Dt 30/Mar/2020

Assignment - 3

Name - Atul Kumar Agrawal
Branch - CSE - I
Roll No - 1602040031

Chapter - 14

Q 14.1 omission of any of the 3 requirements of Defn 14.3 (termination, agreement, non-triviality) for the consensus problem allows a very simple soln. Show this by presenting the 3 simple solns.

Soln:

→ If ~~only~~ termination & agreement are required :-

Each correct process immediately decides on 0.
(This algorithm is trivial because no 1 decision is possible).

→ If only termination & non-triviality are required :-

Each process immediately decides on its input.
(Agreement is not satisfied because different decisions can be taken)

→ If only agreement & non-triviality are required :-

Each process ~~immediately~~ decides should its input, then wait until $\lceil \frac{N+1}{2} \rceil$ messages with the same value have been received, & decides on this value.
(This algorithm doesn't decide in every ~~the~~ correct process if insufficiently many processes shout the same value.)

Q14.2 Give deterministic, 1-crash robust consensus protocols for each of the following restrictions on the input values.

(1) It is given that the parity of the input is even. (i.e. there are an even no. of processes with input 1) in each initial configuration.

(2) There are 2 (known) processes γ_1 & γ_2, & each initial config^n satisfies $x_{\gamma_1} = x_{\gamma_2}$.

(3) In each initial configuration, there are at least $\lceil N/2 + 1 \rceil$ processes with the same input.

Ans (1) The even parity implies that the entire input can be reconstructed if $N-1$ inputs are known.

Soln: Every process shouts its input & waits for the receipt of $N-1$ values (this includes the process' own input). The N^{th} value is chosen so as to make the overall parity even, the process decides on the most frequent input (0 if there is a draw).

(2) Soln → Processes γ_1 & γ_2 shout their input, A process decides on the 1st value received.

(3) Soln → Each process shouts its input, awaits the receipt of $N-1$ values, & decides on the most frequently received value.

Q14.3 Show that there is no 1-initially-dead-solution election algorithm for $t \geq N/2$.

Ans Two sets of processes S & T can be formed such that the processes in each group decide independently of the other group. If one group can reach a decision in which there is a leader, an execution can be constructed in which 2 leaders are elected. If one group can reach a decision in which no leader is elected, an execution can be constructed in which all processes are correct, but no leader is elected.

Q14.4 Show that no algorithm for ϵ-approximate agreement can tolerate $t \geq N/2$ crashes.

Ans Choose disjoint subsets S & T of $N-t$ processes & give all processes in S input 1 & the processes in T input $1+2\epsilon$. The processes in S can reach a decision on their own, because all inputs in this group is 1, so are the outputs. Indeed, the same sequence of steps is applicable if all processes have input = 1, in which case 1 is the only allowed output. Similarly, the processes in T decide on output $1+2\epsilon$ on their own, the decisions of S & T can be taken in the same run, contradicting the agreement requirement.

14.5 Give a bijection from the set

$\{(s,r) : N-t \le s \le N \ \& \ 1 \le r \le s\}$

to the integers in the range $[1 \ldots k]$.

Ans $f(s,r) = \frac{1}{2}(N-t+s-1)(s-(N-t)) + r$.

14.7 Adapt the proof of Theorem 14.1 for the case that G_T consists of k-connected components.

Ans Th 14.1 → there exists no non-trivial 1-crash-robust decision algorithm for a disconnected task T,

→ Call the components $G_1, \ldots, G_k$. After determining that the decision vector belongs to G_i, decide on the parity of i.

14.8 Consider the problem of $[k,\ell]$-election, which generalizes the usual election problem. The problem requires that all correct processes decide on either 0 ("defeated") or 1 ("elected"), & the no. of processes that decide 1 is b/w k & ℓ (inclusive).

(1) What are the uses of $[k,\ell]$-election?

(2) Demonstrate that no deterministic 1-crash robust algo. for $[k,k]$-election exists. ($k < n$)

(3) Give a deterministic t-crash robust algorithm for $[k, k+2t]$-election.

Ans (1) Sometimes a leader is elected to coordinate a centralized algorithm. Execution of a small no. of copies of this algo. (eg. in situations where election is not achievable) may still be considerably more efficient than execution by all processes.

