Relação

A ideia para a resolução do problema proposto se baseou muito nas definições de cada relação binária apresentada na especificação do trabalho e na sugestão dada em uma das aulas no início do ano de se encarar essas ligações por meio de uma matriz de adjacência.
Após lido do arquivo de entrada os nós e as ligações entre eles, bastou criar funções do tipo char que retornavam “V” ou “F” para cada tipo de relação.

Conforme comentado no código, para testar se a relação era reflexiva, bastou checar se a matriz de adjacência possuia todos os elementos da sua diagonal principal iguais a 1. Aqueles que fossem iguais a 0 representariam o par de elementos que faltavam para tornar a relação reflexiva.
No caso da irreflexibilidade, algo semelhante ao teste da reflexibilidade foi feito, no entanto invertia-se os valores. Checava-se se todos os elementos da diagonal principal eram iguais a 0, e aqueles que fossem iguais a 1 representariam o par de elementos que faltavam para tornar a relação reflexiva.
O teste quanto a simetria em essência checa se a matriz de adjacência é igual a sua transposta. Os pares y e x que deveriam ter uma relação também de x e y mas não o tinham, eram os pares que faltavam para tornar a relação simétrica.
Quanto ao teste de anti-simetria, bastou checar na matriz de adjacência se sempre que se tinha uma ligação de x para y e y para x, x=y. Os pares x e y que não satisfaziam essa condição eram aqueles que impediam a relação de ser anti-simétrica.
Para checar se a relação era assimétrica bastou percorrer toda a matriz de adjacência verificando se quando existia uma ligação de x para y, não existia uma ligação de y para x.
O caso do teste da transitividade precisou do uso de três loops aninhados do tipo for para percorrer toda a matriz de adjacência checando se quando existia uma ligação de x para y e y para z, também existia uma ligação de x para z. Os pares x e z que não possuiam uma relação na matriz de adjacência eram aqueles que faltavam para tornar a relação transitiva.
Para saber se a relação era de equivalência, bastou utilizar as funções enunciadas previamente que testavam se a relação era reflexiva, simétrica e transitiva simultâneamente. Enquanto para saber se era de ordem parcial bastava chamar as mesmas funções, no entanto utilizando a função de teste de anti-simetria no lugar da de simetria.
O fecho transitivo da relação sempre será dado pelo conjunto que contém os pares existentes adicionado do menor conjunto que contém os pares restantes para tornar a relação transitiva. Pensando nisso, criou-se uma matriz auxiliar de adjacência que continha a ligação entre os pares x e z que faltavam para tornar a relação transitiva, conforme mencionado anteriormente. Assim, a soma da matriz de adjacência com a matriz auxiliar de adjacência resulta na matriz cujas ligações entre os pares dará o fecho transitivo da relação.

Análise de complexidade

O algoritmo sempre tem que performar os mesmos testes para qualquer entrada com n nós distintos. Assim, os maiores custos a nível de tempo de execução do programa são relacionados a função que testa a transitividade e a parte que cria e preenche a matriz auxiliar de adjacência, pois em ambas existem três laços aninhados do tipo for que vão sempre de 0 até n (número de nós). Então, o custo do programa como um todo a nível de tempo de execução é da ordem de $\boldsymbol{O(n^3)}$ operações.

Quanto a análise da complexidade de espaço, é possível perceber que além das diversas variáveis e vetores que são inicializados, as matrizes são as que mais acabam aumentando o custo de espaço, pois possuem sempre nxn dimensões. Assim, o custo do programa como um todo a nível de espaço é da ordem de $\boldsymbol{O(n^2)}$.