Universidade Federal do Rio Grande do Norte

DIM0152 - Matemática para Computação I

Prof.: Valdigleis S. Costa

Atividade de Reforço 02

Semestre 2025.2

Aluno: Eduardo Teixeira de Moura Silva

Matrícula: 20200047345

Data: 24 de novembro de 2025

1. Introdução

Esta atividade consiste na implementação de algoritmos para manipulação de arranjos e conjuntos, utilizando apenas recursos básicos de uma linguagem de programação.

A linguagem escolhida foi **Haskell**, devido à sua expressividade natural para lidar com listas e recursão, que são fundamentais para problemas combinatórios.

O código fonte completo encontra-se no arquivo `Arranjos.hs` neste mesmo diretório.

2. Soluções Implementadas

2.1. Questão 1(a): Arranjos sem Repetição

Problema: Gerar todos os arranjos sem repetição de tamanho m utilizando elementos de um conjunto X.

Implementação: Utilizamos uma abordagem recursiva. Para gerar um arranjo de tamanho m:

1. Selecionamos um elemento x de X.
2. Recursivamente geramos arranjos de tamanho $m-1$ a partir dos elementos restantes ($X - \{x\}$).
3. Combinamos x com cada sub-arranjo gerado.

```haskell
arranjosSemRepeticao :: Eq a => [a] -> Int -> [[a]]
arranjosSemRepeticao _ 0 = [[]]
arranjosSemRepeticao elements m =
    [ x:xs | x <- elements,
             xs <- arranjosSemRepeticao [e | e <- elements, e /= x] (m-1) ]
```

2.2. Questão 1(b): Arranjos com Repetição

Problema: Gerar arranjos com repetição de tamanho m que utilizam no mínimo n elementos distintos do conjunto X.

Implementação:

1. Geramos todos os arranjos com repetição de tamanho m (produto cartesiano).
2. Filtramos apenas aqueles que possuem quantidade de elementos distintos $\geq n$.

```haskell
questaoB :: Show a => Eq a => [a] -> Int -> Int -> IO ()
questaoB x m n = do
    let todos = arranjosComRepeticao x m
    let validos = [ arr | arr <- todos, contaDistintos arr >= n ]
    mapM_ print validos
```

2.3. Questão 1(c): Maior Arranjo com Soma Fixa

Problema: Encontrar o maior arranjo formado por $X \cup Y$ tal que a soma seja p, com no máximo m elementos de X e n elementos de Y.

Implementação:

1. Geramos todos os subconjuntos de $X \cup Y$.
2. Filtramos os que têm soma igual a p.
3. Verificamos as restrições de quantidade para cada origem.
4. Selecionamos o de maior comprimento.

2.4. Questão 1(d): Maior Subarranjo Crescente

Problema: Dado um arranjo A, encontrar o maior subarranjo contíguo crescente.

Implementação: Percorremos a lista mantendo o subarranjo crescente atual. Se o próximo elemento for maior ou igual ao anterior, estendemos o subarranjo. Caso contrário, iniciamos um novo. Mantemos sempre o maior encontrado até o momento.

```
encontraSubCrescente :: [Int] -> [Int] -> [Int] -> [Int]
encontraSubCrescente (x:xs) atual melhor =
    let ultimo = last atual
    in if x >= ultimo
       then encontraSubCrescente xs (atual ++ [x]) melhor
       else
           let novoMelhor = if length atual > length melhor then atual else melhor
           in encontraSubCrescente xs [x] novoMelhor
```

3. Como Executar

Para executar o código e ver os resultados dos testes para cada questão:

1. Certifique-se de estar no ambiente Nix:

   ```
   nix-shell
   ```

2. Compile o arquivo Haskell:

   ```
   cd Atividade_02
   ghc Arranjos.hs -o arranjos
   ```

3. Execute o programa:

   ```
   ./arranjos
   ```

O programa executará casos de teste para cada uma das 4 questões e imprimirá os resultados no terminal.