IEEE知识和数据工程事务

最长的增加子序列  
 流序列的计算

李友环，雷邹[[1]](#footnote-2)1 张华明2 赵东燕1

1 中国北京大学；

2 美国亨茨维尔阿拉巴马大学

1 (李友环，邹磊，zhaody}@pku.edu.cn[2](mailto:2hzhang@cs.uah.edu) [赫章@cs.uah.edu](mailto:2hzhang@cs.uah.edu)

摘要-在本文中，我们提出了一种数据结构，即四重邻居列表(简称QN列表)，以支持对所有最长增加子序列(L IS)和LIS的实时查询，并对顺序数据流进行约束。 由我们的算法构建的QN列表需要O(W)空间，其中w是时间窗口大小。 构建初始QN-List的运行时间需要O(wlogw)时间.. 应用QN列表，插入新项目需要O(logw)时间，删除第一项需要O(W)时间。 据我们所知，这是第一项支持LIS枚举和LIS的约束计算的工作，使用单个统一的数据结构进行实时顺序数据流。 我们的方法在时间和空间成本上都优于最先进的方法，不仅在理论上，而且在经验上。

索引项-数据流，最长的增加子序列，枚举，约束。

.

1. 导言

­­ 序列数据是由一系列数据点组成的时间序列，这些数据点是通过在一段时间内进行的连续测量得到的。 对序列数据进行了大量的技术研究，如（近似）模式匹配查询[1]、[2]、聚类[3]。 其中，在序列数据上计算最长增加的子序列是一个经典问题。 越来越多了­­1 子序列是一个子序列，其元素按从最小到最大的顺序排序。 请注意，一个序列可能包含多个LIS，它们都具有相同的长度。

除了静态模型(即在给定序列a上计算LIS)外，流模型[4]、[5]还考虑了计算LIS。 给定一个无限的时间演化序列=修1，...，一个^}(A，R)，我们在时间窗口{a诱导的子序列上连续计算LIS­­*;*-\_( *w* 1)，aj\_(w\_2)，.，ai)。 时间窗口的大小是数据流中跨度的项数。 考虑图3中窗口W下的={3，9，6，2，8，5，7}序列（第2节）。 a有四个LIS：{3、6、7}、{3、6、8}、{2、5、7}和{3、5、7}。 除了LIS枚举之外，我们还介绍了LIS的一些重要特性，

1. e.、间隙、重量、斜率、范围和计算具有各种约束的LIS，如具有最大间隙的LIS，其中“间隙”测量LIS的尾部和头部项之间的值差。 在四个LIS中，{3,6，8}和{2，5,7}都是差距最大的LIS。 第2节正式定义了更多的约束。 我们讨论了两个例子来证明LIS在不同应用中的有用性。

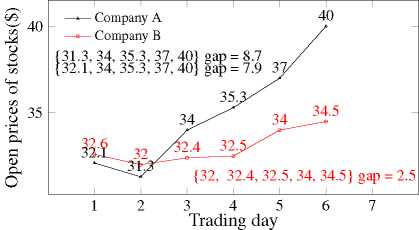


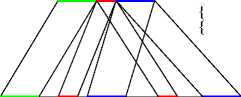
图1 1：不同股票价格序列差距的LIS

**1：实时趋势检测。**

信息系统是排序和趋势分析的经典度量[6]、[7]、[8]。 序列的LIS越长，序列显示的排序越多，这进一步表明序列的上升趋势[6]。 例如，基于LIS的股票趋势检测在Jin等人中进行了研究.. [7]。 一个公司的股票价格形成一个时间变化的序列，实时测量股票趋势对股票分析具有重要意义。 给定一个周期内股票价格的序列a，一个衡量价格上涨趋势的LIS。 我们可以看到，长期LIS的价格序列对股票价格总是表现出明显的上升趋势，即使存在一些价格波动。

虽然LIS长度可以用来测量上升的稳定性，但不同间隙的LIS表明不同的生长强度。 例如，图1显示了两家公司的股票价格序列：A和B。 虽然A和B的序列具有相同的LIS长度（5），但生长

IEEE知识和数据工程事务



基因组序列L

{1,21,31}

{1,21,32}

{1,22,32}

123

1 2131

22 32

图1 2：生物序列比对

A的股票强度明显占B的主导地位，这很容易从A和B中LIS的不同间隙中观察到。 因此，除了LIS长度外，间隙是LIS的另一个特征，它加权了生长强度。 我们认为，具有极端间隙的LIS的计算比随机LIS更有可能被选择作为生长强度的测量。 此外，两个项目之间的斜率（见定义5）也可以用来描述股票价格的上涨强度。

查询成绩单/蛋白质Q

**2：序列匹配。**

在序列匹配[4]，[9]中也使用了LIS，主要用于生物序列查询.. 一个典型的例子是张[9]提出的两步算法(BLAST+LIS)，在人类基因组图中定位转录本或蛋白质序列。 BLAST（基本局部比对搜索工具）[10]算法是在查询转录序列Q和长基因组序列L之间识别高分段对(HSPs)。­ 图2可视化BLAST的输出。 相同颜色（数）的片段表示HSP。 例如，第2段（红色的）在基因组序列L中有两个匹配，表示为21和22。 为了获得全局对齐，基因组序列L中片段1、2、3的匹配应该与查询序列Q中的片段顺序重合，这恰恰构成了图2中列出的LIS(以L为单位)。 例如，LIS(1，21，31}表示Q在序列L上的全局对齐。实际上，L中有三种不同的LIS，如图2所示，它们对应于查询转录本/蛋白质Q和基因组序列L之间的三种不同的对齐。显然，只输出一个LIS可能会错过一些重要的发现。 因此，我们应该研究LIS枚举问题。

我们将上述LIS枚举应用扩展到滑动窗口模型[11]中。 事实上，Q对L的整个对齐结果的范围不应该太长。 因此，我们可以引入一个阈值长度|w|来发现所有跨越不超过|w|项的LIS，即每个时间窗口中大小为w|的所有LIS。 这类似于我们在本文中的问题定义。 此外，有些可能限制两个连续HSP在一定范围内的距离，这对应于定义5中的范围约束LIS。

­ 因此，LIS是有用的，应用程序需要同时有效地计算LIS枚举和约束LIS枚举，但现有的方法都不支持这一点。 例如，[5]中的方法支持LIS枚举，但未能计算约束LIS。 在[12]、[13]和[14]中，该方法可用于计算约束LIS，但不能枚举所有LIS。 支持LIS枚举和约束LIS枚举的统一方法是可取的。

此外，许多工作是基于静态序列的，在这些工作中开发的技术不能处理在数据流上下文中必不可少的更新。 据我们所知，只有三篇研究文章讨论了通过数据流模型计算LIS的问题[4]、[5]、[15]。 它们都不计算约束LIS。 文献综述和我们的方法与其他相关工作的比较研究分别在第6节和第7节中给出。

1. 我们的贡献

从上述例子中观察到，本文提出了一种新的解决方案，即在数据流模型下，用均匀的方法研究具有约束的LIS枚举和计算LIS。 我们提出了一种新的数据结构来有效地支持LIS枚举和具有约束的LIS。 此外，我们还设计了一种有效的更新算法来维护我们的数据结构，以便我们的方法可以应用于数据流模型。 对该算法进行了理论分析，证明了该方法的性能优于实际工作。 我们证明了我们的数据结构的空间复杂度是O(W)，而在[5]中提出的算法需要一个大小为O(W)的空间2 )。 我们的数据结构构建和更新算法的时间复杂性也优于[5]。 例如，[5]需要O(w2 数据结构构建的时间，而我们的方法需要O(^w日志w)时间。 此外，我们还证明了具有约束查询算法的LIS枚举和LIS都是最优输出敏感算法2 。 第6节对我们的结果与以前的结果进行了全面的比较研究。 在真实和合成数据集上的实验结果证实了我们的算法优于现有的算法。 实验代码和数据集可在Github[16]获得。 我们总结了我们的主要贡献如下：­

1. 在数据流模型中，我们首先考虑具有约束的LIS和LIS枚举的计算。
2. 我们引入了一种新的数据结构来处理LIS枚举和所有具有约束的现有LIS的计算。
3. 由于线性更新算法和线性空间代价，我们的数据结构在流模型中是可扩展的。
4. 广泛的实验证实了我们的方法的优越性。
5. 问题的解决

给定序列a=(们，叽-，an}，a的增子序列s是一个子序列，其元素按从最小到最大的顺序排序。 如果a的一个增加子序列s没有其他增加子序列s，则称为最长增加子序列(L IS)，||<s­ z |。 一个序列a可能包含多个LIS，它们的长度都是相同的。 我们表示集合

1. 算法时间复杂度与相应的输出大小呈线性关系..

IEEE知识和数据工程事务

由LIS(A)组成的LIS。 对于序列s，s的头和尾项分别表示为S和S。 我们用|s|表示s的长度.

