避免的会发生以下错误：进程可能会慢、被杀死或者重启，消息可能会延迟、丢失、重复，在基础 Paxos 场景中，先不考虑可能出现消息篡改即拜占庭错误的情况。Paxos 算法解决的问题是在一个可能发生上述异常的分布式系统中如何就某个值达成一致，保证不论发生以上任何异常，都不会破坏决议的共识。一个典型的场景是，在一个分布式数据库系统中，如果各节点的初始状态一致，每个节点都执行相同

的操作序列，那么他们最后能得到一个一致的状态。为保证每个节点执行相同的命令序列，需要在每一条指令上执行一个“共识算法”以保证每个节点看到的指令一致。一个通用的共识算法可以应用在许多场景中，是分布式计算中的重要问题。因此从 20 世纪 80 年代起对于共识算法的研究就没有停止过。

要求：1、请你设计一个小型伪分布式系统模拟 Paxos 算法进行实现并对你的系统进行压力测试

2、系统的 proposer 应不少于 100，同时进行提案的 proposer 应不少于 20

3、应尽可能提升并行性能与吞吐性能

4、可以参考 https://en.wikipedia.org/wiki/Paxos_(computer_science)

4、【X=1.1】数据的清洗，预处理，分类与聚类分析

在数据计算领域，我们需要先对读入的数据进行数据的清洗，从而提高数据质量，同样可能由于隐私等问题，需要对数据进行预处理，将数据的某些细节隐藏掉但不影响对其进行后续的分类和处理。分类和聚类作为两种重要的大数据分析技术,在大数据分析过程中起到重要作用，请你综合利用 Apache Spark 和 Apache Hadoop 这两种大数据计算工具，在 HDFS 中完成对数据的清洗，预处理，分类与聚类分析。

要求:1、清洗过程中主要解决数据中的格式错乱问题与数据缺失问题，缺失值你可以采用多种方法进行解决，比如用平均值填充，删去这一条存储等等（该部分请自行设计数据缺失和格式错乱问题）并验证你的结果

2、预处理过程请使用 PCA(主成分分析)对数据进行降维

3、分类分析请至少使用两种不同算法进行实现并验证你的结果

4、聚类分析请至少使用两种不同算法进行实现并验证你的结果

5、你可以使用 Apache Spark 中的相关机器学习组件进行算法的实现，你也可以自行编写程序实现算法(但请务必使用 Spark 或 Hadoop 进行实现)，这有可能会影响你的最终得分

6、数据以及数据描述请参考 https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

5、【X=1.1】随着许多应用遇到图规则大小的飞速增长，由于缺乏用于评估图处理方法性能的大规模真实图，我们对快速的图生成器的需求越发迫切. 本实验中,

希望你阅读论文，实现一个快速的图生成器，它是一个递归矩阵模型，递归地选择矩阵的象限来生成边. 请你实现该生成模型后，选取文中提到的若干指标，观察其生成的图与原图的对比结果，并验证其时间复杂度. 如有余力，可以实现文中提到的其他算法并完成对比实验，会体现到你最终成绩结果中。
参考论文：R-MAT: A Recursive Model for Graph Mining

6、【X=1.2】在现实生活中，图一直是表示数据的有力工具. 特别是在社交网络、网络、生物信息学和天体物理应用中，它们得到了广泛的应用. 在大数据时代，图的规模也随着数据变得越来越大. 同时，子图匹配问题也成了图问题中的一个重要问题，它在蛋白质的功能表征及其相互作用、社会网络的结构建模、知识图的语义结构发现等方面有着广泛的应用. 阅读参考论文，实现这个基于邻近索引的近似图匹配算法.
参考论文: TALE: A Tool for Approximate Large Graph Matching

7、 【X=1.2】利用 AI 动态为大规模知识图谱建立索引
【题目背景】由于大数据的兴起，计算能力的升级，涌现出以知识图谱为代表的一批大数据时代的产物。知识图谱的查询一直受到学术界和工业界的广泛关注，而其查询效率与知识图谱上建立的索引密切相关。而随着知识图谱数据的不断更新，以及其上查询工作负载的不断变化，最初建立的索引可能不能很好地满足更新后的知识图谱数据及其负载。因此，需要根据动态变化的数据及负载调整知识图谱索引。
【数据来源】DBPedia 知识图谱 :https://wiki.dbpedia.org/develop/datasets DBPedia
工作负载：
http://wifo5-03.informatik.uni-mannheim.de/benchmarks-200801/
【设计要求】利用所给 DBPedia 知识图谱及其负载，设计动态建立知识图谱索引的方法，要求：（1）构造性能评估函数，以判断知识图谱及其负载对知识图谱查询性能的影响；（2）利用 AI 方法动态感知知识图谱及其负载的变化是否对知识图谱查询性能产生影响；（3）对于动态感知获得的结果，设计知识图谱索引调整算法，保证调整后知识图谱查询性能的同时调整索引的代价不宜过大。
【实验要求】对所提出的方法进行合理实验评估，自行设计实验方法，记录（1）知识图谱的查询效率；（2）索引存储空间；（3）动态感知算法的效率。

