大数据计算基础
BIG DATA COMPUTING

刘显敏 海量数据
liuxianmin@hit.edu.cn

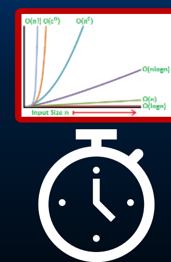
大数据算法

2025年秋

2

亚线性时间算法

Sublinear Time Algorithm



亚线性时间算法

3

高效算法的标准是什么?
- 多项式时间算法
- 线性时间算法：输入大小为n，算法的时间代价是cn

若时间资源受限，甚至不允许读数据一遍，能做什么?
- 无法回答 “for all” 或者 “exactly” 类似的问题
  - 数组中是否所有元素均比10大? 数组中比10大的有多少?
- 或许能够回答下列问题
  - 数组中比10大的元素大概有多少?
  - 数组中的平均数是否以很大概率大于10?

算法的计算目标需要修改

亚线性时间算法

4

算法的计算目标需要修改
- 对于绝大多数问题，我们只能给出近似的答案

什么是 “近似”
- 经典的近似算法概念：
  - 算法的结果有一个度量函数：路径长度
  - 算法结果度量值与最优解度量值 “接近” ：近似比
- Property Testing（判定问题）
  - 算法的结果是yes或者no
  - 算法输出要么与输入 I 对应的答案相同，要么与 I' 对应的答案相同，这里 I' 与 I 非常接近

问题1：点集合的直径

5

点集合的直径问题

输入：m个点，距离用矩阵 D表示

(1) D_{ij}是i点到j点的距离

(2) D是对称的，并且满足三角不等式

$D_{ij} \le D_{ik} + D_{kj}$

输出：(i, j)使得D_{ij}是最大的，D_{ij}是点集合的直径

问题1：点集合的直径

6

亚线性求解直径算法（Indyk’s Algorithm）

/** input D **/

1. 选取任意一个 k，选取 l 最大化 D_{kl};
2. **return** (k, l)

近似比分析：$opt/2 \le D_{kl} \le opt$

$opt = D_{ij} \le D_{ik} + D_{kj}$（三角不等式）

$\le D_{kl} + D_{kl}$（l是最大化选择）

$\le 2D_{kl}$

运行时间：$O(m) = O(\sqrt{n})$

问题2：连通分量的数目

连通分量的数目（#CC）

输入：G = (V, E), ϵ, d = deg(G)

图G用邻接链表表示，|V| = n, |E| = m ≤ dn

输出：y, 令C为#CC

C − ϵn ≤ y ≤ C + ϵn （additive appro.）

#CC可以在线性时间求解（DFS或者BFS）

问题2：连通分量的数目

n_v：顶点v所属的连接分量中的节点数目

A：A ⊆ V是一个连通分量的点集合

$$\sum_{u\in A}\frac{1}{n_u}=\sum_{u\in A}\frac{1}{|A|}=1$$

$$C=\#cc=\sum_{u\in V}\frac{1}{n_u}$$

为什么用这种表示?

- 可以支持高效估计(estimate)

问题2：连通分量的数目

估计$C=\sum_{u\in V}\frac{1}{n_u}$：估计$\frac{1}{n_u}$ ⇒估计$\sum_{u\in V}\frac{1}{n_u}$

估计$\frac{1}{n_u}$

想法：n_u很大，精确计算很难，但此时$\frac{1}{n_u}$很小，可以用一个很小的常量代替 $\frac{1}{n_u}$ （0或者ϵ/2）

$$\hat{n}_u=\min\left\{n_u,\frac{2}{\epsilon}\right\}\qquad \hat{C}=\sum_{u\in V}\frac{1}{\hat{n}_u}$$

引理：∀u ∈ V，有$\left|\frac{1}{n_u}-\frac{1}{\hat{n}_u}\right|\le\epsilon/2$，即$|C-\hat{C}|\le\frac{\epsilon n}{2}$。

问题2：连通分量的数目

估计$C=\sum_{u\in V}\frac{1}{n_u}$ ⇒估计$\frac{1}{n_u}$ ⇒估计$\sum_{u\in V}\frac{1}{n_u}$

$$\hat{n}_u=\min\left\{n_u,\frac{2}{\epsilon}\right\}\qquad \hat{C}=\sum_{u\in V}\frac{1}{\hat{n}_u}$$

计算$\hat{n}_u$算法

1.从 u 开始做先广遍历（BFS）；

2.**while** BFS 访问过的节点数量 ≤$\frac{2}{\epsilon}$ **do**

3. 继续做 BFS 遍历;

4. **if** 没有新节点 **then**

5. **return** BFS 访问过的节点数量;

6.**return** 2/ϵ;

