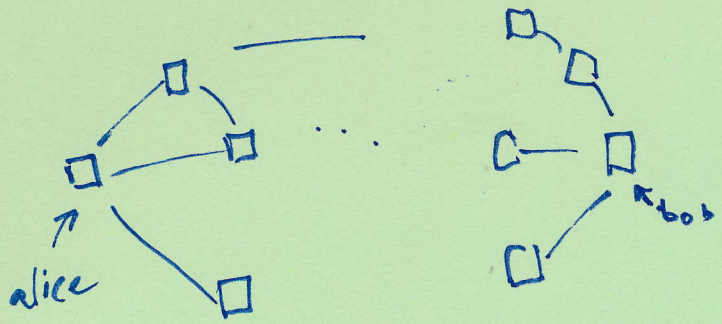
Distributed Networks



all our favourite questions:
- can alice send bob a message?
- what is the shortest route?
- colouring?

(- minimum cut, bipartiteness, etc.)

Model

Synchronised rounds.
Each round, each vertex independently:
- check inbox, read messages,
- perform unlimited computation,
- message neighbours

Each vertex has unlimited storage, a unique ID (known beforehand) and a list of neighbours (known beforehand)

Congest model: $O(\log n)$ bit limit

Colouring a path

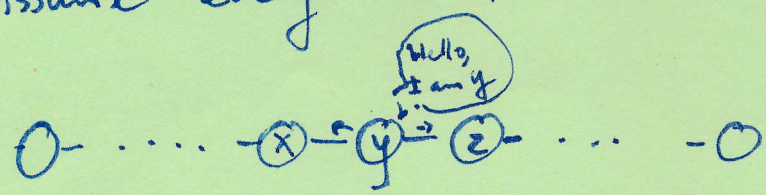
Reminder, colouring:

Each vertex is given a colour, $c : V \to \{1, \ldots, k\}$, such that each edge (v, u) is bichromatic, i.e. $c(v) \neq c(u)$

RAM-model colouring of a path: 1–2–1–2–1–2 ... trivial! But in the distributed model? Less trivial.

Colouring algorithm for 3-colouring a path

Assume every vertex has a unique identifier of $O(\log n)$ bits.



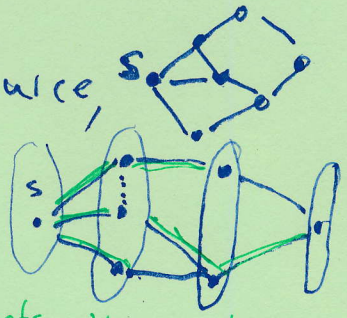
Round 1: send ID to neighbours.

All other rounds:

- read neighbour's IDs from inbox
- if (myID > max{ID of neighbours} and myID > 3)

  then: choose ID $\in \{1, 2, 3\}$.
- send ID to neighbours

Breadth-first search tree

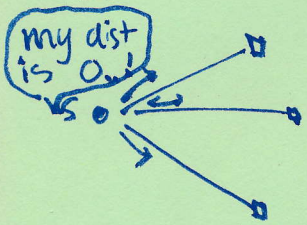


Reminder: Given a graph and a source, we may divide into layers of incr. dist. from s. BFS connects the next layer to the previous.

RAM-model: Starting with $L_0 = \{s\}$, find L_i as $\bigcup_{v \in L_{i-1}} \{\text{neighb. } v\}$. Link $u \in L_i$ to an arbitrary neighbour in L_{i-1}.

Wave algorithm

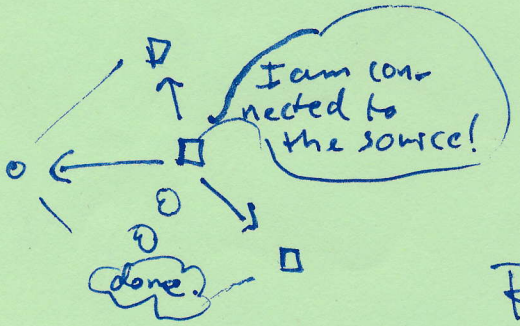
Given a source, s, who initiates the wave (everyone else passive)

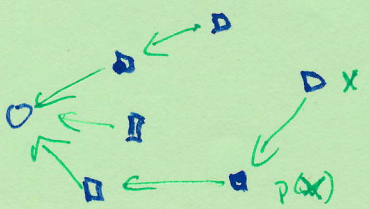


Round 1: s sends message to all its neighbs.
s switches state to "done".

