Relazione Consenso Bizantino

- **Il problema:**

Le scelte dei quattro processi sono memorizzate in una matrice di tre righe e quattro colonne di cui le righe rappresentano le quattro scelte che riceve ogni processo onesto (compresa la propria) e le colonne rappresentano le scelte di ogni singolo processo sottoforma di uni o zeri.

	processo1	processo2	processo3	processo4
processo1	0	1	0	1
processo2	0	1	0	0
processo3	0	1	0	1

La tabella sopra indica che il processo1 sceglie 0 e riceve 1 dal processo2, 0 dal processo3 e 1 dal processo4.

Il quarto processo è quello malevolo e sceglierà il contrario della scelta del processo al quale si riferisce la riga. Per esempio nella riga del processo1 il processo4 sceglierà il contrario della scelta del processo1, nella riga del processo2 sceglierà il contrario della scelta del processo2 e nella riga del processo3 il contrario di quello che ha scelto il processo3.

La condizione di accordo si otterrà nel momento in cui ogni processo onesto riceve almeno 3 bit uguali e il valore di questi bit deve essere lo stesso per ogni processo.

	processo1	processo2	processo3	processo4
processo1	1	1	1	0
processo2	1	1	1	0
processo3	1	1	1	0

La tabella sopra indica una condizione di accordo tra i processi onesti e quindi la fine dell'algoritmo.

Al turno 0 le scelte di ogni processo sono randomiche e può capitare, circa nell'1/4 dei casi, che i processi onesti si trovino già in una situazione di accordo. Questo porta alla risoluzione del problema in 0 turni.

Se così non fosse si giunge al turno 1 in cui ogni processo analizza i bit ricevuti, in base a quelli decide il bit da scegliere per quel turno, e lo comunica agli altri processi.

La scelta del bit da comunicare avviene in base al seguente algoritmo:

$maj(i) \leftarrow$ valore maggioritario tra i ricevuti (incluso il proprio)
$tally(i) \leftarrow$ numero dei valori uguali a $maj(i)$
`if` $tally(i) \geq 2t + 1$
    `then` $b(i) \leftarrow maj(i)$
`else if` $testa$
    `then` $b(i) \leftarrow 1$
`else` $b(i) \leftarrow 0$

Nel caso il cui tally non sia ≥ del doppio del numero dei processi maligni più uno si procede con il lancio della moneta globale della quale risultato viene comunicato a tutti i processi.

L'algoritmo viene iterato fino a quando non si raggiunge una condizione di accordo.

- **Il codice:**

Nel *main()* vengono create le strutture dati necessarie a immagazzinare le scelte dei vari processi e il numero di round necessari per raggiungere un accordo. Poi vi è un ciclo for che si occupa di iterare l'algoritmo un numero di volte definito nella costante ITER e di aggiungere al vettore dei risultati il numero di round utilizzati ad ogni run. L'algoritmo utilizza la funzione *inizio()* per inizializzare la matrice delle scelte con valori casuali e attraverso un while che pone la condizione *accordo() == false* dove la funzione *accordo()* si occupa di vedere se si è raggiunto un accordo esegue la funzione *round().*

La funzione *round()* calcola la tally e la maggioranza delle scelte ricevute da ogni processo attraverso le analoghe funzioni e in base all'algoritmo visto prima sceglie cosa mandare agli altri processi. Ad ogni esecuzione di *round()* viene incrementata la variabile *nRound* la quale indica per l'appunto il numero di round a cui siamo.

Usciti dal for si calcola la media, la varianza e il numero di round dopo il quale la probabilità di raggiungere un accordo è maggiore del 99.9%.

- **Considerazioni finali:**

Iterando codice per 10^5 volte si ha una media di 1.50123 round, una varianza di 2.25328 e che la probabilità di raggiungere un accordo è maggiore del 99.9% dopo 10 round.

Sappiamo che se n ≥ 3t + 1 dove n è il numero di processi e t il numero di processi inaffidabili il consenso può essere sempre raggiunto e in un numero di round dell'ordine di O(t + 1), che nel nostro caso è 2, quindi il valore che ci esce come media conferma la stima.

La probabilità di raggiungere un accordo è maggiore del 99.9% dopo 10 round perché ad ogni round con probabilità 1/2 dopo l'esito del lancio della prima moneta tutti i processi affidabili avranno raggiunto il consenso ed infatti 1-(0.5^10) = 0.999.