(a) The decision vectors of $(0,k)$-election all have exactly k 1's, which implies that the table is disconnected (each node in G_T is isolated). The problem is non-trivial because a crashed process can't decide 1; therefore, in each execution the algorithm must decide on a vector whose 1's are located in correct processes. Consequently, the 14.1s applies.

(3) Every process shouts its identity, awaits the receipt of $N-t$ identities (including its own), & computes its rank r in the set of $N-t$ identities. If $r \leq k+t$ then it decides 1, otherwise it decides 0.

Q M.9 (I) Does the convergence requirement imply that the expected no. of steps is bounded?

(II) Is the expected no. of steps bounded in all algorithms of this section?

Ans (I) NO, it doesn't. The probability distribution on k defined by $Pr[k \geq x] = 1/k$ satisfies $\lim_{k\to\infty} Pr[k \geq k] = 0$, but yet $E[k]$ is unbounded.

(II) In all algorithms the probability distribution on the no. of rounds k is geometric, i.e. there is a constant $c < 1$ such that $Pr[(k > K)] \leq c \cdot Pr[k \geq k]$ which implies that $E[k] \leq (1-c)^{-1}$.

14.10 Show that if all correct processes start round k of the crash, robust consensus algorithm (Algo 14.3) then all correct processes will also finish round k.

Ans If all correct processes start round k, at least N-t processes start in that round, allowing every correct process to receive N-t messages & finish the round.

14.11 (1) Prove, that if more than (N+t)/2 processes start the crash-robust consensus algorithm (Algo 14.3) with input V, then a decision for V is taken in 3 rounds.

(2) Prove, that if more than (N-t)/2 processes start the algo, with $\exists l : v_l = V$, then a decision for V is possible.

(3) Is a decision for V possible if exactly (N-t)/2 processes start the algorithm with SIP V_i^2

(4) What are the bivalent SIP configurations of the algorithm?

Ans (1) In the 1st round, less than $\frac{N-t}{2}$ processes vote for a value other than V, thus implies that every (correct) process receives a majority of votes for V. Hence, in round 2 only V-votes are submitted, so in round 3, all processes witness a majority of V, & decide on V.

(2) Assuming the processes with SIP 'V' don't crash, more than $\frac{N-t}{2}$ processes vote for V, in 1st round.
→ It is possible that every (correct) process receives all V-votes, & less than $\frac{N+t}{2}$ votes for the other value.

then in round 2, only V-votes are submitted, implying that in round 3 all (correct) processes witness for V, & decide on V.

(3) Assuming the process with IP V doesn't crash, exactly $\frac{N-t}{2}$ processes vote for V in 1st round. In the most favourable (for V) case, all correct processes receive $\frac{N-t}{2}$ V-votes & $\frac{N-t}{2}$ votes for the other value as well. As seen from algo 14.3, the value '1' is adopted for the next round in case of a draw. Hence, a decision for V is made the if $v=1$.

(4) An initial configuration is bivalent if the no. of 1's, (#1), satisfies :-

$$\left(\frac{N-t}{2}\right) \leq \#1 \leq \frac{N+t}{2}$$

Q14.11 (1) Prove that, if more than $(N+t)/2$ processes start algo 14.5, with IP V, a V-decision is eventually taken,

(2) Prove that, if more than $(N+t)/2$ processes start algo 14.5, with IP V, & $t < N/5$, a V-decision is taken in 2 rounds.

Ans (1) In the 1st round, more than $\frac{(N+t)}{2}$ correct processes vote for V; $\Rightarrow$ every correct process accepts a majority of V-votes in the 1st round, & hence chooses V. As in proof of Lemma 14.3, it is shown that all correct processes choose V again in all later rounds, implying that a V-decision will be taken

(2) As in point (1), in the 1st round all correct processes choose v_d (the one) in 1st round, & vote for it in the 2nd round. Hence, in the 2nd round each correct process receives at least $N-2t$ v-votes; as $t < N/3 \Rightarrow N-2t > \frac{N+t}{2}$, so it follows that in 2nd round every correct process decides on v.

M.13 Prove that no asynchronous t-Byzantine resilient broadcast algorithm exists for $t \geq N/3$.

