# 海量信息的极大点查询算法优化及应用研究

The Optimization and Application of the Maxima-finding Algorithm for Massive Information

贵向泉/著

兰州大学出版社

# 海量信息的极大点查询算法优化及应用研究

The Optimization and Application of the Maxima-finding Algorithm for Massive Information

贵向泉/著

兰州大学出版社

**图书在版编目(CIP)数据**

海量信息的极大点查询算法优化及应用研究 / 贵向泉著. —兰州:兰州大学出版社,2013.12
ISBN 978-7-311-04376-6

Ⅰ.①海… Ⅱ.①贵… Ⅲ.①计算机应用—情报检索—研究 Ⅳ.①G252.7

中国版本图书馆 CIP 数据核字(2014)第 013700 号

策划编辑 陈红升
责任编辑 郝可伟
封面设计 刘 杰

书 名 海量信息的极大点查询算法优化及应用研究
作 者 贵向泉 著
出版发行 兰州大学出版社 (地址:兰州市天水南路 222 号 730000)
电 话 0931-8912613(总编办公室) 0931-8617156(营销中心)
0931-8914298(读者服务部)
网 址 http://www.onbook.com.cn
电子信箱 press@lzu.edu.cn
印 刷 兰州瑞昌印务有限责任公司
开 本 787 mm×1092 mm 1/16
印 张 5.75
字 数 124 千
版 次 2014 年 1 月第 1 版
印 次 2014 年 1 月第 1 次印刷
书 号 ISBN 978-7-311-04376-6
定 价 12.00 元

# 前 言

近年来，海量的信息已经变得无处不在，面对庞大的数据量，如何有效、及时地对数据进行处理是目前面临的一个重要挑战，否则很多数据就会失去它应有的价值。幸运的是极大点查询就是一个有效的数据处理工具，可以帮助人们快速地从大量的信息中找到感兴趣的内容。极大点查询产生于计算几何，由于其良好的应用前景，其在多目标优化、数据库等领域出现了越来越多的各种应用。但面临的问题是这些应用常常需要巨大的计算资源，使得传统的极大点查询算法很难有效地使用。针对这个问题，本书主要对极大点查询算法的性能优化和应用方面的研究成果进行了介绍。

本书第1章和第2章首先对目前极大点查询算法的研究现状进行了综述性介绍。从内存储算法和外存储算法两个方面，全面介绍和分析了已有的极大点查询算法。在此基础上介绍了一种基于体积优先策略的极大点查询算法和一种线性I/O次数的极大点查询外存储算法以及这两种算法在数据库的Skyline查询处理和金融领域的资产配置问题中的应用。

第3章主要介绍了一种基于体积优先策略的极大点查询算法。通过试验发现算法的效率高于Bently等人提出的经典的MTF算法。通过理论证明，算法的时间复杂度以极高的概率保持线性时间。部分解决了没有线性时间复杂度的极大点查询算法这一公开问题。其高维空间上的理论证明对极大点查询算法在高维空间上的复杂性分析提供了有益的参考。

第4章主要介绍了外存储极大点查询算法的优化问题。当数据超过内存的限制时，经典的算法往往无法很好地处理这些数据，近年来算法研究已经关注于处理海量数据的外存储算法。这时算法的效率通常以内外存之间数据输入输出次数(I/O次数)来衡量。对于极大点查询问题，目前还没有线性I/O次数的外存储算法。对此，本书介绍了一种针对海量数据的线性I/O次数的极大点查询算法。算法针对独立同分布数据，利用了极大点的数目概率特性。并通过实验和理论证明了算法的有效性。

第5章主要介绍了极大点查询算法在数据库查询中的应用。数据库Skyline查询处理是近年来数据库查询领域的研究热点，极大点问题与Skyline查询有着天然的联系。本章介绍了极大点查询算法在数据库的Skyline查询处理中的应用，通过实验反映出整个算法的内存时间复杂度和I/O复杂度均达到了线性时间。

第6章主要介绍了金融网络中社团结构研究和极大点的应用。金融领域存在海量的数据信息，如何从这些数据中找到有价值的信息指导投资决策是现代投资理论的重要问题之一。本书在尝试利用极大点查询指导投资决策的过程中，在研究资产类别的历史表现时发现了一些有趣的经济现象，并首次系统地提出了一种基于复杂网络社团结构的股票分类方法和利用极大点查询进行投资组合的新的策略。

