算法设计与分析

第17讲 随机算法应用

汪小林

北京大学 信息科学技术学院

目录

目录

问题描述

- 给定一个无向无权连通图$G(V, E)$，求其中所有点对之间的最短路径。

目录

普通解法

- 最简单直接的想法，我们可以利用Floyd算法来对该问题进行求解。算法的复杂度是$O(n^3)$的。
- 或者，对于每个顶点i，我们可以用最短路算法或者广搜方法求出其到达其他任何一个顶点的最短路。对所有顶点使用该算法后即可获得所有点对之间的最短距离。由于最短路算法是$O(n^2)$或者$O(m)$的，而在稠密情况下$m = O(n^2)$。故总的算法代价仍然是$O(n^3)$的。

普通解法

- 通过上述的分析我们可以大致知道，用普通算法解决该问题，时间复杂度为$O(n^3)$。
- 下面我们将介绍一种带有随机机制的算法，使得总的问题复杂度降为$O(n^{2.376}log^2 n)$。

目录

APD问题

- 为了更简单的入手分析这个问题，我们首先先求出任意两点之间的最短距离，这个问题我们称之为所有点对之间最短路径长度问题（all-pairs distances, APD）。
- 这一问题其实通过任意一个最短路算法都可以解决，但是为了之后将问题推广到求最短路径，我们介绍一种比较特殊的方法来解决APD问题。

图论知识回顾

- 设A为图G的邻接矩阵，则$A_{ij} = 1$当且仅当i、j在图G中相邻。
- 设矩阵$C = A^n$，则C_{ij}等于图G中在i、j之间距离为n的通路数量。
- 图的直径：图中任意两点间最短路径的最大值。

一些定义

- 设G是一个无向无权连通图，我们将图G中所有距离为1或2的点对之间加一条边，得到新图G’，称G’为G的平方。
- 我们定义G的邻接矩阵为A，距离矩阵为D；G’的邻接矩阵为A’，距离矩阵为D’。问题的目的就是根据G的邻接矩阵A求出G的距离矩阵D。
- 在之后的算法过程中我们会经常用到矩阵A的平方，定义其为Z，即Z=A*A。

矩阵A’的计算

- **定理**：在图G’中，两点i和j相邻当且仅当i和j在图G中的距离为1或2。
- 根据这一性质，我们就可以利用A和Z计算A’了，具体方法为：

$$A'_{ij} = 1 \text{当且仅当} i \neq j \text{且} Z_{ij} \text{和} A_{ij} \text{不都为} 0$$

- 下面我们考虑如何利用A’来计算D。

矩阵D的计算

- 注意到一个性质：当且仅当G的直径不超过2时，G’是完全图，A’是除对角元素为0外其他所有元素为1的矩阵。这种情况我们称之为终止情况。
- 此时，D可以通过A和A’直接计算得到，公式为：

$$D=2A'-A$$

矩阵D的计算

- 对于G的直径大于3，即G’不是完全图的情况，我们称之为一般情况。在一般情况下，可以利用D’来计算D。公式如下：
  - 如果D_{ij}是偶数，则$D_{ij} = 2D'_{ij}$
  - 如果D_{ij}是奇数，则$D_{ij} = 2D'_{ij} - 1$
- 容易证明这种方法的正确性。
- 于是，现在问题被转化为求无向无权连通图G’的距离矩阵D’，这个问题可以递归求解。因为每次对图求平方时，都会使任意两点间的距离缩小近一半，所以经过若干次求平方，图一定会变为一个完全图，这就是上页中提到的递归算法的终止情况，也是递归的终点。

思路整理

- 算法输入：图G的邻接矩阵A
- 算法输出：图G的距离矩阵D
- 1.计算Z=A*A
- 2.计算A’，即得到图G的平方G’
- 3.如果G’是完全图，则利用D=2A’-A得到D，返回。
- 4.否则，递归调用该算法计算，利用A’计算得到D’，然后利用上页中的公式用D’来计算D。

