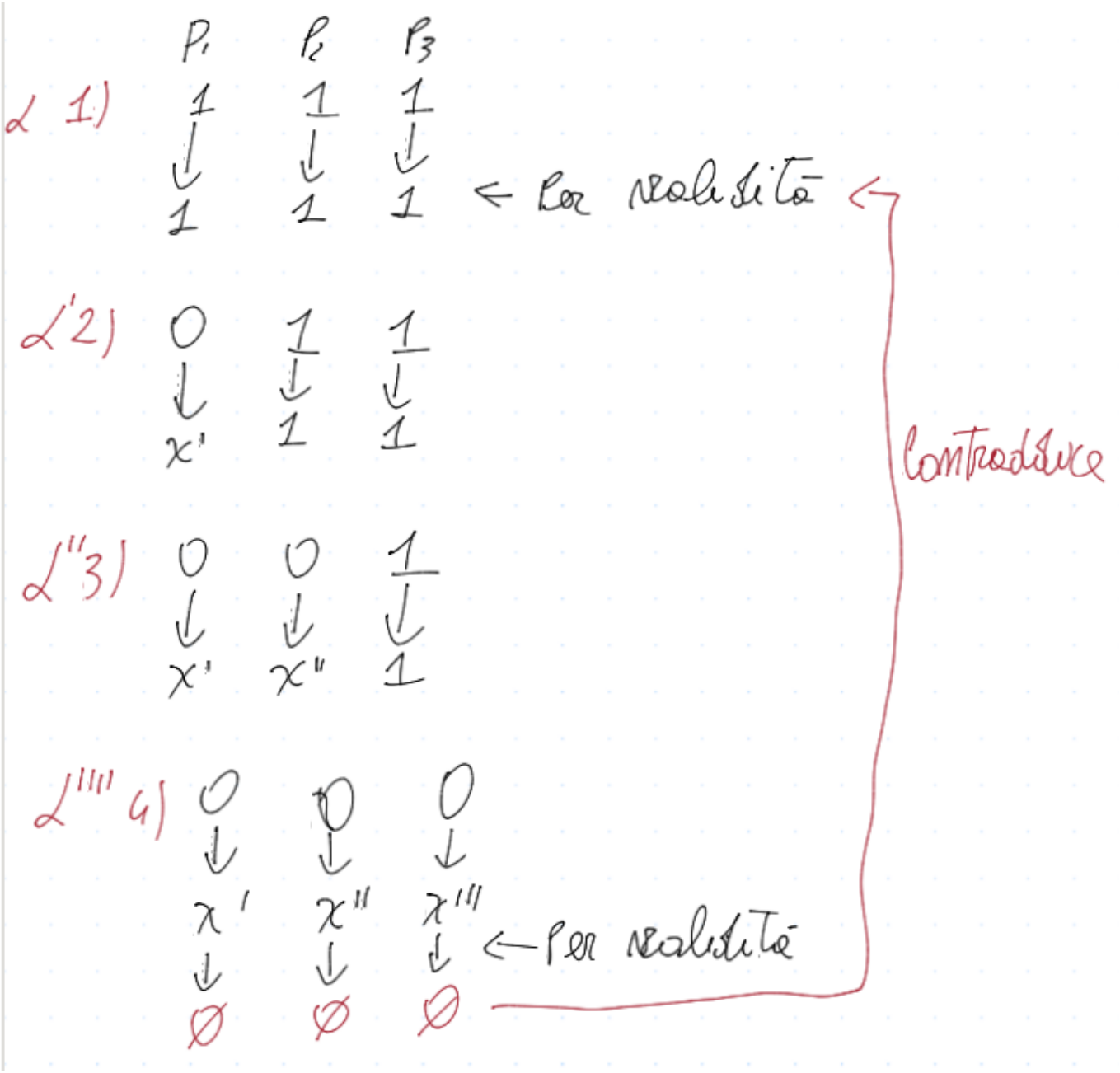
Esercizi6

Es. 1

No è impossibile.
Stessa dimostrazione di $AttaccoCoordinato$.
Togliere le comunicazioni fin quando nessuno comunica ma comunque si deve raggiungere il consenso. Partendo con l'assunzione $v_i = 1 \forall i$, allora modificando $v_i = 0 \forall i$ per la proprietà di validità dovranno scegliere tutti 0, ma in realtà decideranno tutti 1 portando ad una contraddizione. Ogni volta è stato cambiato solo un valore in input. Questo cambiamento, siccome non cioè scambio di informazione verrà notato solo da lui stesso, per gli altri nodi l'esecuzione sarà identica.



Da scrivere meglio

Assumere che esiste un algoritmo A che risolve il consenso.

Siano $\alpha_0, \dots, \alpha_r$ esecuzioni dell'algoritmo A in cui nella configurazione iniziale tutti i processori partono con un valore $v_i = 1$ e lo scambio dei messaggi è completo.

In ognuna delle $\alpha_0, \dots, \alpha_r$ esecuzioni rimuoviamo l'ultima comunicazione fino ad arrivare ad α_0 in cui non ci sono state comunicazioni ma comunque si è deciso 1.

Sia α' l'esecuzione in cui p_1 ha $v_1 = 0$.
in α'_0 per p_2, p_3 non ci sono stati cambiamenti quindi continueranno a scegliere 1.
Sia α'' l'esecuzione in cui p_2 ha $v_2 = 0$.
in α''_0 per p_3 non ci sono stati cambiamenti quindi continuerà a scegliere 1.
Sia α''' l'esecuzione in cui p_3 ha $v_3 = 0$.
Per validità tutti dovranno scegliere 0 ma questo va in contraddizione con il fatto che in α senza comunicazioni si è comunque scelto 1.

