08/12/20

Name: Davi Augusto Neves Leite RA: 191027383

EDII - 2ª Avaliação

(1) A ordenação externa consiste em ordenar arquivos de tamanho maior que a memória interna disponível. Dessa forma, os algoritmos associados devem diminuir o número de acesso as unidades de memória externa.

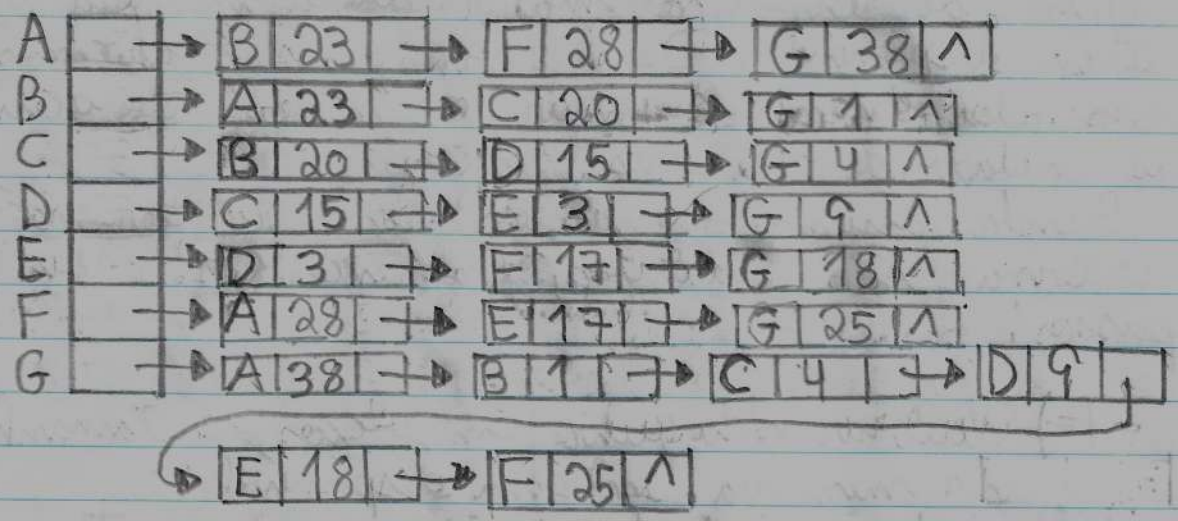
Sendo assim, os algoritmos para ordenação externa são pautados, geralmente, nos seguintes passos:

1º) Quebrar o arquivo em blocos do tamanho da memória interna disponível;
2º) Ordenar cada bloco na memória interna;
3º) Intercalar os blocos ordenados, fazendo varias passadas sobre o arquivo;
4º) A cada passada são criados blocos ordenados cada vez maiores, até que todo o arquivo esteja ordenado.

Resumidamente, essa estratégia básica é dividida em duas etapas: Classificar (gerar as partições ordenadas) e Intercalar (transformar as partições em um arquivo ordenado).

② a) A lista de adjacência, de par (vértice-adjacente / peso-da-aresta), para o grafo é:

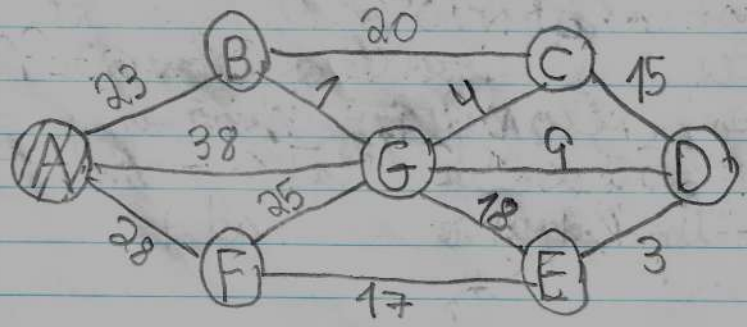
Lista



b) Um exemplo de percurso em profundidade, a partir do vértice A, pode ser:

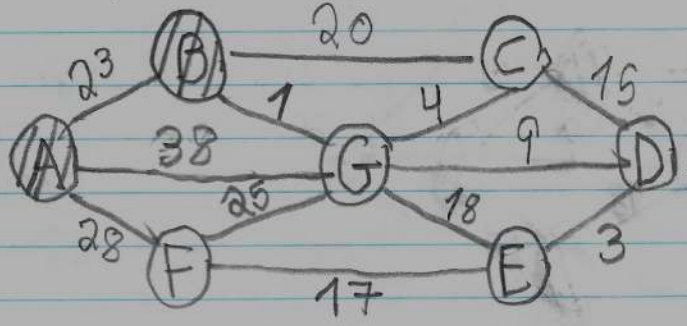
* Considerar: ○ - não visitado
◍ - visitado

1º)



