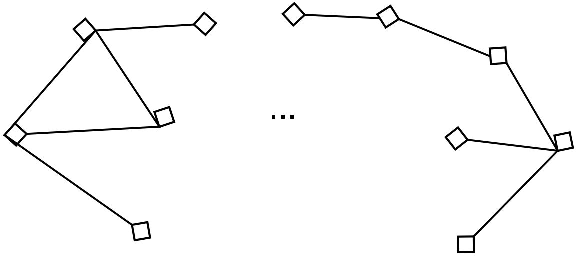
Distributed Algorithms 2

Philip Bille Inge Li Gørtz Eva Rotenberg

The Congest Model – Limited bandwidth

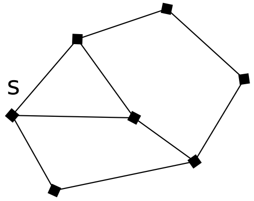


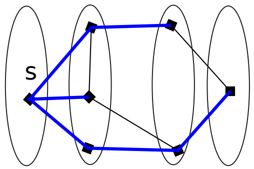
Each round, each vertex

- reads messages,
- performs infinite computation,
- sends messages to neighbours
    - Max size of message $= O(\log n)$ bits.

Question: Algorithms from last lecture – messages of size $O(\log n)$?

Breadth-first search tree





Reminder:

Given a graph and a source, s

We may divide into layers

of increasing distance from s.

BFS-tree connects the next layer to the previous.

RAM-model: Starting with $L_0 = \{s\}$,

find L_{i+1} as $\cup_{v \in L_i} \mathrm{Nb}(v) \setminus (L_{i-1} \cup L_i)$.

Link $u \in L_i$ to arbitrary neighbour in L_{i-1}.

Distributed: ?

Wave algorithm

Given a source s who initiates the wave (everyone else is passive)

Round 1: s sends message to neighbours. s is done.

All other rounds: each v who is not done:

- if some message from some w,
- set $p(v) = w$.
- v is done.
- send message to all neighbours.

Result: every vertex connected to s

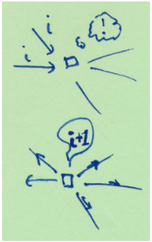
can send a message to s via a shortest path:

$v \to p(v) \to p(p(v)) \to p(p(p(v))) \to \ldots \to s$

Wave with distances

Given a source s who initiates the wave (everyone else is passive)

Round 1: s sends message 0 to neighbours. s is done.



All other rounds: each v who is not done:

- if some message i from some w,
- set $p(v) = w$ and $d(s, v) = i + 1$.
- v is done.
- send message $i + 1$ to all neighbours.

Result: every vertex connected to s knows their distance from s.

Question: So, do we have a representation of a BFS-tree here?

Problem 1: Who are my children?

Problem 2: When are we all done?

BFS-tree algorithm

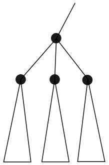
Objective: grow a bfs-tree such that for each edge in the bfs-tree, both endpoints are aware of this edge.

Modification 1: Accepting a parent.

When setting $p(v) = w$, notify w.

Result: w knows $v \in C(w)$, its set of children.

Modification 2: Are we all done? -bit. $a(v)$.



Once a vertex knows its children, its entire subtree is done once those children are all done.

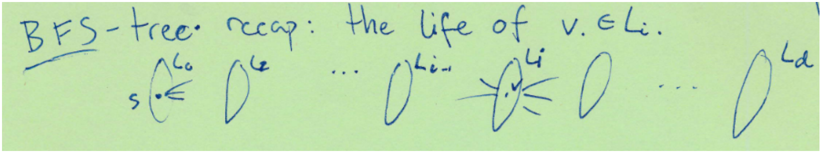
So, keep track of how many of your children have responded that they are done.

Once all children are done, send an acknowledgement to $p(v)$.

Result: At some point, s knows their message has reached every vertex.

Round complexity: If z is of max dist from s. $d(s, z)$ rounds to reach z. $O(1)$ rounds to realise $C(z) = \emptyset$. $d(z, s)$ rounds to send "ack" back to s. Total: $O(\text{diam}(G))$ rounds. Note: Tighter analysis than just saying $O(n)$.

BFS-tree recap: the life of a vertex $v \in L_i$



For rounds $1, 2, \ldots, i-1$, nothing happens.

Round i: $d(v) := i$, $p(v) = u$, $u \in L_i$.

Round $i+1$: $C(v) := \{c_1, c_2, \ldots\}$

Rounds $i+2, \ldots, x$: waiting on children.

Round $x+1$: send ack to $p(v)$.

Further rounds: nothing happens.

How many rounds?

Note that x can be no longer than $\text{diam}(G) + 1 + \text{diam}(G) - i$

Leader Election (Connected Graph)

Goal: Elect any leader.

Method: Elect vertex with lowest ID.

Round 1: Everyone starts BFS from themselves, appending myID.

Round $i + 1$: check msg and update minID.

proceed as BFS from minID.

augment messages with minID.

Result:

For s in G with smallest ID, BFS terminates with "ack" from all children.

For $v \neq s$, there is some child $c_i \in C(v)$ who will never send "ack". So no vertex falsely believe they are the leader. Eventually, everyone will agree on the minID.

How many rounds?

All Pairs Shortest Paths (Connected Graph)

Goal: Want $d(a, b)$ for all $a, b \in G$.

Trivial solution: run BFS from each $v \in V$.

Running this in sequence: $O(n \cdot \text{diam}(G))$ rounds.

Running this in parallel: no guarantees about message size.

Token walk algorithm

Given leader s and BFS-tree from s

Round 1: s passes TOKEN to $c_1 \in C(s)$.

Round > 1: vertex v reads messages. possibly: TOKEN, or Wave(x).

Case TOKEN Initiate wave(v). For $v \neq s$: next round, send TOKEN to successor in $\{p(v), c_1(v), c_2(v), \ldots c_{|C(v)|}\}$ of where the TOKEN came from.

Case wave(x): proceed with wave(x).

Rounds? Correctness?

Token Walk Analysis

Round complexity:

Every edge traversed 2 times. $n-1$ edges. $\Rightarrow O(n)$ token moves.´

After last token move, $O(\text{diam}(G))$ rounds to finish. $O(n)$ total rounds.

Correctness: v does not receive wave(x) and wave(y) the same round.

Assume v gets wave(x) and wave(y) in round i. Assume wave(x) and wave(y) were initiated in rounds t_x and t_y with $t_x < t_y$.

Then, $d(v,x) = i - t_x$ and $d(v,y) = i - t_y$.

The token only moves in odd rounds, so $t_y - t_x \geq 2d(y,x)$.

$i - t_x = d(x,v) \leq d(x,y) + d(y,v) = d(x,y) + i - t_y$

$\Rightarrow t_y - t_x \leq d(x,y)$. So... $2d(x,y) \leq d(x,y)$.

$\Rightarrow d(x,y) = 0$, i.e. $x = y$.