Es. 2

Fissiamo un numero di $round$ finito ≥ 1.

Algoritmo:

```
decisione_processore_i():
        Invio v_i per r round.

        Se non ho ricevuto niente, decido v_i
        Altrimenti
                Se v_i = v_j allora decido v_i
                Altrimenti
                        decido 0
```

Prova della correttezza (Dimostrazione delle 3 proprietà):

- Terminazione: Termina dopo r round con r finito ≥ 1
- Validità:
  - Se $v_1 = 1, v_2 = 1$:
    - Se non sono stati scambiati messaggi, entrambi decideranno $v_i = 1$
    - Se sono stati scambiati messaggi, siccome $v_1 = v_2 = 1$ allora decideranno entrambi 1
  - Se $v_1 = 0, v_2 = 0$:
    - Se non sono stati scambiati messaggi, entrambi decideranno $v_i = 0$
    - Se sono stati scambiati messaggi, siccome $v_1 = v_2 = 0$ allora decideranno entrambi 0
- Accordo: Con probabilità $(1-p)^r$ che non si scambino messaggi e con probabilità $\frac{1}{2}$ che i processori decidano valori diversi. La probabilità che non si raggiunga l'accordo è quindi $\frac{(1-p)^{2r}}{2}$.

Es. 3

Fissiamo un numero di $round$ finito ≥ 1.

Algoritmo:

```
decisione_processore_i():
        Invio v_i per r round.
```

```
        Se non ho ricevuto niente, decido v_i
        Altrimenti
                Se v_i = v_j allora decido v_i
                Altrimenti
                        decido 0
```

Prova della correttezza (Dimostrazione delle 3 proprietà):

