Trabalho Prático de Matemática Discreta

Relações Binárias

Aluno: Roberto Gomes Rosmaninho Neto
N°: 2018054940
Turma: 2018/1

1. Introdução

Seja o conjunto A. Uma relação binária R de A para A é um subconjunto de A × A, ou seja, um subconjunto do produto cartesiano de A por A. Dado um par ordenado (x, y) em A × A, x está relacionado com y através de R, escrito xRy, se e somente se (x, y) pertence à relação R. O termo binário é usado nessa definição para referir ao fato que uma relação é um subconjunto do produto cartesiano de dois conjuntos. Uma relação de A para A pode ser representada por um grafo dirigido. Nesse caso, ao invés de representar A como dois conjuntos separados de pontos, representa-se A somente uma única vez e desenha-se uma aresta de cada ponto de A para cada ponto relacionado.

2. Implementação

O Programa foi escrito em sua totalidade na função Main, no entanto os procedimentos de cada operação foram divididos de acordo com a necessidade de saída e/ou do uso para outro procedimento.

LEITURA

A leitura do Arquivo de Teste foi feita por meio da função fopen() que recebe como parâmetros um arquivo que o usuário fornece no momento que realiza a inicialização do programa e o comando para leitura "r".

CONJUNTO R

O Conjunto de Elementos do problema foi lido e armazenado na TAD (Tipo Abstrato de Dados) Vetor unidimensional de tamanho n fornecido pelo usuário como primeiro elemento do arquivo, os n elementos posteriores são os elementos presentes no Conjunto R.

LEITURA E ARMAZENAMENTO DAS RELAÇÕES BINARIAS NO CONJUNTO R

Cada relação foi dada em uma linha do arquivo e, também, por linha foi lido. Para registrar as relações foi utilizada uma matriz booleana quadrada de tamanho n, tamanho do Conjunto R, onde cada linha corresponde ao índice do primeiro elemento no Vetor do Conjunto R e cada coluna ao do índice segundo elemento do par ordenado no Vetor do Conjunto R.

Dessa forma, foi possível criar uma matriz com o menor tamanho possível com todos os elementos do Conjunto R representados.

Toda a manipulação de dados foi feita por meio do índices correspondentes aos elementos do Conjunto R, assim sempre que fosse necessário uma comparação na matriz booleanda ou realizar uma saída para o usuário, esses dois TAD's eram usados simultaneamente.

Caso haja uma relação entre dois elementos x,y os índices destes no conjunto R serão os mesmos índices na matriz booleana.

Exemplo de aplicação:

Imagem 1

ESTRATÉGIA DE VERIFICAÇÃO DAS PROPRIEDADES DAS RELAÇÕES

Neste trabalho, em todos os procedimentos de verificação, utilizei o princípio da contagem de possibilidades de relações e comparação com as relações fornecidas pelo usuário para determinar se uma propriedade era falsa ou verdadeira.

Para isso, nas propriedades Reflexiva, Simétrica e Transitiva o fecho foi declarado antes da verificação de existência, apesar de sua saída para o usuário ter sido realizada somente no final, como dado no exemplo da documentação fornecida pelo Professor.

Para a implementação destas Propriedades foram utilizadas 4 estratégias diferentes, além das citadas acima.

1. Propriedades Relacionadas e Mutualmente Excludentes: Reflexiva e Irreflexiva

   Para verificar a validade de cada uma das propriedades foi necessário armazenar em uma matriz auxiliar todas relações dadas que satisfazem a primeira propriedade até um índice *aux*, depois deste as relações que não satisfaziam a primeira, mas satisfaziam a segunda propriedade eram armazenados, e assim, utilizando a variável *aux* e uma estrutura *if-else* foi possível verificar a validade de cada propriedade e imprimir o que não satisfazia cada uma.

2. Propriedades não Relacionadas de Implementação Simples: Simetria e Anti-Simetria

   Caso a propriedade fosse verdadeira a saída para o usuário era imediata, no entanto, caso fosse falsa, todos as relações que tornavam essa propriedade falsa eram armazenadas em uma matriz auxiliar e, então a saída na tela era realizada mostrando que a propriedade não era verdadeira e o que a invalidava.

3. Propriedades que são Consequência da Validade de outras: Assimetria, Equivalência e Ordem Parcial

   Nestes casos, apenas foi necessário verificar se as propriedades necessárias para validade destas eram verdadeiras ou não, para isso cada propriedade possuía uma variável *char* correspondente que recebia o valor ‘V’ ou ‘F'dependendo de sua validade.

