Lengoaiak, Konputazioa eta Sistema Adimendunak

7. gaia: Haskell – 1,6 puntu – Bilboko IITUE

2014-01-13

1 Murgilketa (0,300 puntu)

Osoa den x zenbakia emanda, bere zatitzaile bikoitien kopurua zatitzaile bakoitien kopurua baino handiagoa baldin bada $True$ eta bestela $False$ itzuliko duen $bik\_gehiago$ funtzioa definitu behar da. Zatitzaile bikoitien eta bakoitien kopurua berdina bada $False$ itzuli beharko da. x parametroaren balioa 1 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da.

$$bik\_gehiago :: Int \text{ -> } Bool$$
$$bik\_gehiago\ \ x\ \ \ldots$$

Adibideak:

$bik\_gehiago\ 3\ =\ False$	3 zenbakiaren zatitzaile bikoitien kopurua (zero) ez delako zatitzaile bakoitien kopurua (bi, 1 eta 3) baino handiagoa.
$bik\_gehiago\ 4\ =\ True$	4 zenbakiak bi zatitzaile bikoiti (2, 4) eta zatitzaile bakoiti bat (1) dituelako.
$bik\_gehiago\ 6\ =\ False$	6 zenbakiak bi zatitzaile bikoiti (2, 6) eta bi zatitzaile bakoiti (1, 3) dituelako.

Murgilketaren teknika jarraituz, osoak diren x, bm, bik eta bak lau zenbaki emanda, bm-tik hasita x zenbakiaren zatitzaile bikoitien kopurua gehi bik zatitzaile bakoitien kopurua gehi bak baino handiagoa baldin bada $True$ eta bestela $False$ itzuliko duen $bik\_gehiago\_lag$ funtzioa definitu behar da. bm-tik hasita, x zenbakiaren zatitzaile bikoitien kopurua gehi bik eta zatitzaile bakoitien kopurua gehi bak berdinak badira, $False$ itzuli beharko da. x balioa 1 baino txikiagoa baldin bada edo bm balioa 1 baino txikiagoa baldin bada, errore-mezua aurkeztu beharko da. Bestalde, bm balioa x baino handiagoa bada ere, errore-mezua aurkeztu beharko da.

$$bik\_gehiago\_lag :: Int \text{ -> } Int \text{ -> } Int \text{ -> } Int \text{ -> } Bool$$
$$bik\_gehiago\_lag\ \ x\ \ bm\ \ bik\ \ bak\ \ \ldots$$

Adibideak:

$bik\_gehiago\_lag\ \ 3\ \ 1\ \ 0\ \ 0\ \ =\ \ False$
1etik hasita, 3ren zatitzaile bikoitien kopurua gehi 0 ez delako 3ren zatitzaile bakoitien kopurua gehi 0 baino handiagoa.

$bik\_gehiago\_lag\ \ 3\ \ 1\ \ 8\ \ 0\ \ =\ \ True$
1etik hasita, 3ren zatitzaile bikoitien kopurua gehi 8 3ren zatitzaile bakoitien kopurua gehi 0 baino handiagoa delako.

$bik\_gehiago\_lag\ \ 50\ \ 1\ \ 0\ \ 0\ \ =\ \ False$
1etik hasita, 50en zatitzaile bikoitien (2, 10, 50) kopurua gehi 0 ez delako 50en zatitzaile bakoitien (1, 5, 25) kopurua gehi 0 baino handiagoa.

$bik\_gehiago\_lag\ \ 50\ \ 6\ \ 0\ \ 0\ \ =\ \ True$
6tik hasita, 50en zatitzaile bikoitien (10, 50) kopurua gehi 0 50en zatitzaile bakoitien (25) kopurua gehi 0 baino handiagoa delako.

$bik\_gehiago\_lag$ funtzioa $bik\_gehiago$ funtzioa baino orokorragoa da. $bik\_gehiago$ funtzioa $bik\_gehiago\_lag$ funtzioaren bidez definitzerakoan bm parametroa zatitzaileak bilatzerakoan zeharkatu beharreko tartearen beheko muga bezala erabili behar da. Bestalde, bik eta bak parametroak zatitzaile bikoitiak eta bakoitiak zenbatuz joateko erabili behar dira.

2 Bukaerako errekurtsibitatea (0,300 puntu)

Har dezagun honako funtzio hau:

$$
\begin{array}{l}
txertatu :: Integer \text{ -> } [Integer] \text{ -> } [Integer] \\
txertatu\ \ x\ \ [\,]\ \ =\ \ x : [\,] \\
txertatu\ \ x\ \ (y : s) \\
\ \ \mid\ x \leq y\ \ = x : (y : s) \\
\ \ \mid\ otherwise\ \ = y : (txertatu\ x\ \ s)
\end{array}
$$

Osoa den x zenbaki bat eta zenbaki osozko zerrenda bat emanda, x balioa x baino handiagoa edo berdina den zerrendako lehenengo elementuaren aurrean (ezkerreko aldean) kokatuz lortzen den zerrenda itzuliko du $txertatu$ funtzioak.