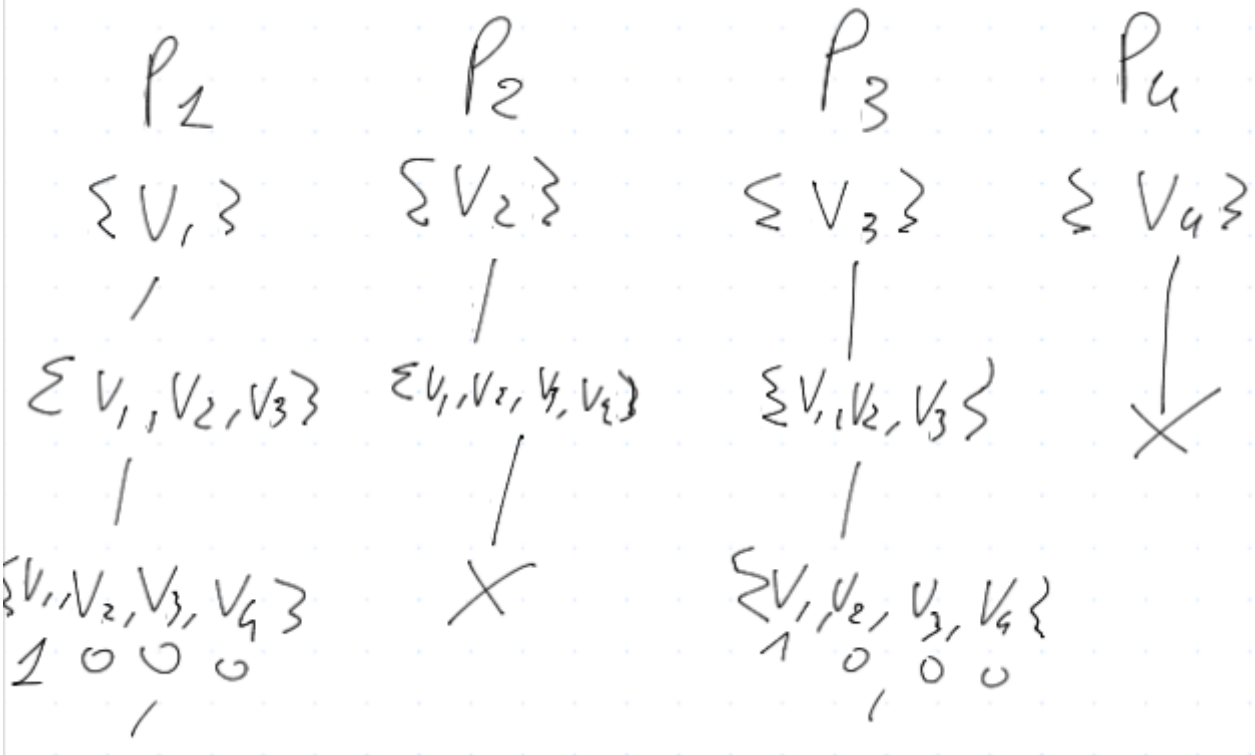
- Terminazione: Termina dopo r round con r finito ≥ 1
- Validità:
  - Se $v_1 = 1, v_2 = 1$:
    - Se non sono stati scambiati messaggi, entrambi decideranno $v_i = 1$
    - Se sono stati scambiati messaggi, siccome $v_1 = v_2 = 1$ allora decideranno entrambi 1
  - Se $v_1 = 0, v_2 = 0$:
    - Se non sono stati scambiati messaggi, entrambi decideranno $v_i = 0$
    - Se sono stati scambiati messaggi, siccome $v_1 = v_2 = 0$ allora decideranno entrambi 0
- Accordo: Con probabilità $((1 - q_1) + (1 - q_2))^r$ che non si scambino messaggi e con probabilità $\frac{1}{2}$ che i processori decidano valori diversi. La probabilità che non si raggiunga l'accordo è quindi $\frac{((1-q_1)+(1-q_2))^r}{2}$.

È uguale all'esercizio 2 ma ora ci sono due probabilità indipendenti che un processore riesca ad inviare un messaggio.

Es. 4

Algoritmo di $flood - set$

numero minimo di round $= f + 1$
Per come è descritto il problema $f = 2 \rightarrow$ numero minimo di round $= 3$
$\alpha_1 =$ Nel primo round si guasta P_4. Prima di guastarsi:
$P_4 \rightarrow P_2$
$\alpha_2 =$ Nel secondo round si guasta P_2. Prima di guastarsi:
$P_2 \rightarrow P_1 P_2 \rightarrow P_3$



A questo punto i processori P_1 e P_3 hanno abbastanza conoscenza per decidere e decideranno lo stesso valore seguendo la proprietà di accordo.