4. Propriedade Complexa e não Relacionada: Transitividade

Para este caso, o fecho da relação foi criado a partir da multiplicação da matriz booleana n vezes, assim, ao final da operação, todos os elementos x,y diferentes de 0 eram parte do fecho transitivo do exemplo. Para verificar se a relação era transitiva ou não, bastou comparar a matriz booleana inicial e o fecho transitivo os elementos do Conjunto R de índices x,y na matriz booleana que possuírem situações diferentes nestas duas matrizes eram armazenados em uma matriz auxiliar e mostrados ao usuário após a mensagem de não validade da relação.

Para melhor explicar essa estratégia, vou utilizar o exemplo dado na Imagem 1.

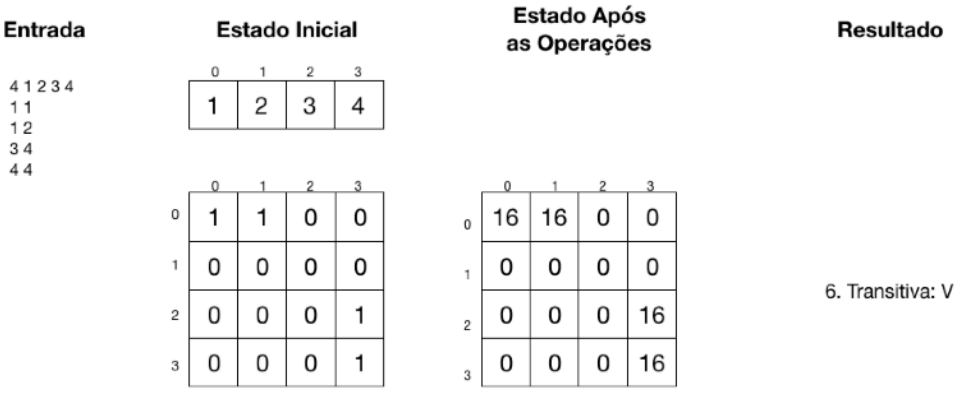


Imagem 2 - Relação Transitiva por *default*

ORGANIZAÇÃO DO CÓDIGO E DETALHES TÉCNICOS

O Código está inteiramente em um único arquivo principal: rb.c como orientado pelo Professor.

O compilador utilizado foi o GCC versão 4.2.1 no Sistema Opereacional MacOS High Sierra Versão 10.13.4.

Para compilar este Código basta ir no diretório do arquivo rb.c e do arquivo de teste, neste exemplo ‘entrada.txt’. Compilar no Terminal por meio da linha de comando “gcc rb.c -o main” e, criado o programa main, executar passando o arquivo entrada.txt como parâmetro “./ main entrada.txt”.

3. Testes

Vários Testes foram realizados para verificar o funcionamento do Programa, nesta documentação serão exemplificados o teste presente no documento fornecido pelo professor e o exemplo presente nas imagens 1 e 2.

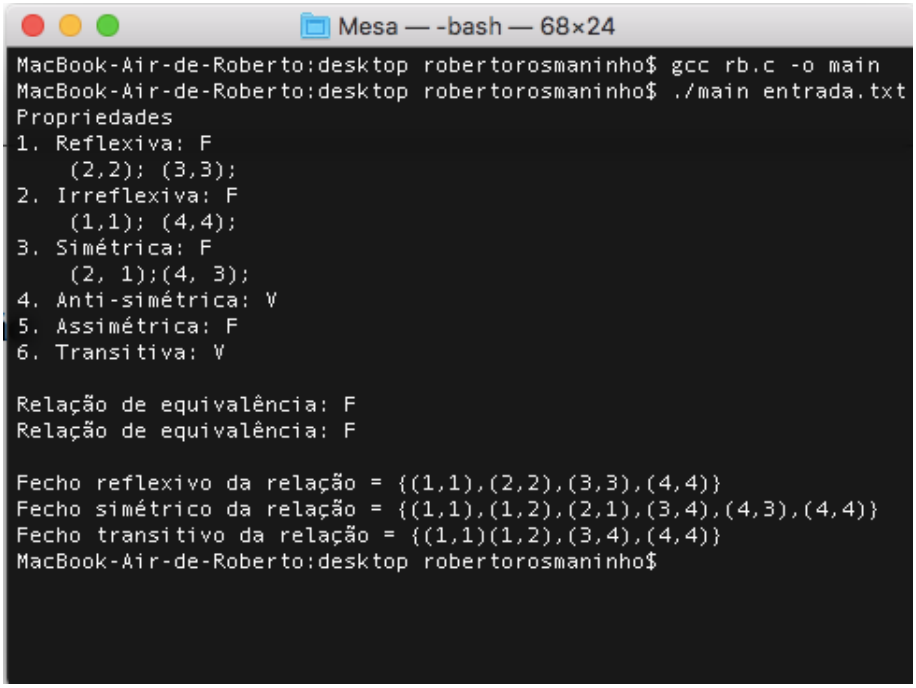
Estes testes foram realizados em um MacBook 2015, i5 de 5geração e 128Gb de SSD.