本书可作为计算机专业研究生的参考读物。对极大点查询问题、外存储算法、复杂网络在金融领域的应用等问题感兴趣的研究人员也可以从本书中得到有意义的参考。

# 目 录

# 第1章 绪 论

## 1.1 研究背景

随着物联网、云计算、移动物联等技术的迅猛发展，无所不在的移动设备、无线传感器每分每秒都在产生数据，数以亿计用户的互联网服务时时刻刻在产生大量的交互数据，随之产生了海量的数据信息。例如沃尔玛公司的客户交易数据库包含了超过2.5 PB(1 PB=$2^{50}$ B)的数据，用来分析指导其战略决策[1]；谷歌搜索引擎通过存储和索引超过460亿的网页来提供快速的文本搜索服务[2]；AT&T每天存储几十亿的电话通讯记录[3]；微软的必应地图(Microsoft Bing Maps)、谷歌地球(Google Earth)等在线地理信息系统存储了以TB(1 TB=$2^{40}$ B)为单位的卫星图片、街道地图和地形数据[4,5]；生物技术的发展使得一些研究中心生成了TB级的基因组序列数据[6]。“我们被信息淹没，但是却渴求知识[7]”，这些海量的数据为科学研究或者商业决策提供了大量的信息，同时使得如何快速、高效地从这些数据中找到有价值的内容成为研究者面临的一个重要课题。极大点查询就是一种有效的数据处理工具，可以帮助我们在海量的数据中找到有价值的信息。

极大点的概念最初产生于计算几何，其具体是指在一个点集中不被其他点支配的所有点的集合。这里的支配是指一个点的各维坐标都大于等于另一个点的相应维坐标。通过将数据点的属性和支配关系进行扩展，极大点查询可以在各领域中有广泛的应用。例如足球教练在挑选球员时，可以根据球员的体能、加速度、身高、耐力等各项其关心的指标构造数据点集，这时数据点的各维坐标就对应球员的各项指标。根据教练的偏好定义支配关系，就可以在大量的信息中快速地缩小搜索范围帮助决策。以图1.1为例，假设一个足球教练希望找一名体能和控球技术都较为出色的中场球员，这时以球员的体能和控球技术作为点的横坐标和纵坐标构造数据点集，以数值大定义支配关系。从图中可以看出球员$S_1$和$S_2$不被其他点支配，而球员$S_3$至$S_9$都会被点集中的某个点支配，$S_1$和$S_2$就构成了这个点集的极大点集。当教练在选择球员时，只需要从这两个球员中做挑选即可，因为球员$S_1$和$S_2$在各方面均优于其他球员，而他们两者之间各有所长。从这个例子可以看出，利用极大点查询可以快速地缩

小信息的搜索范围，找到人们最关心的有价值的信息。

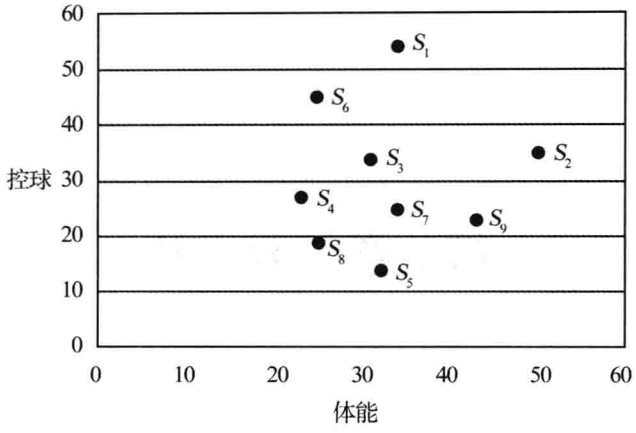


**图 1.1　球员极大点查询示例**

极大点查询在 Pareto 最优、多目标优化、决策支持系统等众多领域均有着广泛的应用。例如 Bentley、Schkolnick、Thompson 等学者在文献[8]中利用极大点集来决定动态规划算法的运行时间。Preparate 和 Shamos 在文献[9]中利用极大点集去解决经济学中的“浮动汇率问题(Floating Currency Problem)”等。特别是 Börzsönyi 等学者在 2001 年数据工程国际会议(ICDE)上将极大点查询的概念引入数据库领域，并命名为 Skyline 查询[10]，使得对极大点查询的研究再一次成为近几年的研究热点。Algorithmica、ACM Transactions on Database Systems (TODS)、VLDB 等众多高水平杂志和会议上涌现了许多高质量的相关论文，呈现了大量的研究成果。