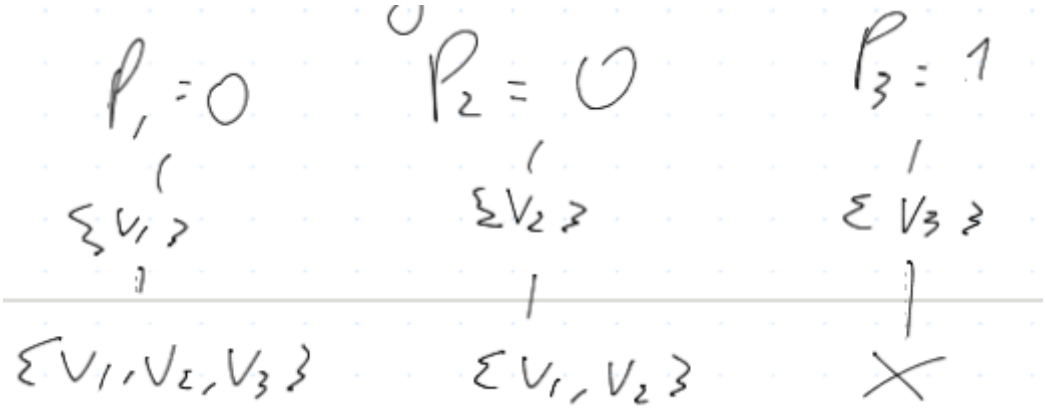
Es. 5

Con f round abbiamo che non si arriva ad un accordo, di seguito l'esempio.

Per l'esempio impostiamo $f = 1$ per semplicità e 3 processori.
$P_1 = 0, P_2 = 0, P_3 = 1$

Al primo round si guasta P_3 e prima di guastarsi $P_3 \rightarrow P_1$.

L'esecuzione si fermerà al primo round siccome
numero di round $= f$



In questo esempio P_2 non è un processore guasto ma non è in grado di decidere in quanto non ha conoscenza del valore v_3.
Quindi con f round non si arriva a consenso.

Se il numero di round è $f+2$ è ridondante in quando dopo $f+1$ round avremo almeno un round in cui tutti i nodi si sono scambiati informazioni.