- 现在的问题的关键就在于：**如何确定D_{ij}的奇偶性**

如何确定D_{ij}的奇偶性

- **引理**：对于图G中任意一对不同的顶点i、j，对于i的任意邻居k，有$D_{ij}-1 \le D_{kj} \le D_{ij}+1$；顶点i存在邻居k使得$D_{kj}=D_{ij}-1$。
- 由引理继而得到下一个性质：
- **性质**：
  - 如果D_{ij}为偶数，则对于顶点i的每一个邻居k来说，$D'_{kj} \ge D'_{ij}$；
  - 如果D_{ij}为奇数，则对于顶点i的每一个邻居k来说，$D'_{kj} \le D'_{ij}$。
- 对于i的每个邻居，对上述等式做累加，通过比较两者的大小就可以判断D_{ij}的奇偶性。
- 至此，APD问题已经完全解决。

最终算法

- 算法输入：图G的邻接矩阵A
- 算法输出：图G的距离矩阵D
- 1.计算Z=A*A
- 2.计算A’，即得到图G的平方G’
- 3.如果G’是完全图，则利用D=2A’-A得到D，返回。
- 4.否则，递归调用该算法计算，利用A’计算得到D’。
- 5.根据D'_{ij}判断D_{ij}的奇偶性，然后计算得到D，返回。

APD问题复杂度分析

- 设$MM(n)$表示两个n阶矩阵相乘的时间复杂度。在现在已经知道的算法中，最优的算法在渐进意义下的复杂度是$O(n^{2.376})$。
- 假设图G的直径为d，则图G'的直径为$[d/2]$。令$T(n,d)$表示APD算法在面对一个有n个顶点，直径为d的图作为输入时的运行时间。
- 当$d=2$时，$T(n,d)=MM(n)+O(n^2)$
- 当$d>2$时，$T(n,d)=2MM(n)+T(n,[d/2])+O(n^2)$
- 而递归的深度至多为$\log n$，故算法的总复杂度为$O(n^{2.376}\log n)$。

目录

APSP问题

- 下面我们在已经求得所有点之间最短距离的基础上，求所有点之间的最短路径。我们称之为无权图的所有点之间最短路径问题（all-pairs shortest paths problem, APSP）。
- 我们将在解决APSP问题时使用随机机制。

后继矩阵

- 对于图G，设矩阵S，其中S_{ij}是顶点i在由i到j的最短路径上的邻居的标号。称S是图G的后继矩阵。
- 如果我们得到了图 G的后继矩阵 S，则对于任意两点i、j，我们就可以通过每次寻找邻居的方法从后继矩阵一步步的得到他们之间的最短路径。
- 于是APSP的问题本质就是根据图G的邻接矩阵A、距离矩阵D来求它的后继矩阵S。

布尔矩阵相乘的“证据”

- 假设$\boldsymbol{A}$和$\boldsymbol{B}$是$\boldsymbol{n}*\boldsymbol{n}$的布尔矩阵，$\boldsymbol{P}=\boldsymbol{AB}$是他们在布尔矩阵乘法下的乘积。
- 元素$\boldsymbol{P_{ij}}$的一个证据是指一个标号$\boldsymbol{k} \in \{\boldsymbol{1}, \ldots, \boldsymbol{n}\}$，使得$\boldsymbol{A_{ik}} = \boldsymbol{B_{kj}} = \boldsymbol{1}$。
- 因为这里使用的乘法是布尔乘法，故
- P_{ij}存在至少一个证据当且仅当$P_{ij} = 1$。 当然， $\boldsymbol{P}$的每个元素都可能拥有多个“证据”。
- 假设$\boldsymbol{C}=\boldsymbol{AB}$是$\boldsymbol{A}$和$\boldsymbol{B}$的整数乘积（非布尔乘积），则$\boldsymbol{C_{ij}}$表示了$\boldsymbol{P_{ij}}$的证据数。

“证据”的图论意义

- 假设A是一个图G的邻接矩阵，$B = A^x$，$X = A^y$，$P = BX$。此处的乘法全部为布尔乘法。
- 则$B_{ik} = 1$表示从i到k存在长度为x的路径，则$X_{kj} = 1$表示从k到j存在长度为的路径。
- $P_{ij} = 1$表示存在一条从i到j的长度为x+y的路径，而P_{ij}如果存在一个“证据”k，则说明$B_{ik} = 1$、$X_{kj} = 1$，其意义为存在一条从i到j且经过k的长度x+y的路径。进一步可知在这条路径上从i到k的距离为x，从k到j的距离为y。
- 这就是“证据”在图论上的意义。