运行时间：$O(d\cdot 1/\epsilon)$

问题2：连通分量的数目

估计$C=\sum_{u\in V}\frac{1}{n_u}$ ⇒估计$\frac{1}{n_u}$ ⇒估计$\sum_{u\in V}\frac{1}{n_u}$

$$\hat{n}_u=\min\left\{n_u,\frac{2}{\epsilon}\right\}\qquad \hat{C}=\sum_{u\in V}\frac{1}{\hat{n}_u}$$

亚线性连通分量数目求解算法（用 $\tilde{C}$ 估计 $\hat{C}$）

1.$r\longleftarrow b/\epsilon^2$；

2.随机从 V 中选取$U=\{u_1,\dots,u_r\}$;

3.计算所有的 $\hat{n}_{u_i}$；

4.**return** $\tilde{C}=\frac{n}{r}\sum_{u_i\in U}\frac{1}{\hat{n}_{u_i}}$;

运行时间：$O\left(d\cdot\frac{1}{\epsilon}\cdot\frac{1}{\epsilon^2}\right)=O\left(\frac{d}{\epsilon^3}\right)=o(|G|)$

问题2：连通分量的数目

估计$C=\sum_{u\in V}\frac{1}{n_u}$ ⇒估计$\frac{1}{n_u}$ ⇒估计$\sum_{u\in V}\frac{1}{n_u}$

$$\hat{n}_u=\min\left\{n_u,\frac{2}{\epsilon}\right\}\qquad \hat{C}=\sum_{u\in V}\frac{1}{\hat{n}_u}$$

亚线性连通分量数目求解算法（用 $\tilde{C}$ 估计 $\hat{C}$）

1.$r\longleftarrow b/\epsilon^2$；

2.随机从 V 中选取$U=\{u_1,\dots,u_r\}$;

3.计算所有的 $\hat{n}_{u_i}$；

4.**return** $\tilde{C}=\frac{n}{r}\sum_{u_i\in U}\frac{1}{\hat{n}_{u_i}}$;

近似性能：$r=O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})\Longrightarrow\Pr[|C-\tilde{C}|\ge\epsilon n]\le\delta$

定理[Chernoff/Hoeffding Bound]

13

$X_1, X_2, \ldots, X_n$为独立随机变量，取值范围$\in [0,1]$, $\mu = \mathbb{E}[\sum_i X_i]$，对任意 $t \geq 0$

$$\mathbf{Pr}[\sum_{i=1}^{n} X_i - \mu \geq t] \leq e^{-\frac{2t^2}{n}}$$

$$\mathbf{Pr}[\sum_{i=1}^{n} X_i - \mu \leq -t] \leq e^{-\frac{2t^2}{n}}$$

$$\mathbf{Pr}[|\sum_{i=1}^{n} X_i - \mu| \geq t] \leq 2e^{-\frac{2t^2}{n}}$$

问题3：近似最小支撑树

14

最小支撑树（Min Spanning Tree）

输入：G = (V, E), ϵ, d = deg(G)

图G用邻接链表表示

边(u, v)的权重是$w_{uv} \in \{1, 2, \ldots, w\} \cup \{\infty\}$

输出：$\hat{M}$，令M为 $\min_{T \text{ spans } G} W(T)$

$(1-\epsilon)M \leq \hat{M} \leq (1+\epsilon)M$

MST问题可以在多项式时间求解，例如Kruskal算法

问题3：近似最小支撑树

15

我们需要重新利用分解的方式定义M

$G^{(i)} = (V, E^{(i)})$，这里$E^{(i)} = \{(u,v) | w_{uv} \leq i\}$

$C^{(i)} = \#CC \text{ in } G^{(i)}$

考虑两个例子

- w = 1: 只有权重为1的边，且连通

  $M = n - 1$
- w = 2: 有权重为1和2的边，且连通

  $M = n - 1 + C^{(1)} - 1 = n - 2 + C^{(1)}$

问题3：近似最小支撑树

16

定理 $M = n - w + \sum_{i=1}^{w-1} C^{(i)}$

α_i：任一MST中权重为 i 的边的数目

$$\sum_{i>l} \alpha_i = C^{(l)} - 1$$

$$\begin{aligned} M &= \sum_{i=1}^{w} i \cdot \alpha_i = \sum_{i=1}^{w} \alpha_i + \sum_{i=2}^{w} \alpha_i + \cdots + \sum_{i=w}^{w} \alpha_i \\ &= C^{(0)} - 1 + C^{(1)} - 1 + \cdots + C^{(w-1)} - 1 \\ &= n - 1 + C^{(1)} - 1 + \cdots + C^{(w-1)} \\ &= n - w + \textstyle\sum_{i=1}^{w-1} C^{(i)} \end{aligned}$$