All other rounds:
for each $v \in$ Network:
if v is not done:

- check messages.
- if there is at least 1 message let u be the neighbour who sent that message (choose arbitrary.) set $p(v) = u$. mark self as "done".
- send message to all neighbs.



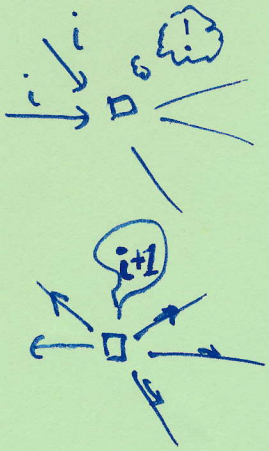


Result: every vertex con. to s can send s a message via a shortest path:
$v \to p(v) \to p(p(v)) \to \dots \to s$.

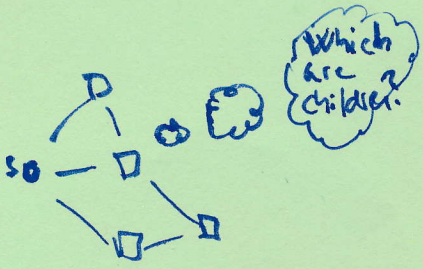
Wave with distances: small modification.

Round 1: s sends 0 to nbs.

Round > 1: let i be the message content received in the arbitrarily chosen msg. send "i+1" to neighbs.



Result: every vertex knows its distance to s.



Problem: Who are my children?
When are we all done?

Breadth-first search tree algorithm

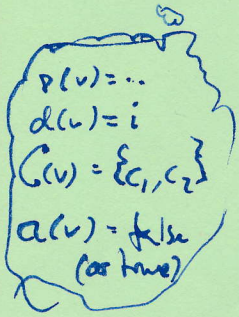
$v \in$ Network has extra information:

$d(v)$ distance to root.

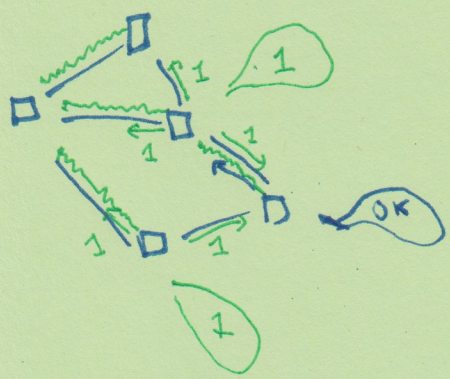
$p(v)$ parent in BFS-tree

$C(v) = \{c_1, c_2, \ldots\}$ children in BFS-tree

$a(v)$ are we done? - bit. Namely:
are all my children's subtrees done.



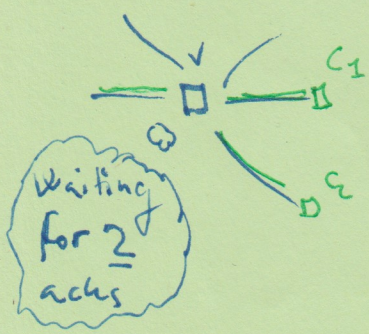
(BFS-tree algorithm, contnd)



Modification 1: Accepting a parent.

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

Result: u knows $v \in C(u)$.



Modification 2: Are we done yet-bits.

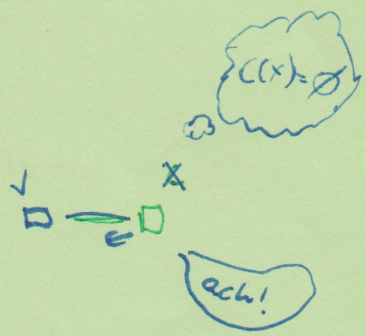
if $C(v)$ has been determined

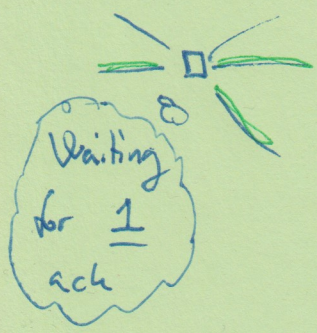
my $a(v)$ is the logical and

$$a(c_1) \wedge a(c_2) \wedge \ldots \quad \text{for } c_i \in C(v)$$

thus: keep booleans $\{a_1, a_2, \ldots\}$

- read messages containing "ack" of children
- update booleans.
- if now all have turned "true" send "ack" to $p(v)$.

