## თავი 1.9. ზარმაცი, იგივე გადადებული გამოთვლები

Haskell-ში გა­მო­სა­ხუ­ლე­ბა­თა შე­ფა­სე­ბა მარ­ტი­ვი მე­თო­დით ხდე­ბა, რომ­ლის მთა­ვა­რი პრინ­ცი­პე­ბი ასე­თი­ა:

* თავის შეკავება ზედმეტი, უსარგებლო გამოთვლებისაგან;
* პროგრამათა მეტი მოდულობის უზრუნველყოფა;
* დაპ­როგ­რა­მე­ბი­სას უსას­რუ­ლო სი­ებ­თან მუ­შა­ო­ბის შე­საძ­ლებ­ლო­ბის შექ­მნა.

გა­მოთ­ვლა­თა ასეთ მე­თოდს გა­და­დე­ბუ­ლი, იგივე ზარ­მა­ცი გა­მოთ­ვლე­ბი ეწო­დე­ბა, ხო­ლო თა­ვად Haskell-ს - ზარ­მა­ცი ფუნ­ქცი­ო­ნა­ლუ­რი ენა.

ძი­რი­თა­დად, გა­მო­სა­ხუ­ლე­ბე­ბი გა­მო­ით­ვლე­ბა ან გარ­და­იქ­მნე­ბა გან­საზღ­ვრე­ბა­თა გა­მო­ყე­ნე­ბით, ვიდ­რე შემ­დგო­მი გა­მარ­ტი­ვე­ბა შე­უძ­ლე­ბე­ლი არ გახ­დე­ბა.

მა­გა­ლი­თად, თუ გა­ვით­ვა­ლის­წი­ნებთ

square n = n\*n

გან­საზღ­ვრე­ბას, მა­შინ square (3+4) გა­მო­სა­ხუ­ლე­ბა შე­იძ­ლე­ბა შე­ფას­დეს გარ­დაქ­მნა­თა შემ­დე­გი მიმ­დევ­რო­ბის გა­მო­ყე­ნე­ბით:

square (3+4)

= square 7

= 7 \* 7

= 49

მაგ­რამ, შე­საძ­ლო გარ­დაქ­მნა­თა მიმ­დევ­რო­ბის გა­მო­ყე­ნე­ბა ერ­თა­დერ­თი გზით არ ხდე­ბა. მა­გა­ლი­თად:

square (3+4)

=(3+4)\* (3+4)

=7 \*(3+4)

=7\*7

=49

ახ­ლა ჩვენ გა­მო­ვი­ყე­ნეთ კვად­რატ­ში აყ­ვა­ნა შეკ­რე­ბის ოპე­რა­ცი­ამ­დე, მაგ­რამ სა­ბო­ლოო შე­დე­გი ისე­თი­ვე მი­ვი­ღეთ.

ფაქ­ტი: Haskell-ში გა­მო­სა­ხუ­ლე­ბის შე­ფა­სე­ბა ორი სხვა­დას­ხვა (მაგ­რამ სას­რუ­ლი) გზით ერ­თსა და იმა­ვე შე­დეგს იძ­ლე­ვა.

ვი­სა­უბ­როთ გარ­დაქ­მნა­თა სტრა­ტე­გი­ებ­ზე. რე­დექ­სის ანუ რე­დუ­ცი­რე­ბა­დი ქვე­გა­მო­სა­ხუ­ლე­ბის (REDu-cible subEXpres­sion) ასარ­ჩე­ვად ორი ზო­გა­დი სტრა­ტე­გია არ­სე­ბობს.

1. ში­და რე­დუქ­ცი­ა. ში­და რე­დექ­სი ყო­ველ­თვის რე­დუ­ცი­რე­ბა­დი­ა;
2. გა­რე რე­დუქ­ცი­ა. გა­რე რე­დექ­სი ყო­ველ­თვის რე­დუ­ცი­რე­ბა­დი­ა.

ისმის შეკითხვა: როგორ ხდება ორი სტრატეგიის შედარება?

**დასრულებადობა**

loop = tail loop

შევაფასოთ fst (1,loop) გამოსახულება რედუქციის ამ ორი სტრატეგიის გამოყენებით:

1. შიდა რედუქცია

fst (1, loop)

= fst (1,tail loop)

= fst (1,tail (tail loop))

= …

ეს სტრატეგია დასრულებადი არ არის

1. გარე რედუქცია

fst (1,loop)

= 1

ეს სტრატეგია იძლევა შედეგს ერთი ბიჯით.