*艾 a2 a. 4a a*

[3 9 6 2 8 5 7]

**i\***  时间窗口  **j**

斜率约束LIS(p=1.5)：{a.*4*=2，^=5，=7}范围约束LIS(L=2，U=3，L*v =1，你 v* =3)：{a]第6=3死=， =8}

具有最大间隙的LIS：{a]=3、a3=6、a$=8}、{a4、2a=5、aq、7}具有最小间隙的LIS：{a1=3、a6、5、a7=7}、{a1=3、a3、6、a7、7}

**图1 3：数据流模型中具有约束的LIS**

考虑无限时间演化序列a 致 ={ai，...，一个^}(aie R)。 在序列a中 致 每个a都有一个唯一的位置i和ai发生在相应的时间点ti，其中“<TJ时，0<我<j。 本文利用元组-基滑动窗模型[11]。 根据元组到系统的到达顺序，元组有一个内部位置，确保在另一个位置较高的输入元组之前尽可能处理输入元组。 滑动窗口W包含{ai，-，a中连续的项块 致 }，W每向a移动一个单一的位置单位 致 不断地。 我们用w表示窗口W的大小，w是窗口内的项数.. 在[ti，ti+1)时间内，滑动时间窗口W内的a项诱导序列{ai\_( *w* \_i)，ai\_(. *w* \_2)，...，ai}，它将用a(W，i)表示。 请注意，在滑动窗口模型中，随着时间窗口不断向a移动 致 在每移动一个单位的速度下，形成的序列和其所有LIS的相应集合也会相应地改变。 在本文的其余部分中，所有考虑的LIS相关问题都在具有滑动窗口的数据流模型中。

*定义1. (L IS-枚举) 给定一个时间演化序列a* 致 ={，...，a 致 *}和大小为w的时间窗口W，LIS-枚举是在窗口W滑移时连续报告LIS(a(W，i)(即W内的所有LIS)。 同一时间窗口中的所有LIS都有相同的长度。*

正如导言中提到的，一些应用程序对使用约束计算LIS很感兴趣，而不是简单地列举所有这些限制。 到目前为止，文献[12]、[13]、[14]提出了八种LIS约束，我们的方法可以很容易地支持它们。 我们研究了LIS的权重（定义2)、间隙(定义3)和宽度(定义4）的以下约束，然后定义了几个具有不同约束的LIS计算问题（定义5）。

*定义2. （重量） 设是序列，s是LIS(A)中的LIS。 定义s的权重为^^. a. e 这是ai，即s中所有项的总和，我们用重量表示它。*

*定义3. （差距） 设是序列，s是LIS(A)中的LIS。 间隙s被定义为间隙=s一s”，即尾s之间的差异 t 还有头 h 的人。*

*定义4. （宽度）[12]。 设一个序列，s是LIS中的LIS(A)，其中s=岡，a‘：...，AIJ(k=|s|)。 宽度定义为宽度=IK一II，即尾项S)与s的头项/Q之间的位置距离。*

*定义5. (约束下的LIS计算)。 给定一个时间演化序列a* 致 ={，...，a 致 *}和滑动窗口W，以下每个问题都是在W幻灯片中连续地报告所有受其自身指定约束的LIS。 考虑一个(W，ti)：s=的LIS {a 二、 a i：，...， a im*}*.*

* *如果s在LIS中的所有LIS中具有最大/最小权重，则s是具有最大/最小权重的LIS(a(W，t) i*)).
* *如果s在LIS中的所有LIS(a(W，ti)之间有最大/最小间隙，则s是具有最大/最小间隙的LIS。*
* *如果s在LIS中的所有LIS中具有最大/最小宽度，则s是具有最大/最小宽度的LIS(a(W，t) i*)).
* *­ 如果对于非负斜率边界p和所有i<k<m，则s中两个连续项之间的斜率不小于p，即D：>p，则s是一个斜率约束LIS(SLIS)。*
* ***如果两个范围[Li，Ui]和[L，则s是一个范围约束的LIS(RLIS)*** *v 其中0<Li<Ui<n，0<L v<UV，李<IK+我\_IK，UI和Ly<AIK*+ *我一AIK<UV(我<k<m)。*

图3给出了整个论文中使用的运行示例，它显示了一个时间演化序列a 致 它的第一时间窗口W.时间窗口内的诱导序列是一个={ai=3，a：=9，一个^=6，A4，2，如=8，a(s=5，a？ =7}。 a有四个LIS：{3，6,7}，{3,6，8}，{2，5,7}和{3，5,7}。 这四个LIS的差距是4，5，5，4。 因此，{3、6、8}和{2、5、7}具有最大间隙，而{3、6、7}和{3、5、7}具有最小间隙。 另外，当我们设置斜率p=i时。 5（如图3)，则{3、6、8}不满足斜率约束，因为^=6之间的斜率和=8之间的斜率是(8一6)/(5一3）=I<I。 5. 同样，{3，6，7}和{3，5，7}不满足斜率约束，只有{2，5，7}是斜率约束的LIS。 对于具有范围[L的范围约束LIS *i* =2、Ui=3]和[Ly=i、Uy=3]、{2，5，7}和{3，5，7}不满足范围约束，因为^=5和^=7之间的位置距离是I<Li2。 同样，A3=6和A7=7也不能构成范围受限的LIS，因为7一3=4>UI=3 {3,6，8}是唯一受范围限制的LIS。

1. 四邻列表L a
   1. 背景和定义

为了便于演示，我们在正式定义四重邻居列表(简称QN-List)之前介绍了LIS的一些概念。 考虑一个序列={ai，a2，...，aw}和ai，ajea 如果我<j和ai<AJ，ai据说与AJ兼容。 我们用人工智能表示它。 此外，我们使用IS *a* 表示以ai结束的a的所有增加子序列的集合，并定义上升长度[5]3 人工智能，表示为RL *a* (ai)，作为iSa(ai)中子序列的最大长度。 例如，考虑序列a={a. i *=3，a.*2 *=9，a.*3 *=6，a.*4 *=2，a.*5 *=*

1. 本文中的上升长度与[5]中定义的高度相同。 我们这里不使用高度来避免混淆，因为高度也被定义为[13]中LIS的头项和尾项之间的差异。

IEEE知识和数据工程事务

图1 4：水平列表和QN列表

横向名单

(b)QN-List La

图1 5：前人的素描

图1 6：达格

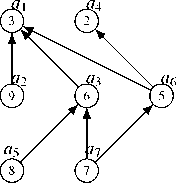
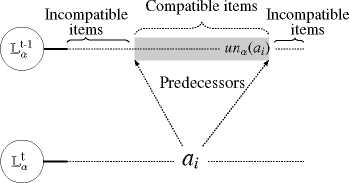
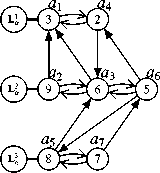


图3中的8，=5，a=7}。 考虑一个=8。 有五个增加子序列{a5=8)，{ai=3，a=8)，{ *a* 3=6， *a 5=* 8)，{a. 4=2， *a* 5= 8)，{a. 1= 3，a 3=6，a5=8)以a、 这些增加子序列的最大长度为3。 因此，RL *a (a5)第3=。*

*定义6. （预测）。 给定一个序列a和ai，对于某些项aj，aj是aiif的前身*

*a*

*<ai和RL a (A J)=RL a (ai)1*

*并将ai的前辈集合表示为Preda(A I)。*

在图3中的运行示例中，^是前身a

自A3<a、rla(A3)（=2）=rla(A5)（=3）1以来，a5。 类似地，ai也是^的前身..

根据上述概念，我们为每个项目a介绍了四个邻居，如下所示：

*定义7. （物品的邻居）。 给定一个序列a和ai，a最多有四个邻居。*

1. ***左邻居LNA(A I)：如果AJ是A‘之前最近的项目，则^A(A I)=AJ。***
2. ***右邻RNA(A I)：如果AJ是AI后最近的项目，则RNA(A I)=AJ，从而使RLA(A I)=RLA(A J)。***
3. *上邻居una(A I)：如果aj是ai之前最近的项，则una(A I)=aj，使得rla(A j)=rla(A I)1。*
4. ***下邻DNA(A I)：如果AJ是ai之前最近的项目，则DNA(A I)=AJ，使得RLA(A J)=RLA(A I)+1。***

显然，如果ai=lna(A J)，那么aj=rna(A I)。 此外，我们还知道，项目ai的左邻右舍（也是右邻右舍）具有与ai相同的上升长度，并且自然地，根据它们的左右邻居关系链接的项目形成了一个水平列表，这在定义8中被正式定义。 图4a显示了a的水平列表。  **定义8***­ （横向清单）。 给定一个序列a，考虑由所有上升长度为k的项组成的子序列：sk=岡，ai，...，ai k }，i1<i2ik。 我们知道1<k<k，一个^ k> =叽⑷八)和AIG=Rnag)。 我们将Sk中的项与左、右邻关系连接起来形成的列表定义为一个水平列表，表示为L k a.*

显然，对于VaieL%，ai的前身必须在L尸（>1）。

***定义9. (四邻名单(QN-名单)。 给定一个序列a=(A1，...，a*** *w }，a(表示为La)上的四重邻居列表是包含a的所有水平列表（定义8）的数据结构，La中的每个项ai也直接链接到其上邻居和下邻居。 从本质上讲，La是通过将a中的所有项与它们的四种邻居关系联系起来来构建的。 尤其是|L* a *|表示La中水平列表的数量。*

图4b显示了运行示例序列a的QN-ListLa（图3），曲线箭头表示左右邻居关系，而直箭头表示上下邻居关系。 很容易理解，a中LIS的长度正好是La中水平列表的数量。 此外，QN列表超过w项的序列a只花费O(W)空间..

* 1. La-Properties

我们讨论了QN-List的一些性质，它将用于第4节中的维护算法和第5节中基于QN-List的各种算法。 定理和引理的证明见附录B4.

*引理1 让一个={a1，a，.，a w }是一个序列。 考虑水平列表L中的两个项ai和aj t a.*

1. *每个水平列表L中的项目 t a 单调递减，而它们的下标(即它们在a中的原始位置)从左到右单调递增。*
2. *在LJ中，ai的所有前辈都形成了一个非空的连续块*1 *(t>1)和una(ai)是最右边的前身。*

在图4a中，我们可以看到每个水平列表中的项从左到右增加，而它们的原始位置正在减少（引理1(1)）。 此外，图5显示了所有的前辈的人工智能 *t a* 从UN形成一个连续的块 *a* 在LJ的左边1 （引理1(2)）。 引理2 *给定序列a和L* a *，V ai e La：*

1. *rl a (a) i =t当且仅当a. i* e L *t a.*
2. *联合国 a (ai)（如果存在）是LJ中最右边的项目*1 *这是在A之前 i 按顺序a。*
3. *DN a(ai)（如果存在）是L^中最右边的项*1 *这是在序列a的ai之前。 此外，dna(ai)>ai。*

例如，在图4b中，a*3* 是在a、inL；a(RL)之前的最右边的项目 *a* (a5)=3)和联合国 *a (a5)=^。*  **引理3** *给定序列a及其L* a *对于1<I，j<|L* a|

