Lista de exercícios 03 - Grafos: árvore geradora mínima, caminhos mínimos, fluxo em redes

Inclui respostas de alguns exercícios.

1. Suponha que é dado um grafo G com custos nas arestas positivos e diferentes. Seja T uma árvore geradora mínima de G. Agora suponha que substituímos o custo cada aresta, c_e, por seu quadrado, c_e^2, criando assim uma nova instância do problema com o mesmo grafo, mas com custos diferentes. Concorde ou descorde: T ainda deve ser uma árvore geradora mínima para a nova instância. Prove ou dê um contraexemplo.

Resposta. Como todas as arestas tem peso diferente, sabemos que T é única (ver questão 3), portanto ela é a solução que o algoritmo de Kruskal retorna. Note que, como os pesos das arestas são todos positivos, para quaisquer arestas e e e', se $c_e < c_{e'}$ então $c_e^2 < c_{e'}^2$. Portanto, quando os pesos são os quadrados, todas as arestas continuam com pesos diferentes e mantém a mesma ordem. Desta forma, o algoritmo de Kruskal constrói a mesma árvore T, pelo que T continua sendo uma árvore geradora mínima do grafo.

Nota 1. Perceba que o argumento acima vale se houver arestas de peso zero.

Nota 2. Quando há arestas de igual peso, T continua sendo uma árvore geradora mínima. Contudo, não há garantias que ela seja o produto de uma execução do algoritmo de Prim ou de Kruskal. Assim, nesse caso, pode provar primeiro que:

Fato 1. Se T e T' são árvores geradoras mínimas de G então para cada peso de arestas w, T e T' tem o mesmo número de arestas com peso w. (uma prova desse fato é muito parecida a outras de exercícios nesta lista).

Depois, pode assumir uma ordenação não decrescente qualquer das arestas de G e perceber que essa mesma ordenação vale para o quadrado das arestas, portanto uma árvore T' que Kruskal retorne com essa ordenação, será a mesma que Kruskal irá retornar para o quadrado das arestas com essa ordenação. Pode usar o *Fato 1* para deduzir que, para cada peso w, T tem o mesmo número de arestas com peso w que T' (retornada por Kruskal) e que, portanto, a soma dos quadrados dos pesos das arestas de T é igual à de T'. Logo, como T' continua sendo árvore geradora mínima, T também.

Nota 3. Quando houver arestas negativas, então não podemos afirmar que T continuará sendo uma árvore geradora mínima (perceba que o quadrado de um aresta negativa leve, não é necessariamente leve). Neste caso, um contraexemplo pode ser aplicado.

2. Seja $G = (V, E)$ um grafo conexo com pesos associados a cada aresta. O professor B. Smart propôs o seguinte algoritmo para encontrar uma árvore geradora mínima de G.

(a) Argumente sucintamente (em poucas linhas) que o grafo obtido é conexo e acíclico.

Resposta. Como G é conexo, inicialmente $H = G$ e só são removidas arestas de H se a remoção delas não desconecta o grafo, temos que o H resultante continua conexo. Para provar que é acíclico, vamos supor o contrário, então H teria um ciclo C, seja e a primeira aresta de C a ser analisada na Linha 5. Temos que $H - e$ continua conexo, pois como visto em aula, a remoção de uma aresta de um ciclo não desconecta um grafo. Logo, e teria sido removida de H pelo algoritmo, o que contradiz que H contenha o ciclo C.

(b) Mostre que se e é uma aresta de G com peso máximo e $G - e$ é conexo, então existe uma árvore geradora mínima de G que não contém e.

```
input    : grafo conexo G = (V, E) e função de pesos nas arestas ω
output   : H subgrafo de G.
begin
    Ordene E em ordem não crescente de pesos
    H ← G
    for e ∈ E em ordem não crescente de pesos do
        if H − e é conexo then
            H ← H − e
        end
    end
    return H
end
```

Algorithm 1: SMART-AGM(G, ω).

Resposta. Considere uma árvore geradora T. Se T não contém e, então a prova está completa, em outro caso e é uma aresta de T. Logo, $T-e$ gera duas árvores T_1 e T_2, como $G-e$ é conexo, temos que o corte $\delta(V[T_1])$ tem mais arestas além de e e como e tem peso máximo, qualquer aresta desse corte tem custo menor ou igual que e. Seja e' uma aresta do corte $\delta(V[T_1])$, temos que $T' = T_1 \cup \{e'\} \cup T_2$ é uma arvore geradora de G com custo:

$$c(T') = c(T_1) + c_{e'} + c(T_2) \leq c(T_1) + c_e + c(T_2) = c(T).$$

Portanto, T' é uma árvore geradora minima de G que não contém e.