ად­გი­ლი აქვს შემ­დეგ ფაქ­ტებს:

გა­რე რე­დუქ­ცი­ამ შე­იძ­ლე­ბა მოგ­ვცეს შე­დე­გი მა­ში­ნაც კი, რო­ცა ში­და რე­დუქ­ცია დას­რუ­ლე­ბა­დი არ არის;

თუ მო­ცე­მუ­ლი გა­მო­სა­ხუ­ლე­ბი­სათ­ვის სა­ერ­თოდ არ­სე­ბობს რე­დუქ­ცი­ა­თა რო­მე­ლი­მე სას­რუ­ლი მიმ­დევ­რო­ბა, მა­შინ გა­რე რე­დუქ­ცია ასე­ვე სას­რუ­ლი იქ­ნე­ბა იმა­ვე შე­დე­გით.

გა­მოვ­თვა­ლოთ რე­დუქ­ცი­ა­თა რიცხ­ვი. კვლავ გან­ვი­ხი­ლოთ შემ­დე­გი რე­დუქ­ცი­ე­ბი:

შიდა გარე

square (3+4) square (3+4)

= Square 7 =(3+4) \* (3+4)

= 7 \* 7 = 7\*(3+4)

= 49 = 7\*7

=49

გა­რე ვერ­სია არა­ე­ფექ­ტუ­რი­ა: 3+4 ქვე­გა­მო­სა­ხუ­ლე­ბა მე­ორ­დე­ბა კვად­რატ­ში ახა­რის­ხე­ბის რე­დი­ცი­რე­ბი­სას და ამი­ტომ შემ­დეგ ორ­ჯერ გვიხ­დე­ბა გა­მარ­ტი­ვე­ბის გან­ხორ­ცი­ე­ლე­ბა.

აქე­დან შე­იძ­ლე­ბა და­ვას­კვნათ, რომ გა­რე რე­დუქ­ცი­ამ შე­იძ­ლე­ბა მო­ითხ­ო­ვოს უფ­რო მე­ტი მოქ­მე­დე­ბა, ვიდ­რე ში­დამ.

ეს პრობ­ლე­მა შე­იძ­ლე­ბა გა­და­იჭ­რას მაჩ­ვე­ნებ­ლე­ბით გა­მო­სა­ხუ­ლე­ბა­თა ერ­თობ­ლი­ვი გა­მო­ყე­ნე­ბის საჩ­ვე­ნებ­ლად გა­მოთ­ვლა­ში.

Square (3+4)

=(**º** \* **º**) აქ **º** არის (3+4)-ის მაჩვენებელი

=(**º** \* **º**) აქ **º** არის 7-ის მაჩვენებელი

=49

ეს რედუქციის ახალ სტრატეგიას იძლევა:

**ზარმაცი გამოთვლა = გარე რედუქცია + ერთობლივად გა­მოყენება**

თუ დავაკვირდებით, შევამჩნევთ შემდეგ ფაქტებს:

* ზარმაცი გამოთვლა არასოდეს ითხოვს რედუქციათა უფრო მეტ ბიჯს, ვიდრე შიდა რედუქცია;
* Haskell იყენებს ზარმაც (გადადებულ) გამოთვლას.

**უსასრულო სიები**

დას­რუ­ლე­ბა­დო­ბის უპი­რა­ტე­სო­ბას­თან ერ­თად, ზარ­მა­ცი გა­მოთ­ვლა მნიშ­ვნე­ლო­ბა­თა უსას­რუ­ლო სი­ე­ბის დაპ­როგ­რა­მე­ბის სა­­შუ­ა­ლე­ბა­საც იძ­ლე­ვა!

განვიხილოთ რეკურსიული განსაზღვრება

ones :: [Int]

ones = 1: ones

რეკურსიის რამდენიმეჯერ განხორციელება იძლევა:

ones = 1:ones

= 1:1:ones

= 1:1:1:ones

= …

ამრიგად მიიღება ერთიანების უსასრულო სია.

კვლავ განვიხილოთ head onesგამოსახულების შეფასება შიდა რედუქციისა და გამოთვლის გამოყენებით:

1. შიდა რედუქცია

head ones = head (1:ones)

= head (1:1:ones)

= (1:1:1:ones)

= ...

ამ შემთხვევაში გამოთვლა დაუსრულებლად გრძელდება.

ზარმაცი გამოთვლა

head ones = (1:ones)

= 1

ამ შემ­თხვე­ვა­ში გა­მოთ­ვლის შე­დე­გია რიცხ­ვი 1.

ამ­რი­გად, ზარ­მა­ცი გა­მოთ­ვლე­ბის გა­მო­ყე­ნე­ბით, ones უსას­რუ­ლო სი­ა­ში, ფაქ­ტობ­რი­ვად, მხო­ლოდ პირ­ვე­ლი მნიშ­ვნე­ლო­ბაა ნა­წარ­მო­ე­ბი, რად­გან მხო­ლოდ იგია სა­ჭი­რო მთე­ლი head ones გა­მო­სა­ხუ­ლე­ბის გა­მო­სათ­ვლე­ლად.