Adibideak:

$$
\begin{array}{l}
txertatu\ \ 5\ \ [4,8,3,6]\ \ =\ \ [4,5,8,3,6] \\
txertatu\ \ 2\ \ [4,8,3,6]\ \ =\ \ [2,4,8,3,6] \\
txertatu\ \ 8\ \ [4,8,3,6]\ \ =\ \ [4,8,8,3,6] \\
txertatu\ \ 10\ \ [4,8,3,6]\ \ =\ \ [4,8,3,6,10]
\end{array}
$$

$txertatu$ funtzioak ez du bukaerako errekurtsibitaterik. Bukaerako errekurtsibitatea edukitzeko, honako bi funtzio hauek definitu behar dira:

- $txertatu$ funtzioak jasotzen dituen x zenbakiaz eta zerrendaz gain, emaitza bezala eraikiz joango den zerrenda gordez joateko erabiliko den bigarren zerrenda duen $txertatu\_lag$ funtzioa. Beraz, $txertatu\_lag$ funtzioak jarraian zehazten diren bi zerrendak elkartuz lortzen den zerrenda itzuli beharko du:
  - Alde batetik, datu bezala emandako bigarren zerrenda.
  - Beste aldetik, datu bezala emandako lehenengo zerrendan x balioa x baino handiagoa edo berdina den lehenengo elementuaren aurrean (ezkerreko aldean) kokatuz lortzen den zerrenda berria.

$$
\begin{array}{l}
txertatu\_lag\ \ 5\ \ [4,8,3,6]\ \ [\,]\ \ =\ \ [4,5,8,3,6] \\
txertatu\_lag\ \ 5\ \ [4,8,3,6]\ \ [12,10]\ \ =\ \ [12,10,4,5,8,3,6] \\
txertatu\_lag\ \ 10\ \ [4,8,3,6]\ \ [12,10]\ \ =\ \ [12,10,4,8,3,6,10]
\end{array}
$$

- $txertatu\_lag$ funtzioari egokiak diren parametroekin deituz $txertatu$ funtzioak egiten duen gauza bera egingo duen $txertatu\_be$ funtzioa.

$$
\begin{array}{l}
txertatu\_be\ \ 5\ \ [4,8,3,6]\ \ =\ \ [4,5,8,3,6] \\
txertatu\_be\ \ 2\ \ [4,8,3,6]\ \ =\ \ [2,4,8,3,6] \\
txertatu\_be\ \ 8\ \ [4,8,3,6]\ \ =\ \ [4,8,8,3,6] \\
txertatu\_be\ \ 10\ \ [4,8,3,6]\ \ =\ \ [4,8,3,6,10]
\end{array}
$$

Beraz, $txertatu$ funtzioak egiten duena $txertatu\_be$ eta $txertatu\_lag$ funtzioak erabiliz egin ahal izango da.

3 Zerrenda-eraketa (1,000 puntu)

3.1. (0,100 puntu) Zenbaki osozko zerrenda bat emanda, zerrendan elementu negatiborik baldin badago $True$ eta bestela $False$ itzultzen duen $negatiborik$ izeneko funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} negatiborik :: [Integer] \text{ -> } Bool \\ negatiborik \quad \ldots \end{array}$$

Adibideak:

$$\begin{array}{l} negatiborik \ \ [7,2,-3,8,-9] \ \ = True \\ negatiborik \ \ [7,2,5,0] \ \ = False \end{array}$$

Aukera bat aurredefinitutako $length$ funtzioa erabiltzea da.

3.2. (0,150 puntu) Zenbaki osozko zerrendez eratutako zerrenda bat emanda, zerrenda bakoitzean elementu negatiborik agertzen al den adierazten duen balio Boolearrezko zerrenda itzultzen duen $neg\_agerpenik$ izeneko funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} neg\_agerpenik :: [[Integer]] \text{ -> } [Bool] \\ neg\_agerpenik \quad \ldots \end{array}$$

Adibideak:

$$\begin{array}{lcl} neg\_agerpenik & = & [[7,2,8],[0,0],[\,],[4,-6,8],[-2,-2]] \ = \ [False, False, False, True, True] \\ neg\_agerpenik & = & [\,] \ = \ [\,] \\ neg\_agerpenik & = & [[2,8],[10],[-1],[4,6]] \ = \ [False, False, True, False] \end{array}$$

Aukera bat aurreko ariketako $negatiborik$ funtzioa erabiltzea da.