(c) Usando os resultados dos itens anteriores, mostre que o algoritmo está correto.

Resposta. Como visto no item (a), o algoritmo retorna um subgrafo conexo e acíclico (uma árvore). Ademais, inicialmente $H = G$ e durante a execução do algoritmo nenhum vértice é removido, assim H é um subgrafo gerador. Logo, o resultado do algoritmo é uma árvore geradora de G.

Para provar que é mínima, note que a prova do item (b) vale mesmo se o enunciado fosse: *"Se e for uma aresta de peso máximo entre aquelas cuja remoção mantém G conexo, então existe uma árvore geradora T que não contém e"*. Observe que a diferencia dos enunciados é que nesta segunda versão, e não precisa ser uma aresta de peso máximo do grafo inteiro, só precisa ser uma aresta de peso máximo dentre aquelas cuja remoção não desconecta o grafo.

A observação acima é suficiente para concluirmos a nossa prova. Para tal, provamos a seguinte invariante de laço: *"Uma árvore geradora mínima de H é também uma árvore geradora mínima de G"*. Antes de começar o laço, a invariante é trivialmente verdade porque $H = G$. Agora analisamos uma execução qualquer do laço. Supomos que, no início da execução a invariante vale. Se $H - e$ desconecta o grafo, e não é removida e como H continua o mesmo, temos que a invariante vale no final dessa execução do laço. Em outro caso, pela escolha de e, temos que e é uma aresta de peso máximo em H tal que $H - e$ é conexo e sabemos que existe uma árvore geradora mínima T de H que não contém e (pela nossa observação sobre o item (b)). Logo, T é árvore geradora mínima de G (pela invariante no início do laço) e de $H - e$ (pois T não contém e). Portanto, temos que uma árvore geradora mínima de $H - e$ também é árvore geradora mínima de G, assim a invariante vale no final dessa execução do laço.

3. Responda as seguintes questões:

(a) Mostre que, se para todo $X \subseteq V$, o corte $\delta(X)$ contém exatamente uma aresta leve, então existe uma única árvore geradora mínima T de G.

Resposta. Seja T a árvore geradora mínima que uma execução de Prim retorna. Suponha, por contradição, que existe uma árvore geradora mínima T' diferente de T. Isso significa que existe uma aresta $e \in T$ tal que $e \notin T'$. Note que, pela escolha de Prim, e é uma aresta leve de algum corte $\delta_G(X)$ em G. Consideremos $T' + e$ e denotemos por C o ciclo que a adição de e cria em $T' + e$. Ao analisar o corte $\delta_{T'+e}(X)$ no grafo $T' + e$, temos que, além de e, pelo menos alguma aresta $e' \neq e$ de C está no corte. Como, $\delta_{T'+e}(X) \subseteq \delta_G(X)$, e é uma aresta leve desse corte e como todo corte tem uma única aresta leve: $c_e < c_{e'}$, ou seja $c_e - c_{e'} < 0$. Assim, $T' + e - e'$ é uma árvore geradora com peso: $c(T') + c_e - c_{e'} < c(T')$, o que contradiz que T' seja mínima.

(b) Suponha que todas as arestas de G têm custos distintos, com exceção de duas $(u, v), (x, y)$, que têm o mesmo custo. Suponha também que todo caminho de u até x contém essas duas arestas. Argumente que existe uma única árvore geradora mínima. Você pode utilizar o fato enunciado no item anterior.

Resposta. Se (u, v) está em todo caminho entre u e x de G, ao fazermos $G-(u, v)$, geramos um grafo com duas componentes, pois além da aresta (u, v) não há caminho entre u e v (caso contrario teríamos um caminho entre u e x sem a aresta (u, v)). Um argumento semelhante nos leva a que $G-(u, v)-(x, y)$ é um grafo com três componentes: C_1, C_2 e C_3. Observe que, todas as arestas da componente C_i $(i \in \{1, 2, 3\})$ são diferentes, portanto qualquer corte dessa componente tem uma única aresta leve. Logo, usando o resultado do item anterior, cada componente C_i tem uma única árvore geradora mínima T_i.

