Ritik Mehra (IIT2019038)
DSS C1 Review Test

Ans 1. ~~Intsi~~

```
for (int i=1; i<n; i++)
    a[i] = b[i] - a[i-1]
```

now, lets divide into 4 threads, and take a[i] for a_0 values be zero.

$$a[i] = b[i] - a[i-1]$$

so, $a_1 = b_1 - a_0$

$a_2 = b_2 - a_1 = b_2 - b_1 + a_0$

$a_3 = b_3 - a_2 = b_3 - b_2 + b_1 - a_0$

$\vdots$

$a_n = b_n - b_{n-1} - b_{n-2} \cdots - b_1 + a_0$

so, if we calculate prefix sum using threads for b array we can calculate 'a' array using thread method. i.e. parallelization

Ans-2.

Given, number of sub problems = 10

number of threads = 10.

work load on each common thread = w

where, w = work ~~= total work~~

and given that 9 threads have same unit and last has 2, therefore ~~work~~

total work = 2w + 9w

= 11w

and work = $\frac{\text{total work}}{11}$

Therefore, last thread determines the actual speed and speedup because

work load on each common thread = $\frac{\text{total work}}{11}$

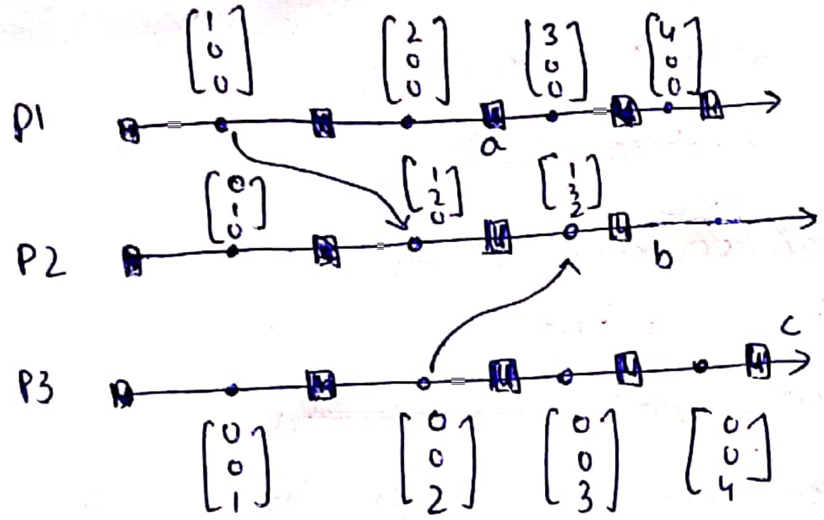
and

work load on last thread = $\frac{2.\text{total work}}{11}$

total speedup = $\dfrac{\text{total work}}{\frac{2\,\text{total work}}{11}} = \frac{11}{2} = 5.5$

Hence, max speedup depends on slowest thread which is 5.5 times speedup one single threaded.

Ans-3.