3.3. (0,150 puntu) Zenbaki osozko zerrendez eratutako zerrenda bat emanda, elementu negatiborik ez duten zerrendak bakarrik mantenduz geratzen den zerrenda itzultzen duen neg_gabe izeneko funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} neg\_gabe :: [[Integer]] \text{ -> } [[Integer]] \\ neg\_gabe \quad \ldots \end{array}$$

Adibideak:

$$\begin{array}{lcl} neg\_gabe & = & [[7,2,8],[0,0],[\,],[4,-6,8],[-2,-2]] \ = \ [[7,2,8],[0,0],[\,]] \\ neg\_gabe & = & [\,] \ = \ [\,] \\ neg\_gabe & = & [[2,8],[10],[-1],[4,6]] \ = \ [[2,8],[10],[4,6]] \\ neg\_gabe & = & [[3,-9],[-20],[-1],[-2,-7,6]] \ = \ [\,] \end{array}$$

Aukera bat 3.1 ariketako $negatiborik$ funtzioa erabiltzea da.

3.4. (0,100 puntu) Osoa den zenbaki bat emanda, zenbaki horren zatitzaileen zerrenda itzuliko duen $zatitzaileak$ funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} zatitzaileak :: Integer \text{ -> } [Integer] \\ zatitzaileak \quad \ldots \end{array}$$

Adibideak:

$$\begin{array}{l} zatitzaileak\ 8 = [1,2,4,8] \\ zatitzaileak\ 7 = [1,7] \\ zatitzaileak\ 0 = [\,] \end{array}$$

3.5. (0,100 puntu) Osoak diren zenbaki positibo denen zatitzaileen zerrendez osatutako zerrenda infinitua aurkeztuz joango de zat_denak funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} zat\_denak :: [[Integer]] \\ zat\_denak\ \ldots \end{array}$$

Adibideak:

$$zat\_denak\ =\ [[1],[1,2],[1,3],[1,2,4],[1,5],[1,2,3,6],\ldots]$$

Aukera bat 3.4 ariketako $zatitzaileak$ funtzioa erabiltzea da.

3.6. (0,100 puntu) Lehenengo osagai bezala osoa eta positiboa den zenbaki bat (zenbakien ohiko ordena jarraituz) eta bigarren osagai bezala lehenengo osagaiaren zatitzaile-zerrendaz osatutako bikoteez eratutako zerrenda infinitua kalkulatuz joango den zat_bikote funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} zat\_bikote :: [(Integer,[Integer])] \\ zat\_bikote\ \ldots \end{array}$$

Adibidea:

$$zat\_bikote\ =\ [(1,[1]),(2,[1,2]),(3,[1,3]),(4,[1,2,4]),(5,[1,5]),(6,[1,2,3,6]),\ldots]$$

Aukera bat aurreko ariketako zat_denak funtzioa eta aurredefinitutako zip funtzioa erabiltzea da.

3.7. (0,200 puntu) Osoa den n zenbaki bat emanda, n baino zatitzaile gehiago dituzten zenbakiei dagozkien zat_bikote zerrendako bikoteen zerrenda infinitua aurkeztuz joango den $zat\_gehiago$ funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} zat\_gehiago :: Integer \text{ -> } [(Integer,[Integer])] \\ zat\_gehiago\ \ldots \end{array}$$

Adibideak:

$zat\_gehiago\ 3\ =\ [(6,[1,2,3,6]),(8,[1,2,4,8]),(10,[1,2,5,10]),(12,[1,2,3,4,6,12]),\ldots]$
Kasu horietan zatitzaile-kopurua 3 baino handiagoa da.

Aukera bat aurreko ariketako zat_bikote funtzioa eta aurredefinitutako $genericLength$ funtzioa erabiltzea da.

3.8. (0,100 puntu) Osoak diren n eta kop bi zenbaki emanda, n baino zatitzaile gehiago dituzten lehenengo kop zenbakiei dagozkien zat_bikote zerrendako bikoteez osatutako zerrenda aurkeztuko duen $gehiago\_finitua$ funtzioa definitu Haskell lengoaia erabiliz.

$$\begin{array}{l} gehiago\_finitua :: Integer \text{ -> } Integer \text{ -> } [(Integer,[Integer])] \\ gehiago\_finitua\ \ldots \end{array}$$

Adibideak:

$gehiago\_finitua\ 3\ 5\ =\ [(6,[1,2,3,6]),(8,[1,2,4,8]),(10,[1,2,5,10]),(12,[1,2,3,4,6,12]),$
$(14,[1,2,7,14])]$
3 zatitzaile baino gehiago dituzten lehenengo 5 zenbakiei dagozkien bikoteak.

Aukera bat aurreko ariketako $zat\_gehiago$ funtzioa eta aurredefinitutako $genericTake$ funtzioa erabiltzea da.