Projeto Matemática Discreta I

Documentação

Alunos: Guilherme A. Motta, Jean Loui, Victor Manoel
Universidade Federal da Bahia

Introdução da situação-problema:

Suponha uma situação em que um programador recém-contratado de uma empresa é chamado para um grande projeto feito em C#. Como um bom programador, ele deseja trabalhar seguindo boas práticas do clean code, visando criar um código com alta coesão e baixo acoplamento. Infelizmente, o código do projeto está completamente desorganizado e suas classes possuem baixa coesão e alto acoplamento, prejudicando a sua escalabilidade e facilitando a ocorrência de “bugs”. Para que esse programador consiga realizar a manutenção do projeto e trabalhar com o seu polimorfismo, é necessário que compreenda a estrutura das classes nele presentes e para isso ele precisa organizá-las. Como ele poderá fazer isso se o código possui mais de 20 mil linhas e não possui nenhuma documentação?

Introdução da resolução:

Para resolver esse problema, serão utilizados os conhecimentos obtidos nas duas primeiras unidades do curso de “Matemática Discreta I”, além de conhecimentos das disciplinas de “Introdução à Lógica de Programação” e “Geometria Analítica”.

A solução consiste em um programa feito na linguagem Python 3 que, para este problema, fará a análise completa de uma dada relação em um arquivo e gerará um diagrama de Hasse em forma de imagem. Ademais, o programa foi feito para lidar com relações de diversas formas, então ele é também capaz de, ao receber uma relação, ordená-la lexicograficamente e, se for possível, gerar o conjunto da classe de equivalência de um valor Y, sua notação matricial e/ou cíclica. Para maior organização, legibilidade e melhor manutenção do código, ele é dividido em vários módulos.

Explicação dos módulos:

1) Módulo “app.py” e “conjunto.py”:

No módulo principal, “app.py”, o programa solicitará um arquivo com uma relação, lerá o seu conteúdo, organizará a relação obtida de forma lexicográfica e a validará. Em seguida, caso os elementos da relação sejam números, o programa perguntará ao usuário se ele deseja verificar se a relação é de congruência. Então, logo abaixo, o programa analisará e obterá todas as propriedades dessa relação e as imprimirá na tela. Com a relação ordenada e com suas propriedades obtidas, o programa verificará se é possível e perguntará ao usuário se ele deseja gerar um “diagrama de Hasse” e uma classe de equivalência, a partir de um valor Y. Posteriormente, o programa perguntará ao usuário se ele deseja gerar a relação ordenada, matricial e/ou cíclica em um arquivo.

No módulo “conjunto.py”, existe a função que obtém uma lista e um nome de arquivo, onde ela será salva em formato de conjunto.

2) Pacote “relação”:

- **Módulo “__init__.py”, “classe.py” e “conversor.py”:**

No pacote relação, um de seus módulos é o “__init__.py”, que contém funções para obter uma relação de um arquivo, para salvar uma relação em um arquivo e para salvar relações na forma cíclica.

Em seu módulo “classe.py”, existe uma função que obtém e retorna o conjunto de uma classe de equivalência da relação passada como argumento.

No módulo “conversor.py”, há uma função que verifica se é possível converter uma relação para a forma cíclica e outras funções responsáveis por converter a relação em matricial e cíclica.

- **Módulo “diagrama.py”:**

O módulo “diagrama.py” é responsável por gerar o diagrama de Hasse. Ele contém uma função que obtém uma relação e retorna uma lista com todas as retas do diagrama em formato de coordenadas e outra que se utiliza dessa para gerar um arquivo de imagem baseado nesses dados. Ele também contém uma função que verifica se existe um par entre dois outros pares que faz uma conexão entre eles e uma que retorna o ponto final de uma reta baseando-se no seu ponto inicial e em seu ângulo. Por fim, há também uma função que retorna a quantidade de pares com um determinado X na relação e outra que ordena a relação com base nessa quantidade.

- **Módulo “ordenador.py” e “propriedades.py”:**

No módulo “ordenador.py”, há uma função para ordenar a relação lexicograficamente em vez de utilizar o método pré-existente “sort”, que faz isso automaticamente. Para isso, ele contém uma função que compara as strings para verificar qual é a menor, utilizando os ID 's de seus caracteres na tabela ASCII. Há também uma função para comparar valores do mesmo tipo, numérico ou string, e verificar se um é menor ou igual ao outro.

Por fim, o código possui o módulo “propriedades.py”, que verifica todas as propriedades da relação. Nele, existe uma função para cada propriedade a ser detectada e uma função que retorna um dicionário com todas as propriedades da relação.