极大点查询作为一个帮助我们发现有价值信息的工具有着良好的应用前景。但针对海量数据信息，查询的速度和效率是影响其应用至关重要的因素。每一个用户都希望尽可能快地得到查询结果。但要提高查询的运行效率还面临着下面两个挑战：

(1)如何进一步降低极大点查询算法的算法复杂度，来应对不断增长的数据。

在许多应用领域数据集的规模增加越来越迅速，而计算能力的增长相对缓慢。加速计算最简单和直接的方法就是利用更强处理能力的计算设备。虽然目前一些大型集群计算环境的计算速度相当可观，但并不是所有的应用环境都有条件和有必要去支持大型的计算设备。所以设计高效的数据结构和算法来加速计算是对这一问题的有效解决手段，也是算法设计人员一直在不断追求的目标。算法复杂度是用来衡量算法运行效率的参数，所以如何设计和改进极大点查询算法，进一步降低算法的复杂度是极大点查询研究人员面临的第一个挑战。

(2)设计高效的极大点查询外存储算法，来应对海量的信息数据。

对于海量的信息数据，仅降低算法的复杂度来加速计算是不够的。由于海量的信息数据通常无法完全放入计算机的内存中，这迫使程序运行时要反复地从外存设备中读写数据。外存设备的读写速度要明显地慢于内存设备，另外，不恰当的外存读写策略会使算法停滞不前，甚至无法运行。所以要加速极大点查询计算还需要设计有效的外存储算法来解决

海量信息极大点查询时的这一瓶颈问题。

本书以这两个问题作为出发点来展开研究工作，首先对极大点查询算法进行了优化，降低了算法的复杂度；其次设计了一个高效的极大点查询外存储算法。

极大点查询虽然是计算几何的一个基本问题，但其在决策支持系统等众多领域具有典型的应用。决策支持系统基于数据库中的数据进行决策分析，要得到有效的分析结果，这些数据库中的记录数以亿计，并且需要频繁地执行数据分析和OLAP(联机分析处理)查询。例如，一个国际销售商(比如沃尔玛)的数据库应该包含全年其所有商店的交易记录。一个很有意义的查询就是数据库Skyline查询。比如需要查询全球哪个商店具有最小的支出和最大的收益。尽管查询的输出很可能仅是很少量的商店，但查询却需要在数据库PB级的海量数据上进行计算。本书第二个方面的工作就是研究了极大点查询在决策支持系统的两个应用：一个是与极大点有着天然联系的数据库Skyline查询；另一个是基于金融数据的股票投资组合问题。

## 1.2 本书的核心内容

为了提高海量数据的查询处理速度，本书在研究了已有的极大点查询算法的基础上，提出了一种采用体积优先策略的极大点查询算法，降低了极大点查询算法的时间复杂度，通过试验和理论证明了算法的时间复杂度是线性。针对极大点查询外存储算法，本书提出了一种线性I/O复杂度的极大点查询算法。结合上述两种算法，将极大点查询问题应用到数据库查询领域，提出了一种内外存时间复杂度均是线性的数据库Skyline查询算法。由于金融领域同样存在海量的数据，通过极大点查询可以得到用户关心的有效信息。将极大点查询应用到金融领域的资产配置问题，通过实证测试发现利用极大点查询进行组合投资是一种可行的投资策略。

本书的核心内容主要包括以下4个方面：

(1)目前极大点查询算法已有很多研究成果，Bently等人在文献[1]中提出的MTF算法是目前高效且易于实现的经典的极大点查询算法。本书在MTF算法的基础上，对启发式策略进行了优化，构造了一种基于体积优先策略的极大点查询算法。通过试验发现算法的效率高于MTF算法。通过理论证明，算法的时间复杂度以极高的概率保持线性时间。部分解决了目前还没有线性时间复杂度的极大点查询算法这一公开问题。由于极大点查询算法在高维空间上分析和证明的难度较大，本书在高维空间上的理论证明结果对这一领域的研究有很好的参考价值。