*尾巴(L. l a <尾(L)一我<j*

*尾巴(L l a)表示列表L中的最后一项 l a.*

1. 所有附录均在提交的补充文件中给出。

IEEE知识和数据工程事务

(c)3,9，6

(d)3，9也

(e)39628

3，a2

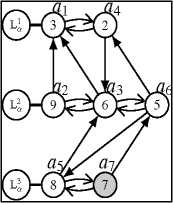
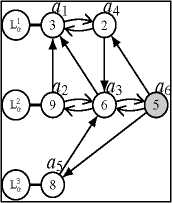
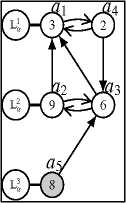
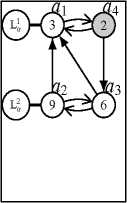
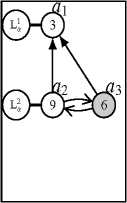
(f)396285

*9，a=6，*  *=2，a..5*

(g)396285]

8，a=5，a=7}

**图1 7：构造QN-序列列表的示例：{ai**



1)

2)

3)

4)

例如，在图4b中，我们可以看到由尾项组成的序列：{a4=2、a6=5、a^7}是升序的。

* 1. La-建筑

升的建设. a 过序列a在于确定a中每个项的四个邻居。 我们讨论La的建设如下。 图7可视化了构造L的步骤 a 为了一个序列a。

**建筑QN-ListLa。**

最初，每个项ai的四个邻居被设置为NULL；步骤1：L i a 在La中创建，在L中添加ai i a;

第二步：如果a2<ai，表示RL. *a (a2)=RL a* (ai)第1=。 因此，我们将a2附加到L ia 。 由于a2在序列a中次于ai，所以我们设置rn *a (ai)=a？* 和ln *a (a2)=人工智能。*

如果a2>ai，我们可以找到一个增加的子序列(ai，a2)，即rla(A2)=2。 因此，我们创建第二个水平列表L2 a 并将A2添加到L中2 a 。 此外，知道ai是a最近的前身是很简单的吗？ 因此，我们将una(A2)设置=a

第一步：假设第一个i项已正确地添加到QN List中(本质上，在a的第一个(i)项的子序列上建立QN列表)，让我们考虑如何将第i项ai添加到数据结构中。 设m表示当前La中水平列表的个数.. 在将ai添加到La之前，让我们首先计算出a的上升长度，考虑一个水平列表L；，我们有以下两个结论：

1. *如果尾(La)>ai，则rla(ai)<t。* 假设RLA(A I)>t。 意味着至少有一个出口

a

项目A J(ELA)，使AJ<ai，即AJ是ai的前身（或递归前身）。 如我们所知，T艾尔(L *t* a 是L中的最小项 *t* a （见引理2）。 尾(L)>ai表示a中的所有项.

拉比a大，这与aj<ajela相矛盾。 因此，RL *a (a[)<t。*

1. *如果T病(L t* a <ai，然后rla(ai)>t.自从tall(L. *t* a )是在a和t的(L *t* a *)<艾，泰尔 t* a 是兼容的ai。 让我们考虑一个越来越多的subseqeunces以尾(La)结束，它的长度是t，因为TAL(L *t* a )的上升长度为t.显然，s的=㊉ai是一个长度-(t+1)增加子序列，以ai结束。 换句话说，ai的上升长度至少为t+即RLA(A I)>t。

此外，我们知道尾(L^)>尾(L，)如果t>t（见引理3）。 因此，我们需要找到第一个列表L *t* a 谁的尾巴T小(L *t*a )大于a *i* 。 然后，我们把ai附加到列表中。 由于所有尾项都在增加，我们可以执行需要O(logm)时间的二进制搜索。 如果没有这样的清单，即TAL(L a *m* <ai，我们创建一个新的空列表Tall(L a *我+我* 并将ai插入TAL(L a *我+我*).

根据引理1，很容易知道ai只能附加到L的末尾 *t* a ，即RNA(T il(L. *t* a *)=ai和lna(ai)=tall(l. t* a )。 此外，根据引理2（2），我们知道 *a* (ai)是L}中最右边的项1 这是在ai之前，然后我们设置 *a (ai)=尾(LJ)* i ). 类似地，我们设置了DN *a (ai)=大九龙^1*)。 到目前为止，我们正确地确定了ai的四个邻居。 我们可以重复上述步骤，直到所有项目插入到La。 在附录A的算法1中给出了建立QN-ListLa的伪码..

显然，建立QN列表的时间复杂度为O(wlogw)..

3.4LIS枚举

让我们讨论如何枚举基于QN-ListLa的序列a的所有LIS。 每个LIS的最后一项必须位于L的最后一个水平列表 a 我们可以通过枚举所有|La|长度增加子序列来枚举a的所有LIS，以L中的项结束？。 为了方便起见，我们使用MISa(A I)表示以ai结尾的所有RLA(A I)长度增加子序列的集合。 考虑最后一个列表La中的每个项目ai a| 。 我们可以用a计算一个结尾的所有LIS *i* 通过迭代搜索上面列表中的ai前辈，从下到上，直到到达第一个列表L i a 。 这是我们的LIS枚举算法的基本思想..

为了简洁起见，我们实际上创建了一个有向无环图(DAG)，以便更直观地讨论La上的LIS枚举。 DAG是基于a中项之间的前身关系定义的。 DAG中的每个顶点对应于a中的一个项。 从a插入有向边. *i* 给一个 *j* 如果是的话 *j* 是a的前身 *i* (a) *i* 还有a *j* 也分别称为父母和孩子)。

*定义10. (DAG G(a))。 给定一个序列a，有向图G表示为G(A)=(V，E)，其中顶点集V和边集E的定义如下：.*

*=(ai\aiea)；E=(ai，aj)\aj是ai}的前身*

IEEE知识和数据工程事务

序列a{3，9，6，2，8，5，7}上的G(A)如图6所示，其中每条长度|L a |对应于LIS。 例如，我们可以找到一条路径，=8t为=6-ai=3，对应于LIS（3，6，8）。 因此，我们可以很容易地设计一个类似于DFS的导线，从L^中的项开始，以输出长度|L的所有路径 a 在G(A)中|。

请注意，我们实际上不需要在我们的算法中构建DAG，因为我们可以等效地在L上进行类似DFS的遍历 a 。 首先，我们可以很容易地访问L中的所有项目 a它们是导线的起始顶点。 其次，类似DFS的遍历中的关键操作是获取顶点的所有前辈。 事实上，根据图5所示的引理1，我们可以通过搜索LJ找到ai的所有前辈1 从联合国 *a* (ai)向左，直至满足与ai不兼容的项目a\*。 搜索过程中所有被触摸的项目(a\*排除)都是ai的前身。

我们从每个项目ai中构造LIS *m* 在L\*（即最后一份清单）中如下。  *艾 m* 首先被推入初始空堆栈的底部。 在每次迭代时，顶部项的向上邻居被推入堆栈。 该算法继续进行，直到它将L%中的项推送到堆栈中，并在堆栈中输出项，因为这是堆栈保存LIS的时候。 然后，该算法开始从堆栈中弹出顶部项，并将当前顶部项的另一个前身推入堆栈。 很容易看出，该算法非常类似于深度优先搜索(D FS)，更具体地说，该算法输出所有LIS如下：（1）L\*中的每个项被推入堆栈；（2)在每次迭代时，每个前身(可以从上邻居扫描到左的水平列表，直到发现不兼容的项)在堆栈中被推送；(3）堆栈内容在满时被打印。 附录A中的算法2给出了LIS枚举的伪代码。

*定理1. 对于LIS枚举的时间复杂度为O(OUT PUT)，其中OUTPUT是所有LIS的总大小。*

1. 维修

当时间窗口滑动时，ai被删除，一个新项目a *w+* 我被附在a的末尾。 很容易看出，四重邻居列表维护包括两个操作：删除第一项ai和插入a *w+* 我到最后。 当我们讨论如何在第3节中构造QN列表时，插入已经被处理(参见附录A中算法1的第2至10行)。 因此，我们只考虑在本节中“删除”。 序列{a？？ ，-，a *w* 从a中删除ai形成的}表示为a 。 我们将四重邻居列表维护的讨论分为两部分：用于更新左、右邻居的水平更新（4.1节)和上下邻居的垂直更新(4.2节）。

4.1横向更新

***定义11. (k-Hoip上邻居)。 让一个={ai，a？*** ，a *w }是一个序列和L* a *是它对应的四重邻居列表。 对于Vai£a，k-跳起来的邻居un！ a(ai)定义如下：*

*()=j a i  k =*0

*尤纳(A I) =\联合国 a unk1 -我 (a) i )k>我*

我们首先用一个运行的例子来说明主要思想和算法的草图。 随后给出了更多的分析和算法细节。­

运行的例子和直觉.. 图8(A)显示了运行示例中序列a的QN列表La.. 删除ai后，La(1<t<m)中的一些项目应提升到上述列表LJ1 其他人还在L *t* a 。 定理2告诉我们如何区分它们。 简单地说，VAeLa(1<t<m)，如果它的(t i)-向上跳邻居是ai（要删除的项），则应该将a提升到上述列表；否则，a仍然在同一列表中。

例如，图8(A)和图8(B)显示了删除ai前后的QN列表。 ­ *{a？*由于}在L中；a和它们的1跳邻居是ai（要删除的项），因此，它们被提升到L的第一个列表 a 。 此外，{as}在L\*中，其2跳邻居也是ai。 也晋升为L\*\_.. 更有趣的是，对于每个水平列表La(1<t<m)，需要提升的项目在L的左侧 *t* a 表示为左(La)，这是图8(A)中的阴影部分。 请注意，左(L^)={ai}。 拉的右（剩余）部分表示为右(La)。 水平更新是耦合左(中尉+ *i* )与右(La)进入一个新的水平列表La。 例如，左(LO)={a？ 作为}加权(L\*)={a^形成L\*\_{a？ 如图8(B)所示，A4}。 此外，图8(A)中的红色粗体线表示从ai开始的左侧和右侧部分之间的分隔线。 附录A中的算法3研究了如何找到分隔线来划分每个水平列表L *t* a 有效地分成两部分。  **分析和算法。**  引理4告诉我们，同一列中两项的上邻关系不交叉，这在定理2的证明中使用..