სა­ზო­გა­დოდ, არ­სე­ბობს შემ­დე­გი სლო­გა­ნი (დე­ვი­ზი, ლო­ზუნ­გი):

**ზარ­მა­ცი გა­მოთ­ვლე­ბის სა­შუ­ა­ლე­ბით გა­მო­სა­ხუ­ლე­ბე­ბი გა­მო­ით­ვლე­ბა ზუს­ტად იმ მო­ცუ­ლო­ბით, რაც აუ­ცი­ლე­ბე­ლია სა­ბო­ლოო შე­დე­გის მი­სა­ღე­ბად.**

ახ­ლა ვხე­დავთ, რომ ones =1 : ones

ნამ­დვი­ლად გან­საზღ­ვრავს პო­ტენ­ცი­უ­რად უსას­რუ­ლო სი­ას, რო­მე­ლიც ფას­დე­ბა ზუს­ტად გა­მო­ყე­ნე­ბის ში­ნა­არ­სის მოთხ­ოვ­ნი­დან გა­მომ­დი­ნა­რე.

შე­იძ­ლე­ბა სას­რუ­ლი სი­ე­ბის გე­ნე­რი­რე­ბა უსას­რუ­ლო სი­ე­ბი­დან სა­მეტყ­ვე­ლო ელე­მენ­ტის გა­მო­ყე­ნე­ბით. მა­გა­ლი­თად:

|  |
| --- |
| Prelude> take 5 ones  [1,1,1,1,1]  Prelude> Take 5 [1..]  [1,2,3,4,5] |

ზარ­მა­ცი გა­მოთ­ვლე­ბი სა­შუ­ა­ლე­ბას გვაძ­ლევს მო­ნა­ცე­მე­ბი გა­მო­ით­ვა­ლის მარ­თვის კომ­პო­ნენ­ტის მოთხ­ოვ­ნის შე­სა­ბა­მი­სად.

გან­ვი­ხი­ლოთ მა­გა­ლი­თი: მარ­ტი­ვი რიცხ­ვე­ბის გე­ნე­რი­რე­ბა. ეს პრო­ცე­დუ­რა ცნო­ბი­ლია „ერატოსფენის საცრის“ სა­ხელ­წო­დე­ბით, ძვე­ლი ბერ­ძე­ნი მა­თე­მა­ტი­კო­სის პა­ტივ­სა­ცე­მად, რო­მელ­მაც პირ­ველ­მა აღ­წე­რა ხსე­ნე­ბუ­ლი Haskell-ალ­გო­რით­მი. ეს ალ­გო­რით­მი შე­იძ­ლე­ბა პირ­და­პირ ჩავ­წე­როთ Haskell ენა­ზე:

|  |
| --- |
| primes :: [Int]  primes = seive [z..]  seive :: [Int] −›[Int]  seive (p:xs) = p:seive [x|x‹−xs, x`mod`p /=0] |

მარტივი რიცხვების გამოთვლის ფუნქციის შესრულებას აქვს სახე:

|  |
| --- |
| Prelude> Primes  [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53, 59,61, 67, … |

ფუნ­ქცი­ის გა­მო­ძა­ხე­ბი­სას შე­იძ­ლე­ბა შე­მო­ვი­ტა­ნოთ სხვა­დას­ხვა სა­საზღ­ვრო პი­რო­ბა. მა­გა­ლი­თად, პირ­ვე­ლი ათი მარ­ტი­ვი რიცხ­ვის ან 15-ზე მცი­რე მარ­ტი­ვი რიცხ­ვე­ბის არ­ჩე­ვა:

|  |
| --- |
| -- პირველი ათი მარტივი რიცხვის არჩევა  Prelude> take 10 primes  [2,3,5,7,11,13,17,19,23,29]  -- მარტივი რიცხვების არჩევა, რომლებიც ნაკლებია 15-ზე  Prelude> take while ( ‹15 ) primes  [2,3,5,7,11,13] |

ზარმაცი გამოთვლები მოხერხებული დაპროგრამებაა!

### სავარჯიშოები

1. გან­საზღ­ვრეთ fibs :: [Integer]პროგ­რა­მა, რო­მე­ლიც წარ­მოქ­მნის ფი­ბო­ნა­ჩის უსას­რუ­ლო [0,1,1,2,3,5,8,13,21,34, ... მიმ­დევ­რო­ბას შემ­დე­გი მარ­ტი­ვი პრო­ცე­დუ­რის გა­მო­ყე­ნე­ბით:

ა) პირველი ორი რიცხვია 0 და 1;

ბ) შემდეგი რიცხვი წინა ორის ჯამია;

გ) გადასვლა ბ)-ზე.

1. გან­საზღ­ვრეთ: fib :: Int−›Integer ფუნ­ქცი­ა, რო­მე­ლიც ან­გა­რი­შობს ფი­ბო­ნა­ჩის n-ურ რიცხვს.