(2)当数据超过内存的限制时，经典的算法往往无法很好地处理这些数据，近年来算法研究已经关注于处理海量数据的外存储算法。这时算法的效率通常以内外存之间数据输入输出的次数(I/O次数)来衡量。对于极大点查询问题，目前还没有线性I/O次数的外存储算法。对此，本书提出了一种线性I/O次数的极大点查询算法。算法针对独立同分布数据，利用了极大点的数目概率特性。并通过实验和理论证明了算法的有效性。

(3)由于Skyline查询在数据库、推荐应用、决策支持等领域良好的应用前景，Skyline查

询处理已成为近年来数据库查询领域的研究热点。数据库的数据量日益庞大,查询的速度和效率至关重要。本书将设计的极大点查询算法应用到数据库的Skyline查询处理中,设计了一种高效、快速的Skyline查询算法,使得查询算法的时间复杂度和I/O复杂度均达到了线性时间。

(4)金融领域包含海量的数据,如何从这些数据中发现有价值的信息,使我们对金融领域有更深入的了解,并指导我们对市场的分析和投资决策是现代投资理论的重要问题之一。本书基于股票市场的数据进行建模分析,实证测试发现利用极大点查询进行资产配置是一种可行的投资组合策略。在对市场数据进行分类时,首次系统地提出了一种基于复杂网络社团结构的股票分类方法,并通过社团结构发现了一些新的金融市场的统计特性。这部分工作对进一步利用极大点查询在金融领域进行决策支持提供了有益的探索。

## 1.3 本书的结构

本书共分6章,结构及各章主要内容如下:

第1章通过一个挑选足球运动员的示例介绍了极大点查询的研究背景和针对海量信息面临的挑战。针对面临的挑战简要介绍了本书的核心内容。最后给出了本书的结构。

第2章详细介绍了极大点查询的定义,从内存储算法和外存储算法两个方面分析了目前极大点查询算法的研究现状,并对外存储算法的概念和特点进行了介绍。

第3章在经典MTF算法的基础上,对算法的启发式策略进行了优化后提出了一种体积优先策略的极大点查询算法。随后介绍了算法的试验结果。最后通过对算法复杂度详细的理论证明,证明了该算法以极大的概率具有线性的时间复杂度,部分解决了没有线性时间极大点查询算法这一公开问题。

第4章结合极大点数目的概率特性,设计了具有线性I/O次数的外存储极大点查询算法。对算法的可靠性进行了试验和分析证明。

第5章首先对数据库Skyline查询的概念和已有的算法进行了介绍,随后将设计的内存储和外存储极大点查询算法进行结合,应用在数据库Skyline查询处理中。使得算法的内存复杂度和外存I/O次数都达到了线性。

第6章首先简单介绍了复杂网络的基本概念。然后系统地介绍了利用复杂网络对金融领域的股票市场进行建模和发现股票社团的方法。对发现的社团结构的一些特性和规律进行了总结。最后应用极大点查询进行了资产配置的实证研究,发现利用极大点进行投资组合是一种可行的策略。

结论中总结了本书的研究工作,并对今后的研究内容和方向进行了阐述。

# 第2章 极大点查询算法研究综述

## 2.1 极大点查询的定义

极大点的概念最初产生于计算几何，其在一个点集的边界上存在重要特性，并作为计算几何中许多问题的基础，是计算几何的一个重要基本概念。其具体是指：在某个 $k$ 维点集中，如果某点 $P=(p_1,p_2,\cdots,p_k)$ 和某点 $Q=(q_1,q_2,\cdots,q_k)$，对于所有的 $i\leqslant k$，存在 $p_i\geqslant q_i$，则称作点 $P$ 支配(dominate)点 $Q$。一个点集 $P_1,P_2,\cdots,P_N$ 中如果没有点 $P_j$ $(P_j\neq P_i)$ 支配点 $P_i$，则点 $P_i$ 称作极大点(maximal point)。所有极大点的集合就称作这个点集的极大点集(maxima)[11]。极大点查询(maximal-finding)问题就是如何快速有效地在一个集合中找到需要的极大点。