Ans Assume that such a protocol exists. Partition the process in 3 groups S, T & U; each of size $\leq t$, with G the general in S.

→ Let γ_1 be the initial configuration where the general has $IP = v$.

→ Because all processes in U can be faulty, $\gamma_0 \rightarrow SUT\,\delta_0$, where in δ_0 all processes in SUT have decided on 0. Similarly, $\gamma_1 \rightarrow SUU\,\delta_1$, where in δ_1 all processes in SUU have decided on 1.

→ Now, assume the processes in S are all faulty. & γ_0 is the initial configuration, first the processes in S cooperate with the processes in T such a way that all processes in SUT decide on 0, then the processes in S restore their state as in γ_1 & cooperate with the processes U in such a way that all the processes in SUU decide on 1. Now the correct processes in S & U have decided differently, which contradicts the statement/agreement.

Prove that during the execution of Algo 14.6 at most $N(3N+1)$ messages are sent by correct processes.

Ans At most N initial messages are sent by a correct process, namely by the general (if it is correct). Each correct process shouts at most one echo, counting for N messages in each correct process. Each correct process shouts at most 2 ready messages, counting for $2N$ messages in each correct process. The no. of correct processes can be as high as N, resulting in the $N \cdot (3N+1)$ bound.

(Adapting the algorithm so as to suppress the 2nd sending of ready messages by correct processes improves the complexity to $N \cdot (2N+1)$.)

Chapter - 15

Q15.1 How many messages are exchanged in the broadcast (N, t) protocol?

Ans Let the no = $M_t(N)$,

$M_0(N) = N - 1$.

$$M_t(N) = (N-1) + (N-1) \cdot M_{t-1}(N-1)$$

→ The 1st term is called initial transmission by the general, and term represents the N-1 recursive calls. The recursion is solved by

$$M_t = \sum_{i=1}^{t} \prod_{j=1}^{i+1} (N-j)$$

Q15.2 What is the highest no of messages sent by correct processes in Algo 15.4 in executions that decide on 0? Answer both for the case where the general is correct & the case where the general is faulty.

Ans → of decision = 0, while general is correct, the general's input is 0.

→ No correct process p initiates or supports any correct process, hence the messages sent by p are supporting a faulty process q; so p sends at most t times; which means sending (N-1).t messages.

→ If the general is faulty, at most t-1 = t correct processes can initiate without embodying a 1-decision, correct process p sends at most 1 <bm, 1> message, at most t <bm, 2> messages for faulty q, & at most t <bm, 0> message for correct q; this means sending at most (N-1)(2t+1) messages.

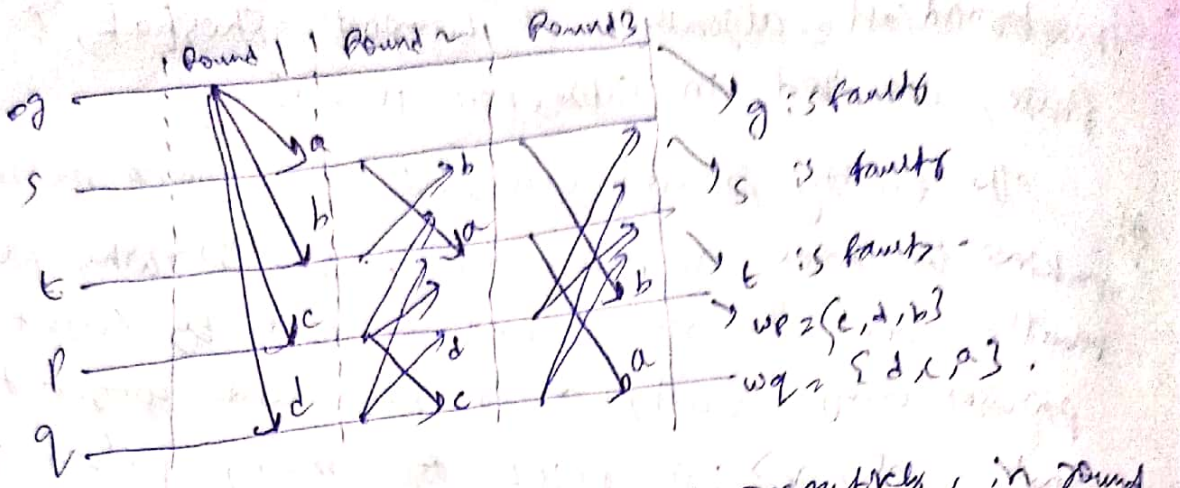
Q 15.3 How many messages are exchanged by the broadcast algorithm of Lamport, Shostak, & Pease, described in Subsection 15.2.1.2?