8、 【X=1.3】利用 AI 动态为大规模知识图谱选择存储结构

【题目背景】由于大数据的兴起，计算能力的升级，涌现出以知识图谱为代表的一批大数据时代的产物。知识图谱的查询一直受到学术界和工业界的广泛关注，而其查询效率与知识图谱的存储结构密切相关。而随着知识图谱数据的不断更新，以及其上查询工作负载的不断变化，最初选择的知识图谱存储结构可能不能很好地满足更新后的知识图谱数据及其负载。因此，需要根据动态变化的数据及负载调整知识图谱的存储结构。

【数据来源】DBPedia 知识图谱 :https://wiki.dbpedia.org/develop/datasets

DBPedia 工作负载：

http://wifo5-03.informatik.uni-mannheim.de/benchmarks-200801/

【设计要求】利用所给 DBPedia 知识图谱及其负载，设计动态选择知识图谱存储结构的方法，要求：（1）选用基于关系模型的知识图谱存储方式，参见参考文献

（http://www.jos.org.cn/jos/ch/reader/create_pdf.aspx?file_no=5841&journal_id=jos）;（2）构造性能评估函数，以判断知识图谱及其负载对知识图谱查询性能的影响；（3）利用 AI 方法动态感知知识图谱及其负载的变化是否对知识图谱查询性能产生影响；（4）对于动态感知获得的结果，设计知识图谱存储结构调整算法，保证调整后知识图谱查询性能的同时调整存储结构的代价不宜过大。

【实验要求】对所提出的方法进行合理实验评估，自行设计实验方法，记录（1）知识图谱的查询效率；（2）存储结构所占空间；（3）动态感知算法的效率。

9、【X=1.3】 随着目前数据的爆炸式增长，对于同一类任务有很多不同的算法被研究出来,（例如：分类、回归）但是对于具有不同特点的数据集而言，并不是所有算法最终的性能都很好，算法之间最后的性能差异有很大的浮动。所以，如何根据数据集和算法的特点，为每个数据集尽可能地选择最适合它的算法成为目前的研究热点。

构建一个模型（可以参考元学习的思想（meta-learning））, 使得其训练后可以：

A． 针对分类任务，为新任务自动推荐最适合它的算法

B． 针对回归任务，为新任务自动推荐最适合它的算法

C． 尽可能提升整个模型的自动化程度，并在实验报告中说明详细。（例如：如果采用 meta-vector 来代表每个历史任务，那么设计一种方法可以自动从一个 candidate meta-feature list 中选择效果最好的 meta-features 来构成

meta-vector）
D． 选择一个自己熟悉的主流大数据框架，使用这个框架实现上述功能，深入思考哪个部分可以采用并行处理，并提升其效率。不要为了使用框架而使用框架，如果使用框架效率降低，将实验结果写明，详细分析原因，写入实验报告。
说明：分类算法和回归算法的实现可以采用现有的算法库，例如：WEKA、sk-learn 等，A、B 两个任务每个任务待选择的算法不少于 15 种。实验数据集可以从 UCI 公开数据集下载 https://archive.ics.uci.edu/ml/index.php 自己进行适当的数据集格式转换。对于算法自动选择来说，不要走入“一个模型可以找到所有最优算法”的误区。我们设计的模型只能达到尽可能接近最优的算法。
除了工程文件和实验报告之外，你需要提交一份说明文档，描述你的算法设计、算法实现和实验结果。不必将思路局限于题目中提及的论文，可以考虑更合理的方式方法。

参考文献：
[1] https://doi.org/10.1007/978-3-030-05318-5_2
[2] Thornton C, Hutter F, Hoos H H, et al. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms[C]//Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013: 847-855.
[3] Cohen-Shapira N, Rokach L, Shapira B, et al. AutoGRD: Model Recommendation Through Graphical Dataset Representation[C]//Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019: 821-830.

10、【X=1.3】 随着目前数据的爆炸式增长，对于同一类任务有很多不同的算法被研究出来,（例如：分类、回归）。但是对于相同的数据集而言，即使采用相同的算法，算法上的不同超参数组合也会导致最后性能的差异较大。所以如何针对算法进行超参数优化，找到对于当前数据集和算法表现最优的超参数组合，目前成为研究热点。
构建一个模型，不是简单实现目前现有的超参数优化算法（贝叶斯优化、遗传算法、随机搜索、网格搜索、BOHB 等），而是在现有超参数优化算法中选择一个算法，针对缺陷（例如遗传算法编码效率问题、收敛慢等）进行改进，使得可以：

A． 针对分类任务，在待选择算法中为每个算法进行超参数调优，输出每个算法的最优超参数组合。
B． 针对回归任务，在待选择算法中为每个算法进行超参数调优，输出每个算法的最优超参数组合。
C． 选择一个自己熟悉的主流大数据框架，使用这个框架实现上述功能，深入思考哪个部分可以采用并行处理，并提升其效率。不要为了使用框架而使用框架，如果使用框架效率降低，将实验结果写明，详细分析原因，写入实验报告。
说明：分类算法和回归算法的实现可以采用现有的算法库，例如：WEKA、sk-learn 等，A、B 两个任务每个任务待选择的算法不少于 15 种。每个算法的超参数自己决定，可以参考 Auto-WEKA2.0。实验数据集可以从 UCI 公开数据集下载（https://archive.ics.uci.edu/ml/index.php）自己进行适当的数据集格式转换。对于超参数调优来说，不要走入“一个模型可以找到所有算法的最优超参数”的误区。我们设计的模型只能达到尽可能接近最优的超参数组合。
除了工程文件和实验报告之外，你需要提交一份说明文档，描述你的算法设计、算法实现和实验结果。不必将思路局限于题目中提及的论文，可以考虑更合理的方式方法。

