O código-fonte foi separado em em módulos assim será mais fácil analisar, documentar e manter toda a estrutura. Detalharemos agora cada módulo (função) do código-fonte:

→ aloc_int

Essa função irá alocar dinamicamente a matriz principal do programa. Isso significa dizer que ela irá ter um custo assintótico de espaço de $O(n^2)$, pois terei uma matriz *nxn* onde será alocado cada posição uma única vez.

Como o procedimentos para a alocação requer apenas uma passagem de 0 a n-1, em tempo teremos então, o custo de $O(n)$. (Obs.: a matriz é alocada com “calloc”, isso significa que ela já está preenchida inteiramente com o valor “0”.)

O custo assintótico para essa função é,

em tempo,

$$aloc(n)_t = O(n)$$

em espaço,

$$aloc(n)_e = O(n^2)$$

→ free_aloc_int

Essa função irá deslocar a matriz principal do programa, para isso teremos o processo inverso descrito em “aloc_int”, então em tempo também teremos $O(n)$, como não há nenhum espaço alocado nesse módulo, o custo será constante isso é, $O(1)$.

O custo assintótico para essa função é,

em tempo,

$$free(n)_t = O(n)$$

em espaço,

$$free(n)_e = O(1)$$

→ rlc_binary

Essa função é o esqueleto de todo o código fonte, ela colhe todos os valores passados no arquivo em um vetor único *“numbers_arc” (*que possui um custo de espaço constante $O(1)$). Após isso ela irá preencher somente a primeira linha e primeira coluna da *“matriz_aloc”* (to custo dessa matriz de espaço já foi detalhado e definido em *“aloc_int”*, o custo de tempo será detalhado posteriormente) com os valores passados como argumento na primeira linha do arquivo. A exemplo teremos:

<u>Arquivo de entrada:</u>

3 1 2 3
1 1
2 2
3 3

A *“matriz_aloc”* terá o seguinte formato inicial: (Obs.: como ela foi alocada com “calloc” ela já está preenchida com 0)

0 1 2 3
1 0 0 0
2 0 0 0
3 0 0 0

Apos isso teremos um *“vetor_aux”* no qual será armazenado apenas os pares ordenados, esse vetor é linear e alocado dinamicamente consumindo um espaço de $O(n)$ e tempo de $O(1)$ pois como é uma alocação linear, o tempo será constante independente da entrada. Assim quando obtivermos esse vetor, teremos os vetores *“par_eixo_x”* e *“par_eixo_y”* onde ficarão respectivamente os valores de x e y de forma sincronizada, ou seja, a posição 0 dos dois vetores terá exatamente no nosso exemplo de entrada os valores (1,1), esses dois pares terão o mesmo custo de tempo e espaço do *"vetor_aux"*, já definidos anteriormente.

Por fim, os vetores *"par_eixo_x"* e *"par_eixo_y"* irão percorrer toda a matriz *"matriz_aloc"* computando os pares e preenchendo-os com o valor "1", desse modo ficaremos com a seguinte matriz:

0 1 2 3
1 1 0 0
2 0 1 0
3 0 0 1

Note que para o preenchimento dessa matriz, é necessário passar *n vezes* em cada posição da minha matriz *nxn*, logo o custo assintótico de tempo gasto nessa parte será $O(n^3)$.

O custo assintótico desse módulo é dado por,

Em tempo,

$rlc(n)_t = O(n^3)$

Em espaço,

$rlc(n)_e = O(n^2)$

→ reflexiva

Definição: $R é reflexiva \Leftrightarrow \forall x em A, (x,x) \in R$

Essa função verifica se a diagonal principal está inteiramente preenchida com o valor "1".

Como é necessário verificar cada posição da matriz nxn, teremos um custo assintótico de tempo representado por $O(n^2)$ e como não há nenhuma alocação teremos custo constante de espaço, ou seja, $O(1)$.

O custo assintótico para essa função é:

em tempo,

$ref(n)_t = O(n^2)$

em espaço,

$ref(n)_e = O(1)$

→ irreflexiva

Definição: $R é irreflexiva \Leftrightarrow \forall x em A, (x,x) \notin R$

Essa função verifica se a diagonal principal possui algum valor diferente de "1".

