Exercício Programa 1

Rede Booleana

funções de dependência → bacia de atração

Faça um programa que receba o nome de um arquivo .txt, leia o arquivo contendo os genes e as funções booleanas e imprima os atratores e o respectivos tamanhos das bacias de atração.

Além disso, considerando a rede do ciclo celular de levedura descrita no artigo de Orlando et al (2008), as declarações abaixo descrevem as dependências da rede:

- MBF é ativado por CLN3.
- Se CLN3 ou MBF for transcrito e pelo menos um dos inibidores YOX1 e YHP1 estiver inativo, o SBF está ativo.
- YOX1 está ativo se ambos os fatores de transcrição MBF e SBF estiverem presentes. O mesmo se aplica ao HCM1.
- YHP1 pode ser ativado independentemente por MBF e SBF.
- SBF e HCM1 em conjunto ativam SFF.
- ACE2 requer que o SFF esteja ativo. O mesmo se aplica a SWI5.
- CLN3 requer a presença de SWI5 e ACE2 e a inatividade de pelo menos um dos inibidores YOX1 e YHP1 para ser ativado.

Escreva as funções a partir dessas instruções e salve-as no arquivo mynetwork.txt

Para a avaliação no VPL serão fornecidos dois arquivos: aula.txt já foi utilizado como exemplo em aula, e cellcycle.txt foi extraído do artigo de Fauré (2006). O terceiro arquivo que será utilizado será mynetwork.txt que você deverá criar e anexar no VPL.
Convenção: Os genes deverão estar em ordem alfabética na representação booleana dos estados. As listas dos atratores que não são singletons devem começar pelo primeiro da ordem crescente. Os atratores também devem ser apresentados em ordem crescente do primeiro estado que o compõe.

No modelo EP1.py fornecido há um cabeçalho que deve ser preenchido com seus dados e caso você tenha utilizado códigos que não são de sua autoria as referências deverão estar listadas logo abaixo do cabeçalho.
Exemplo:

```
'''
A classe Digraph é fornecida no livro Algorithms de R Sedgewick, K Wayne - 2011, versão em Python de Big Poppa e Fred disponibilizado no repositório:
https://github.com/ChangeMyUsername/algorithms-sedgewick-python
```

```
As classes DFSGraph, Graph e Vertex foram extraídas do livro Problem Solving with Algorithms and Data Structures using Python de Brad Miller and David Ranum, disponível em:
https://runestone.academy/runestone/books/published/pythonds/index.html
'''
```

Caso utilize uma estrutura de grafo, o VPL não possui interface gráfica e portanto a importação de bibliotecas gráficas pode gerar erros de execução, devendo ser removidas do código final a ser submetido.

De preferência crie uma classe **`BooleanNetwork`** ou **`BN`** contendo um método que retorne os atratores e o tamanho da bacia. Mas se você for iniciante em programação, poderá resolver tudo com listas mesmo, nem precisa usar grafos.

Exemplo de arquivo (`exemplo.txt`) contendo genes e funções:

```
A,     !A&C
B,     !A&!B
C,     A&B
```

Exemplo de saída do programa:

```
arquivo: exemplo.txt
atrator: ['000', '010'] tamanho da bacia: 5
atrator: ['001', '110'] tamanho da bacia: 3
```

Dica para quem vai resolver com listas:

Os códigos dos exercícios anteriores podem ser todos aproveitados. Tendo a tabela de transição de estados é possível percorrer todos os caminhos partindo de cada um dos estados iniciais e a cada caminho criar uma lista para conferir se não volta para algum dos estados que já foi visitado. Quando encontrar um estado já visitado é porque encontrou um ciclo, que é um atrator.
Armazene esse ciclo numa outra lista (atratores). Cuidado antes de armazenar, arrume a lista para que ela comece no menor valor do ciclo. Não é para ordenar a lista do ciclo, se não o ciclo vai ficar fora de ordem. Ao percorrer os caminhos, se o ciclo já está na lista de atratores não precisa armazenar para não ficar repetido.
Todos os caminhos também podem ser armazenados numa lista para depois calcular o tamanho da bacia: Para cada um dos atratores verifique em cada um dos caminhos aparece um dos estados do atrator (pode ser o primeiro) o que significa que esse caminho está na bacia de atração. Se estiver, guarde em uma lista. Terminando os caminhos 'limpe' essa lista porque ela vai estar com um monte de elementos repetidos. O tamanho da lista sem as repetições é o tamanho da bacia. Repita para os outros atratores.

Referências:

Orlando, D., Lin, C., Bernard, A. et al. *Global control of cell-cycle transcription by coupled CDK and network oscillators*. Nature 453, 944–947 (2008).

https://doi.org/10.1038/nature06955

Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, Denis Thieffry. *Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle*, Bioinformatics, Volume 22, Issue 14, 15 July 2006, Pages e124–e131, https://doi.org/10.1093/bioinformatics/btl210

Bradley N. Miller, David L. Ranum. *Problem Solving with Algorithms and Data Structures using Python*. Pythonds,
https://runestone.academy/runestone/books/published/pythonds/index.html#

Sedgewick, R., Wayne, K. (2011). *Algorithms, 4th Edition.*. Addison-Wesley. ISBN: 978-0-321-57351-3
Python version code:
https://github.com/ChangeMyUsername/algorithms-sedgewick-python