Agora, analisemos uma árvore geradora mínima T de G. Note que, T precisa ter a aresta (u, v), em caso contrário em T não haveria um caminho entre u e x e portanto T não seria conexa. A mesma análise vale para (x, y). Assim, uma árvore geradora mínima T de G se obtém juntando uma árvore geradora mínima T_1 de C_1 com uma T_2 de C_2 e uma T_3 de C_3 mediante a adição das arestas (u, v) e (x, y); isto é: $T = T_1 \cup T_2 \cup T_3 \cup \{(u, v), (x, y)\}$. Como T_1, T_2 e T_3 são únicas, T também é única.

4. Seja $G = (V, E)$ um grafo não direcionado. Um conjunto $F \subseteq E$ de arestas é chamado conjunto de retroalimentação se cada ciclo de G tiver pelo menos uma aresta em F.

(a) Suponha que G não seja ponderado. Crie um algoritmo eficiente para encontrar um conjunto de retroalimentação de tamanho mínimo.

Resposta. Observe que se F é um conjunto de retroalimentação de G, então $G - F$ é uma floresta. Assim o problema seria encontrar uma floresta com maior número de arestas; isto é, uma floresta com uma árvore por cada componente de G. O DFS visto em aula, já retorna uma floresta onde cada árvore corresponde a uma componente de G. Assim, qualquer aresta que não seja de árvore do DFS, será uma aresta de F. Portanto, no DFS de um vértice u, ao analisar a aresta (u, v), se v já foi descoberto e não é o pai de u, então (u, v) não é aresta de árvore do DFS e a adicionamos a uma lista resultante F. O custo computacional desta solução, é igual ao do DFS, $O(V + E)$.

(b) Suponha que G é um grafo não direcionado com pesos positivos nas arestas. Projete um algoritmo eficiente para encontrar um conjunto de retroalimentação de peso mínimo. Argumente que seu algoritmo está correto.

Resposta. Se desejamos minimizar o peso de F, então procuramos maximizar a soma dos pesos das arestas na floresta $G - F$. Isto é, procuramos uma floresta geradora maximal (em número de arestas) X de G que maximize $\sum_{e \in X} c_e$, note que isso é equivalente a minimizar $\sum_{e \in X}(-c_e)$. Assim, estamos procurando uma floresta geradora maximal X de G, de peso mínimo (após multiplicar o peso das arestas de G por -1).

Note que, o algoritmo de Kruskal retorna uma floresta geradora maximal (em número de arestas) de peso mínimo: durante a execução do algoritmo, somente não são adicionadas arestas que gerem ciclos, ou seja, o algoritmo retorna um subgrafo acíclico maximal de G e a prova do peso ser mínimo foi vista em aula. Assim podemos multiplicar as arestas de G por -1 em $O(V)$ e obter uma floresta geradora maximal de peso mínimo usando Kruskal em

$O(E \log E)$. Durante a execução do algoritmo, as arestas que não adicionadas na florestas, podem ser adicionadas em F.

5. Uma empresa possui n filiais que devem ser conectadas, direta ou indiretamente. Cada par de filiais que são conectadas diretamente utiliza um serviço de fibra ótica, ou um serviço de conexão via linha telefônica. A velocidade da conexão (em MBit/s) via linha telefônica entre duas filiais depende da distância e é fornecida pela empresa de telefonia. A velocidade da fibra é sempre constante, 1GBit/s. Devido a restrições orçamentárias, somente f ($f < n$) conexões via fibra serão contratadas. Uma vez instalada a rede, duas filiais podem ser conectadas (indiretamente) por uma sequência de conexões diretas; *a velocidade de conexão entre as duas filiais* é a velocidade da conexão mais lenta nessa sequência. A velocidade de rede é a menor velocidade de conexão entre quaisquer duas filiais. Escreva um algoritmo para encontrar uma rede com a maior velocidade de rede possível. Argumente por que o algoritmo está correto e analise a complexidade.