“证据”与后继矩阵S的关系

- 对于两个节点i、j，假设他们的距离为d。令A为图G的邻接矩阵，矩阵$B = A^{d-1}$，$P = AB = A^{d}$，乘法均为布尔乘法。
- 显然$P_{ij} = 1$，如果它有一个证据k，根据上页的分析，就可以说明存在一条从i到j经过k的距离为d的路径，也就是从i到j的最短路。而且，i到k的距离为1，即k是i的邻居。
- 根据上述性质，这个k就是一个符合要求的S_{ij}。
- 只要我们对任意的i和j都找出P_{ij}的一个证据，我们就能得到后继矩阵。

思路整理

- 算法输入：图G的邻接矩阵A和距离矩阵D
- 算法输出：图G的后继矩阵S
- 1.预处理，得到矩阵A、A^2、...、A^n（布尔乘法）
- 2.对于任意点对i、j：
  - 从距离矩阵D中得到i、j的距离d
  - 得到矩阵$B = A^{d-1}$
  - 得到矩阵P=AB
  - 找到P_{ij}的一个证据k，$S_{ij} = k$
- 3.返回S

- 现在问题的关键为： 如何找到P_{ij}的一个证据。

如何找到P_{ij}的一个证据

- 设$P=AB$。如果P_{ij}只含有一个证据：
- 定义矩阵T，$T_{ik} = kA_{ik}$。
- 设矩阵$W=TB$，则W_{ij}即为P_{ij}的证据。

如何找到P_{ij}的一个证据

- 当然，我们不能保证每一个P_{ij}都只有一个证据。但是我们可以利用随机机制来保证P中充分多的元素具有这一性质。
- 假设 P_{ij} 的证据个数为w，这个值可以从矩阵C中直接得到。如果$w=1$，则可以按照上页的方法直接求得。
- w大于1时：取一个整数r，r满足$\frac{n}{2} \leq wr \leq n$。我们从1~$n$这$n$个数中随机挑选$r$个数构成集合$R$。可以证明，$R$中只包含$P_{ij}$的一个证据的几率$\Pr \geq \frac{1}{2e}$。

$$
\begin{aligned}
\frac{C_w^1 C_{n-w}^{r-1}}{C_n^r} &= w\frac{r!}{(r-1)!}\frac{(n-w)!}{n!}\frac{(n-r)!}{(n-w-r+1)!} \\
&= wr(\prod_{i=0}^{w-1}\frac{1}{n-i})(\prod_{j=0}^{w-2}(n-r-j)) \\
&= \frac{wr}{n}(\prod_{j=0}^{w-2}\frac{n-r-j}{n-1-j}) \\
&\geq \frac{wr}{n}(\prod_{j=0}^{w-2}\frac{n-r-j-(w-j-1)}{n-1-j-(w-j-1)}) \\
&= \frac{wr}{n}(\prod_{j=0}^{w-2}\frac{n-w-(r-1)}{n-w}) \\
&= \frac{wr}{n}(1-\frac{r-1}{n-w})^{w-1} \\
&\geq \frac{1}{2}(1-\frac{1}{w})^{w-1}
\end{aligned}
$$

如何找到P_{ij}的一个证据

- 假设集合R包含元素P_{ij}的唯一证据，我们可以用类似只包含一个证据的方法来求得这个证据。
- 设R表示一个向量，如果k在集合R中，则$R_k = 1$，否则$R_k = 0$。
- 设矩阵A、B的含义同上述定义。设矩阵AA满足$AA_{ik} = kR_kA_{ik}$，矩阵BB满足$BB_{ik} = R_kB_{ik}$。这样的处理与之前的处理唯一的差别就在于只保留了两个矩阵中R_k为1的列。
- 设矩阵PP=AA*BB，则PP_{ij}即为P_{ij}的一个证据。

求证据

- 可以证明求证据的算法所需要的期望运行时间是$\boldsymbol{O(MM(n)log^2n)}$的。