参考文献：
[1] https://doi.org/10.1007/978-3-030-05318-5_1
[2] Kotthoff L, Thornton C, Hoos H H, et al. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA[J]. The Journal of Machine Learning Research, 2017, 18(1): 826-830.

11、【X=1.2】大规模图数据中三角形计数算法的设计与性能调优
大数据时代，对关联（图）数据的处理被广泛应用于社交网络、智能交通、移动网络等领域。三角形的定义是一个包含三个顶点的子图，其中顶点两两相连。对图数据的三角形计数被广泛应用于图数据的特征描绘（如聚集系数、联通度等）、社区结构检索、子图匹配、生物网络等应用。
（1）算法设计要求：要求设计一个分布式算法，快速准确的计算给定的大规模图数据上三角形的总数。
（2）实验要求：要求使用 Hadoop/Spark 系统实现该算法；要求至少与两个现有的三角形计数（Triangle Counting，TC）算法进行对比分析。
（3）超参数调优问题：如果设计的算法中存在超参数，请设计有效的超参数优

化算法，对算法超参数进行最优推荐。
数据来源：

12、【X=1.2】图数据的存储和索引
应用背景：
图数据有多种类型，比如 RDF 图和 native 图，结构上的不同造成了这两种图分别适合不同的存储方案和查询。比如 RDF 图上适合进行简单的连接查询，native 图上合适进行复杂的子图匹配操作。随着数据量的增加，我们越来越需要实现这些数据的高效存储。
数据来源：
http://swat.cse.lehigh.edu/projects/lubm/
http://github.com/facebook/linkbench
实验要求：
1） 熟悉图数据的基本概念，熟悉 RDF，RDF, RDFS, OWL，SPARQL 等基本概念。
2）请以上述两个数据集为基准，设计最佳的图存储方案来使得工作负载的执行效率最高。
3） 考虑时下流行的分布式数据库 hugegraph、OrientDB、Cassandra 等（如果使用单机版数据库会影响你的成绩，可使用伪分布式进行实现）
4）请分析和对比究竟哪些因素影响了图存储，不同的工作负载类型都适合于怎样的存储方案。

13、【X=1.2】XML 数据的高效存储和索引
应用背景：
为了实现 XML 数据的高效存储，人们从多个角度，设计了不同方案将其转换成关系数据来存储，比如从 XML 的结构出发，从 XML 的负载类型出发等等。
实验要求：
1） 熟悉 XML 数据的基本概念
2）请以如下参考文献为基准，在时下流行的分布式关系数据库上复现一篇论文的代码
3） 设计一种基于机器学习的高效转换方法，并与现有工作进行对比，比较不同类型工作负载的效率。
参考文献：

Cost-Driven Storage Schema Selection for XML.

Adaptability of Methods for Processing XML Data using Relational Databases – the State of the Art and Open Problems.

14、【X=1.2】关系数据库物化视图选择

物化视图的存在可以显著提高 SQL 数据库系统的性能，在实际应用场景中，由于空间限制和 SQL 更新需要，存在对物化视图进行选择的问题。

论文中提出了一种为给定数据库和由 SQL 查询和更新组成的工作负载确定一组适当索引和物化视图（及物化视图上索引）的方法。

实验内容：

（1）请仔细阅读参考论文，实现其中提到的物化视图选择及合并部分。

（2）请思考设计一种可能的物化视图选择方法，可以参考论文中同时考虑索引及物化视图的交互。

实验要求：使用 Apache Spark 或其他大数据计算引擎实现如上算法，并选择足够大的数据集进行实验。

参考论文：Agrawal S , Chaudhuri S , Narasayya V R . Automated Selection of Materialized Views and Indexes in SQL Databases[C]// VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt. Morgan Kaufmann Publishers Inc. 2000.

15、【X=1.2】物化视图的选择和利用

物化和利用曾经的查询结果（视图）对于提高查询性能非常重要。在一定的资源约束下，为给定的查询工作负载选择最佳视图集是商业数据库管理和数据仓库系统中最常见的问题之一。在有限的资源（如视图维护成本和存储空间）下，选择实现哪些视图以最小化总查询成本的问题称为视图选择问题，这是一个 NP 完全问题。

请仔细阅读论文，实现论文中提出的基于约束规划（CSP）的方法解决视图选择问题，包括将问题建模以及使用启发式搜索策略（Variable and value selection heuristics）。

实验要求：使用 Apache Spark 或其他大数据计算引擎实现，并选择足够大的合适数据集进行实验。

参考论文：Mami I , Bellahsene Z , Coletta R . A Declarative Approach to

View Selection Modeling[J]. 2013.

