TPFI 2022/23
Hw 3: Strutture Dati Infinite

assegnato: 20 aprile 2023, consegna 7 maggio 2023

Esercizio 1 (INSOMNIA) Scrivere una funzione Haskell che genera la lista infinita di caratteri `insonnia = "1 sheep 2 sheep 3 sheep 4 sheep ..."`. Provare a scrivere un "one-liner", cioè un programma che semplicemente compone opportunamente funzioni. Può essere utile la funzione `show :: Show a => a => String` che trasforma un elemento di qualsiasi tipo che implementa la classe `Show` in una stringa, cioè una lista di caratteri.

Esercizio 2 (TRIANGOLO INFINITO DI TARTAGLIA) Definite in Haskell la lista infinita di liste finite `tartaglia`, tale che `tartaglia!!n` sia l'n-esima riga del triangolo di Tartaglia, e quindi `tartaglia!!n!!k` sia il coefficiente binomiale $\left(\begin{array}{c} n \\ k \end{array}\right)$.

Esercizio 3 (NUMERI FORTUNATI) I *numeri fortunati*, introdotti da Stanislaw Ulam, sono definiti come segue.

Dalla sequenza dei numeri naturali tolgo tutti i *secondi numeri*, cioè i pari.

A quel punto, il *secondo numero rimasto* è il 3 e quindi si tolgono tutti i terzi numeri tra i sopravvissuti (5, 11, 17, ...).

Ora considero il *terzo numero rimasto* cioè il 7 e rimuovo tutti i settimi numeri (il primo è il 19) e così via, fino a ottenere tutti i numeri sopravvissuti a tutte le operazioni di "filtraggio".

Scrivere una funzione Haskell che genera lo stream dei numeri fortunati.

Esercizio 4 (ALBERO DEI RAZIONALI) Tradizionalmente si dimostra che i numeri razionali sono *numerabili* fornendo una biiezione tra i numeri naturali e le coppie di numeri naturali: tuttavia questa biiezione contiene più volte gli stessi razionali (in realtà ogni razionale p/q appare infinite volte, in quanto tutte le coppie di naturali $\{(kp, kq) \mid k > 0\}$ rappresentano lo stesso numero razionale).

L'*albero di Calkin-Wilf* (vedi figura prossima pagina) fornisce un metodo per costruire una biiezione diretta tra numeri naturali e numeri razionali: i nodi dell'albero contengono tutte e sole le coppie di razionali ridotte ai minimi termini. Ci basta sapere che l'albero di Calkin-Wilf ha una semplice regola di

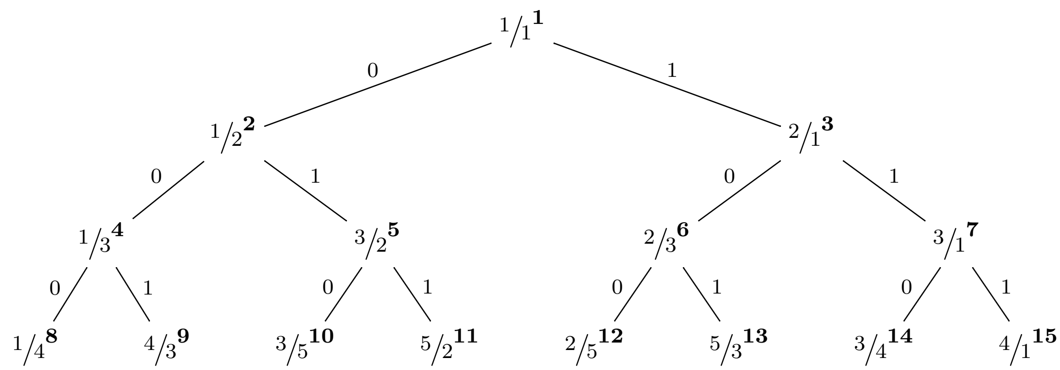

Figura 1: L'albero dei razionali di Calkin-Wilf. In apice e in neretto il naturale corrispondente alla frazione nella numerazione proposta.

costruzione: se (m, n) è il valore in un nodo, il figlio sinistro contiene la coppia $(m, m + n)$ e il figlio destro la coppia $(m + n, n)$.

Un altro modo di vedere la cosa, è che l'algoritmo di Euclide (in forma sottrattiva) per il Massimo Comun Divisore fa esattamente le stesse sequenze di chiamate ricorsive su tutte le coppie $\{(km, kn) \mid k, m, n > 0\}$: i cammini (dal basso verso l'alto) che congiungono $^m/_n$ a $^1/_1$ sono le corrispondenti sequenze binarie che codificano le computazioni di $\mathsf{mcd}(m, n)$ con l'algoritmo di Euclide in versione sottrattiva. I naturali in neretto possono essere interpretati come la somma tra il naturale rappresentato dalla sequenza di 0 e 1 (letta dall'alto verso il basso) sommato all'ultimo numero del livello precedente più 1.

Viene richiesto di:

1. scrivere un'equazione ricorsiva che genera l'albero di Calkin-Wilf;

2. scrivere la funzione `takeNlevels::Int -> BinTree a -> BinTree a` che taglia un albero (eventualmente infinito) ai primi n livelli; stipulando che `takeNlevels 0 b` torni sempre l'albero vuoto `Empty` e `takeNlevels 1 (Node r lft rgt)` torni `Node r Empty Empty`, etc., l'albero in figura si ottiene con la chiamata `takeNlevels 4 calkinWilf`.

3. ★scrivere una funzione `visitaLivelli :: BinTree a -> [a]` che produce la lista dei valori contenuti nei nodi di un albero (finito) nella sequenza ottenuta da una visita per livelli.

Quindi, la chiamata `visitaLivelli (takeNlevels n calkinWilf)` produrrà i primi $2^n - 1$ razionali nell'ordine indotto da questa biiezione