

Exercício 1:

2. Preencha a tabela de execução abaixo

Iteração	Vértice	A	B	C	D	E	F
Inicial	-	0	∞	∞	∞	∞	∞
1	A	0	4	2	∞	∞	∞
2	C	0	3	2	∞	7	∞
3	B	0	3	2	6	7	∞
4	D	0	3	2	6	7	8
5	E	0	3	2	6	7	8
6	F	0	3	2	6	7	8

3. Identifique o caminho mínimo de A para F

A->C->B->D->F

4. Desenhe a árvore de caminhos mínimos resultante

```
A
|
C
| \
B  E
|
D
|
F
```

Exercício 2:

Não há ciclo negativo.

Iteração	S	A	B	C	D
Inicial	0	∞	∞	∞	∞
1	0	3	8	5	2
2	0	3	8	5	2
3	0	3	8	5	2
4	0	3	8	5	2
Verificação de Ciclo Negativo					
Final	0	3	8	5	2

Se usasse Dijkstra nesse caso daria o mesmo resultado, porém em casos gerais tem muita chance de dar errado pois não funciona com vértices negativos.

Exercício 3:

Nesse grafo existe ciclo negativo ($X \rightarrow Y \rightarrow Z \rightarrow X$) pois a soma das arestas da negativo.

Análise do ciclo $X \rightarrow Y \rightarrow Z \rightarrow X$: Peso total = -1

Exercício 4:

São Paulo -> Rio de Janeiro -> Salvador

Eu usaria Dijkstra já que o grafo é ponderado e não possui vértices negativas.

Custo total: $400 + 1200 = 1600$

Iteração	Vértice	SP	BSB	BH	RJ	SSA
Inicial	-	0	∞	∞	∞	∞
1	RJ	0	1000	580	400	1600
2	BH	0	1000	580	400	1600
3	BSB	0	1000	580	400	1600
4	SSA	0	1000	580	400	1600

São Paulo p/ BH: SP->BH

São Paulo p/ RJ: SP->RJ

São Paulo p/ BSB: SP->BSB

São Paulo p/ SSA: SP->RJ->SSA

Exercício 5:

Critério	Dijkstra	Bellman-Ford
Pesos Negativos	Não serve	Serve
Deteção de Ciclo Negativo	Não serve	Serve
Complexidade Temporal	$O(V^2)$	$O(V \cdot E)$
Estrutura de Dados Principal	Heap	Matriz adjacência
Estratégia	Guloso (escolhe vértice c/ menor custo)	Relaxa as arestas repetidamente
Melhor Aplicação	Grafos com arestas positivas	Grafos com arestas negativas

Exercício 6:

Função caminhoMinimo(Grafo G, VerticeOrigem):

negativo = falso

Para cada aresta (u, v) em G:

Se peso(u, v) < 0:

negativo = verdadeiro

Pare

Se negativo = verdadeiro:

resultado = BellmanFord(G, origem)

Se resultado indica ciclo negativo:

Retorne "Ciclo negativo detectado"

Senão:

Retorne resultado

Senão:

resultado = Dijkstra(G, origem)

Retorne resultado

Fim Função