**引理4 让一个={aia.** *w }是一个序列和L* a *是它对应的四重邻居列表。 设m为L中水平列表的个数.* a *。 设ai和aj是La，t>1中的两个项目。 如果ai在AJ的左边，则un^ai)=un%(A J)或una(A i)在u^(A J)的左边，每0<kt。*

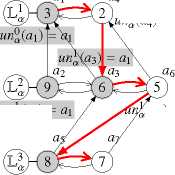
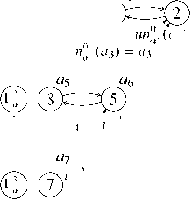
*定理2. ^^一个序列a={a，a？ ，-，a w }和La.. 让我=|L* a *|。 让一个 ={a？ ，-，a w 通过删除ai从a中获取}。 那么对于任何一个ai，2<我<meLa，1<t<m，*

1. *如果联合国 t j i (ai)是ai，然后是RL a -(ai)=RL a (ai)第1一。*
2. *如果联合国 t j i (ai)不是ai，那么RL a (ai)=RL a (ai)*

天真的方法。 利用定理2，更新水平列表的直接方法是计算u^~ *x* 每个人工智能在L *t* a 。 如果联合国 *t* a *i* 是人工智能，将人工智能提升为L *t* a *i* 。 在将项目分组到正确的水平列表之后，我们通过它们的值的递减顺序对每个水平列表的项目进行排序。 根据定理2和引理1（1）可知，上述过程得到的水平列表与重新建立Lafor序列a相同..

优化方法。 每个项目a *i*在洛杉矶 *t* a (我<t<m)在运行示例中，我们在图8a中报告了它的(t i)向上邻居。 阴影顶点表示其(T I)向上邻居在La中是ai的项；其他的在白色顶点中。 有趣的是，这两类项目

IEEE知识和数据工程事务



皿「（。2）=。2

2

第7况h

中尉-H8

(a)5 )=A4

*wna\_(a)*6 )=A4

*6*

“底（.5)=展(.7）

(a)司。

( *a* 6)= *a4*

一个列表形成两个连续的块。 阴影的一个在左边，另一个在右边。

让我们回顾引理4，其中说，同一清单中的两个项目的上邻关系不交叉。 事实上，在删除ai之后，对于每一个eL；，如果 是艾，

然后，对于L；，WNJ中ai左侧的任何项目AJ1 (A J)也是人工智能。 当，如果不 *t1* (ai)不是ai，那么对于任何a项 *k* 在人工智能的右侧，在L%，UN *t~ i* (ai)不是ai。 这两种说法可以由引理4证明。 这就是为什么两类项形成两个连续块的原因，如图8a所示。

删除ai后，我们可以将每个列表L分成两个子列表：左(LO)和右(Right)。 对于左(L^o)的任何项目，un *t-i* (A J)是ai，而对于任何项目，AKe右(L；)，un *t-i* (A K)不是人工智能。 我们提出了一种有效的算法(附录A中的算法3)，将每个水平列表La划分为两个子列表：左(La)和右(L；)，而不是计算每个项的(t i)向上邻居。

让我们考虑L的每个水平列表的划分 a 。 事实上，在我们的除法算法中，L的除法取决于L；1 。 我们先分L；。 很明显，左边 *i x* =(ai}和权利(L；)=(L；)一(ai}。 递归地，假设我们已经完成了L的划分；，我<t，m，有三种情况可以划分L；1 。 注意，对于每个项目，左(L；)，un *t-i* (ai)=ai；而对于每个项目，aie右(L；)，un *t< -我 (ai)，ai。*

1. *如果右(L；)=空，为VAJEL；* i我们有un；(A J)e左(L；)，因此，un；(A J)正是ai。 因此，以下所有列表都设置为左部分。 具体来说，对于任何t. *f >t，我们设置左(L；)=L；右(L；)=NULL。*
2. *如果右(L；)，则NULL和HEAD(右(L；)为AK：*
3. 如果DN；(A K)不存在，即L；1 当AK插入L时是空的 *t*； 然后L中的所有项目 *t*；  i 在AK之后，他们的上邻居要么是AK，要么是AK右侧的项目，因此，L中每个项目的t跳上邻居 *t*；  i 不可能是艾。 实际上，下面所有的列表都设置为正确的部分。 具体来说，对于任何t>t，我们设置左(L；)=NULL和右(L；)=L；。
4. 如果DN；(A K)存在，则DN；(A K)及其左侧的项目先于a *k* 他们的邻居只能在AK的左边(即Left(L) *t*； 因此，DN的t跳上邻居；(A K)或DN左边的项目；(A K)必须是ai。 此外，在DN右侧的项目；(A K)在AK之后，他们的上邻居要么是AK，要么是a右侧的项目。 因此，DN右侧每个项目的t跳向上邻居；不能是ai。 一般情况下，我们设置Left(L *t*；  *i* 从L的头部作为诱导子列表 *t*；  *i* 到DN；(A K)（包括）和设置正确(L *t*；  *i* )作为其余部分，即右(L) *t*；  *i )= t*；  *i 勒夫特(L) t*；  *i* )。 我们对剩余的列表迭代上述过程。

最后，对于我<t<m，左边的子列表Left(L *t*； )应提升至上述名单；及权利(L. *t*； )仍在第t项名单中。 具体来说，我们将右(L；\_)追加到左(L；) *i* 形成L； 在运行示例中，我们将右(L；)=(A2，AS)附加到左(L；)=(A4)以形成L；\_=(A？ 如图8b所示。

*定理3. 将右(L；)附加到左(L；)形成的列表* i *从左到右是严格减少的。*

根据定理2和引理2（1），我们可以证明附加权利(L)形成的列表 *t*； *)到乐富(L) t*；  i 表示为L，包含与L相同的一组项 *t*； 是的。 此外，根据引理1（2）和定理3，L和L *t*； 单调递减，因此，我们可以知道L等效于L *t*； 我们可以导出水平列表调整方法是正确的。

*unl\_(a)7 )=。4*

**(b)删除后。**

**图1 8：维修**

4.2纵向更新

除了调整水平列表外，还需要更新四重邻居列表中的垂直邻居关系，以完成从L；到L；的转换。 在提出我们的方法之前，我们回顾引理2（2），其中表示，对于项aiel；，un；(ai)（如果存在）是L中最右边的项；1 谁在ai之前是顺序；而DN；(ai)（如果存在）是L中最右边的项 *t*；  *i* 谁在人工智能之前的顺序；

运行的例子和直觉.. 让我们回忆一下图8。 调整水平列表后，我们需要处理垂直邻居的更新。 下面的引理6告诉我们，在将L；转化为L；时，将保持哪些垂直关系；.. 一般情况下，当我们推广Left(L *t*； 从上面的水平来看，我们需要改变他们的上邻居，而不是下邻居。 同时，对(L. *t*； 水平更新后仍处于同一水平。 我们需要改变他们的邻居，而不是邻居。

例如，左(^=(AS)被提升到L；\_。 在L；，un；(as)是as，但我们把它改为un；(as)=a4，即L中最右边的项；\_谁在前面，按顺序排列； 。 同样，右(L)=(A^)仍处于L的第二级；  *DN；(a)*6 是AS，但我们将其更改为NULL，因为L中没有项；垂直更新的形式化分析和算法描述之前的\_如下。  **分析和算法。**

*引理5 给定一个序列；和L；，对于任意1<t<m：*

1. *左(L；)，DN；(ai)（如果存在）e左(L； i).*
2. *正确的(L； i )，联合国；(ai)（如果存在）e权利(L；)。*

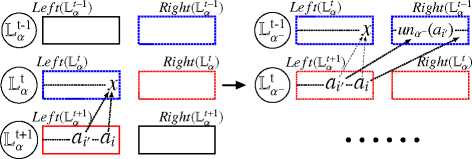
*引理6 让；=(ai，a？，-，aw)是一个序列。 设L；为其对应的四重邻列，m为L中水平列的总数；.. 让=(a)？ -，aw}是从；通过删除ai获得的。 考虑一个项目 t*； *在我<的地方，根据水平列表的调整，ai有两种情况：ai来自Left(L) t*；  *i )或艾是从右边来的 t*； *)。 则，下列主张成立：.*

IEEE知识和数据工程事务

1. *假设艾是从*
2. *DNA(A I)=DN a(ai)（下邻留存）。*
3. *设x是最右边的左边项(L)。 联合国 a (ai)，x，un a (ai)=联合国 a （一)(上邻居仍然）。*
4. *假设ai来自右(L%)*
5. *联合国(A I)=联合国 a (a.i)（即上邻居仍然）。*
6. *让你成为最右边的左(L^1 )。 如果是DN a (ai)，y，dn a (ai)=DN a (ai)（即^^^邻居留下）*

在引理6中，对于一个项aiel%\_，有两种情况需要更新a、a的垂直邻居关系

1. *案例1：ai来自左(L0~1* )。 设x为左(L^的最右项.. 我们需要更新莱夫大学的人工智能邻居 *a* (ai)=x，如图9所示。
2. 案例2：ai来自右(L%)。 让你成为左边最右边的项目(L0+*1* )。 我们需要更新洛杉矶的下邻居 a 如果DN *a* (ai)=y。 图10演示了这种情况。



**(a)删除之前 (b)删除后**

**图1 9：案例1：更新邻居**

案例1：考虑左边的项目(L*: c+ l* )。 根据定理2，左(L­ *t+1* )将被提升为名单L“

让艾是最右边的左边(L *t0+ i 和x=尾（左） t a* ))。 根据引理6(1.b)，如果联合国*a* (ai)更x，然后是un *a (ai)=联合国 a (ai)* 很容易证明：如果联合国 *a* (a.i)黄x然后联合国 *a* 黄x，其中AJ在ai的左边(L^*1* )。 因此，所有项目在左(L^*1* 不要改变垂直关系（见算法4中的第6-8行）。

现在，我们考虑这个案子 *a (ai)=x。* 设^表示左(L)中最左边的项 *t0+ i* )联合国 *a* （人工智能>）为x。 从atoai（包括两者）连续块中的项的向上邻居都是L中的x a (注意，x是左(LO)中最右边的项)，如图9(A)所示。 这些项目的邻居需要调整在L a.