Demonstração do programa:

1) Arquivo “media.txt”:

Para demonstrar o funcionamento do programa, existem duas relações de exemplo referentes à estrutura de classes de um determinado projeto: o “gui.txt”, com as classes responsáveis pela interface gráfica, e o “media.txt”, com as classes responsáveis pela reprodução e gravação de áudio e vídeo.

Ao executar o programa, ele pedirá o nome do arquivo que contém a relação. Será utilizado, para este exemplo, o arquivo “media.txt”. Como os elementos da relação são strings, ele não perguntará ao usuário se ele quer verificar se a relação é de congruência. Primeiramente, ele dará todas as propriedades que a relação possui e as que ela não possui, incluindo o motivo pelo qual ela não possui tal propriedade.

Como a relação possui as propriedades reflexiva, antissimétrica e transitiva, ele perguntará se o usuário deseja gerar o diagrama de Hasse da relação. Digitando Y para gerá-lo, é possível ver a estrutura de classes no diagrama. A sua visualização fica muito mais fácil no diagrama do que direto no código. É perceptível, por exemplo, que a classe “VideoEncoder” herda de “Encoder”, que “VideoDecoder” herda de “Decoder” e que “MP4” herda de “VideoEncoder” e “VideoDecoder”. Com essas informações dá para fazer polimorfismo, utilizando a referência do tipo “Encoder” para um objeto de “MP4”.

De volta ao programa, ele perguntará se o usuário deseja gerar a relação em ordem lexicográfica. Ao digitar Y, ela será gerada. Além disso, ele também perguntará se o usuário quer gerar a forma matricial da relação. Da mesma forma, ela será gerada ao digitar Y. Em um arquivo a relação estará totalmente ordenada e em outro ela estará organizada na forma matricial.

2) Arquivo “gui.txt”:

Ao fazer o mesmo procedimento para a relação “gui.txt” e gerar o seu diagrama de Hasse, é possível ver nitidamente a estrutura de classes referente à interface gráfica do projeto. É visível, por exemplo, que “Style” herda de “Core” e que “Gradient” herda de “Style”. Torna-se perceptível, também, que “GradientFrame” herda de “Gradient”, e isso é um problema de alto acoplamento, já que o “GradientFrame” herda de “Core”, que é uma classe mais relacionada ao sistema da interface gráfica, e também herda de “Frame”, que por sua vez herda de “Widget”. Evidentemente, usar o diagrama de Hasse torna a detecção desses problemas muito mais fácil. Com essas ferramentas, o trabalho do programador citado no começo fica muito mais simples e organizado.

3) Demais recursos com relações numéricas:

O programa também possui outros recursos que serão exemplificados utilizando algumas relações numéricas. Primeiramente, será utilizada uma relação que supostamente é uma relação de equivalência e de congruência módulo 5. Após escrever o nome do arquivo “equiv_congr_mod_5.txt”, o programa perguntará ao usuário se ele quer verificar a sua congruência. Será verificada aqui a congruência módulo 5. O programa retornará as propriedades da relação, que é reflexiva, simétrica e transitiva, e portanto é de equivalência. Ele também informará que essa é uma relação de congruência. Logo abaixo, ele dará uma opção antes indisponível, que é a de obter a classe de equivalência de um valor. Digitando sim e, por exemplo, a classe de equivalência de 16, ele gerará um arquivo com o conjunto desta classe de equivalência.

Por fim, será utilizada uma relação que é supostamente uma permutação. Após colocar o nome do arquivo “permutation.txt”, é possível verificar, por fins de exemplificação, se ela é uma relação de congruência módulo 3. No caso, ela não é uma relação de equivalência e não é uma relação de congruência módulo 3. Para essa relação, a opção de gerar a forma cíclica aparecerá, indicando que essa relação é realmente uma permutação. Ao digitar Y ela será gerada.

Referências bibliográficas:

- Como referência e base dos conceitos usados neste projeto, foi utilizado o livro Fundamentos Matemáticos para a Ciência da Computação, de Judith Gersting;

Créditos:

- Jean Loui: Criação dos módulos app.py, conjunto.py, __init__.py, diagrama.py e ordenador.py; Criação do problema do projeto e seus arquivos de exemplo; Criação de roteiro; Criação de slides; Edição de vídeo; Apresentação.

- Victor Manoel: Criação do módulo propriedades.py; Criação de roteiro; Criação de slides; Apresentação

- Guilherme A. Motta: Criação do módulo classe.py e conversor.py; Organização e correção do roteiro; Criação da documentação em PDF; Apresentação.