Vinícius do Amaral Brunheroto
Atividade 4 – ED1

```
const MaxNós= …..;
tipo Ptr = *vet  //ponteiro que aponta para um vetor de inteiros
tipo Ptr2 = *inteiro;   //ponteiro que aponta para um inteiro
Adjmatriz = vetor [MaxNós, MaxNós] de lógico;
vet = vetor [MaxNós] de inteiro;


//Adj: matriz de Adjacências
//i e j: nós de interesse

//indice é um ponteiro que aponta para o endereço do inteiro definido no programa principal
função Retorno_triplo (Adj:AdjMatriz;i;j;indice:Ptr2);

var k=0,p=0 ,q=0, r=0, L=0,M=0,flag=0: inteiro;  //inicializados com zero
ADJL,ADJM: Adjmatriz
aux1=0,vetor_final=0  : vet   //vetores são inicializados com todas as posições tendo o valor 0


começo


//Passo 1: criar vetor auxiliar 1 que contém todos os caminhos que existem entre J e I

  ADJL = ADJ;
   para L = 1 a MaxNós fazer
      começo
         se (ADJL [j, i] == 1)
               aux1[k]= L
               k=k+1
         ADJL = Prod_Bool (ADJL, ADJ);
       fim;
//Por estar dentro de um laço, supõe-se que o valor de L vá aumentando a cada iteração ( de 1 até N)

// Alocar dinamicamente o vetor final que deverá ser retornado para o programa principal
AlocDinam(vetor_final,MaxNós)


//Passo 2: verificar todos os caminhos que existem entre I e J e preencher o vetor final com os
comprimentos daqueles que possuem retorno triplo.

  ADJM = ADJ;
  para M = 1 a MaxNós fazer
    começo
   flag = 0 // reseta flag
```

```
        se (ADJM [i, j] == 1)
              q = 3*M
              p = 0 // resetar variavel aux p
              enquanto (flag == 0 e p <= MaxNós)
                    //verifica se o triplo do comprimento de I para J se encontra no vetor auxiliar
                     se (q == aux1[p])
                           vetor_final[r] = aux1[p]
                           r= r+1
                           flag=1
                               // se já encontrou o elemento uma vez, não precisa continuar procurando
               p=p+1; //alterando indice de p

        ADJM = Prod_Bool (ADJM, ADJ);
    fim;


//r possui a info da quantidade de elementos que foram colocados no vetor final
*(indice)= r
//muda-se o conteúdo do inteiro índice no programa principal

//Por estar dentro de um laço, supõe-se que o valor de M vá aumentando a cada iteração ( de 1 até
N)

return (vetor_final)
//vetor_final contém os comprimentos dos caminhos de I a J que possuem retorno triplo, esse vetor
//deve retornar ao programa principal

fim; //final da função



programa_principal:
var i, indice: inteiro
Pont: Ptr //ponteiro do tipo vetor de inteiros

começo
// (…)
 //ponteiro criado aponta para vetor_final que retornou da função
// variável índice passada por referência para saber quantos elementos foram colocados dentro do
//vetor_final

 Pont =  Retorno_triplo (Adj:AdjMatriz,i,j,Endereço(indice));

//Endereço(indice) indica que se quer passar o inteiro por referência
//com o valor de indice preenchido na função, printa-se apenas o que foi preenchido dentro do vetor
 para i = 0 a i = indice fazer
    printar(Pont[i])

//Após utilizar esse vetor, desalocar região que foi alocada dinamicamente
liberaRegião(vetor_final)
```

```
// (…)
fim;
```