*定理4. 给定一个序列a和L* a *假设那个联合国 a (ai)=x和a？ 表示匚£烏°‘中最左边的项目，其中un a （人工智能<）为x。 如果是DN a (x)，NULL，则^正好为Rn a (DN) a (x))；否则，av就是头(L) 拉^ 艾')*

利用定理4，我们可以很容易地在O（1）时间内找出一个^。 然后，我们首先调整L中^的向上邻居 a 。 最初，我们设置了一个\*=联合国 *a* (ai>)=x。 然后，我们在L^中一步地向右移动a\*1 直到找到最右边的项目，其位置在a之前？ 按顺序a 。 最后，我们设置了联合国 *a* (a/)=a\*(见附录A算法4第14行)。

在运行示例中，当删除图8a中的A1时，左(LO)={a*5* }和un *a* 正如尾项是左(Ll)的^，因为 l\*\_ 是的  *{a^* = 9, = 6, = 2), 由...形成

将右(L0)({A2=9，A^=6))附加到左(L%)({A4=2))，A4是L\*中最右边的项目\_他以前和A一样 然后我们就出发了 *a* (AS)为A4=2(见图8b)。

换句话说，我们考虑一个^右边的项目。 实际上，下一项向上邻居的调整可以从a的当前位置开始\* 。 知道更新邻居的时间复杂度(附录A中的算法4)是O(|LU|)是很简单的，因为L‘中的每个项1 最多扫描一次。

*a*

案例2：考虑所有项目的权利(L)。根据水平调整，右(L^)项的向下邻居是左(L^的尾项（即最右项）*1* )或右边的物品(L^*1).*

实际上，案例2与案例1是对称的。 我们强调以下一些重要步骤。 让ai是右(L)中最左边的项目，让y是尾(L)*: c+1* )，即y是左(L)中最右边的项 *t0+ i* )。 很明显，DN *a* (ai)=y（算法3）。 然后我们扫描右(L)从ai到最右边的项目a^，其中DN *a* （人工>）是y。 从ai到^的连续块中的项的向上邻居（包括两者）都是y(参见图10(A)。 ^右边的项目不需要改变他们的下邻居，因为他们的下邻居在L a 不是y(见引理6(2.b)。

我们只考虑从ai到^的连续块（见图10）如下。 一，我们调整a的下邻？？­ 在洛杉矶 a 。 最初，我们设置了一个\*=大K^左(U^})，即最右边的左边项目(L *t0+ i* )。 然后，我们在LR中一步地向左移动一个\*1 直到找到位置在^之前的最右边的项目。 最后，我们将dna(ai，)设置=a\*(见附录A算法5中的第8行)。

在运行示例中，删除A1=3时，右(L%)是{A4=2)，其头项为A4。 我^ *a* (A4)是第6=^，是左(LO)的尾项。 然后，最初，我们设置DN *a* (A4)为左(LO)的尾项，即DN *a* (A4)=和扫描L\*\_从右到左，直到找到一个最右边的项目，谁之前的A4在A 。 由于L\*\_中没有这样的项目，我们将DN(A4)设置为NULL。

&奶皿%)

*我件舟i* (^0^1  *s)和IAL-*

*— y* ―►（^0^5“纨-(£）。 i i

*左(中尉+*2)  *对(中尉+*2)

……

**(a)删除之前 (b)删除后**

**图1 10：案例2：更新邻居**

换句话说，我们考虑^左边的项目。 实际上，向下邻居的调整可以从a的当前位置开始\*(附录A算法5第11行)。 因此，算法5的时间复杂度为O(|L01 |)，因为我^1 最多扫描两次。

最后，我们可以看到，在a序列中处理头项A1删除的解决方案包括两个主要短语。 第一个短语是划分每个列表L；(1<t<m，然后通过附加权(L)完成水平更新*: a 向左(U^1* ). 第二个短语是进行垂直更新。 处理删除的伪码在附录A的算法6中给出。

*定理5. 我们删除算法的时间复杂度为O(w)，其中w表示时间窗口大小..*

IEEE知识和数据工程事务

5 计算具有约束的LIS

在本节中，我们考虑第2节中定义的各种约束，并计算序列a和L上具有不同约束的LIS­ a 。 由于空间限制，我们在附录C中给出了具有最大/最小重量/间隙/宽度的LIS的计算，这在我们以前的会议文件[17]中已经讨论过，我们重点讨论了斜率约束LIS和范围约束LIS的计算。

1. 斜率约束LIS(SLIS)

­ 根据SLIS的定义（定义5），我们可以发现斜率只约束LIS中的每两个连续项。 因此，斜率在本质上是对序列中一个项目的前辈的约束。 对于项%，满足斜率约束的前辈称为斜率-性质的前辈，因此，SLIS的朴素解决方案是在LIS枚举的计算过程中验证斜率约束。 然而，反复访问没有坡度的项目可能是浪费的。 一个可能的优化是标记那些没有斜率属性的项目，并避免在LIS枚举期间访问它们。 然而，每个项目最多可能有O（|一个|）前辈，标记计算是昂贵的。 此外，具有斜率属性的前辈的项目可能不在SLIS中。 例如，对于只有一个斜率-属性前身aj的项ai，如果aj没有slopeproper前身，则ai将永远不存在于SLIS中。 在这些观察的基础上，我们提出了一种动态规划算法来颜色项（白色或黑色），以确定谁在SLIS计算中被忽略。 我们将首先介绍我们的着色算法（着色短语)，然后讨论如何根据着色结果(输出短语）有效地枚举所有SLIS。

着色过程从QN列表的第一级开始.. 最初，L中的所有项目都是白色的。 我们迭代地处理从L到L^的其他级别1 。 如果ai至少有一个斜率-属性的前身，我们将ai颜色为白色，这是以前有色的白色。 排序后，对于任何白色项ai，必须存在一个以ai结尾的递增子序列，其中s中的每个项都是白色的。

*定理6. 给定一个序列a和ai，ajel^*1 *。 假设a‘，aj>eL；分别是ai和aj的最左边的白色斜率。 如果aj在ai的右侧，那么^要么是^，要么在^的右侧..*

我们可以知道，找到一个最左边的白色斜率-适当的前身为人工智能；1 足以证实人工智能是白色的。 用定理6，在确定了最左边的白色前身aofai之后，搜索了rn的最左边的白色前身. *a* (ai)可从L的权利处进行； 因此，在L；中着色项目后，我们可以在L；中着色项目1 通过扫描L；和L；1 只有一次(附录A算法13中的第3-12行)。

输出SLIS后，为了输出SLIS，我们可以找到一个白色项aiel\*(m=|L；|)，并访问最左边的白色斜率-属性的前身ai。 递归地，我们可以得到一个SLIS（算法13中的第13-21行）。 我们的方法可以很容易地扩展到支持输出所有SLIS。 它类似于LIS枚举方法，但它只在枚举期间访问QN列表中的白色项。 在运行示例上的SLIS计算在附录D中给出。SLIS的颜色成本O（||）时间，而输出SLIS的成本O(L)时间，其中l是SLIS长度。 因此，SLIS计算超过L；代价O(W)时间。

1. 范围受限的LIS(RLIS)

考虑一个序列；两个范围[Lj，Ui]和[Lv，Uv]，其中0<Lj<Ui<n，0<L *v* <乌夫。 对于一个^的人；和一个人；1 。 我们叫a？ 作为ai的范围属性前身，如果aie[Lv，Uv]和ie[Lj，Ui](注意，由于0<Lj和0<Lv，那么我们可以很容易地知道这一点

；

艾，<艾和我<我，即，a？  *是一个*

­ 颜色类似于SLIS计算的解，我们还为每个项目分配颜色。 对于每个项ai，如果RL；(Ai)=1，ai应该是白色的；否则，ai将是白色的当且仅当ai至少有一个白色范围-属性的前身。 非白色物品称为黑色物品.. 我们设计了一种只花费线性时间的高效着色算法.. 最初，项目在L1； 都是白色的。 我们迭代地处理从L；到L；‘的其他级别。 假设我们已经完成了第一个t横向列表中的着色项目(即L；，L；，...，L；)，让我们讨论如何在L中对项目进行着色；1.

对于一个项目来说1 还有一个^，如果是的话？ E[Lv，UV]和I-I *f* e[Lj，Uj]，然后等效地，ai-Uv<一个^<a=lv和iUj，我<iLj。 根据引理1（1），项在L. *t*； 从左到右单调递减，而它们的位置在；单调递增。 因此，对于艾莱 *t*； 1 我们可以在L中找到最左边的物品 *t* ； 这样就<了一个LV和我UJ<L。 显然，在ai的右边，ai<一个^的Lv和i-Uj<l。 同样，我们使用ar来表示最右边的项，使aiU *v* <AR和R<ILJ。 显然，从ai到ainL的项；形成一个连续的块，其中包含ai的所有范围属性前辈(参见附录D中的图23)。 如果ar在al的左边，那么ai没有范围属性的前身；否则，al是a的最左边范围属性的前身 *i* 而a *r* 是最右边的。 然而，a. *l* 可能是黑色的项目，这对RLIS计算是无用的。 因此，我们关注的是最左边的白色范围-属性的ai的前身..  **定理7** *给定序列；和aiel；*1 *假设ay是L中最左边的白色项目，这样一个^<aiLv和我Uj<I。 然后，我们可以得出结论，要么^是最左边的白色范围-属性的ai的前身，要么ai没有白色范围-属性的前辈(即，ai应该是黑色的)。*  **定理8** *给定序列；和L；。 考虑ai，ajel；*1 *。 假设一个^，aj\*el；是ai和aj最左边的白色前辈，使得^<aiLv和i-Uj f 而AJ><AJLv和jUJ<j。 然后，如果aj在ai的右侧，aj要么是^，要么在^的右侧。*

与定理8和定理7，确定最左边白色项a后？？ 在L；这样的a？ <aiLv和iUj<i，如果ai正好是ai的范围属性前身，我们将ai颜色为白色，并将从ai到^的指针设置为最左边的白色范围属性前身；否则，如果^不是ai的范围属性前身，则将ai颜色为黑色。