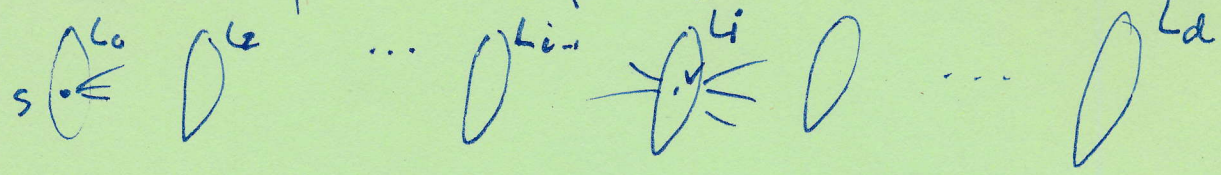




Result: s knows at same point that her message has reached every vertex.

Round complexity: Let z be a vertex of maximal distance to s. It takes $d(s,z)$ rounds to reach z. Then $O(1)$ rounds for z to realize $C(z)=\emptyset$. Then $d(s,z)$ rounds to send back an "ack". Total: $O(\text{diam}(G))$

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



for rounds 1, 2, ... $i-1$: nothing happens

round i: $d(v) := i$. $p(v) := u$, $u \in L_{i-1}$.

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

rounds $i+1$ to $x-1$: waiting to hear back from children.

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

further rounds: nothing happens.

Round analysis: Notice that x can be no larger than $d + 1 + (d - i)$.

Leader election: finding the minimum ID in G.

Round 1: everyone starts BFS from themselves.

Round > 1:
- check msg and update min. ID
- proceed as BFS from source minID
- augment messages with minID.

Result:
- for s with minim. ID in G, BFS terminates with ack from all children.
- for $v \neq s$, there is $c_i \in C(v)$ who never sends "ack". Thus, every vertex knows whether they lead

All pairs shortest paths

Reminder: We want $d(a,b)$ for every $a,b \in$ network.

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

rounds: $n \cdot O(\text{diam})$ where n = number of vertices. (or $O(n^2)$ in RAM)

idea: run "Wave with distance" in parallel.

challenge: message size $O(\log n)$ in congest model.

Token walk algorithm

Given a leader s and a BFS-tree from s.

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

Round > 1: • v checks messages. They contain either

- TOKEN

or

- dist-wave from some $x \in C_v$, where $d(x,v)$ is unknown

(or nothing/irrelevant)

1) case TOKEN:
if not done before, {
initiate Wave(v). }
set wait := true
~~set next := smallest i such that~~
set i to be the sender of TOKEN in $p(v), c_1, c_2, \dots, c_{|C(v)|}$

2) case Wave(x):
v proceeds with Wave(x).

3) case nothing/irrelevant:
if wait is true, then {
send TOKEN to number $i+1$ in $p(v), c_1, \dots, c_{|C(v)|}, p(v)$, and
set wait := false. }

Token walk analysis

(n is the numb. of vertices)

Round complexity: every tree-edge is traversed 2 times. There are $n-1$ tree-edges. There is one wait per move:

$O(n)$ token movements.

After the last token movement, it takes $O(\text{diam}(a))$ rounds to finish.

$O(n)$ rounds. $\Leftarrow$

Correctness:

We boldly claimed that vertex v received at most 1 distance-update in each round.

Let us prove this statement by contradiction: assume v receives the wave from both x and y for the first time in round i.

Assume Wave(x) was initiated before Wave(y), at times t_x and t_y, respectively.

Then, our distance to x is $i - t_x$ and to y is $i - t_y$.

But because the token is slow, $t_y - t_x = 2 \cdot \text{dist}(x, y)$.

$$i - t_x = \text{dist}(x, v) \le \text{dist}(y, x) + \text{dist}(y, v) = \text{dist}(y, x) + i - t_y$$

$$\Rightarrow t_y - t_x \le \text{dist}(x, y). \Rightarrow 2 \cdot \text{dist}(x, y) \le \text{dist}(x, y)$$

$\Rightarrow \text{dist}(x, y) = 0$ and thus $x = y$, a contradiction. □