Custo assintótico de tempo e espaço já detalhados em *"reflexiva"*.

O custo assintótico para essa função é:

em tempo,

$irr(n)_t = O(n^2)$

em espaço,

$irr(n)_e = O(1)$

→ simetrica

Definição: $R é simetrica \Leftrightarrow \forall x e y em A, se (x,y) \in R, então (y,x) \in R$

Essa função verifica se o elemento da matriz[i][j] == matriz[j][i] para toda a matriz.

Custo assintótico de tempo e espaço já detalhados em *"reflexiva"*.

O custo assintótico para essa função é,

em tempo,

$sm(n)_t = O(n^2)$

em espaço,

$sm(n)_e = O(1)$

→ anti_simetrica

Definição: $R é anti\text{-}simetrica \Leftrightarrow \forall x e y em A, se (x,y) \in R e (y,x) \in R, então x = y$

Essa função verifica se o elemento da matriz[i][j] != matriz[j][i] a menos que i e j sejam iguais.

Custo assintótico de tempo e espaço já detalhados em *"reflexiva"*.

O custo assintótico para essa função é,

em tempo,

$anti(n)_t = O(n^2)$

em espaço,

$anti(n)_e = O(1)$

→ assimetrica

Definição: $R é simetrica \Leftrightarrow \forall x e y em A, se (x,y) \in R, então (y,x) \notin R$

Essa função verifica se o elemento da matriz[i][j] != matriz[j][i] . Se i = j, então não pode haver conexão.

Nesse caso, como há somente verificação de condicionais, custo de tempo e espaço são constantes.

O custo assintótico para essa função é,

em tempo,

$ass(n)_t = O(1)$

em espaço,

$ass(n)_e = O(1)$

→ transitiva

Definição: $R é transitiva \Leftrightarrow \forall x, y, z em A, se (x,y) \in R, e (y,z) \in R, então (x,z) \in R$

Essa função verifica se a $matriz[i][j]{=}{=}1 \wedge matriz[j][z]{=}{=}1 \wedge matriz[i][z]{=}{=}1$

Como precisamos analisar as posições *x,y, z* em uma matriz *nxn* é necessário passar *n vezes* por toda a matriz, isso significa que teremos um custo de *nxnxn* o que pode ser representado por $O(n^3)$.

O custo assintótico para essa função é,

em tempo,

$tran(n)_t = O(n^3)$

em espaço,

$tran(n)_e = O(1)$

→ main

A função main irá efetuar os procedimentos para que a saída esperada seja executada, isto é, analisar se a entrada de dados corresponde a uma das relações propostas no trabalho. Caso seja falso, ela irá imprimir os pares ordenados que seriam capazes de tornar aquela entrada verdadeira.

Detalhando o custo assintótico no bloco principal notamos que nada é alocado diretamente nele, isso acontece pois temos um módulo especifico que aloca a matriz, com isso o custo de espaço do modulo main é constante, representamos isso como $O(1)$. Em tempo, com a exceção da relação *"assimetrica"* ,*"relação de equivalência"* e *"relação de ordem parcial"*, todas as outras relações caso seja *falso* é necessário verificar as posições da matriz *nxn* para imprimir os pares ordenados que fariam aquela sentença ser verdadeira, com isso temos um custo de tempo de $O(n^2)$, entretanto, na relação de transitividade, como trabalhamos com *três variáveis (x,y,z)* é necessário repetir o processo descrito acima *n vezes*, fazendo com que o custo de tempo dela especificamente seja $O(n^3)$.

O custo assintótico para essa função é,

em tempo,

$main(n)_t = O(n^3)$

em espaço,

$main(n)_e = O(1)$

Então, concluímos que o custo assintótico no pior caso do código-fonte *"relacao.c"* de tempo e espaço respectivamente é da ordem de:

$relacao(n)_t$**= *O(n³)***

$relacao(n)_e$= ***O(n²)***