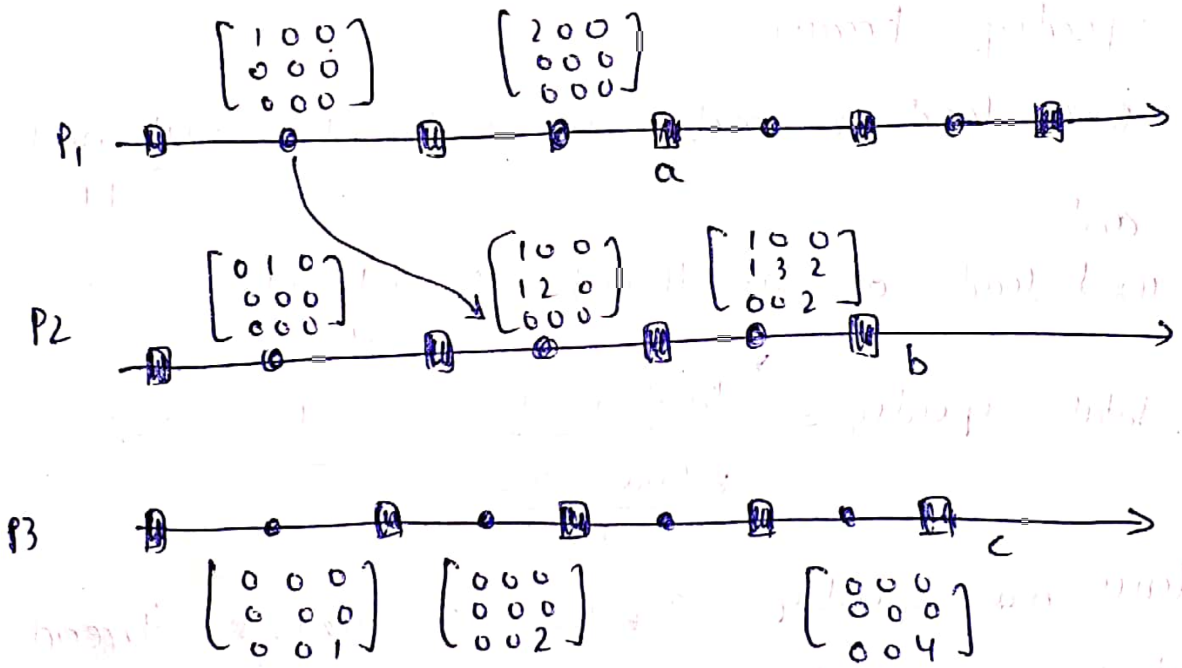


Therefore, $vts(a) = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$

$vts(b) = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$

$vts(c) = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix}$

(b)



Therefore, $wts(a) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$wts(b) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 3 & 2 \\ 6 & 0 & 2 \end{bmatrix}$

$wts(c) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{bmatrix}$

Ans 4. Singhal - kshemakalyani's differential is based on

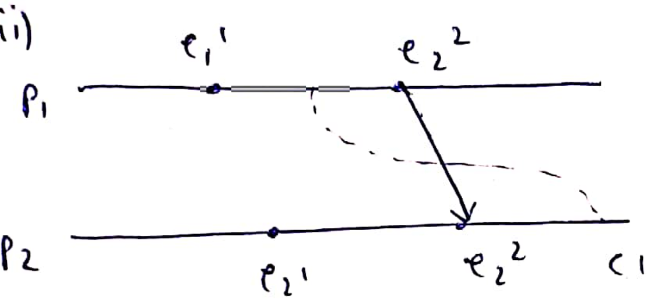
(i) FIFO communication channel -

for each process this method includes two other vectors → (i) last sent

(ii) last updated

The channel should deliver the first message before the next message otherwise last update will reflect the incorrect value, but last sent will receive the correct value which will produce store inconsistency.

(ii)



(inconsistent due to receive in the past).

No,

To disprove -: Here P1 and P2 are taken with two events each has two events.

e_1^2 is send

e_2^2 is receive

$t[e_1^2] = 2$

$t[e_2^2] = 2$

now, if we take a global cut such that e_1^1, e_2^1 and e_2^2 are in the past.

e_1^2 is in the future.

Therefore, $lts(G[1]) = 1$ and

$lts(G[2]) = 2$ and

But we, have already proven that the cut is inconsistent, because a message is received in the past.

Since this true for inconsistent stated by contradiction we have disproved the formal arguement.

Ans-5. This algo assumes that the clocks are initially synchronised and that they are resynchronised often enough so that no two non faulty process class differs by more than 'δ'. Let, $c_x[p]$ denotes the clocks i^{th} reading of clock k's value. Each clock i repeat execute these steps

Step 1: Read the value of every clock in the system

Step 2: Discard the outliers and also substitute the value by local clock, therefore if

$$|c_i[i] - c_j[i]| > \delta, \text{ then}$$

$$c_j[i] : c_i[i]$$

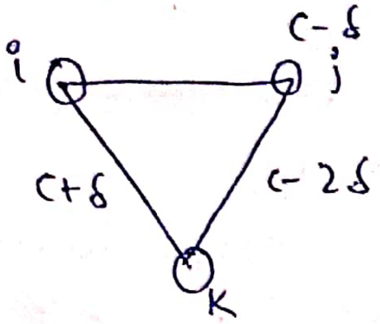
Step 3: update the clock reading using the average of these values.

This algorithm also guarantees that in a system of n process the clock remain in sync. even if there are at most 2 fault clocks when $n > 3t$. To verify this lets consider two distinct non-faulty clocks i & j reading a third clock k.

Two cases are →

① clock is non faulty then $C_k[i] = C_k[j]$

② Clock is faulty, then they can produce avg. reading, however a faulty clock can make other clock ~~&~~ accept their readings are ~~well~~ good.



The following ~~esn~~ values constitute a feasible set of readings accepted to every clock.

$C_i[i] = c$

$C_k[j] = c+\delta$

$C_j[i] = c-\delta$

$C_k[j] = c-2\delta$

now, assuming that at most out of the n clocks are faulty and remaining are non faulty.
The max difference found is $(c+\delta) - (c-2\delta) = 3\delta$.
Therefore, the max difference b/w the avg computed by any two non faulty clock is $\frac{3+\delta}{n}$.