16、【X=1.2】分布式机器学习的梯度压缩问题

在分布式机器学习的问题中，单节点上的计算开销和结点间的通信开销是消耗整个分布式计算任务资源的两大主要模块。而节点通信的开销和有限网络通信带宽的矛盾是限制分布式机器学习可扩展性的主要原因。而对于机器学习任务，结点间的通信开销主要是每轮迭代过程中的梯度信息的传输和模型信息的传输。针对这样的问题，梯度压缩通过减少传输内容从而成为了减少通信开销的实用方法，参考 https://www.aclweb.org/anthology/D17-1045.pdf。)。请你根据这一思路，选择 Apache Spark 或者其他你喜欢的大数据分布式计算平台，解决以下问题：

问题 1：分布式机器学习实现

针对 MNIST 数据集，实现同步梯度传输的伪分布式/真分布式机器学习模型，并训练至准确率超过 95%

问题 2：系统可扩展性分析

针对问题 1 建立好的分布式计算系统系统，对比 4 节点、8 节点和 16 节点的收敛时间与单机收敛时间的差异，并作图比较。

问题 3：梯度压缩方法实现

基于参考文献中提到的稀疏化梯度压缩算法，只针对 MNIST 数据集，自定 drop ratio，记录梯度压缩后模型准确率达到 95%所需时间，并与使用梯度压缩算法前的训练时间进行对比，分析该方法的优势，并请尝试设计其他方法，在不降低模型准确率的前提下缩短训练时间。

17、【X=1.2】分布式机器学习的同步/异步更新问题

对于分布式机器学习问题，不同工作节点之间需要传递梯度和模型信息，据此便产生了一些同步策略。比如批量同步并行 bulk synchronous parallel (BSP)、异步并行 asynchronous parallel (ASP)，以及延迟同步并行方法 stale synchronous parallel (SSP)。

问题 1：请查阅相关资料，了解以上三种并行策略的优势和劣势，并选择合适的分布式计算框架和数据集来完成一种广义线性模型（LR、SVM、FM 等）的机器学习并行实现。

问题 2：在达到相似准确率的前提下比较三种并行策略的不同，尤其是总训练时间的差异。

问题 3：对于延迟同步并行方法 stale synchronous parallel (SSP)，分析其等待轮数阈值 s 的改变对训练时常的影响，并针对问题 1 所选定的机器学习任务，选择出最优 s。

18、【X=1.1】
流数据（stream data）是一类重要的数据模型，其基本假设是数据以很高的速率源源不断地到来。因此，在设计处理流数据的算法时，要求对每个新到的数据对象处理的时间要尽可能短。请你设计实现一个算法对流数据进行聚类，要求：
聚类结果必须能够对数据的变化进行增量更新而不必遍历所有历史数据；
能够保留聚类结果随时间变化的信息。
请你使用 Apache Spark 或者你喜欢的其他大数据计算引擎实现你设计的聚类算法。你可以参考论文 http://www.vldb.org/pvldb/vol11/p393-gong.pdf。你需要考虑如何使用静态数据集模拟流数据（比如使用时间序列数据集）。请选择足够大的数据集进行实验。

19、【X=1.1】
集合的基数（cardinality）的估计是数据库系统设计的关键问题，比如，其解决方案是基于代价的查询优化器（cost-based optimizer）的基础。基数估计问题是要求一个多重集合（multiset）中所有不同元素的个数。精确基数估计方法对于大数据来说是不可接受的。因此我们需要一个近似估计方法。
一个可行的方案是对原数据集进行抽样，用样本的基数推断总体的基数。请你根据这一思路设计实现一个基数估计算法。你需要保证估计值和真实值的误差在一定的限度内。你可以参考相关的文献。
另外一个可行的方案是基于草图（sketch）的方法。其中最有名的两个数据结构是 BloomFilter 和 HyperLogLog。请你实现基于这两个数据结构的基数估计算法，并与上面你设计的算法相比较，分析它们的优劣（包括但不限于性能上的）。
请使用 Apache Spark 或者你喜欢的其他大数据计算引擎实现上述三个算法。请选择足够大的数据集进行实验。

20、【X=1.2】异构数据的分析长久以来都是一大难题。最近，有人提出了将所有数据对象统一转换为向量，用向量的距离衡量数据的相似度的方法进行数据集成。
a) 请你阅读论文“Termite: A System for Tunneling Through

Heterogeneous Data”，实现这一想法。你可以使用大规模机器学习框架，如 PyTorch 等来实现。
b) 请你进行实验，验证上述想法的效果，并与至少一个现有的数据集成方法进行比较。

21、【X=1.2】判断两个不同形式的查询是否等价是查询优化的一个关键问题。最近，有一些研究者正在尝试使用代数方法形式化地证明两个查询的等价性。请你阅读论文“Automated Verification of Query Equivalence Using Satisfiability Modulo Theories”，并选择一个有查询重写（Query Rewriting）功能的分布式 SQL 查询处理引擎，如 Spark SQL，实现论文中提出的等价性判定算法，并以此判定使用 SQL 查询处理引擎内置的各种不同的查询重写规则得到的 SQL 查询的等价性。注意：由于现有的 SQL 查询引擎只使用它认为等价的规则进行重写，对于某一个 SQL 查询，其重写的结果可能只有一个（甚至不会进行重写）。你需要对其进行修改，尽可能多地生成不同的查询重写结果，然后进行等价性判定。
d) 请你进行实验，对于一个给定的 SQL 查询，比较各种不同的等价的查询重写结果的效率，并思考造成效率差异的原因，然后尝试设计并实现一种生成效率更佳的重写结果的方法。你的方法可以只针对某些查询而不必是通用的方案。

