Matemática Discreta - Trabalho Prático

Amanda Cristina Rodrigues Tavares

21/09/2020

Implementação

Como é necessário ler o arquivo teste (que é um .txt), o caminho do arquivo é passado como argumento para a função. A primeira linha do arquivo é quem define como será a matriz, sendo o primeiro número quem define tanto o número de colunas quanto o número de linhas. Por ser necessário ter controle de quando pula linha, foi usado $fgetc$ até que se encontrasse o final do arquivo.

Cada char lido é passado para uma variável. É esperado que após ler um algarismo (um char que seja diferente de espaço ou final do arquivo ou nova linha ou tabulação) ele seja precedido por um espaço para assim formar o número completo que será tratado ao montar a matriz. Cada algarismo é inserido num array auxiliar. Quando é encontrado um char diferente de um algarismo, o array é desmontado e o número é tratado de acordo com outras relações.

A estrutura da matriz é montada de forma que a primeira linha e coluna irá conter todos os números que fazem parte do problema. Sendo assim, a matriz tem linhas e colunas igual a $n+1$. As próximas linhas após a primeira vão definir como popular a matriz por indicar a relação entre os números (inserindo 1 na linha e coluna), sendo que o primeiro número seria a posição linha, e o segundo número a posição coluna na matriz. É usada uma função para encontrar o verdadeiro index de onde está cada número dentro da estrutura da matriz, isto é, qual a sua posição na primeira linha, que consequentemente é sua posição na coluna. Assim, é possível popular a matriz corretamente com as relações.

Algumas análises precisam que seja impresso os pares ordenados que a tornam Falsa, então foi criado uma struct para representar o par e uma matriz (com 6 linhas e n^2 colunas) que terá todos os casos a serem testados. Para cada caso existe um contador, que é incrementado quando se acha um par que torna o teste do caso Falso.

Cada caso foi analisado a partir das informações dadas previamente para resolver o problema. Para imprimir os testes, verifica se o contador daquele caso é igual a 0, que determina que é verdadeira a propriedade. Caso contrário, imprime que é falso e imprime os pares dependendo do caso. Já a relação de equivalência e ordem parcial verifica os casos anteriores. E para imprimir o fecho transitivo, a matriz é transformada para sua versão transitiva se for necessário.

Análise de Complexidade

A parte do algorítmo responsável pela leitura do arquivo tem taxa de crescimento variada. Na parte de desmontar o número, a taxa depende do tamanho do número (sendo que o limite é de 6 algarismos), portanto é $O(n)$. Já a montagem da matriz é $O(n+1)$, mas ocupa espaço de $(n+1)^2$ por ser uma matriz com número de linhas e colunas igual. A parte para encontrar o index do número na montagem das relações é $O(n)$, já que é preciso encontrar na primeira linha da matriz o índice dos dois números que compoem a relação.

A montagem da matriz com as propriedades é $O(n)$, mas ocupa espaço n^2, já que o número de colunas é n^2.

A análise da propriedade reflexiva e irreflexiva é de crescimento $O(n)$, já que precisa validar apenas

as diagonais (olhar a posição matriz[i][i]).

A análise da propriedade simétrica e anti-simétrica é de crescimento $O(\log_n n^2)$, já que precisa percorrer todas as linhas e colunas, verificar qual é o par não simétrico e validar se já não existe o par encontrado dentro da matriz de propriedades.

A análise da propriedade assimétrica é de crescimento $O(n^2)$, pois percorre todas as linhas e colunas para fazer sua validação.

A análise da propriedade transitiva é de crescimento $O(\log_n n^3)$, pois percorre três vezes o tamanho da matriz e ainda valida se o par encontrado já não pertence a matriz de propriedades.

A impresão do resultado de boa parte dos testes é de crescimento $O(n)$, exceto quando é necessário transformar a matriz em transitiva, tendo taxa de crescimento $O(n^3)$, e taxa $O(n^2)$ para a impressão do fecho transitivo por percorrer a matriz inteira.