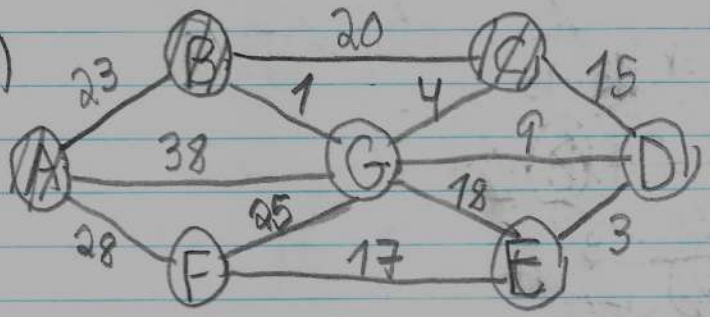
Percurso atual: A

2º)



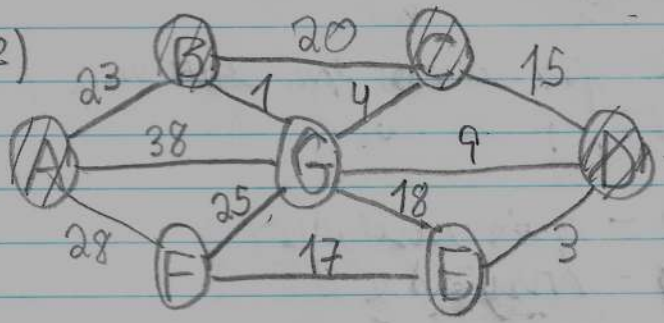
Percurso atual: A → B

3º)



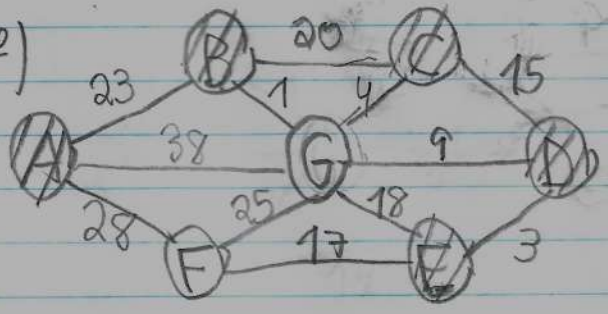
Percurso atual: A → B → C

4º)



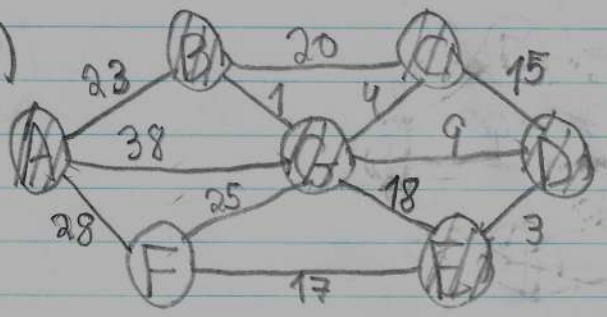
Percurso atual: A → B → C → D

5º)



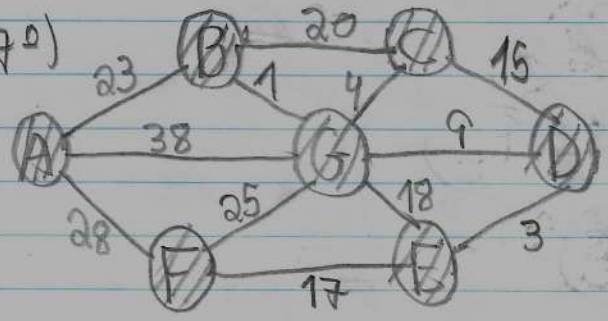
Percurso atual: A → B → C → D → E

6º)



Percurso atual: A → B → C → D → E → G

7º)

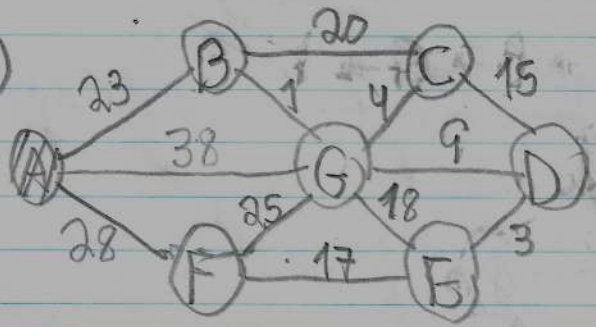


Percurso atual: A → B → C → D → E → G → F

c) Um exemplo de percurso em largura, a partir do vértice A, pode ser:

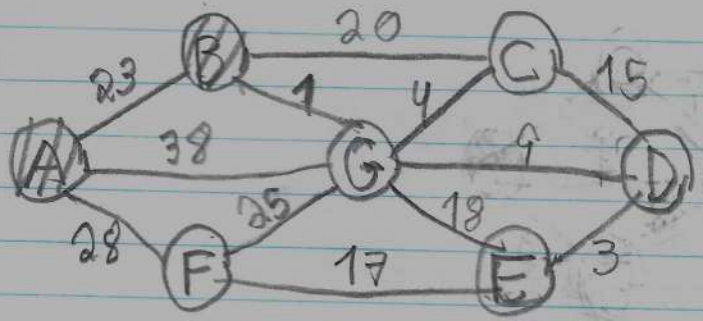
* Considerar: ○ - não visitado
◍ - visitado

1º)



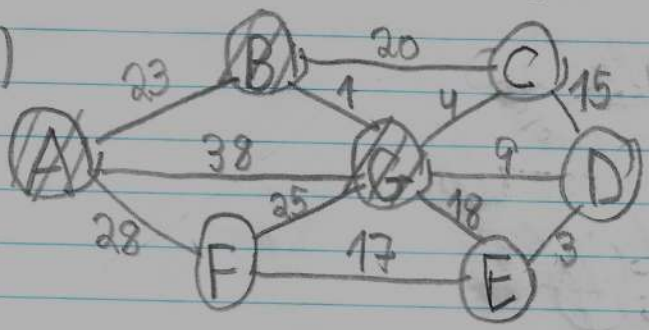
Percurso atual: A

2º)



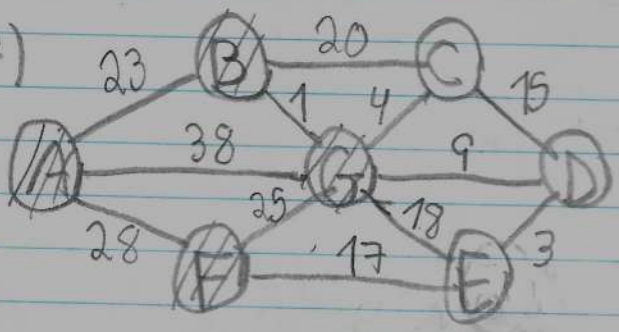
Percurso atual: A → B

3º)



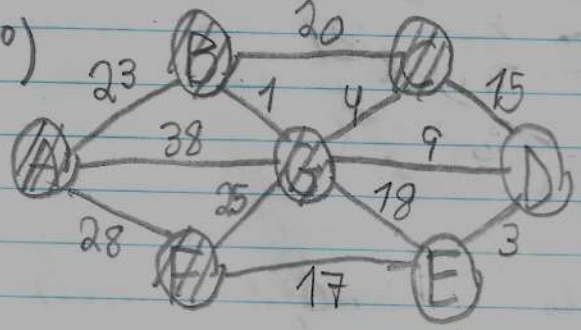
Percurso atual: A → B → G

4º)



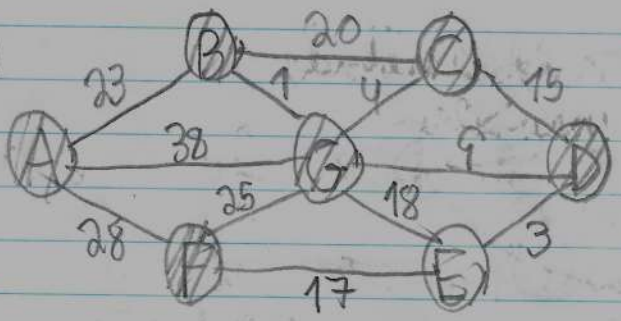
Percurso atual: A → B → G → F

5º)



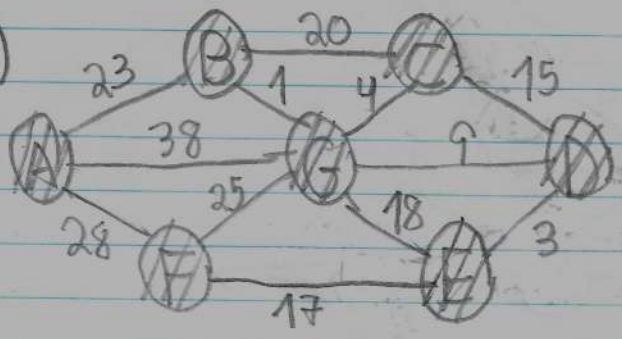
Percurso atual: A → B → G → F → C

6º)



