第二部分 分布式算法

第四次课

中国科学技术大学计算机系

国家高性能计算中心（合肥）

§2.5 不指定根时构造DFS生成树

算法2.2和2.3构造连通网络的生成树时，必需存在一个特殊的结点作为启动者(Leader)。当这样的特殊结点不存在时，如何构造网络的一棵生成树？但本节算法须假定：**各结点的标识符唯一，不妨设是自然数，§3.2仍需此假定。**

不指定根构造DFS生成树，和后面的领导者选举问题一样，都是破对称问题。

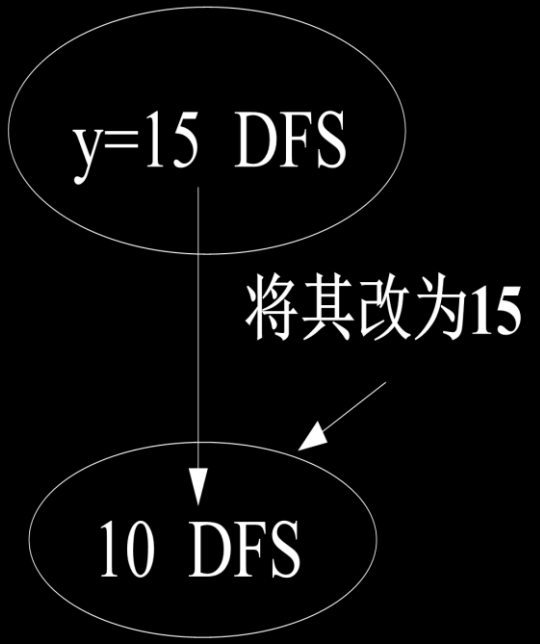
1. 基本思想

- 每个结点均可自发唤醒，试图构造一棵以自己为根的DFS生成树。若两棵DFS树试图链接同一节点(未必同时)时，该节点将加入根的id较大的DFS树。
- 为了实现上述思想，须做：
  - 每个结点设置一个leader变量，其初值为0，当P_i唤醒自己时，$leader_i=id_i$；

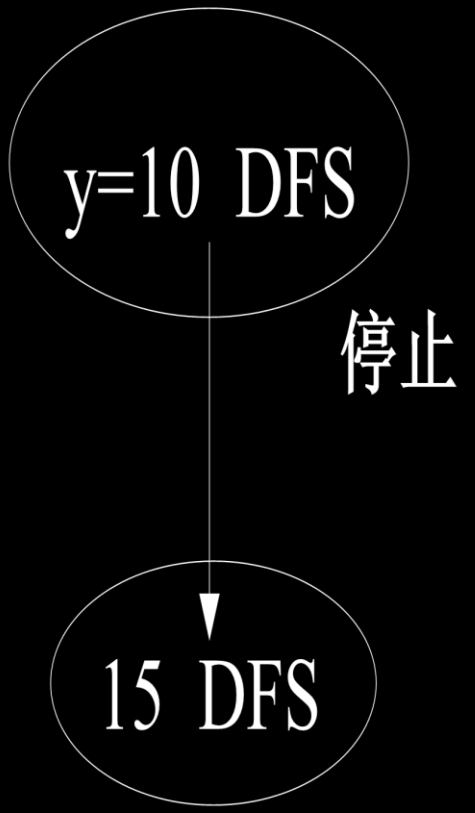
§2.5 不指定根时构造DFS生成树

- 当一结点自发唤醒时，它将自己的id(leader)发送给某一邻居；
- 当一结点P_i收到来自邻居P_j的标识符y时，P_i比较y和$leader_i$：

①若$y>leader_i$，则y可能是具有最大标识符结点的DFS子树的标记。因此，将$leader_i$置为y，并令P_j是P_i的双亲。从P_i开始继续生成标记为y的DFS树。Note：要修改原P_i所在的DFS子树中所有结点的leader。



§2.5 不指定根时构造DFS生成树



② 若$y<leader_i$，则标记为y的DFS树中最大id(y)小于目前所看到的最大标识符。此时无须发送msg，停止构造标记为y的DFS。等待最终某个更大的id的leader消息到达标记为y的树中结点时，再将该节点连接到树中。(至少标记为$leader_i$的msg会到达标记为y的树)

③ 若$y=leader_i$，则P_i已属于标记y的DFS树中。

§2.5 不指定根时构造DFS生成树

2. 算法

Alg2.4 构造生成树，不指定根

Code for Processor P_i $0 \le i \le n-1$

```
Var parent:      init nil;
    leader:      init 0;
    children:    int φ;
    unexplored:      init all neighbors of Pi;
1: upon receiving no msg:  //wake up spontaneously
2:    if parent = nil then {
          //若非空，则Pi在某子树上，则Pi失去竞选机会
3:        leader := id;  parent := i;//试图以自己为根构造树
4:        ∀ Pj∈unexplored;
5:        将Pj从unexplored中删去；
6：       send <leader> to pj;
      }
```

§2.5 不指定根时构造DFS生成树

想像：有m个人竞选领袖，id是他自身的素质分，不想竞争人的id不参与比较。

竞争规则：将自己的id(如讲演片)传递给一个熟悉的人，由他再传给另一人(一次只能送一人。)

7: upon receiving <new-id> from neighbor P_j:

8: if leader<new-id then { //将P_i所在树合并到P_j所在树中

9: leader := new-id; parent := j;

//令P_i的双亲为P_j，可能是修改，而非对nil赋值

//并不一定能停止较差的竞选者传播msg

10: unexplored := all the neighbors of P_i except P_j;