Ans The faulty processes can send any no. of messages, making the total no. unbounded; this is why we usually count the no. of messages sent by correct processes only; faulty g, P_1, P_2 could conspire to send $\langle val, v \rangle : g : P_1 : P_2$ for many values v to correct p to trap p into sending many messages. Therefore, a message that repeats another message's sequence of signatures is declared invalid.

→ then, a correct general sends at most $N-1$ messages in pulse 1, & for each sequence $g, P_1, P_2 \cdots P_i$ at most $N-i$ messages are sent in pulse i by a process (P_i). Thus, the no. of messages sent by correct processes is at most $\sum_{i=1}^{t+1} T_{j=1}^{i} (N \neq j)$.

Q 15.4 :- Give an execution of the protocol of Dolev & Strong, described in subsection 15.2.1, where correct processes p & q end with $W_p \neq W_q$.

Ans Let p & q be correct; the 1st & 2nd value accepted by p or q are forwarded, & also accepted by the other. So, we let them both receive a 3rd value from faulty processes, which is not received by the other.



→ p & q forward the values c, & d, respectively, in round 2 & that they forward in round 3 their locally received value, namely d, & c, respectively.

→ The 3rd received value (b & a, respectively) is not forwarded in the next round.

Q15.5 Show that order-preserving renaming in the range 1 through N can be solved when interactive consistency is achieved.

Ans The processes achieve interactive consistency on the vector of inputs; the name of p will be the rank of xp in this vector.

→ Byzantine processes may refuse to participate, but then their default values are assumed to have the highest ranks in the list. They may also cheat by submitting other values than their input, for eg, the name of a correct process. Then this name appears multiple in the list & the correct process decides on the smallest i for which (in the sorted vector) V[i] = xp.

→ Because all correct processes decide on the same vector & thus decision vector contains the xp of each correct process (by dependence), the decisions of correct processes are different & in the range 1 ... N.

Q 15.6 Assume that P signs 2 messages M_1 & M_2 with the Elgamal signature scheme, using the same value of a, show how to find P's secret key from the 2 signed messages.

Ans Bob as Z receives signed messages M_1 & M_2 from P i.e. the triples (M_1, γ_1, S_1) & (M_2, γ_2, S_2), where $\gamma_i = g^{a_i} \bmod p$ & $S_i = (M - d\gamma_i) a_i^{-1} \bmod (p-1)$.

→ Reuse of a, i.e. $a_1 = a_2$ is detected by observing that $\gamma_1 = \gamma_2$ in this case. Subtracting S_2 from S_1 eliminates the unknown term $d\gamma_i$ as $S_1 - S_2 = (M_1 - M_2) a^{-1}$

i.e. $a = (M_1 - M_2)(S_1 - S_2)^{-1} \pmod{p-1}$.

→ Next, we find $d = (M_1 - aS_1)\beta^{-1}$.

Q 15.7 Let n be the product of 2 large prime nos, assume that a black box is given that computes square roots. That is, given a quadratic residue y, the box outputs an x with $x^2 = y$ (eqn is modulo n). Show how the box can be used to factor n.

Ans In rings with zero-divisors, i.e. the axiom $a \cdot b = 0 \Rightarrow a = 0 \vee b = 0$ applies, a square is the square of at most two elements. Indeed, $x^2 = z^2$ reads $(x-z)(x+z) = 0$. Hence, by quoted axioms, $(x-z) = 0$ or $(x+z) = 0$.

i.e. $z = x$ or $z = -x$.

→ In particular, if p = prime, $\mathbb{Z}_p$ is free of zero-divisors, & $x^2 = y^2 \pmod p$ implies $y = \pm x \pmod p$.