22、【X=1.2】基于 NoSQL 的知识图谱管理查询引擎设计
应用背景：
知识图谱是一种揭示实体之间关系的语义网络，可以对现实世界的事物及其相互关系进行形式化地描述。主流知识图谱数据通常使用 RDF 三元组（主语，谓词，对象）的形式表示。随着知识规模的不断扩大，单机系统已经很难实现知识图谱的管理分析。
1) 熟悉知识图谱基本知识，熟悉 RDF, RDFS, OWL，SPARQL 等基本概念。基于 NoSQL 设计一个知识图谱分布式的管理查询引擎
2) 请使用 LUBM 生成合适大小的数据集，测试你的知识图谱管理查询引擎。
3) 对比和单机管理引擎的性能差距，例如 RDF-3X。（性能差距，可以从同一算法在同一数据集上，两个系统查询效率作为衡量标准）
数据来源：http://swat.cse.lehigh.edu/projects/lubm/

23、【X=1.2】集合覆盖问题及其质量的研究

大数据时代，数据中的关系往往呈现多元关系，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。但是对于元素-集合来说，我们利用其特性，能描述更为复杂的关系信息。所谓超图（Hyper-Graph），是一种广义的图，其特点是一条超边可以连接多个点。超图上的问题中，超边覆盖问题，又称集合覆盖问题，作为经典的 NP 完全问题，在设备选址、信号覆盖等领域有着很多应用。请你选用 Apache Spark 或其他你喜欢的大数据处理框架，对以下问题进行系统实现。

问题 1：海量集合数据管理

请你选择足够大的数据,设计基于集合-元素的表述和存储方式，实现基本的数据添加、删除、修改和查询操作。

问题 2：集合覆盖算法的实现

集合覆盖是指，求取一个集族，使得每一个元素都被至少一个集合覆盖。k-最小集合覆盖问题定义为：给出集族S和上界k，判断S是否有不超过k个集合的集合覆盖。请基于你选用的大数据处理框架，设计并实现大规模数据上的 k-最小集合覆盖问题算法。

问题 3：集合覆盖质量评价的探索

传统集合覆盖方法给出的最优解往往不尽如人意，因此，如何定义高质量集合覆盖是一个有着很大研究价值的问题，一种高质量评价方法要求每一个元素被覆盖的次数至少为θ次，请你基于你选用的大数据处理框架，设计并实现此限制条件下的k-最小集合覆盖问题算法，我们鼓励对于θ和k以及数据量的关系，或者对于其他集合覆盖质量评价方法的探索。

除了工程文件之外，你需要提交一份说明文档，描述你的算法设计、工程实现和实验结果。

24、【X=1.2】海量数据频繁项集挖掘的研究

频繁项集挖掘是数据挖掘研究课题中一个很重要的研究基础，它可以告诉我们在数据集中经常一起出现的变量，为可能的决策提供一些支持。频繁项集挖掘是关联规则、相关性分析、因果关系、序列项集、局部周期性、情节片段等许多重要数据挖掘任务的基础。因此，频繁项集有着很广泛的应用，例如：购物篮数据分析、网页预取、交叉购物、个性化网站、网络入侵检测等。但是，随着大数据时代的到来，海量数据对于以频繁项集挖掘为代表的很多传统数据挖掘算法的有效性和效率提出了挑战。请你基于以下问题，展开海量数据频繁项集挖掘的研究。

问题 1：传统频繁项集挖掘方法的限制

请你选用一种你熟悉的编程语言，选择足够大的数据集从中抽取不同规模的样本进行一种传统频繁项集挖掘算法（如 apriori、FP-growth 等）的实现，测试并记录这些传统频繁项集挖掘算法面对大规模数据时的表现和限制。你可以合理调用已有的函数库进行实验。

问题 2：频繁项集挖掘方法的改进

请你根据问题 1 中传统频繁项集挖掘算法的限制，设计面向海量数据的频繁项集挖掘算法，使其能够在不损失过多效果的同时，合理解决大规模数据带来的问题。请选用 Apache Spark 或其他你喜欢的大数据处理框架，对这一问题进行工程实现。你可以考虑参考文献[1]等已有工作，使用随机算法的思路设计算法。

除了工程文件之外，你需要提交一份说明文档，描述你的算法设计、工程实现和实验结果。

25、【X=1.1】存储过程的设计

存储过程（Stored Procedures，简称 SP）是使用数据库系统支持的过程式语言，其编写的目的是对数据库系统现有的功能进行扩充和封装，提升数据库系统易用性和处理复杂任务的性能。为分布式数据库系统添加存储过程的支持可以采用存储模块和计算模块分离的方式，使其拥有能与传统单机数据库媲美的易用性和性能。但是存储计算分离架构的缺陷在于其引入的额外通信开销降低了查询处理的性能。为了减小通信开销，一个思路是尽可能让存储节点使用部署在同一台机器上的计算节点来执行存储过程。于是，计算节点的分配就成为了一个重要的课题。

