Home Assignment 2 - Project Groups 88 – Souptik Paul, Venkata Sai Dinesh Uddagiri

3 Self-stabilizing α-maximal-partitioning (due: September 29, 2022)

Consider a network, T := (P,E), whose topology is of a tree. The nodes in the network

are anonymous, i.e., they have no globally unique identifiers and they are run the same

algorithm. Let α >∆ be a positive integer, where ∆ is an upper bound on the node

degree. Let P1,P2,...,Pk be a partitioning of P, such that ∀x,y:x=yPx = Py, S

x Px = Pas well as each Px = ∅,|Px|≤α, and Tx := (Px,Ex) is a (connected) tree, where Sis thesetunion operator, |A| is the cardinality operator applied to the set A, x ∈{1,…,k}and Ex ⊆E∩(Px ×Px). In this case, we say that P1,P2,...,Pk is an α-partitioning of T.Suppose that there are no x,y ∈{1,...,k}: x = y, such that P1,P2,...,(Px ∪Py),…,Pkis also an α-partitioning of T. In this case, we say that P1,P2,...,Pk is an α-maximal-partitioning of T. Your task is to design a self-stabilizingalgorithm for constructing anα-maximal-partitioning of T.

# Please provide clear and exact pseudo-code for the entire protocol that you pro-

# pose. Please prove the correctness of your proposed solution. That is, show that when

# starting from any system state (configuration), the algorithm constructs an α-maximal-

# partitioning of T.

# Hint: start by assuming that the network includes a distinguished processor, i.e., the

# network is semi-anonymous. Then, use a self-stabilizing algorithm for finding the center

# of a tree, such as the one by Datta, Larmore, and Masuzawa [1]. If the center includes a

# single processor, the task is done. If the center includes two processors, each processor

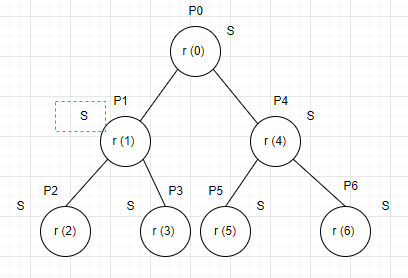
# is a the root of a tree. Solve the problem for each of these trees via a divide and conquer

# approach.

# For the sake of simpler solutions, you may assume that the tree is rooted by a dis-

# tinguished processor. Note that α refers to a bound on the number of elements in each

# partition rather than the number of partitions.



# Assumption:-

1. We assume that for each processor P(i) in the tree T, there exist a register r(i), which is a pointer register, that can store(or point) the address of another Processor. Also, for each P(i), there exist a shared register S, which is readable and writeable by all processors in the tree T.
2. We assume that the tree T is traversed using Pre-Order traversal(root->left child->right child).
3. In an given tree or diagram we have, a processor is considered as part of the α-partioning if, the register r(i) of the processor points towards the **root** node(for the root node to be a part of the α-partioning the r(i) register of the root node also has to point to itself !). We are assuming that the address of the **root** node, is known to all the processors (in terms of programming language, it is saved a global variable, **root**).
4. To solve this problem we are only considering trees with ∆ = 3, however the solution will work even for ∆=2, where ∆ is the upper bound on the node degree and α = 4. We can obtain the value ∆ using the self stabilizing node counting algorithm. Hence we are assuming the value of α is pre-defined for the pseudo code.

# Pseudo Code

1 do forever

2 For P(i), where i = 0 to n-1, in Pre-Order Traversal

3 S = (S + 1) mod n

4 if (S == 0 or S >α)

5 r(i) = Null

6 if (S <= α)

7 r(i) = P(0)

# Legal Executions:-

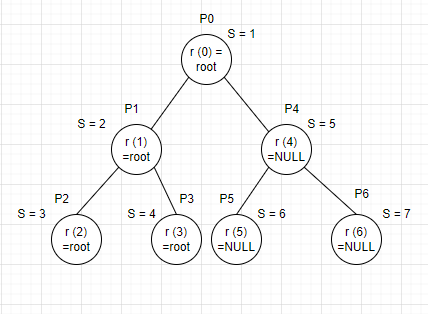
Let LE(MM) be the set of all legal executions in which the processors can access and update the register infinitely often.

The task MM can be defined as :-

1. Exactly one processor can change the value of its registers(state), in any one configuration.

2. Every processor can change the value of its registers(state) in infinitely many configurations in every sequence in ME.

3. After a certain set of configurations, the safe state should be achieved, where we can get the the maximal matching for the graph.



# Proof

Let us take a configuration every having only themselves in there one set

To prove the functioning of the algorithm, let us define the safe state of this α-partitioning algorithm. For n processors, c(n) configurations, there should be a subtree T’ of tree T, such that the number of nodes in T’= α. Then T’, is the α-partitioning of tree T.

Let us consider the Tree in the above diagram. In order for the algorithm and the proof to work, we are considering the tree T is being traversed using Pre-Order traversal(root->left child-> right child). Hence the traversal of the tree T will be in the order P(0), P(1), P(2), P(3), P(4), P(5), P(6). For the above tree the ∆ = 3 and α = 4.

To demonstrate the proof, of this algorithm, let us consider the functioning of this algorithm in a few configurations.

As shown in the diagram the, the execution starts at an arbitrary configuration c, at P(0), where S = 0. In this configuration c, the value of the S register is updated by the equation, S = (S + 1) mod n. In this configuration, the condition S ==0 or S >α is false. Then in configuration c, the step or action will be that the register r(0), of P(0) will point towards itself(root node).

Let c1 be the configuration, after the register of P (0) points towards itself. In this configuration c1, the value of the S register is updated by the equation S = (S + 1) mod n. In this configuration, the condition S == 0 or S >α is false. Then in configuration c1, the step or action will be that the register r(1), of P(1), will point toward P(0)(root node).

Let c2 be the configuration, after the register of P (1) points towards the root node. In this configuration c2, the value of the S register is updated by the equation S = (S + 1) mod n. In this configuration, the condition S = 0 or S >α is false. Then in configuration c2, the step or action will be that the register r(2), of P(2), will point toward P(0)(root node).

Let c3 be the configuration, after the register of P (2) points towards the root node. In this configuration c3, the value of the S register is updated by the equation S = (S + 1) mod n. In this configuration, the condition S == 0 or S >α is false. Then in configuration c3, the step or action will be that the register r (3), of P (3), will point toward P(0)(root node).

Let c4 be the configuration, after the register of P (3) points towards the root node. In this configuration c4, the value of the S register is updated by the equation S = (S + 1) mod n. In this configuration, the condition S ==00 or S >α is true. Then in configuration c4, the step or action will be that the register r (4), of P (4), will be NULL.

Let c5 be the configuration, after the register of P(4) points towards the root node. In this configuration c5, the value of the S register is updated by the equation S = (S + 1) mod n. In this configuration, the condition S =0 or S >α is true. Then in configuration c5, the step or action will be that the register r (5), of P (5), will be NULL.

Let c6 be the configuration, after the register of P (5) points towards the root node. In this configuration c6, the value of the S register is updated by the equation S = (S + 1) mod n. In this configuration, the condition S ==0or S >α is true. Then in configuration c6, the step or action will be that the register r (6), of P (6), will be NULL.

Let c7 be the configuration, after the register of P (6) points towards the root node.

Now, as we have defined earlier, the nodes in the tree T, point to the root node are part of the T’, which is the α-partitioning of tree T. In configuration c7, we see that the node {P(0),P(1),P(2),P(3)} point towards the root node. Hence we can say that, T’ = {P (0),P (1),P (2),P (3)}. Also we can see that, number of nodes in T’ = α. Hence we can say that T’ is the α-partitioning of tree T. Subsequently, we can also say that, c7 is a safe state and the time taken to reach state c7 is n.

Now, let us say that c7=c’. If we run the, second iteration of the loop with same set of steps for each of the processors from configurations c’ to c’(n). We will find that the c’(n) is also a safe state and c’(n)= c7. Hence we can say that the algorithm is self stabilizing.