→ Now, let $n = p \cdot q$ be a composit; the ring $\mathbb{Z}_n$ has zero-divisors because $a \cdot b = 0$ holds if p divides a & q divides b, or vice versa. Each square can be the square of upto 4 nos;

if $y = x^2$, then also $y = z_i^2$ for

z_1 s.t. $z_1 = x \pmod p$ & $z_1 = y \pmod q$
z_2 s.t. $z_2 = x \pmod p$ & $z_2 = -y \pmod q$
z_3 s.t. $z_3 = -x \pmod p$ & $z_3 = y \pmod q$
z_4 s.t. $z_4 = -x \pmod p$ & $z_4 = -y \pmod q$

→ The four no.s form 2 pairs of opposites: $z_1 = -z_4$ & $z_2 = -z_3$ (possessing z_i & z_j such that $z_i = \pm z_j$ is useless. However, if we possess z_i & z_j from different pairs, compute $a = z_i + z_j$ & $b = z_i - z_j$ & observe that neither a nor b equals 0, while $a \cdot b = 0$. Hence, each of a & b is divided by one of p & q, so $\gcd(a, n)$ is a non-trivial factor of n.

→ The square-root box can be used to produce such a pair, select a random no. x & use the box to produce a square root z of $y = x^2$.

→ Because x was selected randomly, there is a probability of $\frac{1}{2}$ that x & y are from different pairs, revealing the factors of n.

(the main work in modern factoring methods is spent in finding a non-trivial pair of no.s with the same sq.)

Q.15.8 Show that if clocks C_p & C_q have ρ-bounded drift & are δ-synchronized at real time t, then they are $\delta(1+\rho)$-synchronized at clock time $T = C_p(t)$.

Ans → The clock C_q has ρ-bounded drift, meaning that its advance in the real time from t_1 to t_2 (where $t_2 \geq t_1$) differs at most a factor of $1+\rho$ from $t_2 - t_1$ i.e.

$$(t_2 - t_1)(1+\rho)^{-1} \leq C_q(t_2) - C_q(t_1) \leq (t_2 - t_1)(1+\rho)$$

→ This implies that the amount of real time in which the clock advances from T_1 to T_2 (where $T_2 \geq T_1$) differs at most a factor $1+\rho$ from $T_2 - T_1$ i.e.

$$(T_2 - T_1)(1+\rho)^{-1} \leq C_q(T_2) - C_q(T_1) \leq (T_2 - T_1)(1+\rho)$$

→ Using only the 2nd inequality we find, for any T, T':

$$|C_q(T) - C_q(T')| \leq (1+\rho)|T - T'|$$

→ Consequently, for $T = C_p(t)$, we have :- (as $C_q(t) = t = C_q^{-1}(C_q(t))$)

$$|C_q(T) - C_p(T)| = |C_q(C_p(t)) - C_q(C_q^{-1}(t))|$$

$$\leq (1+\rho) \cdot |C_p(t) - C_q(t)| \quad \text{(by the bounded drift of } C_q \text{ as } C_p, C_q \text{ are } \delta\text{-sync. at real time } t\text{)}$$

$$\leq (1+\rho) \cdot \delta$$

Q 15.9 The probabilistic clock-synchronization protocol is efficient if many messages have a delay close to δ_{max} (i.e. $F(\delta_{max} + \epsilon)$ moves away from 0 quickly even for small ϵ).

→ Give a "dual" protocol that works efficiently if many messages have a delay close to δ_{max} (i.e. if $F(\delta_{max} - \epsilon)$ stays sufficiently far from 1 even for small ϵ).

→ Show that the expected message complexity is bounded by $2 \cdot [1 - F(\delta_{max} - \epsilon)]^{-2}$ & prove a bound on the expected running time.

Ans The process p asks the server for the time, the server replies immediately with a (time, T) message, & p measures the time I between sending the request & receiving the reply. As the delay of the (time, T) message was at least $I - \delta_{max}$ (this is because at most δ_{max} of the I time interval was used for the request message) & at most δ_{max}, it differs at most $\delta_{max} - I/2$ from $\frac{1}{2}I$. So, p sets its clock to $T + \frac{1}{2}I$, achieving a $\delta_{max} - I/2$ precision; if ϵ is smaller than $\delta_{max} - I/2$, the request is repeated.