IEEE知识和数据工程事务

无论a是什么颜色，aj的颜色都=rn *a* 可以通过搜索最左边的白色项AJ进行，从ai到右边，这样AJ，<AJ一L *v 和jUI<j* 它非常类似于SLIS的着色过程(附录A算法14中的第3-15行)。

输出RLIS这个短语和输出SLIS的短语是一样的(见附录A算法14中的第16-24行)。 在附录D中给出了运行示例上的RLIS计算。同样，在La成本上输出RLIS 时间到了。

6实验

­ 我们实验评估我们的解决方案与比较方法。 所有方法，包括比较方法，都由C实现++并在默认设置下由g++（5.2.0）编译。 每一种比较方法都是尽我们最大的努力根据相应的论文实现的。 实验是在窗口进行的

1. (Intel(R)i7-47903.6GHz，8G)。 所有代码，包括比较方法的代码都在Github[16]中提供。
2. 数据集

我们在实验中使用了四个数据集：真实世界的股票数据、基因序列数据、电力使用数据和合成数据。 股票数据是过去二十年微软合作公司的历史公开价格5 多达7400天。 该基因数据集是一个由4，525个匹配位置组成的序列，这些位置是通过mRNA序列的BLAST输出来计算的6 针对基因数据集7 根据[9]中的过程。 电力使用数据集是[18]中使用的公共电力需求数据集。 它包含35,040个功耗值。 合成数据集是一个时间序列基准[19]，包含100万个数据点。 由于空间限制，功率和合成数据的实验结果见附录F。

1. 比较方法

­ 我们将我们的方法(表示为QN-列表)与几种比较算法进行了比较，包括我们以前的方法(表示为QN-Prev)在[17]中。

­ 在“流模型”中，LISS ET[5]是唯一提出LIS枚举的方法。 它枚举每个滑动窗口中的所有LIS，但它无法计算具有不同约束的LIS，例如具有极端间隙的LIS和具有极端权重的LIS。

MHLIS[13]是寻找最小间隙的LIS，但在数据流模型下不起作用。 为了实现比较，我们实现了MHLIS的两个流版本：MHLIS+Re和MHLIS+I/D，其中MHLIS+Re是在每个时间窗口从零开始重新计算LIS，而MHLIS+I/D是在MHLIS中应用我们的更新方法。

在[12]中提出了一系列静态算法，包括最小/最大权重/间隙/宽度的LIS(表示为VARIAN T)。 为了比较，我们实施­

1. [http://finance.yahoo.com/quote/MSFT/history？ltr=1](http://finance.yahoo.com/quote/MSFT/history?ltr=1)
2. ftp：/ftp.ncbi.nih.gov/refseq/B\_taurus/mRNA\_Prot/
3. ftp：/ftp.ncbi.nlm.nih.gov/genbank/

两个流版本的VARIANT：VARIANT+Re和VARIANT+I/D，其中VARIANT+是从零开始重新计算每个时间窗口和VARIANT+I/D是应用我们的更新方法在VARIANT。

我们在比较研究中包括了经典的动态规划(表示为DP)算法。 标准DP算法只计算LIS长度和单个LIS。 为了枚举所有LIS，我们在确定以它结尾的递增子序列的最大长度时保存每个项的所有前辈。

Yang等人。 [14]提出了两种不同的斜率约束LIS计算方法(表示为YangS)和范围约束LIS计算方法(表示为YangR)。 它们侧重于静态序列，并使用范围最大/最小查询(R MQ)[20]来支持有效的范围约束检查。 它们只找到一个RLIS(SLIS)，而我们的方法可以找出满足给定约束的所有LIS。 因此，为了进行比较，本节中的SLIS/RLIS计算只侧重于找到一个合格的LIS。 此外，为了启用比较，YangS/YangR的流版本是通过从零开始为每个窗口重新计算SLIS/RLIS来实现的。­

李松[4]计算了LIS长度，并在滑动模型中输出了LIS。 当更新发生时，他们保持了杨的第一排桌子。 第一行的长度正好是窗口中序列的LIS长度。

mxv為dS

103

**102 “(S m)UI[**

100

103

图1 11：空间

DP 杨S

杨瑞松

**哦，穆奥**

\*DP0Yang S

杨瑞松

QN-列表/QN-前^ 利塞特

变体和mhlis

1

2 3 4

窗户尺寸

(a)库存

10°

*-*

QN-列表/QN-预^LISSET-VARIANTV^MHLIS

12 3 4

窗户尺寸

**(b)基因**

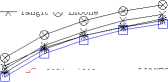
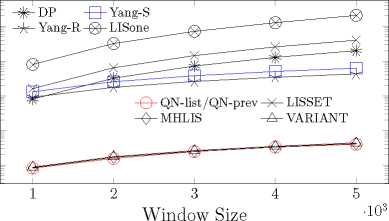
**图1 12：建筑**

103

6.3实验评价

­数据结构比较。 数据结构的评价侧重于空间、施工时间和更新时间。 由于我们的方法(QN-列表)比我们以前的方法(QN-Prev)的优化在于维护，空间成本和施工时间与以前版本的相同[17]。

每种方法的空间成本如图11所示。 由于每种方法的数据结构的空间成本只取决于序列（窗口）的大小，所以空间



IEEE知识和数据工程事务

在不同的数据集上，成本将是相同的，我们只在库存数据集上呈现空间成本。 我们可以看到，我们的方法比LISSET、DP、YangS/YangR和LISONE的内存成本低得多，而MHLIS和VARIANT的内存成本略高于MHLIS和VARIANT，这是由于QN-List中的额外成本，以支持具有约束的高效维护和计算LIS。 请注意，没有一种比较方法可以同时支持LIS枚举和具有约束的LIS；但是我们的QN列表可以以统一的方式支持所有这些LIS相关问题（表3）。

我们构造每个数据结构五次，并在图12中给出它们的平均构造时间。 同样，我们的方法比LISSET、DP和LISONE运行得快得多，因为我们的施工时间是线性的，但LISSET、DP和LISONE具有平方时间复杂度（见表2）。 我们的施工时间比VARIANT稍慢，但比MHLIS、杨S和杨R快，因为它们具有相同的施工时间复杂性（表2）。

数据结构更新性能。 此外，在图14中，我们给出了QN列表的保存项在QN-Prev上的比率来显式地度量优化（见定理4）。

s

第100皀。5

s

日100

QN清单 QN-prev

货物+I/D■货物+是

。目10°

503

1

2 3 4

窗户尺寸

s

8嶷

1

2 3 4

窗户尺寸

库存最大重量

*0)b J)U*

—

>

v101

QN清单 QN-prev

和0瓦里亚特+I/D目0瓦里亚特+是

1 Windjw Sizt-5*03*

**(b)基因的最大重量**

QN清单 QN-prev

瓦里安特+I/D®0瓦里安特+是的

|  |  |  |  |
| --- | --- | --- | --- |
|  | ‘-Q-QN-列表QN-Prev. |  | “QN-列表WQN-Prev |
| 105 | 角\_ \*消失€-消失 | 105  a | d 克里塞特忌-利索 |
| n  d  苛104 |  | m104 |  |
| p  会103 | 、  \*DP EYang S\*Yang R | |1o3 | 知7^DP每Yan知 |
| -^瓦里安特+R^p MnLLS，Re\* |  | \*\*■  兽医+R^^MHLIS+Re\* |
| 102 | 4种变异+I/domhlis+I/d | 102 | 4种变异+I/domhlis+I/d |
|  | 1 2 3 4 5 |  | 1 2 3 4 5 |
|  | 窗户尺寸 ,103 |  | 窗户尺寸 ,103 |
|  | **(a)库存** |  | **(b)基因** |

图1 13：维修

·1q5

503

**库存重量最小**

**图1 16：和LIS在一起**

2. 3 4

窗户尺寸

503

**基因的最小重量**

**太重了**

窗口大小1Q3

(a)库存

·1q5

1. 第1054如

1. 24..105

**65**

00

**43**

0

吕一> b 是°的

吕一> b 是°的

第四季度 m n° h

1. 我..5

1.29,

10

0.2273

0.2123

0.2023

唱4 0.1691

-Q-QN-列表-G-QN-Prev

**第四季度 m 没有JQ h**

**65**

*a°*

**43**

问：

窗口大小1Q3

**(b)基因**

**图1 14：QN-列表的优化**

020100

（巻）111Ee AV

QN清单

SB QN-PrevDqdp

目目lisset

102

101

100

11QN-列表SB QN-PrevDqdp

~~罪雷蚩毫書雷fialm，~~

1 2 3 4 5 s 1 2 3 4 5 s

窗户尺寸 ,10 窗户尺寸，10

**(a)库存 (b)基因**

**图1 15：LIS枚举**

­ 没有一个MHLIS，VARIANT，DP或YangS/YangR解决维护问题。 为了实现比较，我们实现了MHLIS和VARIANT的两个流版本。 首先是在每个时间窗口(MHLIS+Re，VARIANT+Re)中重建数据结构。 第二，将我们的更新思想应用于MHLIS和VARIANT(MHLIS+I/D，VARIANT+I/D)。 更新效率由吞吐量来衡量，即每秒处理的项数，而不回答任何查询。 图13显示，我们的方法明显快于比较方法

00 t-2

洛（南）①UI①

QN清单 HBQN-prev.

货物+信用证眼货物+是

窗户尺寸

**库存最大差距**

101

s

£

尊10。

h

”“没有

>

QN清单 QN-prev

莫利斯+I/D 厚生+重

申请人+身份证+是

s

皀

日100

中

b十)

2

中

敖\*

11QN-列表目目QN-Prev

可瓦里安特+我/^0瓦里安特+是的

1

2 3 4

窗户尺寸

503

基因的最大差距

11QN-列表 0BQN-prev.