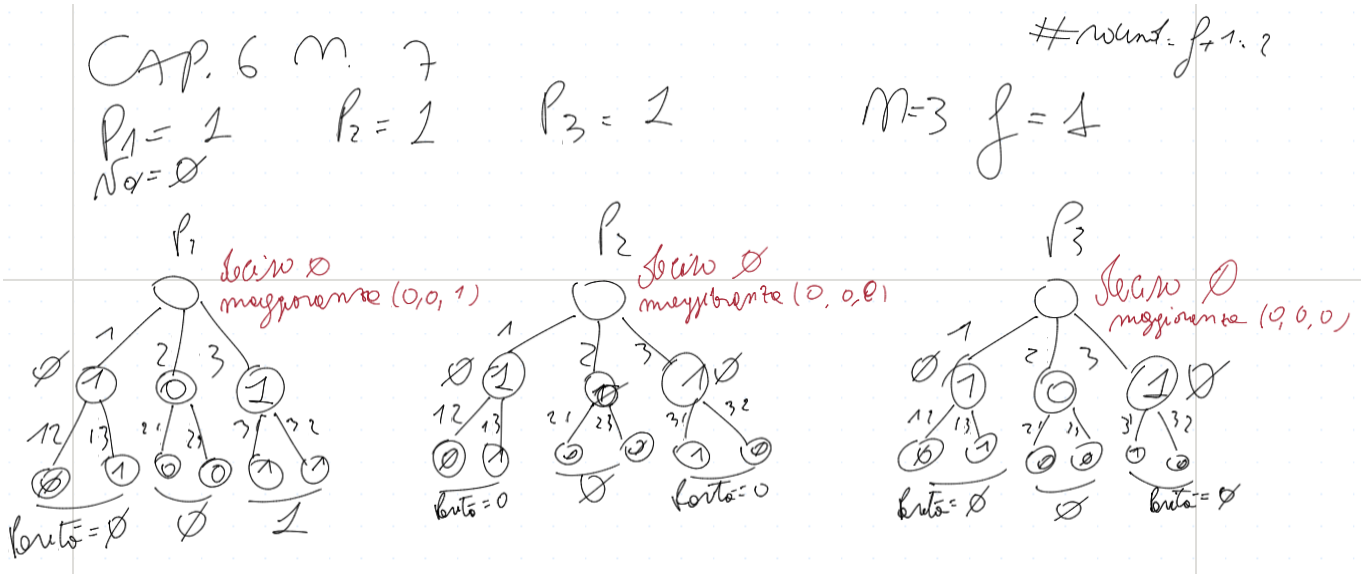
Es. 6

Terminazione: vengono effettuati $f+1$ round.

Validità: Tutti partono con $v_i = x$, il minimo sarà sempre x, tutti decidono x.

Accordo: Siccome esiste un minimo globale e dopo $f+1$ round tutti si saranno scambiati le informazioni, ogni nodo avrà ricevuto il minimo globale e lo deciderà seguendo la regola di decisione.

Es. 7



Questo è un esempio di esecuzione dell'algoritmo EIG_{BYZ} in cui tutti per validità devono decidere 1 mentre P_1 e P_2 (i nodi

onesti) sceglieranno 0 violando la validità.

Poi ricordiamo che $n > 3f$ dove n è il numero di nodi e f è il numero di guasti, quindi se abbiamo 3 nodi e 1 guasto EIG_{BYZ} non può risolvere niente.

Es. 10

Si usa l'algoritmo di FloodSet.
Bisogna tenere in considerazione una variabile ($flag$) che ci dice se in quel round si sono verificati dei guasti.
Se per due round di fila l processore non ha cambiato la sua conoscenza, $flag = TRUE$
Se per tutti i processori il $flag$ è $TRUE$,sono passati $f' + 1$ round e si può arrivare in consenso.

Es. 11

Si utilizza l'algoritmo EIG_{BYZ}.
Bisogna tenere in considerazione una variabile ($flag$) che ci dice se in quel round si sono verificati dei guasti.
Se per due round di fila il processore ha ricevuto kn messaggi vuoti dove k è il numero di messaggi vuoti ricevuti nel round precedente e n è il numero di nodi ancora funzionanti, $flag = TRUE$
Se per tutti i processori il $flag$ è $TRUE$,sono passati $f' + 1$ round e si può arrivare in consenso.

Es. 12

Semplicemente basta rifare l'algoritmo di $BenOr$, in cui se un processore arriva a decidere un valore v, allora invece di effettuare i 2 round dell'algoritmo, effettuerà un solo round in cui invierà ($first$,v,v). Questo è corretto secondo il lemma $6.5.5$ e per la proprietà di accordo.