Resposta. Como $f < n$ e entre todo par de filiais deve haver um caminho, temos que a solução deve ser uma árvore geradora T do grafo G em que os vértices são as filiais e os pesos das arestas a velocidade da conexão. Desejamos escolher T de forma a maximar a velocidade da conexão. Note que a velocidade da conexão de T é a menor velocidade entre duas filias, que por sua vez é a velocidade da aresta de menor velocidade no caminho entre essas filiais. Assim, encontrar T que maximiza a velocidade de conexão equivale a encontrar T que maximiza o peso da menor aresta. Se multiplicamos os pesos das arestas por -1, nosso problema é encontrar uma árvore geradora T de G que minimiza o peso da maior aresta.

Perceba que qualquer árvore geradora mínima, minimiza o peso da maior aresta. Para provar isso, considere uma árvore geradora mínima T e uma árvore geradora T' que minimiza o peso da maior aresta. Seja e uma aresta de maior peso de T, ao fazermos $T - e$ geramos duas árvores T_1 e T_2, consideremos o corte $\delta_G(V[T_1])$ em G. Perceba que toda aresta e' em $\delta_G(V[T_1])$ tem peso maior ou igual que o de e, caso contrário $T - e + e'$ seria uma árvore geradora de G com menor peso que T, contradizendo que T seja mínima. Como T' é árvore geradora de G, deve ter pelo menos uma aresta no corte $\delta_G(V[T_1])$. Portanto, em T' há pelo menos uma aresta com peso maior ou igual que e. Assim a aresta de maior peso de T' tem peso maior ou igual que a de T, o que implica que T minimiza a aresta de maior peso.

Portanto é suficiente definir o grafo G a partir das filiais e as conexões, multiplicar os pesos das arestas por -1 e usar um algoritmo para árvore geradora mínima (Prim ou Kruskal).

6. Considere um grafo direcionado $G = (V, E)$ cujas arestas têm pesos 0 ou 1. Projete um algoritmo de tempo $O(V + E)$ que obtém uma árvore de caminhos mínimos a partir de um vértice s.

7. Dado um grafo ponderado e direcionado $G = (V, E)$ sem ciclos de peso negativo, seja m o máximo, entre todos os vértices $v \in V$, do número mínimo de arestas em um caminho mínimo da fonte s para v. (Aqui, o caminho mínimo é por peso, e não por número de arestas.) Reescreva o algoritmo Bellman-Ford para ele termine em $m + 1$ passos, mesmo que m não seja conhecido com antecedência.

Resposta. A propriedade de relaxamento de caminhos nos garante que se existe um caminho mínimo $(s = v_0, v_1, v_2, \ldots, v_k = v)$ de s até v, relaxando as arestas do caminho em ordem, obtemos $d[v] = dist(s, v)$ (isso, podendo fazer quaisquer outras relaxações no meio). Considere um vértice $v \in V$ qualquer, note que existe um caminho mínimo de s a v com no máximo m arestas, logo, se relaxamos todas as arestas de G, m vezes, teremos conseguido $d[v] = dist(s, v)$. Como isso é verdade para qualquer vértice, após executar o *loop* principal do Bellman-Ford m vezes, teremos que $d[v] = dist(s, v)$ para todo $v \in V$. Pela propriedade de convergência sabemos que uma vez alcançado o valor $dist(s, v)$, $d[v]$ não muda mais, assim na execução $m + 1$ do *loop* não haverá nenhum valor $d[v]$ alterado. Desta forma, conseguimos parar o *loop* principal do Bellman-Ford quando for detectado que em uma iteração (após tentar relaxar todas as arestas) nenhum valor $d[v]$ foi modificado. Para o tipo de grafos descrito no exercício, isso acontece no pior caso, na iteração $m+1$. Assim, mesmo sem saber o valor de m, podemos adaptar o Bellman-Ford para que o ciclo principal não execute mais do que $m+1$ passos, parando quando for detectado que nenhum $d[v]$ foi atualizado após tentar relaxar todas as arestas.

8. Escreva um algoritmo que verifica se há ciclos negativos em um grafo direcionado e, se houver, devolva um tal ciclo.

Resposta. Seja $G = (V, E)$ um digrafo. Adicione um novo vértice s, com arcos unitários de s para cada vértice de G, isto é, para cada vértice $u \in V$, adicionamos o arco (s, u) com peso $c(s, u) = 1$. Observe que essa modificação não adiciona ciclos negativos em G, de fato nenhum novo ciclo é adicionado a G (pois só foi adicionado um vértice e arcos saindo dele, nenhum chegando). Ademais, como s alcança todo vértice de G, temos que se houver um ciclo negativo em G, então s alcança tal ciclo. Portanto, o algoritmo de Bellman-Ford desde s como origem, retorna se existe ou não um ciclo em G. Adicionar s como descrito consome $O(V)$ enquanto Bellman-Ford consome $O(VE)$, portanto a complexidade do algoritmo para encontrar ciclos negativos resulta $O(VE)$.