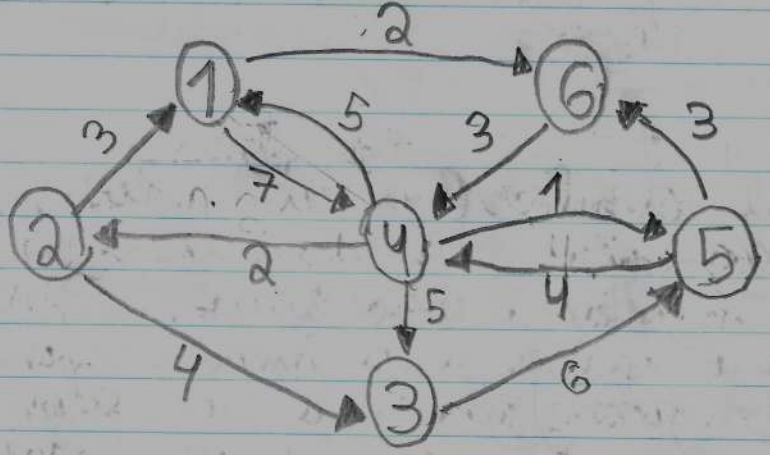
Percurso atual: A → B → G → F → C → D

7º)



Percurso atual: A → B → G → F → C → D → E

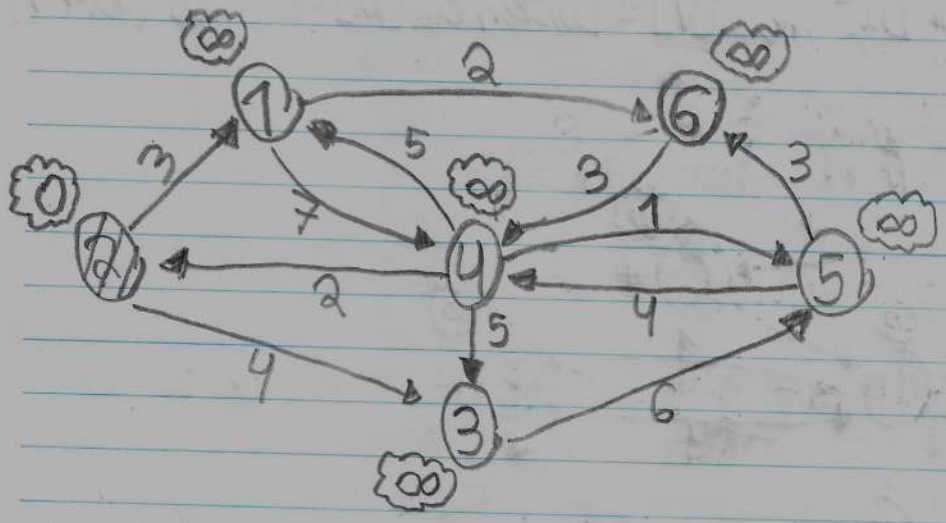
③ Grafo atual:



Considerando o vértice 2 como origem, tem-se os seguintes caminhos mínimos (aplicando Dykstra):

* Considerar: 0 - não visitados
1 - visitados
(X) - distância X até o vértice atual

1º) Iniciar o vetor distância com infinitos, exceto a posição inicial (valor 0), como mostra abaixo



Distância [] = { INF, INF, 0, INF, INF, INF, INF }

2º) Atualizar a distância dos vértices adjacentes somente se a distância do vértice atual (ou seja, até ele) mais o peso da aresta entre o vértice atual e o adjacente analisado for menor (caminho mínimo) do que a distância atual do vértice adjacente analisado.

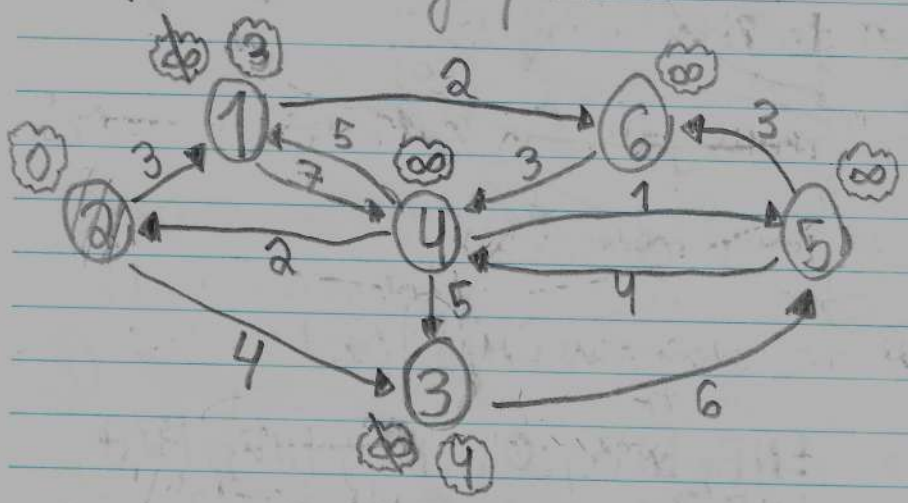
↳ Em outras palavras: seja "u" o vértice atual, tem-se o seguinte algoritmo:

⇒ Para cada vértice "V" adjacente a "u":

⇒ Se Distancia[u] + peso(u→V) < Distancia[V]:

⇒ Distancia[V] = Distancia[u] + peso(u→V)

Aplicando ao grafo:



Distancia[] = {INF, 3, 0, 4, INF, INF, INF}

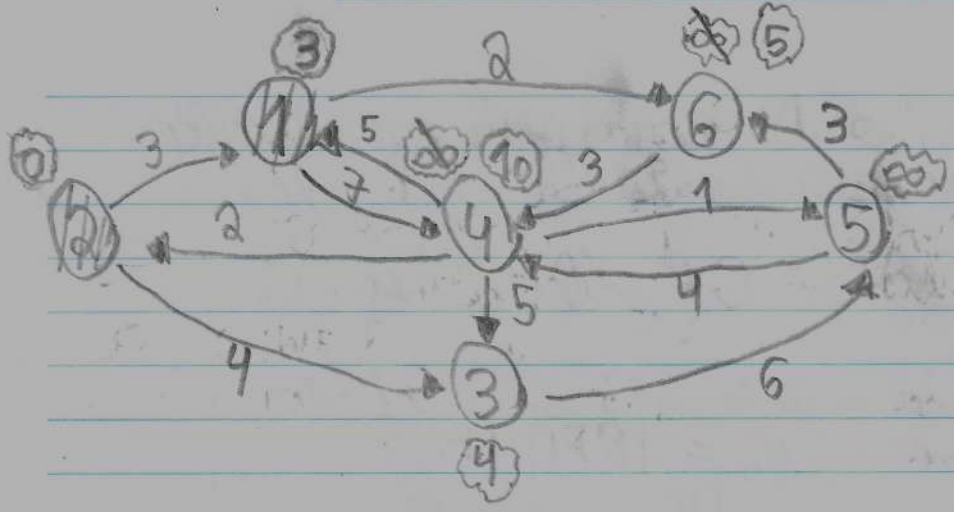
3º) Para otimizar a repetição necessária ao passo 2 (passando por todos os vértices adjacentes) e ter um critério de parada melhor definido, utiliza-se uma fila de prioridade que consiste em uma fila ordenada por prioridade (valores crescentes ou decrescentes).

Dessa forma, pode-se adicionar o par de (-distancia[adjacente], adjacente) na fila de prioridade (decrescente) e realizar o passo 2 a partir do valor de início da fila.

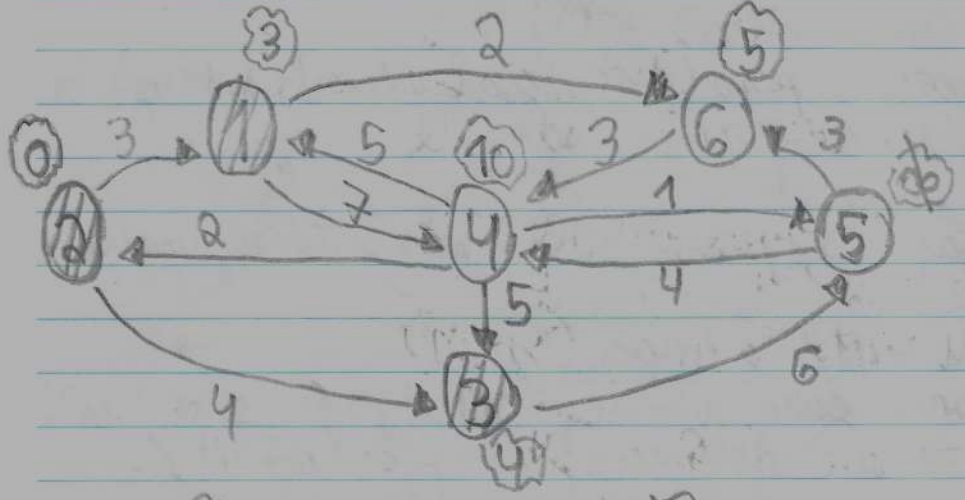
Dessa forma, o Dykstra pode ser descrito como o seguinte algoritmo:

- Dykstra (grafo G; vértice início):
  - Para cada vértice "u" em G:
    distancia[u] = INFINITO;
  - distancia[início] = 0;
  - Adicionar o par (distancia[início], início) a uma fila de prioridades Q
  - Repetição: enquanto Q não estiver vazia:
    - u = retira Início (Q)
    - Para cada vértice "V" adjacente a "u":
      - Se distancia[u] + peso(u→V) < distancia[V], então:
        - distancia[V] = distancia[u] + peso(u→V)
        - Adicionar (-distancia[V], V) à fila de prioridades Q
  - Fim do Algoritmo