EXEMPLO 1:

ENTRADA:

```
4 1 2 3 4
1 1
1 2
3 4
4 4
```

SAÍDA:

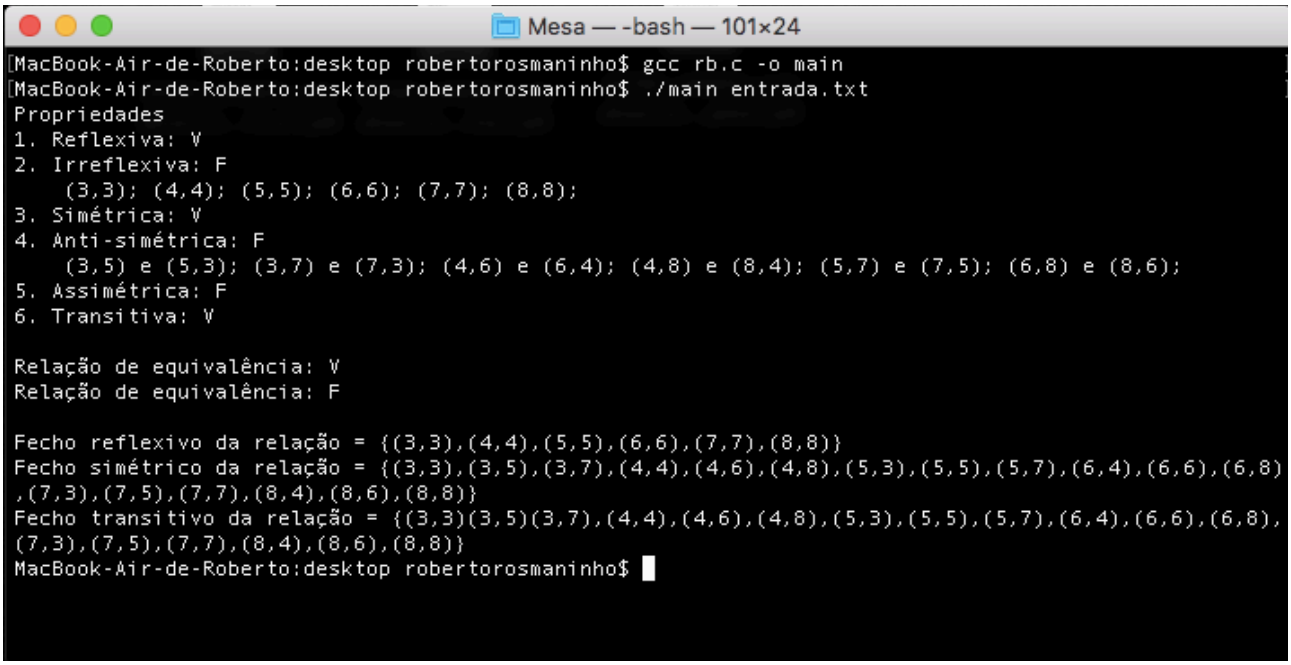


EXEMPLO 2:

ENTRADA:

```
6 3 4 5 6 7 8
3 5
5 7
7 3
5 3
7 5
3 7
4 6
6 8
8 4
6 4
8 6
4 8
3 3
4 4
5 5
6 6
7 7
8 8
```

SAÍDA:



4. Conclusão

Neste trabalho foi possível aprimorar os conhecimentos de programação na linguagem C e aplicar de forma prática os conhecimentos sobre Conjuntos e Relações adquiridos em sala de aula. Após a finalização do trabalho, em sua fase de testes, foram utilizados os exemplos dados pelo Professor nos Slides e nas Listas de Exercícios disponibilizadas na Página Pessoal desse, e sem maiores problemas todos os testes obtiveram o resultado esperado. A maior dificuldade apresentada durante a implementação do problema foi criar funções para o problema sem que ocorressem grandes redundâncias no código, por isso optei por escrever todo o código dentro da função principal, main.