DIMHLIS+I/D 恥MHLIS+Re

如变体+I/D目的变体+r

100

<101

1 2 3 4 5.3 1 2 3 4 5.3

窗户尺寸 ,10 窗户尺寸，10

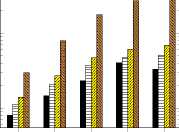
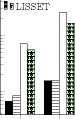
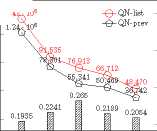
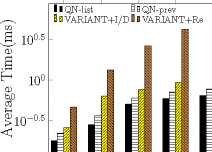
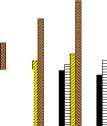
**(c)存量的最小差距(d)基因的最小差距**

**图1 17：LIS与极端间隙**

­ LIS枚举。 我们将我们的LIS枚举方法与LISS ET和DP进行了比较。 在滑动窗口模型下，LISS ET是以前唯一可以用来枚举LIS的工作。 LIS的数量相当庞大8 每个方法只返回每个窗口不超过10,000个LIS。 我们在图15中报告平均查询响应时间。 在数据流模型中，总体查询响应时间包括两部分，即数据结构更新时间和在线查询时间。 我们的方法比LISS ET和DP都快，随着时间窗口大小的增加，性能优势更加明显。

最大/最小重量的LIS。 瓦里安特[12]是以前唯一的最大/最小重量的LIS工作。 图16证实了我们的方法在VARIANT(VARIANT+Re和VARIANT+I/D)方面的优越性。  **具有Max/MinGap的LIS。**  瓦里安特[12]计算最大和最小间隙的LIS，而MHLIS[13]

8. 我们在附录E中讨论这个问题



IEEE知识和数据工程事务

只计算具有最小间隙的LIS。 不同方法的每个窗口的平均运行时间在.

优于[14]中的方法和基于LIS枚举的解决方案。

图17 我们可以看到，我们的方法优于其他方法

方法明显。

我的QN-列表即QN-Prev

10瓦里安特+身份证+是

100.5

8

l

10

*<*

100

102

101

100

s

皀

日100

中

b十)

2

a

<101

11QN-列表 目ElQN-Prev0□Lisone

血MHLIS+I/D口GHLIS+关于VARIANT+I/DMAVARIANT+关于杨S &国杨R

dddp

m1

*m*

~ 2 3 4

窗户尺寸

**库存的最大宽度**

1-0.5

吉

5o3

2 3 4

窗户尺寸

**(a)库存**

库存的最小宽度

图1 18：LIS具有极宽

基因的最大宽度

1

沃夫·西兹特 503

基因的最小宽度

s

邑

日100

h

中

b十)

冬

v101

*s*

g102

w10

我的QN-列表目日QN-Prev.□Lisone口口MHLIS+I/D口□MHLIS+关于VARIANT+I/D，VARIANT+R&I杨S

冃0阳ODDP  n 「

2 3 4

窗户尺寸

**(b)基因**

**图1 21：LIS长度**

5 。 io3

最大/最小宽度的LIS。 瓦里安特[12]是以前唯一在LIS上最大/最小宽度的工作。 图18证实了我们的方法在VARIANT(VARIANT+Re和VARIANT+I/D)方面的优越性。

(a)库存

J[io0

8

1 温多·W·西兹4 膈

(b)基因

LIS长度(输出LIS). 我们比较了我们的方法与李松[4]的输出LIS（长度直接出来）。 我们还将其他比较工作添加到比较中，它们可以很容易地支持输出LIS。 由于RLIS和SLIS中有用户定义的参数，我们为RLIS设置了足够大的相应范围，为SLIS设置了足够小的斜率，以保证LIS的输出。 图21显示，我们的方法比计算LIS长度和输出单个LIS的比较方法更有效。

图1 20：范围受限的LIS

­ 斜率/变化约束LIS[14]是以往关于斜率约束LIS(SLIS)和范围的唯一工作

约束LIS(RLIS)。 我们设置了三个不同的范围(RL=

*{l=1，u/=20，l v* =0，你 *v =50}，R2={l=20，u=*

40，Lv=50，Uv=100}，R3={信用证40，U/=60，L *v* =100，Uv=150})和三个不同的斜率(S1=0，S2=0。 第5条第3款=第1款。 0)来评估绩效。 我们使用三个范围/斜率下的平均运行时间进行比较。 此外，由于着色算法将引入额外的成本，我们还将我们的方法与LIS枚举期间验证斜率/范围约束的解决方案进行了比较(表示为QN-nocolor)。 从图19和图20中我们可以看到我们关于这两个问题的方法

7相关工作

7.1解决办法展望

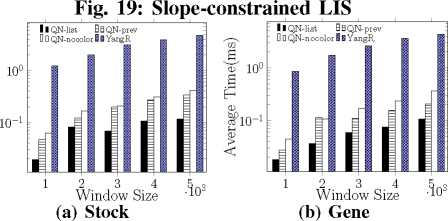
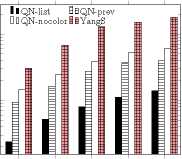
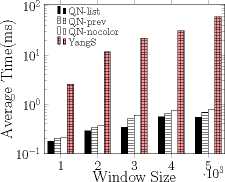
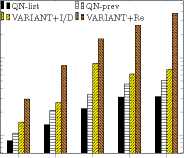
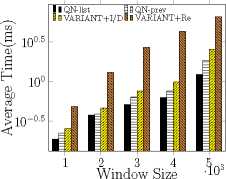
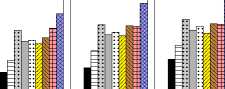
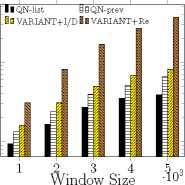
一般来说，现有的LIS计算方法可分为以下三类：

1. 基于动态编程的。 动态规划是计算LIS长度的一种经典方法。 给定一个序列a，假设ai表示由a的第一个i项组成的前缀序列，则基于动态编程的方法是在计算a、how的LIS后计算ai+1的LIS，但是基于动态编程的方法代价O(W)2 时间，其中w表示序列a的长度。 在[14]中，计算范围/斜率约束的解也是基于动态规划的，即ai+1的范围/斜率约束LIS是从基于D、动态规划方法的LIS中计算出来的，也可以很容易地扩展到枚举代价为O(W)的序列中的所有LIS­2 )空间。
2. 杨的桌子。 [21]提出了一种基于Young的基于tableau的解决方案，用于在O(wlogw)时间内计算LIS。 在一个序列a上建造的杨的第一排桌子的宽度正好是Albert等人的LIS的长度。 [4]按照Young的基于表格的工作计算滑动窗口中的LIS长度。 当窗口滑动时，他们保持了杨的第一行表，称为原则行。 对于窗口中的序列a，有n个=|一个|后缀子序列，[4]中的素数思想是压缩所有

1041-4347(c)2017年IEEE。 个人使用是允许的，但重新发布/重新分配需要IEEE的许可。 见http://www.ieee.org/publications\_standards/publications/rights/index.html

更多

信息。



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **方法** | **LIS枚举** | **最大重量的LIS** | **最小重量的LIS** | **最大间隙的LIS** | **与最小间隙的LIS** | **斯利斯** | **rlis** | **莉斯**  **长度** |  | **方法** | **空间**  **复杂性** | **时间的复杂性** | | |
| **建筑工程** | **插入** | **删除** |
| QN-列表/QN-prev | *没有（输出）* | *没有（输出）* | *没有（输出）* | *O(w+输出)* | *O(w+输出)* | *O(w)* | *O(w)* | o（1） | QN-列表/QN-prev | *O(w)* | *O(w日志w)* | O(logw) | *O(w)* |
| 利塞特[5] | *没有（输出）* | - | - | - | - | - | - | o（1） | 利塞特[5] | *O(w)*2) | *O(w)*2) | *O(w)* | *O(w)* |
| 莫利斯[13] | - | - | - | - | *O(w+输出)* | - | - | o（1） | 莫利斯[13] | *O(w)* | *O(w日志w)* | O(logw) | - |
| 变体[12] | - | *没有（输出）* | *没有（输出）* | *O(w+输出)* | *O(w+输出)* | - | - | o（1） | 变体[12] | *O(w)* | *O(w日志w)* | O(logw) | - |
| DP | *没有（输出）* | - | - | - | - | - | - | o（1） | DP | *O(w)*2) | *O(w)*2) | *O(w)* | - |
| 杨S[14] | - | - | - | - | - | *O(w)* | - | o（1） | 杨S[14] | *O(w)* | *O(w日志w)* | O(logw) | - |
| 杨R[14]. | - | - | - | - | - | - | *O(w)* | o（1） | 杨R[14]. | *O(w)* | *O(w日志w)* | O(logw) | - |
| 利松[4]. | - | - | - | - | - | - | - | o（1） | 利松[4]. | *O(w)*2) | *O(w)*2) | *O(w)* | *O(w)* |

IEEE知识和数据工程事务

表2：数据结构

**表1：在线查询的理论比较**

这些后缀子序列的原则行成数组，当更新发生时，可以及时更新。 此外，它们还可以输出具有树数据结构的LIS，其成本为O(W)2 )空间。

3. 基于分区。 还有一些工作通过在序列[5]、[12]、[15]、[13]中划分项目来计算LIS。 它们将项目分类为l分区：P\，P2...，PI，其中l是序列的LIS长度。 每个项目a在P中 *k* (k=1，l)，以a结尾的递增子序列的最大长度正好是k。因此，当构建分区时，我们可以从Pl中的项开始，然后扫描P-(1<k<l)中的项来构造LIS。 在不同的方法中，分区被称为不同的名称，如[12]、[15]中的贪婪覆盖，[5]中的反海因。 请注意，[12]和[13]在静态序列上进行分区，以有效地计算具有约束的LIS。 [15]使用基于分区的方法作为子程序，找出n个w窗口中最大的LIS长度，其中w是大小为n的序列a上滑动窗口的大小。他们的核心思想是避免在LIS长度小于先前发现的窗口上构造分区。 事实上，他们重新计算了每个窗口中没有从零开始过滤的贪婪覆盖层。 基于分区的解决方案都没有解决[5]预期的数据结构维护问题。 [5]是研究流模型中LIS枚举的唯一方法。 它们的插入和删除算法都花费了O(W)时间[5]。 此外，它们用O(W)指针分配每个项，因此它们的方法代价是O(W)2 )空间。

我们的方法属于基于分区的解决方案，其中每个水平列表（定义8）都是一个分区。 然而，我们的数据结构只为每次更新花费O(W)空间和O(W)时间，并且支持具有各种约束的LIS枚举和LIS。 下面第7.2节将讨论现有工作的更多理论分析和可行性比较。