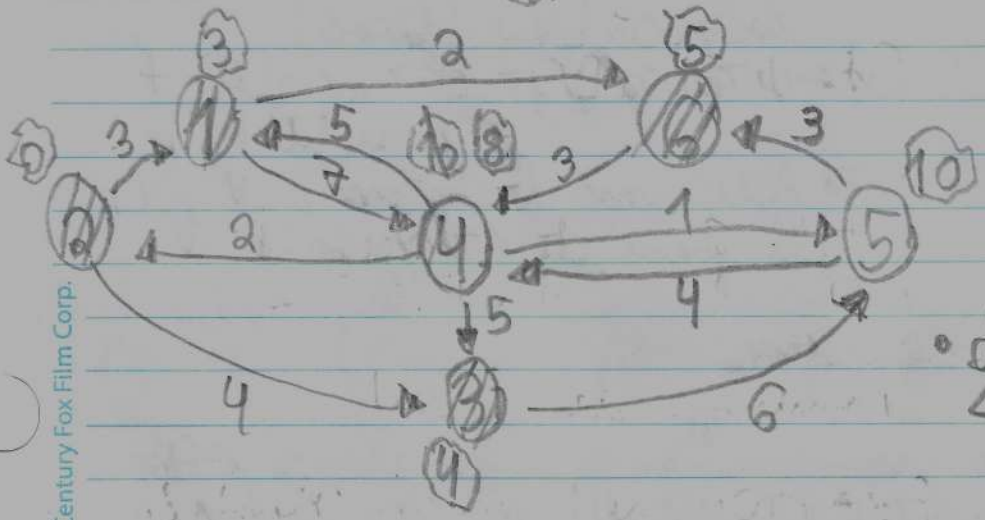
Sendo assim, aplicando o algoritmo acima no grafo presente na parte 2, tem-se:



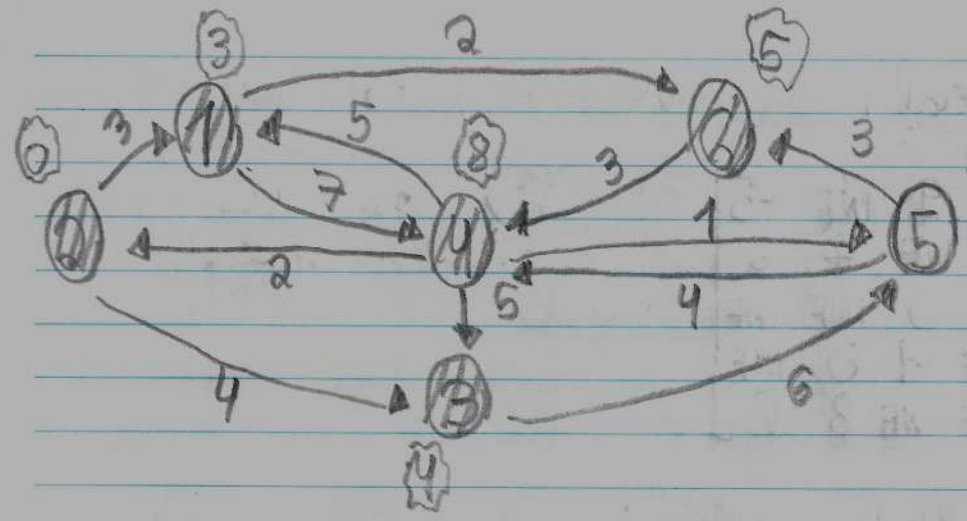
- dist[1] + peso(1→6) < dist[6]
- dist[1] + peso(1→4) < dist[4]



- dist[3] + peso(3→5) < dist[5]

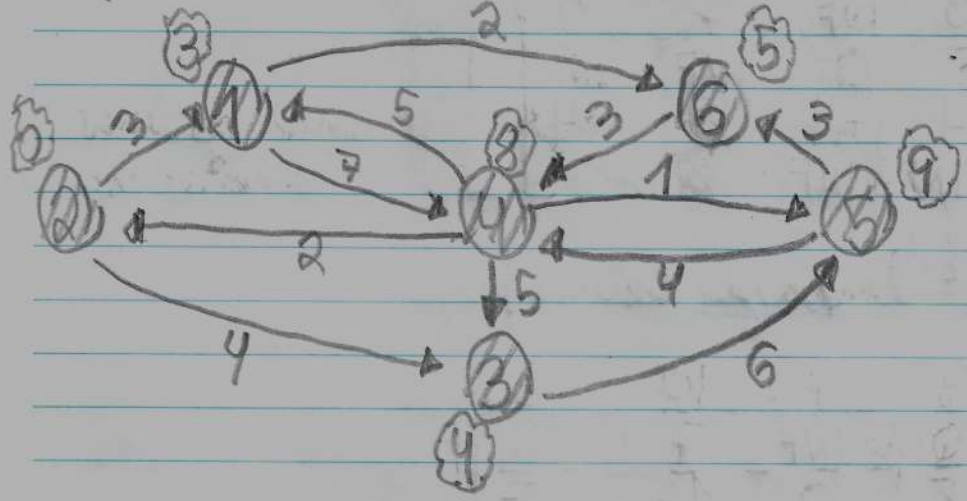


- dist[6] + peso(6→4) < dist[4]