请根据题目要求设计算法解决如下（包括但不限于）三个问题：（1）如何就近分配计算节点（2）如何平衡计算节点的负载（3）如何确定需要的计算节点的数量

26、【X=1.2】多查询统一索引结构构建研究（分布式系统环境）

应用背景：

众所周知，图数据是一种广泛应用的数据类型。图查询算法也一直是研究人员研究的热点问题，如可达查询、最短路径查询、强连通分量计算、图拓扑查询、关键字查询、图匹配、pagerank、simrank、k-core、k-truss 和 clique 等。针对特定的查询问题，目前的研究方法可概括为：提出相应的查询处理算法，并构建索引结构来加速查询。

然而 ,现实应用中需求的多样化以及图数据规模爆炸式的增长为该研究方法带来了两方面挑战。第一，同一个图数据在应用中会涉及多种查询，但针对不同查询问题的处理机制和索引结构均不相同 ,因此在设计图数据库时需构建多个索引和相应的查询算法；第二，索引的规模通常比原图数据的规模大，多个索引同时存在会占用大量的系统空间，导致图数据库的性能急剧下降，不能被真正的应用。

1) 了解可达查询、最短路径查询、关键字查询、图匹配、pagerank、simrank、k-core、k-truss 和 clique 等大图查询。了解大图分布式存储方式。
2) 至少选择上述查询的四种查询 ,对其在分布式系统环境下构建统一的索引结构，设计在该索引结构下的快速查询算法。*如果选择更多种类的查询可在报告中重点标注*
3) 使用 LUBM 生成合适大小数据集，选择合适的分布式图计算引擎，测试算法效率
    a) 索引构建时间
    b) 索引的空间代价
    c) 各种查询查询效率

数据来源：http://swat.cse.lehigh.edu/projects/lubm/

27、【X=1.3】超图顶点覆盖

大数据时代，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，我们利用其特性，能描述更为复杂的关系信息。所谓超图（Hyper-Graph），是一种广义的图，其特点是一条超边可以连接多个点。超图上的问题中，顶点覆盖问题作为经典的 NP 完全问题，在设备选址、信号覆盖等领域有着很多应用。请你选用 Apache Spark 或其他你喜欢的大数据处理框架，解决以下问题：

问题 1：大规模超图管理

请你选用足够大的超图数据集[1]，使用超图表征其关系，并对该大规模超图进行存储，并实现基本的超边、顶点的增加和删除功能。

问题 2：超图顶点覆盖问题

所谓顶点覆盖，是顶点的一个集合，使得超图中每条超边所包含的顶点集都与该集合有非空的交集。超图顶点覆盖问题定义为：给出超图G和上界k（k必须是在从 0 到G的顶点数减一之间），判断G是否有不超过k个顶点的顶点覆盖。请

基于你选用的大数据处理框架，设计并实现大规模超图上的顶点覆盖问题算法。

问题 3：超图上的受限顶点覆盖问题

大规模超图上顶点覆盖存在一个严重的问题，即囿于过大的超图规模，精确顶点覆盖问题的耗时和解的规模都给下游任务带来了很大的负担，该问题的一种解决思路是，选用不超过k个顶点覆盖c%的超边作为近似解输出给下游任务。请设计算法，基于你选用的大数据处理框架进行实现[2]。

除了工程文件之外，你需要提交一份说明文档，描述你的算法设计、工程实现和实验结果。

[1]你可以使用文献 http://www.vldb.org/pvldb/vol13/p698-whang.pdf 中用到的数据集进行实验。

[2]问题 3 的思路可以参考文献 https://ieeexplore.ieee.org/document/7113341。

28、【X=1.3】时间序列上的依赖规则研究

随着传感器网络深入人们的生活，每时每刻采集的数据形成了海量高维时间序列，如何挖掘这些时间序列中的依赖规则，并通过这种关系维护数据质量是一个备受关注的问题。在单维时间序列的异常段检测、聚类等问题上，一类经典的方法是 SAX[1]方法，这类方法将时间序列转化为字符串的方式来解决问题。在高维时间序列上，近年来，有学者提出了时间序列规则挖掘的方法[2]。请你参考这两种思路，选取足够大的高维时间序列数据集，基于时间序列上的依赖规则展开时间序列数据清洗问题的研究。

问题 1：单维时间序列分段符号化

对于单维时间序列数据进行合理分段，参考 SAX 类方法的思路，选用合适的时间序列聚类方法，将单维时间序列进行符号化。在本问题中，每个维度时间序列的分段情况可以完全相同。你可以选用 Apache Spark 或其他你喜欢的大数据处理框架，对这一问题进行工程实现。

