Aluno: **Lucas João Martins** - Matrícula: **15100752**

Complexidade de algoritmos - Floyd Modificado

A seguinte disposição da numeração das linhas no algoritmo de Floyd Modificado foi utilizada para a realização dos cálculos:

```
01. FloydModificado()
02. início
03.    para i = 1 até n faça
04.        para j = 1 até n faça
05.            A[i,j] <- D[i,j];
06.            R[i,j] <- 0;
07.       fim para
08.    fim para
09.    para i = 1 até n faça
10.        A[i,i] <- 0;
11.    fim para
12.    para k = 1 até n faça
13.        para i = 1 até n faça
14.            para j = 1 até n faça
15.                se A[i,k] + A[k,j] < A[i,j] então
16.                   A[i,j] <- A[i,k] + A[k,j];
17.                   R[i,j] <- k;
18.           fim para
19.        fim para
20.    fim para
21. fim
```

Análise linha por linha (considere a sigla ut como unidade de tempo):

- Linhas 01, 02, 07, 08, 11, 18, 19, 20 e 21 consomem 0 ut;
- Linha 3 gasta 1 ut para atribuição de i, n + 1 ut para verificação e n ut para incremento do i (no total 2n

+ 2);

- Linha 4 gasta 1 ut para atribuição de j, n + 1 ut para verificação e n ut para incremento do j (no total 2n + 2);
- Linha 05 e 06 gastam 1 ut cada (2 uts), mas são executadas n vezes devido a linha 4 (2n uts), que por sua vez é executado n vezes devido a linha 3 ($2n^2$ no total)
- Linha 9 gasta 1 ut para atribuição de i, n + 1 ut para verificação e n ut para incremento do i (no total 2n + 2);
- Linha 10 gasta 1 ut, mas que é executada n vezes (1n ut no total);
- Linha 12 gasta 1 ut para atribuição de k, n + 1 ut para verificação e n ut para incremento do k (no total 2n + 2);
- Linha 13 gasta 1 ut para atribuição de i, n + 1 ut para verificação e n ut para incremento do i (no total 2n + 2);
- Linha 14 gasta 1 ut para atribuição de j, n + 1 ut para verificação e n ut para incremento do j (no total 2n + 2);
- Linha 15 é um condicional que consome 2 uts;
- Linha 16 e 17 gastam 3 uts (2 uts para a 16 e 1 ut para a 17) e são a maior expressão do condicional da linha 15. Além disso, esses 3 uts são executados n vezes devido a linha 14 (3n), que por sua vez é executado n vezes devido a linha 13 ($3n^2$), que por fim também é executado n vezes devido a linha 12 ($3n^3$ no total).

No fim, o algoritmo possui $T(n) = 3n^3 + 2n^2 + 13n + 14$ em unidades de tempo. Isso corresponde a uma complexidade assintótica de $O(n^3)$ (cúbica).