• dist[4] + peso(4→5) < dist[5]

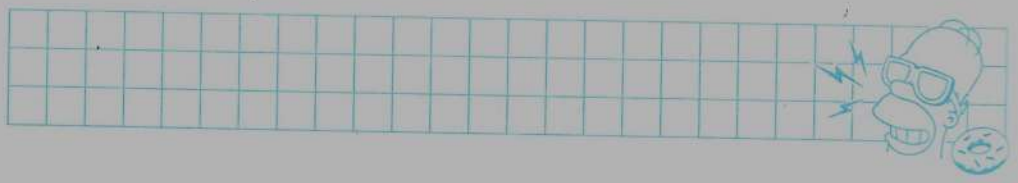
Grafo final com distâncias mínimas a partir do vértice 2:



Distancia [ ] = { INF, 3, 0, 4, 8, 9, 5 }
Vértice → 0 1 2 3 4 5 6

(4) Aplicando o algoritmo de Floyd:

* Considere: MX = Matriz de Distâncias (Pesos) da iteração "X"



• X = 0 (inicial)

$$M_0 = \begin{bmatrix} 0 & 3 & 9 & INF & -5 \\ INF & 0 & INF & 7 & 2 \\ INF & 5 & 0 & INF & INF \\ 1 & INF & -4 & 0 & INF \\ INF & INF & INF & 8 & 0 \end{bmatrix}$$

• X = 1 (1ª linha e coluna)

$$M_1 = \begin{bmatrix} 0 & 3 & 9 & INF & -5 \\ INF & 0 & INF & 7 & 2 \\ INF & 5 & 0 & INF & INF \\ 1 & 4 & -4 & 0 & -4 \\ INF & INF & INF & 8 & 0 \end{bmatrix}$$

• X = 2 (2ª linha e coluna)

$$M_2 = \begin{bmatrix} 0 & 3 & 9 & 10 & -5 \\ INF & 0 & INF & 7 & 2 \\ INF & 5 & 0 & 12 & 7 \\ 1 & 4 & -4 & 0 & -4 \\ INF & INF & INF & 8 & 0 \end{bmatrix}$$

• X = 3 (3ª linha e coluna)

$$M_3 = \begin{bmatrix} 0 & 3 & 9 & 10 & -5 \\ INF & 0 & INF & 7 & 2 \\ INF & 5 & 0 & 12 & 7 \\ 1 & 1 & -4 & 0 & -4 \\ INF & INF & INF & 8 & 0 \end{bmatrix}$$

• X = 4 (4ª linha e coluna)

$$M_4 = \begin{bmatrix} 0 & 3 & 6 & 10 & -5 \\ 8 & 0 & 3 & 7 & 2 \\ 13 & 5 & 0 & 12 & 7 \\ 1 & 1 & -4 & 0 & -4 \\ 9 & 9 & 4 & 8 & 0 \end{bmatrix}$$

• X = 5 (5ª linha e coluna)

$$M_5 = \begin{bmatrix} 0 & 3 & -1 & 3 & -5 \\ 8 & 0 & 3 & 7 & 2 \\ 13 & 5 & 0 & 12 & 7 \\ 1 & 1 & -4 & 0 & -4 \\ 9 & 9 & 4 & 8 & 0 \end{bmatrix}$$

∴ Matriz Final de Distâncias

$$M_5 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 3 & -1 & 3 & -5 \\ 8 & 0 & 3 & 7 & 2 \\ 13 & 5 & 0 & 12 & 7 \\ 1 & 1 & -4 & 0 & -4 \\ 9 & 9 & 4 & 8 & 0 \end{bmatrix} \end{array}$$

(5) Para otimizar a infraestrutura de rede da empresa basta encontrar a chamada árvore geradora mínima do grafo formado pela infraestrutura de rede(G). A árvore geradora mínima nada mais é do que um subgrafo de G (sendo G um grafo não orientado, conexo e ponderado) que conecta todos os vértices de G e com a menor

peso total final. Em outras palavras, para realizar a otimização da infraestrutura de rede, basta encontrar um grafo que conecte todos os vértices da infraestrutura, tendo o menor custo total.

Uma das formas de se encontrar essa árvore geradora mínima está na aplicação do algoritmo de Prim. Esse algoritmo consiste nos seguintes passos:

1º) Iniciar num vértice arbitrário;
2º) Percorrer as arestas incidentes em busca da aresta segura, sendo esta uma aresta de peso mínimo que conecta a árvore a um vértice não presente na árvore.
3º) Inserir o vértice na árvore.