→ The probability that the delay of the request (or response) exceeds $\delta_{max} - \epsilon$ is $1 - F(\delta_{max} - \epsilon)$, so the probability that both delays exceed $\delta_{max} - \epsilon$ is $[1 - F(\delta_{max} - \epsilon)]^2$. This implies that the expected no. of trials is at most $[1 - F(\delta_{max} - \epsilon)]^{-2}$.

→ Hence, the expected no. of messages is at most $2 \cdot [1 - F(\delta_{max} - \epsilon)]^{-2}$ & the expected time is at most $2\delta_{max}[1 - F(\delta_{max} - \epsilon)]^{-2}$.

8.15.10 Does the set AP in Algo 15.6 satisfy width (AP) $\leq \delta$ for all correct P?

Ans No, certainly not. Assume N-t processes submit the value x, & in addition t receives the values $x-\delta$ & $x+\delta$. Although at least one of these is erroneous (may differ by more than δ), neither of them is rejected because both have N-t submissions that differ δ from it. This lowerbounds width (AP) to 2δ.

Chapter 13

Q 13.1 Prove:-

Assume that:

(1) all terminal configurations belong to $\mathcal{L}$,

(2) $\exists$ a fun^{n} $f : C \to W$, where W is a well-founded set, & to each partition $\gamma \to \delta$, $f(\gamma) > f(\delta)$ or $\delta \in \mathcal{L}$ holds.

then $S = (C, \to)$ satisfies convergence

Ans Consider an execution $E = (\gamma_0, \gamma_1, \cdots)$ of S.

If E is finite, its last configuration is terminal & belongs to $\mathcal{L}$ by (1). Otherwise, consider the sequence $(f(\gamma_0), f(\gamma_1), \cdots)$; as there are no infinite decreasing sequences in W, there is an i such that $f(\gamma_i) > f(\gamma_{i+1})$ doesn't hold. By (2), $\gamma_{i+1} \in \mathcal{L}$.

Q 13.2 Prove that Dijkstra's token ring reaches a legitimate configuration in $O(N^2)$ steps. Shorten the analysis by giving a single norm function, quadratically bounded in N, that decreases with every step of the algorithm.

Ans In the initial state F is at most $\frac{1}{2}N(N+1)$ & every step of process 0 increases F by at most $N-1$ (if it gives privilege to process 1). So the no. of steps by processes other than 0 before the $(N+1)^{th}$ step of process 0 is bounded by $\frac{1}{2}N(N+1) + N(N-1)$ which brings the total no. of steps to at most $\frac{3}{2}N(N+1) - N$.

→ A suitable norm function is defined as follows [DHS+94 Sec. 8.2]. Call configuration γ single-segment if there exists $j \in N$ such that for all $i \leq j : \delta_i = \delta_0$ & $\forall i > j : \delta_i \neq \delta_0$.

Define,

$$A(\gamma) = \begin{cases} 0, & \text{if } \gamma \text{ is single segment} \\ \text{the smallest } i \text{ s.t. } \delta_0 \neq i \\ \text{doesn't occur in } \gamma, & \text{otherwise,} \end{cases}$$

& $W(\gamma)$ by $W(\gamma) = N \cdot A(\gamma) + F(\gamma)$. For each step, $\gamma \to \delta$, $[W(\gamma) > W(\delta) \; \forall \; \delta \in \Sigma]$ holds.

Q17-4 Modify the ring-orientation algorithm (Algo 17.1) so that it will terminate after establishing its postcondition.

Ans Modify action (2) so that the idle process p will receive the token only if p & q are oriented differently; if they are oriented the same way, the sending process becomes idle.

Actions	q	p	p
(1a)	→	← I	→ R
(1b)	← I	← S	→ ← I

→ Action (1a) is the same silent step as before, but although it will kill the token where originally it would be forwarded to a process that already has the same orientation. This ensures that token circulation ceases, yet we are still guaranteed to have tokens as long as the ring is not legitimate.

Q176 Show that the maximal-matching algorithm can be implemented uniformly in the link-register read-all model.

Ans → Replace the variable $pred_p$ by a 3-valued variable c_{pq} for each neighbour q, with value *you* if p has selected q, *other* if p has selected $r \neq q$, & *none* if p has not made a selection. These variables satisfy a local consistency:

$$lc(p) \equiv (\forall q \in Neigh_p : c_{pq} = none) \vee (\exists q \in Neigh_p : c_{pq} = you \wedge \forall r \neq q \in Neigh_p : c_{pr} = other).$$

→ An additional "clearing" action C_p, executed at most once by every process, establishes local consistency.