问题3：近似最小支撑树

17

利用连通分量数目估计的算法来估计MST大小

$$\tilde{O}\left(\frac{d}{\epsilon^3}\right) \Longrightarrow \mathbf{Pr}[|C - \tilde{C}| \leq \epsilon n] \geq 1 - \delta$$

亚线性近似最小支撑树算法（计算$\hat{M}$）

1.**for** i = 1 to w − 1 **do**

2. $\widehat{C^{(i)}}$ = appro. #CC of $G^{(i)}$ within $(\epsilon' = \frac{\epsilon}{2w}) \cdot n$;

   with probability $\geq 1 - \delta' = 1 - \frac{\delta}{w}$

3.**return** $\hat{M} = n - w + \sum_{i=1}^{w-1} \widehat{C^{(i)}}$;

单次时间：$\tilde{O}\left(d \cdot \frac{1}{\epsilon'^3}\right) = \tilde{O}\left(\frac{dw^3}{\epsilon^3}\right)$

共计时间：$\tilde{O}\left(\frac{dw^4}{\epsilon^3}\right) = o(|G|)$

近似性能：$\mathbf{Pr}[|\hat{M} - M| \geq \epsilon M] \leq \delta$

问题4：Vertex Cover

18

顶点覆盖（Vertex Cover）

输入：G = (V, E)、最大度数d，图G用邻接链表表示

C ⊆ V是一个VC (Vertex Cover)当且仅当

∀(u, v) ∈ E有u ∈ C或者v ∈ C

输出：最小VC的大小 |VC|

- $|VC| \geq \frac{|E|}{d}$
- NP-完全问题
- 存在集中式的2近似算法

问题4：Vertex Cover

19

在亚线性时间内，能有多好的近似算法？

- 乘性的近似比 ($\frac{\widetilde{VC}}{VC}$)：No

  0条边的图 ⇒ |VC| = 0；　1条边的图 ⇒ |VC| = 1

  区分上述两种情况需要 Ω(n) 时间
- 加性的近似比 ($|\widetilde{VC}| - |VC|$)：Hard
- 乘性近似比的下界是 1.36，很可能是 2

定义[(α, ϵ)-近似]对于最优解是y的最小化问题，如果

$$y \le \hat{y} \le \alpha y + \epsilon$$

则称$\hat{y}$是该问题的(α, ϵ)-近似。

问题4：Vertex Cover

20

设计想法：利用分布式算法设计亚线性算法

分布式网络

- 最大度数 d
- 每个节点是一个处理器，知道它的邻居是谁
- 同步计算、通信

每一轮（同步）

- 每个节点计算基于它的输入、随机生成位、通信中收到的信息
- 向每个邻居节点发送消息
- 从每个邻居节点收取消息

问题4：Vertex Cover

21

设计想法：利用分布式算法设计亚线性算法

分布式顶点覆盖（Vertex Cover）问题

- 输入：网络图即是要计算顶点覆盖的图 G
- 输出：每个节点知道自己是否属于 VC

Fast Distributed Alg. ⇒ SubLinear Time Alg.

- k 轮分布式算法中，节点 v 最多依赖于距离为 k 的节点，至多 d^k 个
- 集中式算法可以在 d^k 时间内，模拟分布式算法，计算 v 是否属于VC

如果是随机算法，需要知道所有 d^k 个节点的随机位

问题4：Vertex Cover

22

- 存在fast VC distribute alg (local distributed alg)?
- 如何利用分布式算法设计亚线性算法?

顶点覆盖的亚线性算法

1. 独立均一地从 G 中选取 $s = \frac{8}{\epsilon^2}$ 个节点构成 S;
2. 为每个 v ∈ S，构造 k 步邻居导出的子图 $G_k(v)$;
3. 为每个 v ∈ S，在 $G_k(v)$ 上模拟分布式算法 D;
4. 如果 D 返回 v 为覆盖节点之一，$X_v = 1$;
5. **return** $|\widetilde{VC}| = \frac{n}{s} \cdot \sum_{v \in S} X_v + \frac{\epsilon}{2} n$;

运行时间分析：$O(s \cdot d^k) = O(\frac{d^k}{\epsilon^2})$

问题4：Vertex Cover

23

- 存在fast VC distribute alg (local distributed alg)?

分布式顶点覆盖算法 D

1. $\widetilde{VC} \leftarrow \emptyset$;
2. **for** i = 1 to $\log d$ **do**
3. 　　$\Delta \leftarrow \{v: v \notin \widetilde{VC}, \deg(v) \ge d/2^i\}$;
4. 　　$\widetilde{VC} \leftarrow \widetilde{VC} \cup \Delta$;
5. 　　从 G 中删除所有与 Δ 邻接的边;
6. **return** $\widetilde{VC}$;

- 运行时间：$d^{O(\log d)}$；近似比：$(O(\log d), \epsilon n)$

24

大数据平台的并行算法