Matemática Discreta - Trabalho Prático

Documentação

Projeto

Para implementar as soluções para os problemas de verificação de propriedades em relações binárias definidos na especificação do trabalho, estas relações são representadas através de matrizes de adjacência (MA) e todos os algoritmos implementados recebem essa estrutura de dados como input e retornam um booleano indicando se a propriedade é válida sobre a relação ou não.

São utilizados dois tipos de dados compostos ao longo do projeto, `InputElements` e `AdjacencyMatrix`, encapsulando os dados sobre o formato de entrada e a matriz de adjacência da relação descrita respectivamente.

Cada propriedade possui uma função verificadora, exceto no caso reflexiva/irreflexiva, que é verificado por uma só função. Todas as funções aderem à mesma interface: recebem uma referência à `AdjacencyMatrix` e retornam um `bool`. No caso da equivaência e ordem parcial, utilizo resultados já computados pelas outras funções e portanto a assinatura das funções é `(bool, bool, bool) → bool` para simplificar a composição.

Os algoritmos usados para a maioria das funções são todos verificações triviais e exaustivas iteradas sobre a matriz de adjacência da relação, e portanto não são conhecidos por nomes específicos, com exceção do fecho transitivo direto que é baseado no algoritmo de Warshall.

Além dos verificadores, também foram implementadas funções auxiliares para manipular `InputElements` e `AdjacencyMatrix` e I/O.

Análise de Complexidade

As complexidades de tempo de pior caso dos algoritmos são:

- Reflexividade/Irreflexividade: $\Theta(N)$ - O algoritmo checa apenas a diagonal da MA
- Simetria: $\Theta(N^2)$ - checamos apenas uma metade triangular da MA
- Anti-simetria: $\Theta(N^2)$ - mesmo caso de simetria
- Assimetria: $\Theta(N^2)$ - é preciso verificar todos os elementos da MA
- Transitividade: $\Theta(N^3)$ - é preciso seguir todas as possíveis conexões de cada elemento da MA, $N^2 \times N$
- Equivalência: $\Theta(N^3)$ - a combinação Simetria$\wedge$Reflexividade$\wedge$Transitividade, e portanto o maior custo domina sua complexidade

- Ordem Parcial: $\Theta(N^3)$ - mesmo caso de equivalência, porém para Anti-Simetria∧Reflexividade∧Transitividade
- Feicho Transitivo: $\Theta(N^3)$ - mesmo caso de transitivade

E a complexidade de espaço para todos os algoritmos é N^2, que é o espaço necessário para representar a MA.

Obs.: Para analisar os custos de tempo e espaço foram ignorados todos os custos associados exclusivamente às funções de escrita para `stdout` do programa, já que não fazem parte do algoritmo em si e são apenas necessárias para as necessidades específicas de saída do trabalho prático.