LT Informatica 2013/2014
Sistemi Operativi 1
Progetto: Torneo di Calcio
11/01/2015

Federico Francescon
fedefrancescon@aol.com
Matr: 146995

Football Tournament

Il software realizzato genera il calendario di un torneo di calcio e lo svolge giocandolo in modo virtuale.
Il calendario viene generato rispettando le modalità del girone all'italiana che comprende partite di andata e di ritorno.
Le informazioni necessarie sono fornite al programma in formato testo da un file di configurazione nel quale sono inserite le variabili: nome ed abilità delle squadre partecipanti.

Una volta generate, le partite vengono "giocate" od eseguite automaticamente dal programma
Ogni 30' di gioco virtuale, cioè tre volte a partita, le squadre hanno la possibilità di segnare un gol, la probabilità è funzione dell'abilità assegnata alla singola squadra ed è determinata dall'algoritmo seguente:

$$Goal = \frac{50}{100} \times \frac{[abilità]}{[abilità\ massima]}$$

Esempio:

$$\left.\begin{array}{r} abilità = 2 \\ abilità\ massima = 10 \end{array}\right\} \Rightarrow Goal = \frac{50}{100} \times \frac{2}{10} = 10\,\%$$

L'agoritmo fornisce il valore percentuale di probabilità della squadra di segnare un goal all'avversario che viene convalidato se questo è maggiore del valore random, compreso tra 1 e 100, generato dal programma.

Esempio:

$$\left.\begin{array}{r} Goal = 10\,\% \\ Random = 6 \end{array}\right\} \Rightarrow 6 < 10 \Rightarrow \texttt{GOAL!}$$

$$\left.\begin{array}{r} Goal = 10\,\% \\ Random = 77 \end{array}\right\} \Rightarrow 77 > 10 \Rightarrow \texttt{Non è goal}$$

Alla fine di ogni partita, in funzione del risultato ottenuto, alle squadre sono attribuiti i punteggi seguenti:

- **Vittoria**: **3** punti
- **Parità**: **1** punto
- **Sconfitta**: **0** punti

Generazione Calendario

Il Calendario delle partite è generato in modo che ogni squadra affronti tutte le avversarie, ma sia impegnata solo in una partita al giorno.
Per ottimizzare il calendario si inizia assegnando gli incontri alla prima squadra in modo che ogni giorno incontri una squadra diversa (la squadra 1 giocherà contro la squadra 2 nella prima giornata, contro la squadra 3 nella seconda ecc.)
Gli incontri delle altre squadre sono assegnati iniziando dal giorno ***n*** in cui ***n*** corrisponde all'indice della squadra avversaria; se una delle due squadre è impegnata nella stessa giornata, si posticipa l'incontro alla giornata successiva.
Questo controllo viene ripetuto sino ad identificare una giornata libera per entrambe le squadre; raggiunto l'ultimo giorno si riprende il processo iniziando dalla prima giornata.

Esempio:

$$S2 - S3 \Rightarrow Giorno\,3 \Rightarrow \text{Squadre libere? Sì} \Rightarrow \text{Giorno 3 confermato}$$

$$S3 - S4 \Rightarrow Giorno\,4 \Rightarrow \text{Squadre libere? No} \Rightarrow \text{Giorno 5}$$
$$\text{Squadre libere? Sì} \Rightarrow \text{Giorno 5 confermato}$$

$$S4 - S5 \Rightarrow Giorno\,5 \Rightarrow \text{Squadre libere? No} \Rightarrow \text{Giorno 1}$$
$$\text{Squadre libere? No} \Rightarrow \text{Giorno 2}$$
$$\text{Squadre libere? Sì} \Rightarrow \text{Giorno 2 confermato}$$

Il numero di giorni e di partite viene calcolato secondo le seguenti formule:

$$[n_{squadre}]\ PARI:$$
$$\text{Num Giorni} = [n_{squadre}] - 1 \qquad \text{Num Partite} = \frac{[n_{squadre}] \times [n_{squadre} - 1]}{2}$$

$$[n_{squadre}]\ DISPARI:$$
$$\text{Num Giorni} = [n_{squadre}] \qquad \frac{\text{Partite}}{\text{Giorno}} = \frac{\text{Num Partite}}{\text{Num Giorni}} = \frac{[n_{squadre}]}{2}$$