1. 问题视角

我们简要地介绍了我们的问题在现有的工作LIS计算在计算任务和计算模型。 一，LIS计算任务有三类.. 首先是计算LIS的长度，并按顺序a[4]、[15]、[22]、[23]、[21]输出单个LIS（而不是枚举所有）。 第二个是LIS枚举，它在一个序列a[24]，[5]中查找所有LIS。 [24]只计算要求为{1，2，...，n}排列的序列而不是一般序列（如运行示例中的{3，9，6，2，8，5，7}）的LIS枚举。 最后一个计算任务研究具有约束的LIS，如GAP、权重[12]、[13]。 此外，每个LIS计算任务都有两个计算模型。 一种是静态模型，假设序列a没有变化[12]、[21]、[25]、[13]、[14]。 另一个模型是最近一些工作[4]、[5]中考虑的数据流模型。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **方法** | **小溪**  **模型** | **­ LIS枚举** | **有着极大的重量** | **与极端差距的LIS** | **斯利斯** | **rlis** | **长度** |
| QN-列表/QN-prev | **/** | **/** | **/** | **/** | **/** | **/** | **/** |
| 利塞特 | **/** | **/** | **x** | **x** | **x** | **x** | **/** |
| 莫利斯 | **x** | **x** | **x** | **/** | **x** | **x** | **/** |
| 变体 | **x** | **x** | **/** | **/** | **x** | **x** | **/** |
| DP | **x** | **/** | **x** | **x** | **x** | **x** | **/** |
| 杨S | **x** | **x** | **x** | **x** | **/** | **x** | **/** |
| 杨R | **x** | **x** | **x** | **x** | **x** | **/** | **/** |
| 利索 | **/** | **x** | **x** | **x** | **x** | **x** | **/** |

表3：支持计算任务的Compaison

表3说明了考虑计算任务和计算模型的现有工作。 我们可以看到，对于所有与LIS相关的问题，如LIS长度、LIS枚举和具有约束的LIS，没有现有的统一解决方案。 此外，没有算法支持在流环境中使用约束计算LIS。 因此，我们工作的主要贡献在于，我们提出了一个统一的解决方案（相同的数据结构和计算框架），在流环境中所有与LIS相关的问题。

我们还提出了现有工作的理论比较（比较工作的更多细节可在6.2节）对这些LIS计算任务。

­ 数据结构比较。 我们比较了我们的数据结构的空间、施工时间和更新时间与表2中其他工程的数据结构（时间复杂性是基于最坏的情况分析）。 我们可以看到，我们的方法比任何度量的任何比较工作都好或不差。 我们的数据结构在空间和时间复杂度上都优于LISS ET。 此外，我们的方法中的插入时间O(logw)也优于LISS ET中的时间复杂度O(W)。 此外，MHLIS、VARIANT、DP或YangS/YangR都没有解决数据结构更新问题。 因此，它们需要O(^w日志w)(O(w)2 )在每个时间窗口重新构建数据结构的时间。 显然，我们的比他们的好。

在线查询比较。 表1显示了不同方法的在线查询时间复杂性。 数据流模型中的在线查询响应时间由在线查询时间和更新时间组成.. 我们可以看到，我们的在线查询时间复杂性与比较相同

IEEE知识和数据工程事务



一些。 然而，我们的方法中的数据结构更新时间复杂度优于其他方法。 因此，从理论角度看，我们的总体查询响应时间优于比较时间。

8结论

­ 在本文中，我们提出了一种统一的数据结构来支持在顺序数据流上枚举具有特定约束的所有LIS和LIS。 我们提出的数据结构只需要线性空间，可以在线性时间内更新，这使得我们的方法在处理高速顺序数据流方面具有实用性。 据我们所知，我们的工作是第一个提出统一解决方案（相同的数据结构和计算框架），以解决数据流场景中所有与LIS相关的问题.. 我们的方法不仅在理论上，而且在经验上都优于最先进的工作..

参考资料

1. C.Faloutsos、M.Ranganat han和Y.Manolopoulos，SIGMOD中的“时间序列数据库中的快速子序列匹配”。­ 1994年，pp。 419—429.
2. *X.Lian，L.Chen和J.X.Yu，“隐形时间序列上的模式匹配”，载于第24届国际数据工程会议记录，ICDE2008，2008年4月7日至12日，墨西哥坎昆，2008年，第页。 1462-1464.*
3. 廖伟T.，“时间序列数据的聚类一项调查”，模式识别，第二卷。 38，不。 第11页 1857-1874,2005.
4. M.H.Albert，A.Golynski，A.M.Hamel，A.Lopez-Ortiz，S.Rao和M.A.Safari，“滑动窗口中最长的增加子序列”，理论计算机科学，第二卷。 321.不。 2-3页 405-414，2004年8月。 [在线]。 可查阅：http:/linkinghub.elsevier.com/retrieve/pii/S0304397504002142
5. 陈E.，杨立阳和袁海元，“基于典型反海因分区的窗口中最长的增加子序列”，“理论计算机科学”，第二卷。 378，不。 3，第3页。 223-236，6月。 2007. [在线]。 可用：http://linkinghub.elsevier.com/retrieve/pi/S0304397507001235
6. P.Gopalan，T.Jayram，R.Krauthgamer和R.Kumar，“估计数据流的排序”，第十八届年度ACM-SIAM离散算法研讨会论文集，第327页，2007年。 [在线]。 现有：http://portal.acm.org/citation.cfm？id=1283417
7. R.，S.McCallen和E.Almaas，“趋势主题：分析动态复杂网络的图形挖掘方法”，在IEEE第七届国际数据挖掘会议上。 IEEE，2007年，pp。 541-546.
8. L.Bonomi和熊L.，“关于数据流中差异私有最长增子序列计算”。 数据隐私交易，第二卷。 9，不。 1，第1页 73-100,2016.­
9. 张H.，“基于最长增子序列算法的BLAST高分段对对齐”，生物信息学，第二卷。 19，不。 第11页 1391-1396,2003.
10. W.M.E.Altschul，Stephen；Gish和D.Lipman，“基本局部对准搜索工具”，“分子生物学杂志”，第二卷。 215，不。 3，第3页。 403-410,1990.
11. N.Jain、S.Mishra、A.Srinivasan、J.Gehrke、J.Widom、H.Balakrishnan、U.Cetintemel、M.Cherniack、R.Tibbetts和S.Zdonik，“实现流式SQL标准”。 《Vldb捐赠学报》，第二卷。 1，不。 2，第2页。 1379-1390,2008.*­*
12. S.多罗维奇，“关于最长的一些变体

增加次序列问题，理论和应用信息学，第二卷。 21，不。 3，第3页。 135-148,2009. [在线]。 现有： <http://projekty.iitis.gliwice.pl/uploads/File/taai/3X_2009/>

2009\\_3\\_4\\_art01\\_deorowicz.pd f

1. 曾C.，C.。 杨和海。 安，“最小高度和序列限制最长增加次序列”，“互联网技术杂志”，第二卷。 10页 173-178,2009.
2. I.-H.Yang和Y.C.Chen，“约束最长增长子序列问题的快速算法”，载于第25期组合数学和计算理论讲习班论文集，2008年，pp。 226-231.*­*
3. S.Deorowicz，“一种基于覆盖合并的算法，用于滑动窗口问题中最长增加子序列，”计算和信息学，第二卷。 31，不。 第6页 1217-1233,2013.
4. GitHub.com/Vito幻想/lis\_stream/。
5. 李Y.，邹国强，张海华，赵国强，“计算序列数据流上最长增加的子序列”，VLDB捐赠学报，第二卷。 10，不。 3，第3页。 181-192,2016.
6. E.Keogh，J.Lin，S.H.Lee和H.VanHerle，“寻找最不寻常的时间序列子序列：算法和应用”，知识和信息系统，第二卷。 11，不。 1，第1页 1-27,2007.
7. ­ *E.J.Keogh和M.J.Pazzani，《科学和统计数据库管理》，1999年。 第十一届国际会议。*  IEEE，1999年，pp。 56-67.
8. T.Shibuya和I.Kurochkin，“cDNA映射的匹配链算法”，在生物信息学算法国际讲习班上。 斯普林格，2003年，pp。 462-475.*­*
9. G.Rabson，T.Curtz，I.Schensted，E.Graves和P.Brock，“最长的增加和减少子序列，”Canad。 j. 数学，第二卷。 第13页 179-191,1961.
10. M.L.Fredman，“关于最长增加子序列长度的计算”，离散数学，1975年。
11. D.Liben-Nowell，E.Vee和A.朱，“在流数据中找到最长的增加和公共子序列”，在“组合优化杂志”，第二卷。 11，不。 2006年，第2页。 155-175.*­*
12. S.Bespamyatnikh和M.Segal，“计数最长的增加子序列和耐心排序”，信息处理信函，第二卷。 76.不。 第1-2页 7-11,2000. [在线]。 可查阅：http:/www.sciencedirect.com/science/article/pii/S0020019000001241
13. G.D.B.罗宾逊，“关于对称群的表示”，“美国数学杂志”，pp。 745-760,1938.

­­ 李友环于2013年获得北京大学电力工程与计算机科学学士学位。 他目前正在攻读博士学位。 北京大学计算机科学与技术学院D.学位，专注于流式数据管理。­

­ 邹磊获得学士学位和博士学位。 华中科技大学计算机科学D.学位分别于2003年和2009年。 现在，他是北京大学计算机科学与技术研究所的副教授。 他的研究兴趣包括图形数据库和语义数据管理。­­

­ 张华明于1992年获得安徽师范大学数学学士学位。 他获得了硕士学位和博士学位。 分别于2002年和2005年在布法罗纽约州立大学获得计算机科学和工程D.学位。 他现在是亨茨维尔阿拉巴马大学计算机科学系副教授。 他的研究兴趣包括算法设计和图论。

­ 赵东燕获得学士学位、硕士学位和博士学位。 分别于1991年、1994年和2000年获得北京大学D.学位。 现在，他是北京大学计算机科学与技术研究所的教授。 他的研究兴趣是信息处理和知识管理。­­

1. 本文不要求增加子序列是严格单调递增的，a中的所有项也可以是任意数值。 [↑](#footnote-ref-2)