→ Qualifications below run over $Neigh_p$.

var c_{pq} : (you, other, none);

C_p : $\{ \neg lc(p) \}$

forall q do $c_{pq} := none$

M_p : $\{ lc(p) \wedge c_{pq} = none \wedge c_{qp} = you \}$

forall r do $c_{pr} := other$; $c_{pq} := you$

S_p : $\{ lc(p) \wedge c_{pq} = c_{qp} = none \wedge \forall r \neq q : c_{pr} \neq you \}$

forall r do $c_{pr} := other$; $c_{pq} := you$

W_p : $\{ lc(p) \wedge c_{pq} = you \wedge c_{qp} = other \}$

forall r do $c_{pr} := none$

→ Each process is locally consistent after almost one step, & the algorithm then behaves like the original one.

Q 12.2 Show that the maximal - matching algorithm stabilizes in $N^2 + O(N)$ steps & show that there is an $\Omega(N^2)$ lower bound on the (worst-case no. of steps).

Ans → For each matching step (action M_p) the value (+f+w decreases by at least 2, & the other steps decrease the value of the second component of F.

→ Hence, the function $F = (\lfloor (E+f+w)/2 \rfloor, 2c+f)$ is also a norm, & provides stabilization within $N^2 + \frac{1}{2}N$ steps.

→ To show the lower bound we inductively construct an execution of $\frac{1}{2}k^2$ steps on a clique containing k (even) free processes P_1 & P_2, first processes P_3, through P_{k-1} select P_k, then process P_k matches P_{k-1}, & finally processes P_1 & through P_{k-2} unchain from P_k. After these $2k-2$ steps, the NLW contains $k-2$ free processes P_3 through P_{k-2} unchain from P_k. After these $2k-2$ steps, the NLW contains $k-2$ free processes, so by induction the execution can be continued for another $\frac{1}{2}(k-2)^2$ steps.

Q 19.8 Design a stabilizing algorithm to construct a maximal independent set & compute the maximal no. of steps before stabilization -

Ans Assume the state read - all model A have a boolean var M_P for each P, indicating if P or it currently in the set M, there are 2 actions :-

(1) if both P & a neighbour are in M,
$m_P := false$;

(2) if neither P nor any neighbour is in M,
$m_P := true$.

→ Call P black if :- (1) applies, gray if :- (2) applies & white if neither applies.

→ Action (1) makes the black process white without making any other process black (a white neighbour could become gray) & action (2) makes a gray process white without darkening any other process.

→ So, the pair formed by the no. of black & gray processes decreases lexicographically with every step & after less than n^2 steps a terminal configuration is reached. In a terminal configuration, no 2 neighbours are in M, & every node not in M has a neighbour in M, hence M is a maximal independent set.

Q 19.10 Prove that outerplanar graphs are 3-colorable. Design a stabilizing algorithm that computes a 3-coloring of an outerplanar graph.

Ans An outerplanar graph has a node of degree 2 or less; So, replacing '6' by '3' & '5' by '2', we get the answer.

Q 12.12 Show that the update algorithm can be used for election & computation of a BFS spanning tree by giving an appropriate path-cost function.

Ans Let the cost of $\pi = \langle p_0, p_1 \ldots p_k \rangle$ be the pair (p_0, k). Ordering is lexicographically, but to get length increase, paths of length N or longer are longer than paths of length below N.

Q 12.13 Give a stabilizing algorithm for computing the network size.

Ans When a spanning tree is known, the size is computed by summation per subtree i.e. each node assigns its count the sum of the counts of its children plus one. Construction of a spanning tree without a priory knowledge of the network size was solved.

Q 12.14 Show how to compute the depth of a tree with the update algorithm.

Ans The update algorithm can compute the longest upward path for each node with the following choice of c, f, & the order. Let $cp = 0$, for each process p, & let $f_{pq}(x) = x+1$ if p is the father of q & $-\infty$ otherwise; the order is reversed, i.e., longer paths are smaller than shorter ones.