如图2.1是一个二维极大点的示例，图中 $P$ 点 $X$ 坐标、$Y$ 坐标都大于点 $Q$ 的相应维坐标，所以点 $P$ 支配点 $Q$。同时 $P$、$P_1$、$P_2$、$P_3$ 都不能被点集中的其他点支配，所以 $P$、$P_1$、$P_2$、$P_3$ 就组成了这个二维点集的极大点集。

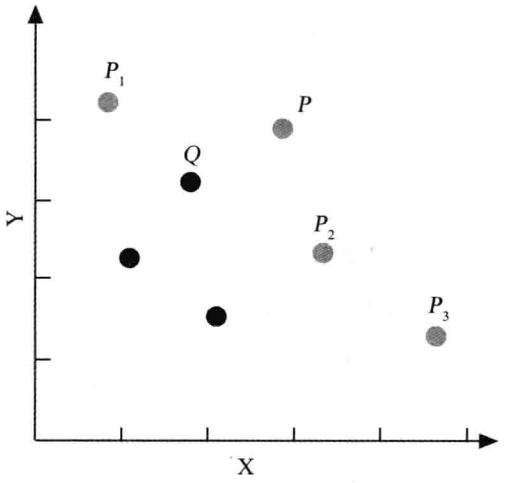


图2.1 一个二维极大点示例

作为计算几何的一个基本问题，极大点查询问题在计算几何的很多问题中占有核心地位或者作为重要组成部分，例如凸包、区域最小化、最近点对、Voronoi图等[9,12,13,14]。由于其本质上是一个多目标优化问题，将点的属性或者支配的概念进行扩展，就可以使极大点在不同领域有着广泛的应用，例如决策支持系统[15,16,17,18]、遗传算法[19,20]、网络[21,22,23,24]、数据挖掘[25,26]、

图着色[27,28]、基因序列分析[29,30]等。对极大点的更多应用可以参阅C. A. Coello维护的网页"List of References on Evolutionary Multiobjective Optimization"[31]和V. Mousseau维护的网页"Multiple Criteria Decision Aid Bibliography"[32]。本书首先关注的是极大点查询算法在复杂度上的优化,所以我们首先从内存储算法和外存储算法两个方面对极大点查询算法的研究成果进行分析和介绍。

## 2.2 极大点查询内存储算法

极大点查询算法的研究始于1966年Barndorff-Nielsen等人对极大点数目的研究[33]。目前对极大点查询的研究已有大量的成果,存在相当多的算法来寻找集合中的极大点。根据算法在1维情况下的模式(退化为排序算法)可以将极大点查询算法分为表2.1中的类别。

表2.1 极大点查询算法分类

| 1维情况(排序算法) | 大于1维情况(极大点查询算法) |
|---|---|
| 连续(Sequential)算法 | 连续(Sequential)算法[11,34] |
| 分治(Divide-and-conquer)算法 | 分治(Divide-and-conquer)算法[34,35,36] |
| 桶(Bucket-select)算法 | 桶(Bucket)算法[37,38] |
| 快速查找(Quick select)算法 | 挑选(Selection)算法[39] |
| 筛(Sieve)算法 | 筛(Sieve)算法[11] |

根据表2.1中算法的处理分类模式,下面对各类算法进行简单的论述。

连续类算法是最直接的极大点查询算法,不需要数据的预处理过程,算法易于实现。以下是该类算法的伪代码:

```
Algorithm SEQUENTIAL( )
//input={a1,a2, ···, aN}//
// aj∈R^d, j=1,···,N//
M:={a1}
for i:=2 to N
if ai支配M中的一些点
then删除这些点;
M:=M∪{ai}
else if ∀b∈M ,ai和b不可比较 then
    M:=M∪{ai}
//output M是集合{a1,a2,···,aN}的极大点集 //
```

算法中点$p$和点$q$不可比较(incomparable)是指点$p$和点$q$互相无法支配。很容易看出算法最坏情况下的复杂度是$O(N^2)$。算法中$M$用来存放已搜索到的极大点,对于$M$的数据结构,Kung等人利用平衡搜索树去存储[34]。Bently等人则利用链表[11],并且当新检测点被支配时,将支配其的极大点移动到链表的开头,从而提高算法的效率,这就是经典的MTF(Move-To-Front)算法。实验发现算法的速度明显快于其他算法,特别是针对独立同分布数据,算