//重置未访问的邻居集

11: if unexplored ≠φ then {

//因为new-id大，使原P_i所在DFS树修改各自id

12: $\forall P_k \in$ unexplored;

13: 将P_k从unexplored中删去；

§2.5 不指定根时构造DFS生成树

14: send <leader> to P_K;
15: }else send <parent> to parent; // unexplored =φ
}else // leader≥new-id
16: if leader=new-id then send <already> to P_j;
//表示自己已经传出过此录像带，无需重传。已在同一树中
//若leader>new-id，则new-id所在DFS停止构造
//以前收到的竞选者优于new-id，不传送，使之停止传播。

17: upon receiving <parent> or <already> from neighbor P_j:
18: if received <parent> then add j to children;
19: if unexplored=ϕ then { //无尚未访问的邻居
20: if parent≠i then send <parent> to parent //返回
21: else terminates as root of the DFS tree; //根终止
22: }else { //有尚未访问的邻居

§2.5 不指定根时构造DFS生成树

23: $\forall P_k \in$ unexplored;
24: 将P_k从unexplored中删去；
25: send <leader> to P_k;
}

3. 分析：

- 只有生成树的根显式地终止，其它结点没有终止，始终在等待msg。但可修改此算法，使用Alg2.1从根结点发送终止msg
- 正确性

该算法比前面的算法更复杂，这里只给出粗略的证明。

设P_m是所有自发唤醒结点中标识符最大者，其标识符为id_m。消息id_m总是被传播，而一旦一个结点收到id_m，则该节点(P_m除外)上所有msgs被忽略。因为消息id_m的处理和Alg2.3求DFS树一致，因此产生的parent和children变量的设置是正确的。因此有：

§2.5 不指定根时构造DFS生成树

Lemma2.12 设P_m是所有自发唤醒结点中具有最大标识符的结点。在异步模型的每次容许执行里，算法2.4构造根为P_m的一棵DFS树。

Note：因为在容许执行中，网络里的所有自发唤醒结点中最大标识符结点最终会自发启动，故建立的DFS树的根是P_m

可通过广播算法从P_m发出终止msg，即使不广播，所有非P_m结点最终也会因为收到P_m的标识符而停止。因此，不可能构造一棵根不是P_m的生成树。

Lemma2.13 在异步模型的每个容许执行里，只有一个处理器终止作为一生成树的根。

§2.5 不指定根时构造DFS生成树

- 复杂性

定理：对于一个具有m条边和n个节点的网络，自发启动的节点共有p个，其中ID值最大者的启动时间为t，则算法的消息复杂度为$O(pn^2)$，时间复杂度为O(t+m)。

消息复杂性：简单地分析，最坏情况下，每个处理器均试图以自己为根构造一棵DFS树。因此，Alg2.4的msg复杂性至多是Alg2.3的n倍：O(m*n)

时间复杂性：类似于Alg2.3的msg复杂性O(m)。

§2.5 不指定根时构造DFS生成树

Ex.

2.1 分析在同步和异步模型下，convergecast算法的时间复杂性。

2.2 证明在引理2.6中，一个处理器在图G中是从P_r可达的，当且仅当它的parent变量曾被赋过值

2.3 证明Alg2.3构造一棵以P_r为根的DFS树。

2.4 证明Alg2.3的时间复杂性为O(m)。

2.5 修改Alg2.3获得一新算法，使构造DFS树的时间复杂性为O(n)，并证明。

补充

§ 2.6 小结

§2.6 小结

Introduction to distributed alg

- 分类：
  - 单源alg：一个启动者。又称centralized alg。
  - 多源alg：任意进程(结点)子集均可是启动者，又称decentralized alg
  - 启动者(initiator)：自发地执行局部算法，即由一内部事件激发其执行
  - 非启动者：由接收一个msg(外部事件)触发其执行局部进程。
- 复杂性：
  - Msg复杂性：msg总数目
  - Bit复杂性：发送msg中bit的总数目，当msg在发送过程中其长度随时间增长时

§2.6 小结

❖ 时间复杂性

① 一个分布式算法的**时间复杂性**是满足下述两个假定的一个计算所耗费的**最大时间**

T1：一个进程在零时间内可计算任何有限数目的事件

T2：一个msg的发送和接受之间的时间至多为1个时间单位

缺点:针对一算法的所有计算，其结果可能是极不可能发生的计算。

② 一个分布式算法的**one-time复杂性**是满足下述假定的一个计算的**最大时间**

O1：同T1

O2：发送和接收一个msg之间的时间恰好是1个单位时间

缺点：某些计算可能被忽略，而其中可能有极其耗时的计算

§2.6 小结

表面上，1-time复杂性至少等于时间复杂性，因为T2假定下的最坏时间不会高于O2假定下的时间。但事实并非如此，而往往O1和O2假定之下的1-time复杂性是前一种时间复杂性的一个下界。

例如：在echo算法里1-time复杂性是O(D)，时间复杂性是Θ(N)，即使直径为1的网络。

③ **两种复杂性的折中：α-复杂性**

假定每个msg延迟介于α-1之间(α≤1常数)

对echo 算法α复杂性为O(min(N, D/ α))

④ **概率分析**：msg延迟服从某种概率分布，由此可获得精确的时间复杂性度量

§2.6 小结

⑤ 基于msg chains的分析

任何计算中最长消息链的长度。

链上msgs：$m_1,m_2,\ldots m_k$序列中，m_i因果关系领先于m_{i+1}。

- 先验知识：邻居的id，全局id等。链路FIFO假定等

下次继续！