Laboratorio di Programmazione a.a. 2017-18

Problema 12

11 Aprile 2018

Descrizione – Parte I

Il protocollo della classe `RoundTable`, che realizza il dato astratto *"tavola rotonda"*, è definito dal costruttore di una tavola con n cavalieri e dai seguenti metodi: `numberOfKnightsIn`, per conoscere quanti cavalieri sono rimasti a tavola; `knightWithJugIn`, per conoscere l'etichetta del cavaliere con la brocca di sidro; `afterNextKnightQuits`, per determinare la configurazione successiva dopo un passo della conta, nel corso del quale il cavaliere con la brocca serve il cavaliere alla sua sinistra, che esce, e passa la brocca al nuovo vicino alla sua sinistra. La realizzazione discussa a lezione è basata su una lista in cui i cavalieri rimasti attorno al tavolo, identificati da numeri interi, sono ordinati in verso orario; il primo cavaliere della lista, inoltre, ha in mano la brocca.

Per realizzare le operazioni previste dal protocollo in modo più efficiente è possibile organizzare le informazioni che definiscono lo stato interno utilizzando tre variabili di istanza, due liste e un intero:

- La prima lista, che non può essere mai vuota, contiene il cavaliere con la brocca e una parte di quelli che seguono, dove l'ordine della lista corrisponde al verso orario attorno al tavolo;
- La seconda lista, che invece può essere vuota, contenente i restanti cavalieri, ma in ordine rovesciato rispetto al verso orario;
- L'intero registra il numero complessivo di cavalieri.

Strutturando le informazioni in questo modo, nella maggior parte dei casi, per costruire la configurazione successiva a una mossa è sufficiente spostare un cavaliere dall'inizio della prima lista all'inizio della seconda (più qualche altra semplice operazione). Solo saltuariamente, circa ad ogni dimezzamento del numero di cavalieri, la prima lista dovrà essere completamente ricostruita rovesciandovi gli elementi della seconda.

Più specificamente, la rappresentazione della configurazione iniziale con n cavalieri è data da:

- Prima lista: $(\ 1,\ 2,\ 3,\ \ldots\ n{-}2,\ n{-}1,\ n\ )$
- Seconda lista: $()$
- Intero: n

Inoltre, per determinare la rappresentazione della configurazione successiva a una mossa si devono gestire tre casi, a seconda che il numero di elementi contenuti nella prima lista sia $k > 2$, oppure 2, oppure 1 (vedi anche l'esempio in calce a questo documento):

Prima della mossa		Dopo la mossa
• Prima lista: $(\ c_1,\ c_2,\ c_3,\ \ldots\ c_{k-1},\ c_k\ )$ • Seconda lista: $(\ c_n,\ c_{n-1},\ \ldots\ c_{k+2},\ c_{k+1}\ )$ • Intero: n	→	• Prima lista: $(\ c_3,\ \ldots\ c_{k-1},\ c_k\ )$ • Seconda lista: $(\ c_1,\ c_n,\ c_{n-1},\ \ldots\ c_{k+2},\ c_{k+1}\ )$ • Intero: $n{-}1$
• Prima lista: $(\ c_1,\ c_2\ )$ • Seconda lista: $(\ c_n,\ c_{n-1},\ \ldots\ c_4,\ c_3\ )$ • Intero: n	→	• Prima lista: $(\ c_3,\ c_4,\ \ldots\ c_{n-1},\ c_n,\ c_1\ )$ • Seconda lista: $()$ • Intero: $n{-}1$
• Prima lista: $(\ c_1\ )$ • Seconda lista: $(\ c_n,\ c_{n-1},\ \ldots\ c_3,\ c_2\ )$ • Intero: n	→	• Prima lista: $(\ c_3,\ c_4,\ \ldots\ c_{n-1},\ c_n,\ c_1\ )$ • Seconda lista: $()$ • Intero: $n{-}1$

Realizza la classe `RoundTable` in ottemperanza alle specifiche illustrate sopra. Verifica quindi il programma che risolve il problema di Giuseppe Flavio (definito nella classe `Josephus`) con questa realizzazione alternativa del dato astratto *"tavola rotonda"* e controlla che i risultati ottenuti siano corretti. A tal fine, la classe `Josephus` non deve essere modificata in alcun modo.

Descrizione – Parte II

Sperimenta il programma che risolve il problema di Giuseppe Flavio con le due realizzazioni alternative della classe `RoundTable` — quella sviluppata a lezione e quella richiesta sopra — utilizzando il codice riportato nella classe `Test`, disponibile online a fianco di questo documento. In particolare, confronta la differenza dei tempi di esecuzione forniti dalla procedura `TimeCost` (in millisecondi) — e i tempi di attesa percepiti! — per valori di *n* dell'ordine di qualche migliaio.

Esempio: alcuni passi dell'evoluzione dello stato interno per *n = 12*.

< (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), (), 12 >	stato interno relativo alla configurazione iniziale
< (3, 4, 5, 6, 7, 8, 9, 10, 11, 12), (1), 11 >	dopo la prima mossa
< (5, 6, 7, 8, 9, 10, 11, 12), (3, 1), 10 >	dopo la seconda mossa
< (7, 8, 9, 10, 11, 12), (5, 3, 1), 9 >	e così via...
< (9, 10, 11, 12), (7, 5, 3, 1), 8 >	
< (11, 12), (9, 7, 5, 3, 1), 7 >	
< (1, 3, 5, 7, 9, 11), (), 6 >	*reverse* [< (), (11, 9, 7, 5, 3, 1), 6 > non ammissibile]
< (5, 7, 9, 11), (1), 5 >	
< (9, 11), (5, 1), 4 >	
< (1, 5, 9), (), 3 >	*reverse* [< (), (9, 5, 1), 3 > non ammissibile]
< (9), (1), 2 >	
< (9), (), 1 >	*reverse*