法平均时间复杂度达到了线性。随后Golin在文献[34]中证明了对于二维的独立同分布数据，算法以极大的概率保持线性，对于高于二维的情况目前还没有理论成果。

分治算法是一种常用的算法设计方法。其在极大点查询算法中也有很好的应用，算法伪代码如下：

```
Algorithm DIVIDE-AND-CONQUER
//input: P:={a_1,…,a_n},a_j∈R, j=1,…,N//
maxima{a_i,…,a_j} (j>=i)
    if i=j then return a_i
m_1:=maxima{a_1,…,a_⌊j/2⌋}
m_2:=maxima{a_⌊j/2⌋+1,…,a_n}
return merge(m_1,m_2)
//output: maxima{a_i,…,a_N} //
```

算法递归地将数据分成两部分，分别计算其上的极大点集，直到每一部分足够小到能够快速地计算出极大点集(例如仅有一个点)，再将这两部分进行归并。具体的分治和归并规则有很多，典型的算法是文献[34]中，Kung等人提出的双向分治算法DD&C (double divide and conquer)。DD&C算法在数据数$N$和维度数$k$上进行双向分治。首先，算法对数据点在某一维上进行排序，例如在维度$d_{k-1}$上排序。然后将排序后的数据进行递归地二分，直到结果集合小于一个界限(比如仅剩一个点)。在递归树的最底层，每一个集合(单个点)都是这个集合的极大点集。随后这些集合自下而上进行归并。在归并的过程中要删除非合并集合的极大点的数据，假设集合$A$和集合$B$进行归并，并假设集合$A$在维度$d_{k-1}$上大于集合$B$（由于数据是在维度$d_{k-1}$上进行过排序的)，这样只需要在维度$d_0,\cdots,d_{k-1}$上递归地调用DD&C算法即可。通过这样的方式达到了在维度和数据上双向分治的目的。Kung等人证明了算法的复杂度在维度数$d\leqslant 3$的情况下是$O(N\log N)$,在维度数$d>3$的情况下是$O(N\log^{d-2}N)$。

桶算法，当输入点集满足某种假设条件后可以利用桶选择算法提高查询速度，例如假设各维度上数据是由一个小范围的整数组成的[37,38]。桶算法针对某些特殊的数据集，算法复杂度可以达到线性时间以下，但不同于连续算法和分治算法，其对输入点集的属性存在很大的敏感性，适应面很有限。

挑选算法，2维数据的算法伪代码如下：

```
Algorithm SELECTION( )
//input: P:={a_1,…,a_N},a_j∈R^2, j=1,…,N//
selection(P)
if |P|=1 then return P
在P中选择一个极大点x作为筛点
删除被x支配的点
M:={x}
   for 每个非空x的象限Q do
```

$M$:=$M$∪selection($Q$)

return $M$

//output $M$:=maxima{$a_1$,$a_2$,⋯,$a_N$} //

二维数据挑选算法的预期复杂度是线性的[42]。但对于二维以上的点集需要类似分治算法中的额外的归并过程,这使得该类算法的复杂度与分治算法相同。Clarkson在文献[39]中讨论了另外一种顺序选择算法。通过比较每一个不可比(incomparable)点$p$来构造极大点集。如果$p$点被支配,则删除$p$,否则$p$仍然属于不可比集合。将其与剩余点进行比较来查询其他的极大点并在$p$被支配的时候及时地将其删除。其过程类似于选择排序。Clarkson的算法可以针对任意维的数据集,但当极大点数目比较多的时候算法效率非常低。

筛算法、挑选算法对寻找第一个好的筛点有一些作用。但是当我们知道更多的输入数据信息时(除维数以外的,比如输入集的分布),可以在不花费额外代价的情况下,人为地构造一个具有强大能力的筛点,使得大部分的非极大点能够很快地被删除掉。算法伪代码如下:

Algorithm SIEVE

//input: $P = \{a_1,\cdots,a_N\}$, $a_j \in R^d$, $j=1,\cdots,N$ //

$M := \emptyset$

选择一个筛点$x$ //$x \notin P$//

for $i$ := 1 to $N$

if $a_i$不被$x$支配 then $M := M \cup \{a_i\}$

if 所有极大点都在$M$中 then

　　在其中查找极大点

else

　　利用其他的极大点查询算法

//output maxima{ $a_1,\cdots,a_N$ }//

例如如果输入数据是(0,1)上的均匀分布,可以选择筛点$x$的各维坐标为$1-(\log N)/N$,这时筛点最后不被支配掉的概率为$1/N$。如果发生筛点没有被支配掉的情况则只能选择其他算法寻找极大点集。一个好的筛点可以快速地支配掉大部分的非极大点以加速计算,使得算法复杂度接近线性,但由于筛点是人为构造的(并非实际数据点),使得算法存在很大的不确定性。

降低算法的时间复杂度一直是算法设计与分析领域的一个主要目标,同时也对算法实际的应用有显著的推动作用。虽然目前有很多极大点查询算法,但构造可以普遍应用的线性时间的极大点查询算法还是一个未解决的公开问题[11]。

本节介绍的算法均是针对输入数据可装入内存的情况。随着信息化的发展,数据量日益庞大,当数据量太大以至无法装入计算机内存时,现代操作系统就会借助虚拟存储技术将数据通过页面的方式在内外存之间进行置换。单纯借助操作系统管理的页面置换往往会使算法性能严重下降,系统I/O所花费的时间将成为制约算法效率的瓶颈。在算法的层面去控

制数据页面的调入调出，而不是单纯地借助操作系统，是解决这种瓶颈问题有效的手段，随之产生了外存储算法这一概念。下一节首先对外存储算法进行简单的介绍。

## 2.3 外存储算法简介

计算模型是对实际的计算机工作方式进行抽象，以便为我们设计和分析算法提供一个基础。目前算法设计的计算模型是RAM(Random Access Machine)模型。RAM模型假设计算机拥有无限大的物理内存，并且任何内存地址的访问都是一个常数的时间。这种模型为研究很多问题的计算复杂性提供了一个有效的工具，但是其不能反映算法在现代计算机多层次存储结构上执行的实际表现。

因为出于成本效率的考虑，现代计算机的存储系统通常由一系列层级存储器组成，不同层级的存储器其自身造价和性能都不相同。一般来说，计算机系统中的存储层次可分为高速缓冲存储器、主存储器、辅助存储器三级。高速缓冲存储器由寄存器和缓存组成，其存储速度最快但成本也最高。主存储器也就是通常所说的内存储器，通常是由动态随机访问存储器(DRAM)组成。再高一级就是辅助存储器，一般以硬盘、光盘等造价较低同时存储速度较慢的存储介质组成。辅助存储器可以通过光纤网络(例如光纤通道或者互联网小型计算机接口iSCSI)提供更大的外部存储容量。图2.2描述了一个典型的存储器层级和相应特点。

