Nome e Registro Acadêmico dos Autores:

- André Augustho Sarti Trindade Silva - 2021.1.08.001
- Vinícius Eduardo de Souza Honório - 2021.1.08.024
- Ygor Jianjulio Nassif - 2020.2.08.002

A atividade consistia no desenvolvimento de um programa em C ou C++ capaz de ler um arquivo de texto contendo as coordenadas dos vértices de um grafo, montar sua representação computacional, encontrar a árvore mínima deste e encontrar seu maior subgrafo de 3 vértices.

O esboço seguinte ajuda ilustrar o método que foi utilizado para encontrar o maior subgrafo de 3 vértices:

MAIOR GRAFO DE 3 PONTOS

AreaMaior <- 0

Para todo {a, b, c | a,b,c pertencem aos Naturais; a,b,c <= Número de Vértices; a != b AND b != c AND a != c}

    Se AREA(a, b, c) > AreaMaior

        AreaMaior <- AREA(a, b, c)

AREA(a, b, c)

    retorna area

AREA OF A TRIANGLE

The area of a triangle with vertices P (x_1, y_1), Q (x_2, y_2), and R (x_3, y_3) is given by :

$$\Delta = \frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = \frac{1}{2}\left| x_1y_2 - x_2y_1 + x_2y_3 - x_3y_2 + x_3y_1 - x_1y_3 \right|$$

Já para árvore mínima, o raciocínio pensado foi:

ÁRVORE MÍNIMA

MatrizOrdenada[n * n]: Matriz que contém, em ordem crescente, todos as arestas do grafo, cada uma com seu par de vértices

ArvoreMinima[n-1]: Estrutura que armazenará cada uma das arestas que forma a árvore mínima do grafo

PaiAtual[n]: Lista com vértices cinzas

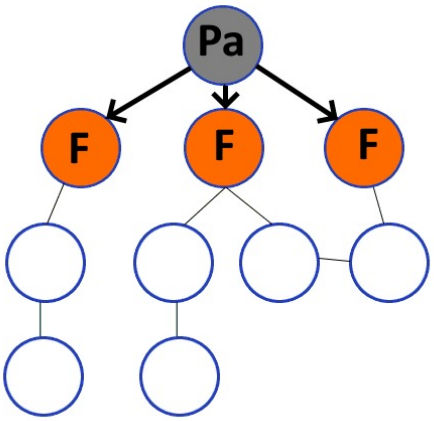
PaiAnterior[n]: Lista com vértices pretos

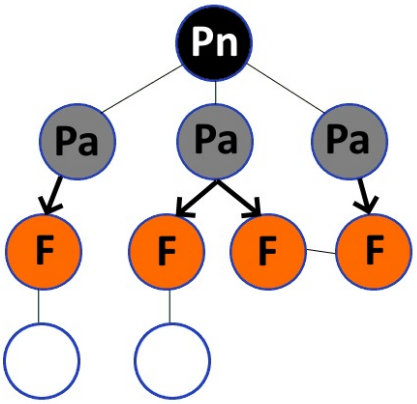
Fila[n]: Lista usada para fila

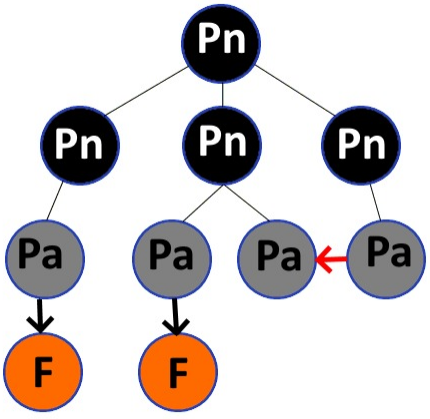
```
ArvoreMinima[1] <- MatrizOrdenada[1]
i <- j <- 2
enquanto i < tamanho(ArvoreMinima)
        ArvoreMinima[i] <- MatrizOrdernada[j]
        se VERIFICACICLO(ArvoreMinima) for falso
                i++
        caso contrário
                apaga(ArvoreMinima[i])
        j++

VERIFICACICLO(ArvoreMinima)
para cada vértice A contido em ArvoreMinima
        PaiAtual <- A
        Para cada vértice B contido em ArvoreMinima
                se éAdjacente(B,A) então
```

```
                    Fila <- Fila u B
        Enquanto Fila != Vazio
            PaiAnterior <- PaiAnterior u PaiAtual
            esvazia(PaiAtual)
            PaiAtual <- Fila
            esvazia(Fila)
            Para cada ponto B contido em ArvoreMinima
                se éAdjacente(B,PaiAtual) então
                    se B pertence a PaiAtual então
                        retorna verdadeiro // existe ciclo
                    se não pertence a PaiAnterior então
                        Fila <- Fila u B
retorna falso
```


Pa
F
F
F


Pn
Pa
Pa
Pa
F
F
F
F


Pn
Pn
Pn
Pn
Pa
Pa
Pa
Pa
F
F

O maior desafio do grupo foi encontrar uma forma de se fazer a verificação da presença ou ausência de ciclos na árvore mínima criada.

Um fato curioso foi de que a função de árvore mínima tinha sido implementada e sua lógica parecia impecável, porém persistia um erro ao tentar-se executar. Isso tudo foi devido a uma das repetições conter como limite de seu laço a expressão “l+1” em vez de “l-1”. Corrigido o problema: o código funcionou aparentemente sem erros.

Saídas Retornadas pelo Programa em cada Grafo Exemplo

a28

2 101 107

1073741824

att48

16 40 44

15876718

att532

1 87 95

1073741824

berlin52

21 49 32

1073205888

bier127

12 123 126

1073741824

brd14051

não compilou

“terminate called after throwing an instance of 'std::bad_alloc'

what(): std::bad_alloc”

burma14

0 7 3

22779020

d15112

não compilou

“terminate called after throwing an instance of 'std::bad_alloc'

what(): std::bad_alloc”

eil101

64 81 97

1073741824