TPFI 2021/22
Hw 2: Efficienza, Tipi & Prove

docente: I. Salvo, Sapienza Università di Roma

assegnato: 25 marzo 2022, consegna 12 aprile 2022

Esercizio 1 (Efficienza)

Definire una funzione Haskell che implementa una versione di *mergeSort* "iterativa": la richiesta può sembrare paradossale, visto che in Haskell ho solo ricorsione, ma provate a scrivere una funzione che implementa l'idea descritta nel seguito:

> *Presa una lista xs, creare una lista di liste lunghe 1, ciascuna contenente un elemento di xs, e poi fondere a due a due le liste ordinate (lasciando inalterata un'eventuale ultima lista se la lunghezza fosse dispari), finchè non rimane una lista di liste lunga 1.*

Ad esempio, avendo come parametro la lista `[5,3,4,2,1]` tale funzione calcolerebbe le seguenti liste: `[[5],[3],[4],[2],[1]]`, poi `[[3,5],[2,4],[1]]`, poi `[[2,3,4,5],[1]]` e infine `[[1,2,3,4,5]]` da cui viene estratto il risultato finale `[1,2,3,4,5]`.

Ragionare sulla complessità di tale algoritmo.

Esercizio 2 (Alberi binari & Funzionali)

Scrivere i funzionali `mapBT` e `foldrBT` che generalizzano agli alberi binari `BinTree` (vedi def. Lezione **9**) gli analoghi funzionali `map` e `foldr` sulle liste.

Scrivere poi le seguenti funzioni usando `foldrBT`:

1. numero dei nodi di un albero binario;
2. altezza dell'albero (= lunghezza in numero di archi del più lungo cammino radice-foglia)
3. massimo indice di sbilanciamento (= massima differenza tra altezza sottoalbero destro/sinistro)

Ovviamente, cercare di ottenere un algoritmo lineare nel numero dei nodi per tutti e 3 i problemi

Esercizio 3 (Alberi ennari & Funzionali)

Scrivere i funzionali `mapT` e `foldrT` che generalizzano agli alberi ennari `Tree` (vedi def. Lezione **10**) gli analoghi funzionali `foldr` e `map` sulle liste.

Esercizio 4 (Alberi binari/ennari ed Efficienza)

Il diametro di una albero è il più lungo cammino che collega due nodi dell'albero (direi anche deve trattarsi di due foglie). Non è viceversa vero che tale cammino più lungo passi per la radice (costruire un esempio di tale fatto). Usare tupling e parametri accumulatori per scrivere una funzione *lineare* che calcola il diametro di un albero.

Scegliere a piacere se si preferisce definirla sui `BinTree` oppure sui `Tree`.

Esercizio 5 (Derivazioni di programmi) Considerare la seguente specifica della funzione `scanr`:

```
scanr f e = map (foldr f e) .  tails
```

Derivare una definizione efficiente (cioè lineare nella lunghezza della lista) facendo manipolazioni algebriche, in analogia con quanto visto per `scanl` (slide lezione **7**).