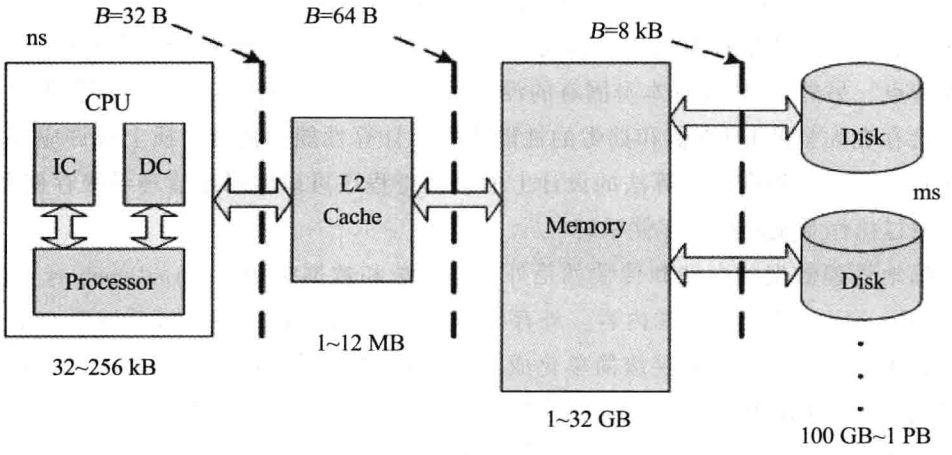


图2.2 一个典型的系统存储层级图

如图2.2所示，高速缓冲存储器包括寄存器、指令高速缓冲存储器、数据高速缓冲存储器，二级缓存；主存储器即内存(Memory)；辅助存储器即图中的磁盘(Disk)。存储器读取的时间从寄存器和高速缓冲存储器的不到1纳秒($10^{-9}$秒)到磁盘的几毫秒($10^{-3}$秒)，相差100万倍以上。不同层级存储器的存储容量显示在图2.2的下面。图顶部的$B$表示在两个相邻的存储层之间传递数据块大小的典型值。图中的数据单位分别为字节(B，Byte)、千字节(kB，$10^3$B)、兆字节(MB，$10^6$B)、千兆字节(GB，$10^9$B)、拍字节(PB，$10^{15}$B)。在图中，磁盘与内存之间物理传输块的大小8 kB仅是一个指示值，在某些批处理应用中通常会使用明显比其更大的数据块。

目前大多数的现代编程语言都基于RAM计算模型,即内存被认为是一个统一的地址空间的编程模式。借助虚拟存储的概念,编程时的地址空间可以远远大于实际计算机内存的大小。程序员有一个自然的趋势就是假设其使用的所有存储器都具有相同的存取速度。在大多数情况下,这种假设是合理的,特别是在数据集不大的情况下。这种编程模式的实用和优雅在很大程度上是其广泛应用的原因,并且对软件产业产生了非常大的贡献。

但事实上,显然并不是所有的存储器存储速度都是一样的。编程时的大地址空间会跨越多个层级的存储器,在最底层的存储器上存储数据要比在高一级的存储器上存取数据快几个数量级。例如,CPU在一个数据寄存器上的存取数据花费几分之一纳秒($10^{-9}$秒)的时间,内存花费几纳秒;但磁盘要花费数毫秒($10^{-3}$秒)的时间,大约相差100万倍!在实际应用中面对海量数据的处理时,在存储器不同层级之间的输入/输出通信(I/O communication)往往成为影响程序执行时间的瓶颈问题。

许多计算机程序在其数据的存取模式中会呈现一定程度的局部性(Locality),也就是特定的数据在一段时间内被反复地存取,之后程序的注意力会转移到其他一组数据中。现代操作系统利用了这种局部性的特点,通过跟踪程序所谓的"工作集"来提高速度。如果工作集较小,就可以将其存储在较高速的存储层级中以节省存取时间。缓存技术和启发式的预载被用来减少"缺页"的发生次数。"缺页"是指引用的数据不在缓存中,只能从高一级的存储设备中调入。例如,在一个内存缺页中,需要通过1次I/O通讯将一页数据从磁盘调入内存。

缓存和预载方法是针对通用目标而设计的,所以不能像预想的一样利用每一种计算的局部性特点。另外,一些计算本身固有的没有局部性,即使是无所不知的缓存管理决策技术也注定会存在大量的I/O次数和拙劣的性能表现。计算性能的提升本质上或许应该通过将局部化特点直接应用在程序算法的设计上面。使得程序可以明确地管理各级存储器内容,而不是通过操作系统的虚拟存储系统。

明确地管理数据的存储和传输就是外存储算法和数据结构(external memory algorithms and data structures)主要的考虑内容。外存储算法有些文献也称作I/O算法或者Out-of-core算法。这类算法通常将存取层级简单化成两级模式,即只关注随机存储器的内存和磁盘的外存之间的I/O通信,因为它们之间的存取速度差别相对最明显。同时以内外存之间的I/O通信次数作为衡量算法效率和复杂度的主要标准。

## 2.4 极大点查询外存储算法

目前已有少量极大点查询外存储算法,例如BNL算法[10]、SFS算法[43]。本节对这些经典的极大点外存储算法进行介绍。

BNL(block-nested-loops)算法:该算法对待测数据点建立一个临时表T,T由多个临时子表$T_i$组成,待检测的第1组输入数据放入临时表$T_0$中。程序在内存中维持一个窗口,保存到当前为止确定的极大点,窗口初始化为空。算法开始时,第1个读取的点自然地被放入窗口中。每当从当前临时表$T_i$中读入一个点$x$时,就用该点和窗口队列中已有的所有点依次进行比较,可能出现以下3种情况: