EDA387: Computer Networks - Lab 2.2
Vertex Colouring

Group 5: Haitham Babbili and Olalekan Peter Adare

27th September 2020

Assumptions

- The number of processors is n, p_0 to p_{n-1}, and are all connected in a ring.
- The color numbers start from **0** to **n**.
- The execution start from processor p_0 to p_{n-1}. Also, at any given time, only a single processor is executing a computational step.i.e. it is an asynchronous shared memory system.
- Each processor has its register separated to two parts $l.r_{i-1}$ where can write the previous processor value and r_i where we can read it's value.

Algorithm

The given constraint in vertex coloring is that the adjacent vertices cannot have the same colour. The solution is based on a ring connection of the processors. It also depends on the number of the processors, **n**, which can be an even or odd, with a unique chromatic number (smallest number of the colours). For each processor having an maximum of 2 neighbour connections, the maximum number of colours needed is 3. This follows the rule of $\Delta + 1$, where Δ is the maximum number of neighbours each processor can have. If **n** is an even number, the chromatic number is 2, and will be 3 otherwise. Each processor has a register of the colours to be used. All processors are connected in a ring starting from root processor, p_0 , and every processor p_i has shared register memory r_i , where every processor read the value of the previous register r_{i-1} and then set its color to 0 if the previous color is 1 or 2, and 1 otherwise.

---

if $i = 0$ is the root processor **then**
    do forever
    $l.r_{n-1}$:= **read**(r_{n-1})
    if $l.r_{n-1} = 0$ **then**
        $color_0 := 1$
    else
        $color_0 := 2$
    end if
    write r_0 **:=** $color_0$
else$(i \neq 0)$:
    do forever
    $l.r_{i-1}$:= **read**(r_{i-1})
    if $l.r_{i-1} > 0$ **then**
        $color_i := 0$
        write r_i **:=** $color_i$
    else
        $color_i := 1$
        write r_i **:=** $color_i$
    end if
end if

Proofs

Lemma 1.1: N processors that form a ring connection requires a maximum of 3 colors for their corresponding graph.

Safe Configuration

Hypothetically, the color of a processor depends only on its state and that of its predecessor. We say that configuration **c** is safe with regard to a legal execution, if every fair execution of the algorithm that starts from **c** belongs to the legal execution.
In a ring network, the color of a processor depends only of its own state and that of its predecessor. The configuration c=$\{r_0$,r_1,r_2,...,$r_{n-1}\}$ in this algorithm will reach a safe configuration state when every two neighbour processors, p_i and p_{i-1}, do not have the same colour value in there register r_i and r_{i-1}. Also, for legal execution, we need to ensure that the $r_0 > 0$ while $r_i < 2$.

Convergence

Based on the assumption that we have **n** processors and that every processor can have in its own register the value of color numbers {0,1,2}. Also, there is the assumption that the root processor is p_0 and it would have written in its own register, r_0, color number 1 or 2 after reading the value of r_{n-1}, when it completes at least one (1) asynchronous cycle laying of the value of n if it's odd or even. Then, r_i to r_{n-1} values 0 or 1. This simply means that for every processors in $p_n = \{p_0, p_1, p_2, ..., p_{n-1}\}$. After **n** asynchronous cycle have colour values displayed in their register to be different from each other, i.e colour value in $r_i \neq r_{i-1}$ and the stats will be fixed.
To prove this, let us consider a unidirectional ring connection of $\{p_0, p_1, p_2, ..., p_{n-1}\}$ of size **n**, using a finite number of states. Also, taking that every node executes a uniform deterministic self-stabilizing vertex coloring algorithm. If we take **n**=5. After five asynchronous cycles, the processor p_0 will read r_4 and write in $l.r4$. If $l.r4 > 0$ and depending on assumption that the value of $r_0 > 0$ then it will be that r_0 =2 so **c** =$\{2, 0, 1, 0, 1\}$ and so the value of $r_0 \neq r_4$. Also, r_i will always have the same color number, except there is a new change in the number of processors,**n**.
If **n** = 6 and after six asynchronous cycles, the processor p_0 will read r_5 and write it in $l.r_5$ and then the value of r_0 =1. So **c** =$\{1, 0, 1, 0, 1, 0\}$ and so the value of $r_0 \neq r_5$. Also, r_i will always have the same color number, except there is a new change in the number of processors,**n**.
If **n** = k and after k asynchronous cycles, the processor p_k will have read the value of r_{k-1} and write it in $l.r_{k-1}$. If $l.r_{k-1} > 0$, then $r_k = 0$ and, if $l.r_{k-1} = 0$, then $r_k = 1$, so in both ways $r_k \neq r_{k-1}$. In any case, processor,r_i, will always have the same color number, self-stabilized, until there is a change in the number of processors. Therefore, we can say that starting from an arbitrary state, the system is guaranteed to eventually reach a stable state.

Closure

To define closure, let us build on the previous result that our system will converge to a safe configuration. Let us suppose that after one asynchronous cycle of the execution **E=E′ E′′**, where **E′′** is the suffix of **E** that follows **E′**. After one complete execution of **E′**, every processor executes at least one complete iteration and the next cycle of E will be **E′′**. If processor P_i want to execute configuration $c_i = \{c_0, c_1, ..., c_n\}$ so p_i will read from $l.r_{i-1}$ where ever P_i start to read and the value of $l.r_{i-1}$ will not change, beside that when P_i want to writ to r_i to reach the next configuration the value of r_i will not change also. That mean the next configuration also reached to safe configuration which mean whenever P_i arrive to safe configuration and all other **n** processors will be the same and

the system in self-stabilization and legal execution stage.
Consequently, once the system eventually reaches a stable state, it cannot become unstable again, provided that no fault occurs. It has thus self-stabilized.

Stabilization Period

We can bound the stabilization period, by first assuming a synchronous scenario. Therefore, to have a deterministic algorithm, we can create an asynchronous self-stabilizing algorithm. The synchronization of the system will eventually create self-stabilizing asynchronous algorithm. Building on our previous proofs that the system will end-up self-stabilizing, we can bound the stabilization period to a total of **n** asynchronous cycles. Hence, the time of **n** asynchronous cycle is stabilization period starting from the first asynchronous cycle.

Optimal Number of Colors

Lemma 1.2: The optimum number of colors is the maximum degree of connection of the processors, plus one.

$$\chi(G) \leq \Delta(G) + 1$$

Proof: By induction on the number of processors **n**, and the degree of each processor, P_i, let the maximum degree be represented as $\Delta(G)$. The chromatic number, $\chi(G)$, is such that $\chi(G) \leq \Delta(G) + 1$. In our ring network, the maximum degree for each processor is 2 and the upper bound on the number of colors is then 3, denoted as color numbers {0,1,2}. Intuitively, the algorithm will use 2 colors provided n mod 2 = 0, and will use 3, otherwise.For the ring connection, when **n** is even, optimal number of colors is 2, and when **n** is odd, the optimal number is 3. Also, this can be achieved in the order of **O(log n)** time.(Logarithm to base 2)

Number of States

Since we have found the chromatic number (smallest number of the colours) is 2, when **n** is even, and this is the same the optimal number of the colours and based on the assumption that every processor has 2 part in his register $l.r_{i-1}$ and r_i and each part can take 3 values $\{0, 1, 2\}$ so the number of states will be in the order of $3^2 = 9$ states for each processor.