问题 2：高维时间序列依赖规则挖掘

在高维时间序列的研究中，时间序列间的依赖规则逐渐受到重视。直观上，若某一特定时间序列T_1在某一时间区间$[t_1, t_2]$呈现某一特定状态θ_1，则另一特定时间序列T_2在与$[t_1, t_2]$相关的区间$[t_3, t_4]$上往往呈现另一特定状态θ_2，这种二元关系被称为高维时间序列的依赖规则。请你参考文献[2]对这一问题的进一步介绍，根据你选择的数据和背景严谨定义这一依赖规则，并根据依赖规则，在你所选择的数据集上展开依赖规则挖掘的研究。在本问题中，需要根据你选择的具体

数据和挖掘结果决定每个维度时间序列的分段情况是否完全相同。本问题不要求使用大数据处理框架来进行工程实现。

问题 3：基于高维时间序列依赖规则的数据清洗

借用依赖规则解决关系数据的数据清洗问题是一种经典的思路[3]。请你尝试利用问题 2 中挖掘的依赖规则，在你所选用的时间序列数据集上自行替换一些时间序列段并进行标注，展开异常时间序列段检测和清洗的研究。本问题不要求使用大数据处理框架来进行工程实现。

除了工程文件之外，你需要提交一份说明文档，描述你的算法设计、工程实现和实验结果。不必将思路局限于题目中提及的论文，可以考虑更合理的方式定义时间序列上的依赖规则、制定基于这些依赖规则的数据清洗方法。

[1] https://ieeexplore.ieee.org/document/1565683

[2]https://xueshu.baidu.com/usercenter/paper/show?paperid=bc4d9e324cf252e6b7f7e1a472c0d433&site=xueshu_se

[3] https://dl.acm.org/doi/10.1145/1066157.1066175

29、【X=1.1】大数据中频繁元素查找

背景：频繁元素查找是数据管理及分析领域的重要问题。频繁元素查找对应大数据管理中十分重要的 Top-k 查询，同时频繁元素及其频率在网络安全监控中常作为某些异常事件的衡量指标，在电商平台、推荐系统中也有广泛应用。

实验内容：请学习 Space-Saving [3]算法，在 Spark 引擎上实现算法，并提出改进方法。要求：

（1）在 Spark 上实现 Space-Saving 算法。

（2）在 Spark 上设计并实现 Top-k 高频元素查找算法（可基于 Space-Saving 做改进或设计完全不同的新方法），高准确性的找到数据中的 heavy hitter，估计这些元素的频率，并评估算法的准确性、空间代价、和效率（与真实结果和 Space-Saving 算法对比）。

数据：自行选择不少于两个（包括两个）足够大的数据集进行测试，真实数据或模拟数据均可，模拟数据需标明生成方式及数据分布特征，若两个均为模拟数据则需具有不同的分布特征。

参考文献：

Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. An integrated effcient solution for computing frequent and top-k elements in data streams. ACM Trans. Database Syst., 31(3):1095–1133, 2006.

30、【X=1.2】大数据频率估计问题

背景：数据频率是数据最基本的统计量和摘要形式，同时也是网络监控、异常检测中的重要指标。然而，在大规模数据上统计准确的数据频率会占用极大的空间，为此，研究者们提出通过亚线性空间的草图（sketch）来近似存储和估计数据频率。数据草图具有体积小，精度高，高效查询的特征，能够估计被广泛应用于频率估计和流数据处理上。

实验内容：请基于文献 [1]及 Spark 中的 *CountMinSketch* 函数学习 Count-Min sketch 的结构和频率估计方式，在 Spark 上实现其改进算法 Augmented sketch [2]，并设计新的 Count-Min 改进算法。要求：

（1）请阅读文献，在 Spark 上实现 Augmented sketch。

（2）请基于 Count-Min sketch 在 Spark 上设计新的大数据频率估计算法，要求能够近似估计任意元素（无论是否出现在数据集中，例如，未出现的元素的频率估计结果应接近 0）的频率，并与 Count-Min 算法在准确性、空间代价、频率查询效率（此外，可增加自定义的对比指标）等方面进行比较，同时分析影响算法性能的因素。

数据：自行选择不少于两个（包括两个）足够大的数据集进行测试，真实数据或模拟数据均可，模拟数据需标明生成方式及数据分布特征，若两个均为模拟数据则需具有不同的分布特征。

参考文献：

LI YGraham Cormode and S. Muthukrishnan. An improved data stream summary: The countmin sketch and its applications. In LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings, pages29–38, 2004.

Pratanu Roy, Arijit Khan, and Gustavo Alonso. Augmented sketch: Faster and more accurate stream processing. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016, pages 1449–1463. ACM, 2016.