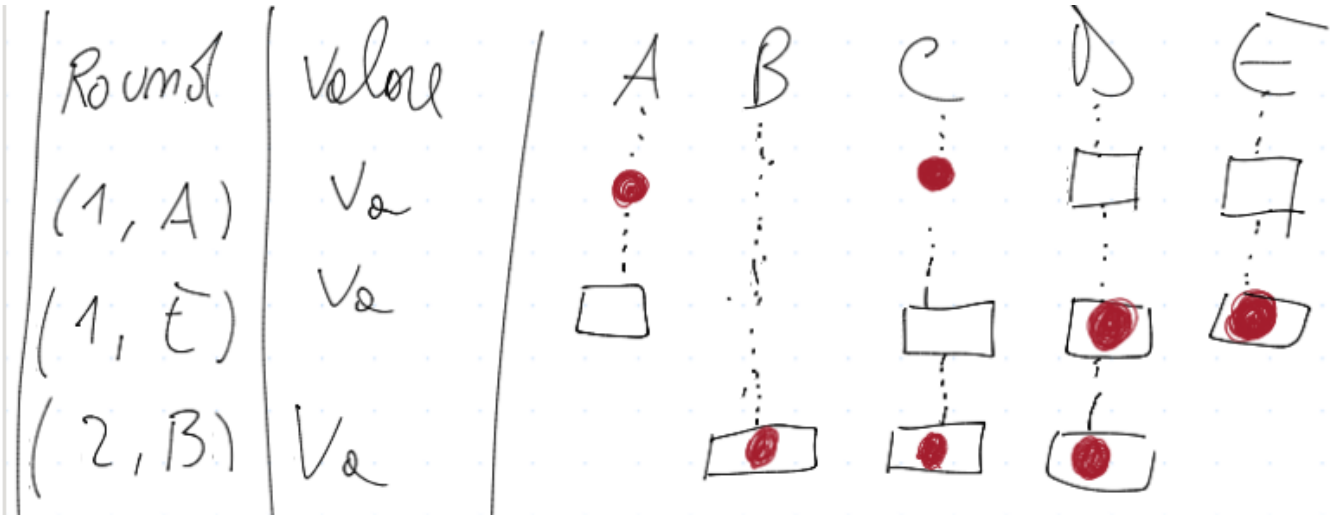
Es. 13

- Validità: Per la proprietà di validità se tutti i processori hanno un valore in input $x = v$ allora tutti i processori decideranno v.
  Nel primo round avremo che $n - f$ processori spediranno il valore v.
  Ogni processore riceverà almeno $n - f$ messaggi e per la prima condizione, siccome di questi messaggi f possono essere bizantini con $n - 2f$ avremo che tutti i processori riceveranno $n - 2f$ volte v.
- Accordo: Sia p_i il primo processore che decide nella fase s il valore v. Per definizione, siccome ha deciso, deve aver ricevuto almeno $n - 2f$ messaggi contenenti il valore v. A questo punto $\forall p_j, i \neq j$ che non ha ancora deciso, riceverà almeno $n - 2f$ messaggi contenenti v (Quelli che sono stati spediti anche a p_i) di cui f sono eventualmente bizantini meno eventuali f guasti.
  $n - 2f - f - f = n - 4f$.

Siccome $n - 2f \leq n - 3f \leq n - 4f$, avremo che il processore p_j nella fase s imposterà il valore $x = v$ e nei round successivi almeno $n - f$ processori invieranno il valore v nel messaggio $second$.

- Terminazione: Facciamo che i processori decidano della fase s il valore v. Significa che nella fase $s - 1$ avremo che tutti i processori nel messaggio second invieranno il v. Per inviare il valore v significa che i processori o hanno ricevuto il messaggio v da $n - 4f$ processori o che sia stato scelto a caso. Analizzando il caso peggiore, ovvero che tutti i processori decidano il valore v a caso avremo che questo si verifica con probabilità $\frac{1}{2^n} \cdot 2$. Quindi la probabilità che non si decide in un round è $(1 - \frac{1}{2^{n-1}})$ e per s round $(1 - \frac{1}{2^{n-1}})^s$.

Es. 14



Nel round $(1, A)$, A e C accettano in quanto non hanno nessun vincolo su eventuali altri round. Verrà deciso v_a in

quanto il leader è A.

Nel round $(1, E)$ avrà valore v_a in quanto tra i nodi che hanno risposto con un messaggio di $Last$ c'è C che è a conoscenza del valore del round precedente.

Nel round $(2, B)$ viene deciso v_a in quanto si raggiunge una maggioranza e c'è sempre C che è a conoscenza del valore conosciuto nel round precedente.