Self-Stabilizing spanning tree

1 Algorithm

Consider a graph where the nodes $\{p_0, \dots, p_{n-1}\}$ represents a set of n processes. Two nodes are connected if the corresponding processes can communicate. The node p_0 is a distinguished process, and is referred to as the root. Process p_i communicates with its neighbour p_j by writing to a shared register r_{ij} and reading from r_{ji}. We say that the process p_i owns the registers in which p_i writes. That is, p_i owns r_{ij} for all neighbours p_j of p_i. Each register r_{ij} comprises two fields. The field $r_{ij}.dis$ holds an integer value representing the distance from the root p_0 to p_i. This integer value is bounded by some large constant K, and any assignment of a larger value to the distance field results in the assignment of K. The field $r_{ij}.parent$ holds a binary value: if p_i considers that p_j is a parent, then $r_{ij} = 1$; otherwise $r_{ij} = 0$.

Finally, each process p_i holds the following local variables: for each neighbour m, the variable lr_{mi} is of the same type as r_{mi}, and is used to store values read from that register; an integer-valued variable *dist*; a boolean variable F.

It is assumed that each process p_i knows the list N_i of its neighbours, and that this list is ordered once and for all. Alg. 1 shows the pseudo-code of the spanning tree construction algorithm. Note that, the lines "for each $j \in N_i$" mean that we take the successive neighbours of p_i in the predefined order.

Algorithm 1: Self-stabilizing spanning tree construction

```
case Node p_0
    while true do
        foreach j ∈ N_0 do
            r_ij.(parent, dis) ← (0, 0)
        end
    end
case Node p_i ≠ p_0
    while true do
        foreach j ∈ N_i do
            lr_mi ← read(r_mi)
        end
        F ← false
        dist ← 1 + min{lr_mi.dis, m ∈ N_i}
        foreach j ∈ N_i do
            if F = false and lr_ji.dis = dist − 1 then
                r_ij.(parent, dis) ← (1, dist)
                F ← true
            else
                r_ij.(parent, dis) ← (0, dist)
        end
    end
```

Question 1. Consider a ring of $n = 4$ processes. Assume that, initially, all processes are in the same state, and the registers contain the same values. Show that, if p_0 executes the same code as the non-root processes, then it is possible to define an ordering of neighbours for each process, and an execution such that: (i) every process takes steps infinitely many times, (ii) the system does not converge. (*Hint: use the fact that opposite nodes "sees the same thing"*)