Esempio:

$$[n_{squadre}] = 8$$
$$\text{Num Giorni} = 8 - 1 = 7$$
$$\text{Num Partite} = \frac{8 \times 7}{2} = 28$$
$$\frac{\text{Partite}}{\text{Giorno}} = \frac{28}{7} = 4$$

$$[n_{squadre}] = 7$$
$$\text{Num Giorni} = 7$$
$$\text{Num Partite} = \frac{7 \times 6}{2} = 21$$
$$\frac{\text{Partite}}{\text{Giorno}} = \frac{21}{7} = 3\,\text{newline}$$

N.B: Tutti i valori ottenuti vanno moltiplicati per due per considerare i ritorni.

Assegnazione dei giorni

TEAMS	T1	T2	T3	T4	T5	T6
T1		Giorno 1	Giorno 2	Giorno 3	Giorno 4	Giorno 5
T2	Giorno 1		Giorno 3	Giorno 4	Giorno 5	Giorno 2
T3	Giorno 2	Giorno 3		Giorno 5	Giorno 1	Giorno 4
T4	Giorno 3	Giorno 4	Giorno 5		Giorno 2	Giorno 1
T5	Giorno 4	Giorno 5	Giorno 1	Giorno 2		Giorno 3
T6	Giorno 5	Giorno 2	Giorno 4	Giorno 1	Giorno 3	

Giornate di ritorno

Il calendario delle giornate di anadata e ritorno è generato contemporaneamente in quanto la differenza tra le partite di andata e di ritorno è l'inversione dell'ordine di gioco delle squadre:

- Andata: Squadra 1 = 'Juventus' Squadra 2 = 'Milan'
- Ritorno: Squadra 1 = 'Milan' Squadra 2 = 'Juventus'

Durante la generazione del calendario le giornate di ritorno inserite momentaneamente in una lista separata che viene accodata solo dopo aver concluso tutte le partite di andata.
L'assegnazione delle partite ai rispettivi giorni è la fase computazionalmente più complessa e dispendiosa in termini di risorse (CPU) e tempo.

Interattività

Il programma supporta una modalità interattiva in cui la prima squadra indicata nel file di configurazione viene gestita dall'utente a richiesta.
L'utente può scegliere se tirare('Shoot') o bloccare ('Block') il tiro, qualora si verifichi per la squadra controllata uno dei seguenti eventi:

- **Subisce un attacco avversario**: l'utente ha la possibilità di parare il goal (Block)
- **Azione d'attacco all'avversario**: l'utente ha la possibilità di segnare un goal (Shoot)

Azioni predefinite

L'utente ha due secondi di tempo per selezionare l'azione desiderata. In caso di timeout il computer sceglie automaticamente un'impostazione di default:

- **Attacco**: Impostazione predefinita: '**S**'
- **Difesa**: Impostazione predefinita: '**B**'

ATTACCO	`ATTACK: '[userTeam]' >> '[oppositeTeam]'. (2sec) Your action: (S)hoot or (B)lock?`
	Premendo '**S**' o '**s**' la squadra [*userteam*] segna un goal.
DIFESA	`DEFEND: '[userteam]' << '[oppositeTeam]'. (2sec) Your action: (S)hoot or (B)lock?`
	Premendo '**B**' o '**b**' paro il goal della squadra [*oppositeTeam*].

Multithreading – Singlethreading

Per impostazione predefinita, come da specifiche del progetto, ogni giornata viene eseguita in un thread separato e per ognuna l'esecuzione delle partite è demandata, ad un ulteriore thread figlio.
Considerata l'indipendenza delle singole partite, si possono eseguire più giornate in parallelo, dividendole su ***n*** threads paralleli.
I thread condividono la stessa lista dei giorni e per ogni thread si procede come segue:

1. Blocco del mutex associato alla lista di giorni
2. Lettura del giorno da elaborare
3. Modifica del puntatore a lista di giorni per puntare al giorno successivo
4. Sblocco del mutex associato alla lista dei giorni
5. Ripeto dal punto [**1**] finché il puntatore al giorno corrente diventa NULL

In questo modo ogni thread processa indipendentemente ogni singola giornata, con le relative partite.
Per testare la crorrettezza dell' implementazione, sono state aggiunte le seguenti modalità di test che permettono di verificare la corretta esecuzione delle partite con il multithreading abilitato:

- **Max Teams Off**: disabilita il numero massimo e minimo di squadre configurabili(4 - 8).
- **Single Thread Mode**: multithreading disabilitato, le partite vengono eseguite nel thead principale.
- **Test Mode**: tutte le partite terminano 1-0, in questo modo si possono lanciare, con lo stesso file di configurazione, più esecuzioni di test del programma essendo certi che la classifica finale è sempre la stessa avendo tutte le squadre con lo stesso punteggio.
  L'ordine nella ranking list rispecchia quello del file di configurazione.
  Si stampano delle informazioni aggiuntive: variabili di configurazione, lista delle squadre importate e dei match generati(informazioni NON salvate sul file di output, solo stampate a video).

Performance ed osservazioni

Analizzando le performance del programma è emersa che **le prestazioni peggiorano con il multithreading**.
Probabilmente questo è imputabile alla maggior complessità del codice e all'overhead necessario allla

creazione dei threads figli. L'esecuzione delle partite è un'operazione molto semplice e veloce, suddividerla in più threads non fornisce alcun vantaggio.
La generazione dei match potrebbe essere più efficiente se eseguita su più thread, tuttavia non è stato identificato un algoritmo che non richieda sequenzialità per l'assegnazione di giorni e partite.
Un'analisi più approfondita richiederebbe un maggior numero di test in un ambiente dedicato, ma questi dati sono sufficienti per delle osservazioni qualitative riassunte nella tabella seguente:

			Teams n° **10**	Teams n° **50**	Teams n° **100**	Teams n° **200**	Teams n° **400**
N° of Threads	Single	1	3307823ns	74573946ns	573351110ns	22s 604823935ns	9m 51s 358649094ns
		2	1931181ns	64332349ns	585155615ns	25s 101768594ns	10m 8s 151012429ns
		3	2554026ns	67308635ns	593075897ns	23s 69179401ns	10m 7s 932022758ns
		avg	**2'597'678 ns**	**68'738'310 ns**	**583'860'874 ns**	**23s 591'923'977 ns**	**10m 2s 480'561'427 ns**
	2	1	4812344ns	71988073ns	617581808ns	25s 90207626ns	10m 7s 932022758ns
		2	2598376ns	70447303ns	589370537ns	25s 382179630ns	10m 6s 246129850ns
		3	3425157ns	71982415ns	631950257ns	23s 126108223ns	10m 12s 210895869ns
		avg	**3'611'959 ns**	**71'472'597 ns**	**612'967'534 ns**	**24s 532'831'826 ns**	**10m 8s 796'349'492 ns**
	5	1	2464071ns	88081741ns	592747992ns	23s 112230689ns	10m 12s 413108824ns
		2	2707188ns	79653294ns	622273538ns	24s 572823764ns	10m 2s 604152599ns
		3	29940402ns	92021418ns	683115847ns	24s 771269759ns	10m 47s 411506115ns
		avg	**11'703'887 ns**	**86'585'484 ns**	**632'712'459 ns**	**24s 152'108'071 ns**	**10m 20s 809'589'179 ns**
	10	1	21175809ns	68852967ns	735047526ns	24s 159624879ns	10m 45s 476335674ns
		2	10615038ns	88800756ns	598883340ns	25s 185773419ns	11m 13s 837692074ns
		3	4441068ns	86447524ns	628052275ns	23s 682786332ns	11m 5s 709908671ns
		avg	**12'077'305 ns**	**81'367'082 ns**	**653'994'380 ns**	**24s 34'272'821 ns**	**11m 1s 674'645'473 ns**

Note finali

Liste

È stata utilizzata una semplice e personale implementazione di liste doppiamente concatenate, in quanto risulta molto comoda e veloce, tenendo presente la possibilità di notevoli miglioramenti applicabili.

Requisiti

Per poter essere compilato, sono richiesti i seguenti due pacchetti (e relative dipendenze):

- gcc
- make

Test di funzionamento

Distribution	Version	Architecture	gcc	make	Success
Debian	squeeze	amd64	4.4.5	3.81	✔
		i386	4.4.5	3.81	✔
	wheezy	amd64	4.7.2	3.81	✔
		i386	4.7.2	3.81	✔
Ubuntu	Vivid Vervet (15.04)	x86_64	4.9.2	4.0	✔
Arch	10/01/2015	x86_64	4.9.2	4.1	✔