31、【X=1.2】大规模时间序列数据的相似性搜索

基于相似性的时间序列搜索是时序数据查询和分析(如分类、聚类等)的关键支撑技术，在金融、医疗、生物科技、行为识别、语音信号处理、IOT 数据分析等领域具有广阔的应用前景。一个时间序列是一系列有序排列的数据对象的列表。考虑一个时间序列数据$T = (t_1, t_2, t_3, \dots)\ t_i \in R$，其长度记为|T|。 T 的任意一个片段$(t_i, t_{i+1}, \dots, t_{(i+L-1)}$称为 T 的一个长度为 L 的子序列。对于一个查询序列 Q(其长度小于 T 即$|Q| < |T|$)，Q 在序列 T 上的相似性搜索是要在 T 的所有子序列中查找 Q 的最近邻(k-Nearest-Neighbor，其中 k=1)。

1.1 在时间序列相似性搜索中 ,用于度量序列间距离的最基本度量是欧拉距离。请实现基于欧拉距离的相似性搜索算法，分析其复杂度，并测试其随时序数据 T 及查询 Q 长度增长的可扩展能力。(参考参数：|Q|取 128，|T|可取 1,000/ 10,000/ 100,000/ 1,000,000….直至运行时间不可接受，如超过 24h ;|T|取 10,000,000 ,|Q|取 16/32/64/128/512/1024/2048/4096)【容易】

1.2 欧拉距离采用点对点的方式计算序列间距离 ,因此对于在时间尺度上具有伸缩的两个序列(如不同语速的相同发音形成的语音序列)度量效果不好。DTW（Dynamic Time Warping）是一种能够在时间伸缩的序列上度量距离的方法，经大量证实在诸多场景中具有很好的表现。请对 1.1 中的算法进行改进，使其支持基于 DTW 的相似性搜索，分析其复杂度并验证其扩展能力，分析结果并于 1.1 的结果进行比较。实验参数可参考 1.1 【较容易】

（注：DTW 距离能够度量两个任意长度序列，而非限制两序列等长度）

1.3 从实验 1.2 中或许可以发现，DTW 距离计算具有很高的计算代价，这也是其最主要的限制。因此，有诸多研究旨在提高 DTW 计算和基于 DTW 搜索的速度。请参考论文《Searching and mining trillions of time series subsequences under dynamic time warping》 关于距离计算和相似性搜索的加速策略，实现能够支持超长序列的高效搜索算法，分析其复杂度并验证其扩展能力。将实验结果与 1.1 和 1.2 的结果相比较；采用控制变量的准则对这些加速策略分别进行测试 ,分析这些加速策略行之有效的原因及其对可扩展性能的贡献。【困难】

32、【X=1.2】

存储过程（Stored Procedures，简称 SP）是使用数据库系统支持的过程式语言，其编写的目的是对数据库系统现有的功能进行扩充和封装，提升数据库系统

易用性和处理复杂任务的性能。近年来，分布式数据库系统被广泛应用。如果为分布式数据库系统添加存储过程的支持，就可以使得其拥有能与传统单机数据库媲美的易用性和性能。但由于数据被划分到不同的机器上，复杂性显著提升。这给为其添加存储过程的支持造成了困难。一种可行的方法是，将系统的存储模块和计算模块分离，部署到不同的节点中。具体到数据库系统来说，存储模块负责数据的存储和读写，而计算模块则负责 SQL 查询的编译、优化、和执行。
实验要求：设计并实现传统分布式架构下的存储过程执行引擎，达到可以动态的创建计算模块的目的。（可参考 TiDB）

33、近似等高直方图【X=1.1】
背景：数据直方图是数据管理中一种基础且常用的数据统计方式，在近似查询处理，查询优化等多方面均有应用。等高直方图（Equi-depth histogram）将数据划分成 k 个桶，使每个桶中的数据量相同。直方图的构建在于确定桶的边界位置，而等高直方图中桶的边界对应着数据中的分位数。
实验内容：请学习 Spark 中 *approxQuantile* 函数的算法(该算法源于文献[4])，并提出改进算法。要求：
设计一维等高直方图的近似划分方法，在 Spark 上实现该算法，并与 Spark 中的 *approxQuantile* 进行性能对比，对比内容包括但不限于算法准确性和效率。
设计二维等高直方图的近似划分方法，在 Spark 上实现该算法，并评估其准确性和效率。
数据：自行选择每种不少于 2 个（包括 2 个）且足够大的一维及二维数据集进行测试（一共至少 4 个，一维数据集 2 个、二维数据集 2 个），真实数据或模拟数据均可，模拟数据需标明生成方式及数据分布特征。
参考文献：
[4] Greenwald M.B., Khanna S. (2016) Quantiles and Equi-depth Histograms over Streams. In: Garofalakis M., Gehrke J., Rastogi R. (eds) Data Stream Management. Data-Centric Systems and Applications. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-28608-0_3

34、【X=1.1】挖掘数据流模式检测多维时间序列数据中的异常
应用背景：
时间序列是一种工业数据中一种常见的数据类型。有效的时间序列异常检测

算法已经广泛应用于现实世界的很多领域，如量化交易、网络安全检测和大型工业设备的日常维护等。在这些场景中，一些场景对异常检测的时效性要求特别高，这就需要在实时的流数据中实现准确高效的的异常检测算法。传统的异常检测算法需要长时间的模型训练，显然无法达到这样的要求。

实验目标：

选择使用一种大数据流计算系统，实现一种实时有效的流计算异常检测模型。可以参考这篇论文：Streaming Pattern Discovery in Mutiple Time-Series。利用实验数据，设计对比实验。采用合适的指标检测算法的时空性能（包括实时检测的能力，F1 score, AUC 值等）。数据集可以在 UCR Archive 或 Kaggle 上搜集。