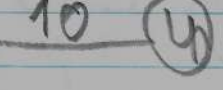
Dessa forma, aplicando o algoritmo de Prim na infraestrutura de rede da empresa tem-se:

↳ Selecionando o vértice 4 como inicial

AGM:

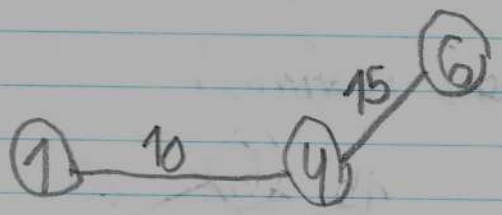
↳ Basta analisar qual a aresta segura, neste caso a que conecta o vértice 4 ao vértice 1 (custo 10)

AGM:



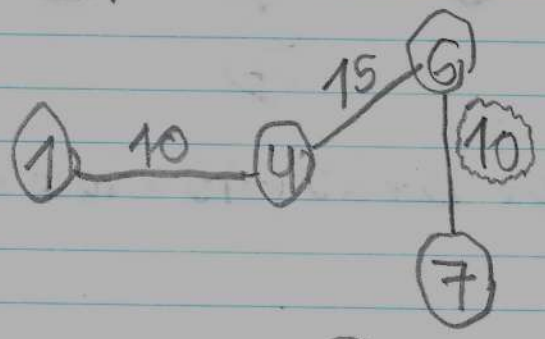
↳ Agora, necessita analisar todas as arestas dos vértices 1 e 4 em busca da aresta segura. Encontra-se duas com mesmo peso de 15: vértice 4 ao 6 e vértice 1 ao 3. Insere arbitrariamente.

AGM:

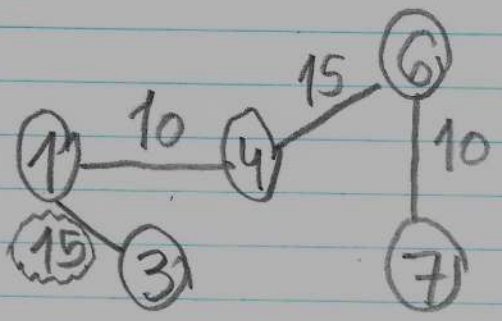


↳ Repetindo os processos:

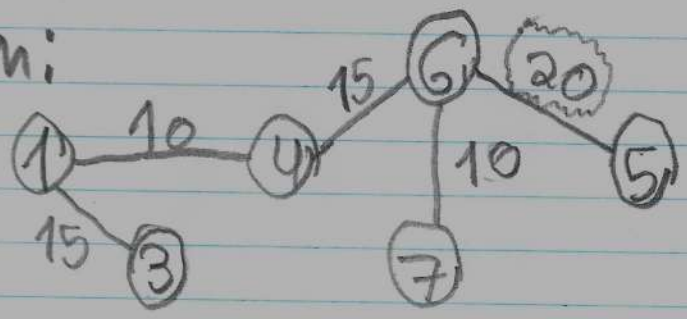
AGM:



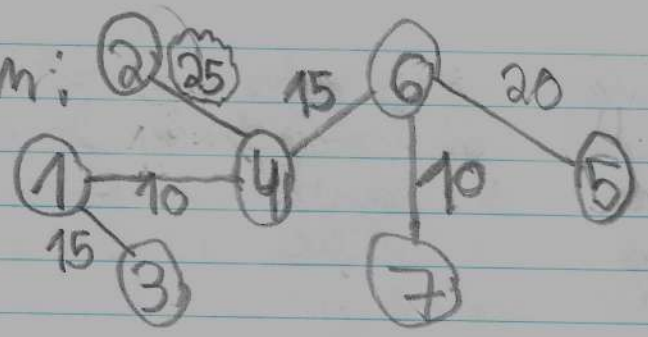
→ AGM:



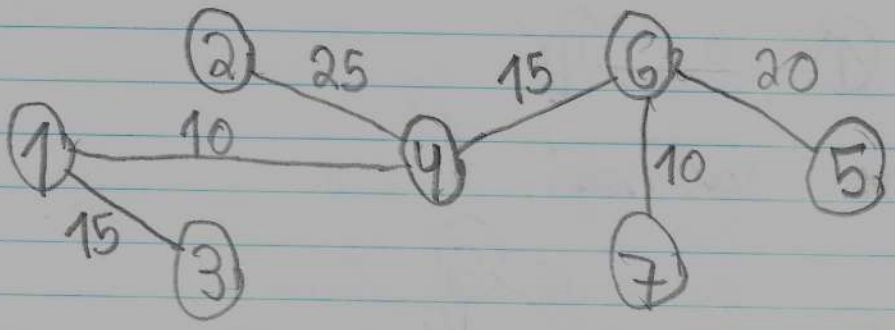
→ AGM:



→ AGM:



∴ Árvore Geradora Mínima:



Peso total: 15 + 10 + 25 + 15 + 10 + 20 = 95