Filippo Manini Matricola: 4798004

Relazione Monte Carlo Byzantine General

TESTO:
Considera il caso di un sistema distribuito costituito da *n*=4 processi di cui il quarto è inaffidabile.
I tre processi affidabili seguono fedelmente il protocollo *MonteCarlo* mentre il processo inaffidabile, a ogni *round*, spedisce al processo affidabile *i* (con *i*=1,2 e 3) il bit 1-*b(i)*.
Implementa i tre processi affidabili e calcola media e varianza del numero di *round* necessari per raggiungere l'accordo.
Determina empiricamente il numero di *round* dopo il quale la probabilità che l'accordo è raggiunto è più grande del 99.9% e cerca di dare una spiegazione al risultato ottenuto.

ESECUZIONE:
g++ -std=c++17 MCByzantine.cpp

FUNZIONAMENTO:
inizializzo i processi (o generali) tramite la funzione *sceltaStart(…)* che mi permette tramite un parametro in input di scegliere se i processi affidabili partono allo stesso modo oppure no.

```
scelta se processi affidabili partono allo
stesso modo == [0] or [1]: 0
STAMPA DEI GENERALI
G1: 1 0 0 0
G2: 1 0 0 0
G3: 1 0 0 0
GX: 1 0 0 0
```

```
scelta se processi affidabili partono allo
stesso modo == [0] or [1]: 1
STAMPA DEI GENERALI
G1: 0 0 0 0
G2: 1 0 0 0
G3: 1 0 0 0
GX: 0 0 0 0
```

all'interno del ciclo while (che parte con bool loop = true) vado ad eseguire la parte di trasmissione tramite la funzione *trasmissione(…)* passi della funzione:

- Durante ogni round ogni processo spedisce un bit a ogni altro processo.
- Prima dell'inizio di un nuovo round ogni processo ha ricevuto un bit da ogni altro processo
- Nello stesso round ogni processo affidabile spedisce lo stesso bit a ogni altro processo
- il processo inaffidabile, a ogni round, spedisce al processo affidabile i (con i=1,2 e 3) il bit 1-b(i).

calcolo per ogni processo il maj(i): valore maggioritario tra ricevuti (incluso il proprio) tramite la funzione *valoreMaggioritario() "tenendo presente che il valore del processo inaffidabile non è corretto perché il valore che trasmette non è quello con cui viene inizializzato"*

```
dopo trasmissione
STAMPA DEI GENERALI
G1: 1 0 0 0
G2: 0 1 0 1
G3: 0 1 0 1
GX: 1 1 0 0
VALORI MAGGIORITARI
G1: 0
G2: 1
G3: 1
GX: 1
```

caso in cui i genarli affidabili non partono uguali

calcolo per ogni processo il tally(i): numero dei valori uguali a maj(i) tramite la funzione *tallyFunz(…)*

```
TALLY:
G1: 3
G2: 2
G3: 2
GX: 2
```

controllo per ogni processo se gTally >= (2*t +1) oppure l'lancio la moneta globale e salvo il risultato di quel processo tramite la funzione *checkData(…)*

infine controllo che tutti i genarli affidabili abbiano raggiunto il consenso tramite la funzione *checkEnd(…)* se questo è garantito esco dal ciclo while

dopo avere eseguito per 10^5 volte l'algoritmo mi calcolo
il valore medio semplicemente prendendo la somma dei #round e dividendo per il numero di esecuzione: *valoreMedio(…)*
varianza calcolata come: *varianza(…).*

$$\sigma_M^2 = \frac{1}{M-1}\sum_{m=1}^{M}(X_m - \langle X\rangle_M)^2$$

```
--------------------
MEDIA dei round: 2.47243
VARIANZA: 2.14632
--------------------
```

calcolo poi la frequenza con la quale l'algoritmo ottiene il consenso utilizzando una mappa con chiave: #round per il consenso e valore: quante volte accade quel determinato numero di round trami te la funzione *calcFreq(…)*

```
--------------------
frequenza del #round:
#ROUND  FREQUENZA
1       25200
2       38127
3       18529
4       9176
5       4389
6       2432
7       1128
8       523
9       248
10      149
11      49
12      30
13      20
--------------------
```

Calcolo empiricamente il numero di round dopo il quale la probabilità che l'accordo è raggiunto è più grande del 99.9%.
Mi calcolo la probabilità sottraendo a 100 quante volte compaiono valori maggiore della media diviso il numero di volte che ho eseguito MC.
poi incremento la media fino a quando la probabilità non supera il 99.9%, a questo punto stampo il numero di round.

```
--------------------
#round dopo il quale la probabilita' che l'accordo e' raggiunto
e' piu' grande del 99.9%
media #round: 2.47243  probabilita': 99.6333
media #round: 2.57243  probabilita': 99.6333
media #round: 2.67243  probabilita': 99.6333
media #round: 2.77243  probabilita': 99.6333
media #round: 2.87243  probabilita': 99.6333
media #round: 2.97243  probabilita': 99.6333
media #round: 3.07243  probabilita': 99.8186
media #round: 3.17243  probabilita': 99.8186
media #round: 3.27243  probabilita': 99.8186
media #round: 3.37243  probabilita': 99.8186
media #round: 3.47243  probabilita': 99.8186
media #round: 3.57243  probabilita': 99.8186
media #round: 3.67243  probabilita': 99.8186
media #round: 3.77243  probabilita': 99.8186
media #round: 3.87243  probabilita': 99.8186
media #round: 3.97243  probabilita': 99.8186
media #round: 4.07243  probabilita': 99.9103

#round finale: 4.07243
```

facendo test con, numero di volte che ho eseguito l'algoritmo da 10^3 a 10^6 ottengo sempre #round finale all'incirca 4

nel caso i processi affidabili partono allo stesso modo la media è sempre 1

```
#round dopo il quale la probabilita' che l'accordo
 e' raggiunto e' piu' grande del 99.9%
media #round: 1  probabilita': 100

#round finale: 1
```

Tutti i processi affidabili dal primo round raggiungono il consenso confermando il valore iniziale v_0, dove per ogni processo affidabile *i* risulta che *tally(i)* ≥ 2t + 1.

Dai dati raccolti possiamo dedurre che dopo all'incirca 4 round abbiamo quasi la certezza che i processi saranno d'accordo.