Weekplan: Distributed Algorithms

Philip Bille Inge Li Gørtz Eva Rotenberg

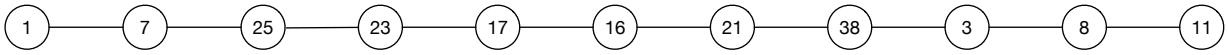
References and Reading

[1] Distributed Algorithms (section 1.1., 1.2, 1.3, 5.1, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7). By Jukka Suomela.

Exercises

1 Path 3-coloring

1.1 [w] Run the P3C algorithm on the following example:



1.2 [w] Give an instance where the algorithm PC3 runs in n rounds.

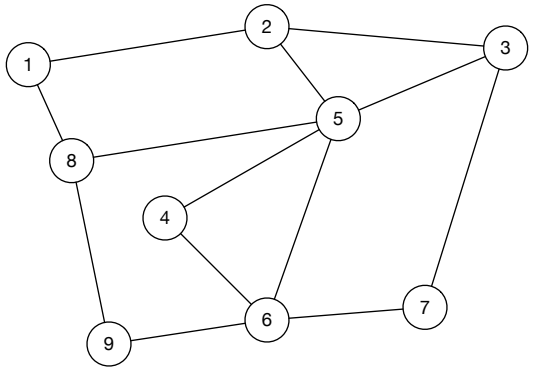
1.3 Rewrite the algorithm so nodes do not keep sending messages.

2 Maximal independent set (Ex. 1.1(a) from [1]) A *maximal independent* set is a set of nodes I that satisfies the following properties:

- for each node $v \in I$, none of its neighbors are in I,
- for each node $v \notin I$, at least one of its neighbors is in I.

Design a distributed algorithm that finds a maximal independent set in any path graph.

3 [w] **BFS tree** Run the BFS tree algorithm on the graph below. Let $t(v)$ be the round in which $a(v)$ was set to 1. For each node maintain $d(v)$, $C(v)$, $a(v)$, and $t(v)$. Indicate $p(v)$ by marking the edge from v to $p(v)$. Assume that a node u with $d(u) = \perp$ always accept the proposal from the node with the smallest identifier.



4 Edge counting (Ex. 6.2 from [1]) The *edge counting problem* is defined as follows: each node has to output the value $|E|$, i.e., it has to indicate how many edges there are in the graph. Assume that the input graph is connected. Design an algorithm that solves the edge counting problem in the CONGEST model in time $O(\text{diam(G)})$.

5 Detecting bipartite graphs (Ex. 6.3 from [1]) Assume that the input graph is connected. Design an algorithm that solves the following problem in the CONGEST model in time $O(\text{diam(G)})$:

- If the input graph is bipartite, all nodes output 1.
- Otherwise all nodes output 0.

6 MST Design an algorithm that computes the minimum spanning tree in $O(n \log n)$ rounds in the CONGEST model. *Hint:* Maintain a spanning forest and recursively merge connected components.

7 Faster 3-coloring on a path Assume the path that we want to 3-color is directed, in the sense that each node a predecessor and a successor (except the start and end that only has one of them). The nodes can still exchange messages in both directions. Consider the following algorithm: Initially we interpret the unique identifiers as colors. Let $c(u)$ be the color of u and let $c_s(u)$ be the color of u's successor (initially $c_s(u) = \perp$).

Algorithm 1: Reduce colors(u)

Send color to predecessor;
Receive message m from successor and set $c_s(u) = m$;
Set $i(u) =$ the index of the least significant bit where the binary representation of $c(u)$ and $c_s(u)$ differs;
Set $b(u) =$ the value of bit $i(u)$ in $c(u)$;
Set $c(u) = 2i(u) + b(u)$;

Let x be the number of bits needed to represent the colors before the algorithm is run. That is, the number colors is 2^x.

8.1 Show that after running Algorithm 1 the number of different colors is at most $2x$.

8.2 Show that after running Algorithm 1 the coloring is still a proper coloring.

8.3 Explain how to use Algorithm 1 and P3C to obtain an $O(\log^* n)$ algorithm for computing a 3 coloring of a path.