9. Milda é a presidenta de um determinado país B. Esse país é dividido em estados e cada estado possui uma capital. Milda quer reestruturar o sistema de estradas e ferrovias e precisa da sua ajuda. As ferrovias são muito antigas e seu custo de manutenção é alto. Seu trabalho é ajudar a presidenta a decidir quais ferrovias podem ser desativadas. No entanto, como B é um país democrático, uma ferrovia só pode ser desativada se isso não piorar a qualidade do sistema de transporte, isso é, uma ferrovia pode ser desativada apenas se a distância de cada cidade à capital mais próxima não for modificada. Considere que o país B tem n cidades e que se uma estrada ou ferrovia liga a cidade i à cidade j, então ela pode ser utilizada para ir tanto de i para j como de j para i. Obtenha um algoritmo eficiente para descobrir quantas ferrovias podem ser desativadas.

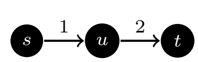
10. Quais das seguintes afirmações são verdadeiras e quais são falsas? Justifique sua resposta apresentado uma prova ou um contra-exemplo.

(a) Se f é um fluxo máximo de uma rede então $f(a) = 0$ ou $f(a) = c(a)$ para toda aresta $a \in E$.

Resposta. Falso. (ver exemplo em aula para Bola Feliz, ou no item (b) desta questão).

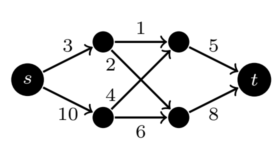
(b) Toda rede possui um fluxo máximo f tal que $f(a) = 0$ ou $f(a) = c(a)$ para toda aresta $a \in E$.

Resposta. Falso. Considere a rede na figura abaixo. O fluxo máximo nessa rede tem valor 1 e o arco (u, t) não pode ser saturado.



(c) Se todas as arestas têm capacidades diferentes então existe um único corte mínimo.

Resposta. Falso. Considere a rede abaixo:



Perceba que o fluxo máximo tem valor 13 (obtido saturando todos os arcos). Logo, 13 é a capacidade de um corte mínimo. Note que há vários cortes com capacidade 13, por exemplo: $(\{s\}, V \setminus \{s\})$ e $(V \setminus \{t\}, \{t\})$.

(d) Suponha que (S,T) é um corte mínimo em uma rede (G,c,s,t). Se multiplicarmos as capacidades de todas as arestas por um número $\lambda > 0$ então (S,T) também é um corte mínimo na nova rede $(D,\lambda c,s,t)$.

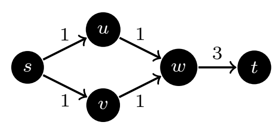
Resposta. Verdadeiro. Seja (S,T) um corte mínimo e (S',T') qualquer corte da rede. Temos que:

$$\begin{aligned} c(S,T) &\leq c(S',T') \\ \sum_{u\in S}\sum_{v\in T} c(u,v) &\leq \sum_{u\in S'}\sum_{v\in T'} c(u,v) \\ \lambda\sum_{u\in S}\sum_{v\in T} c(u,v) &\leq \lambda\sum_{u\in S'}\sum_{v\in T'} c(u,v) \quad \text{(multiplicando por } \lambda > 0\text{)} \\ \sum_{u\in S}\sum_{v\in T} \lambda c(u,v) &\leq \sum_{u\in S'}\sum_{v\in T'} \lambda c(u,v) \end{aligned}$$

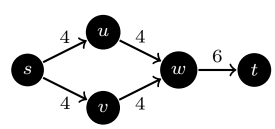
Como isso vale para qualquer corte (S',T'), temos que (S,T) continua mínimo após multiplicar por $\lambda > 0$ as capacidades da rede.

(e) Suponha que (S,T) é um corte mínimo em uma rede (G,c,s,t). Se aumentarmos as capacidades de todas as arestas somando um número $\lambda > 0$ então (S,T) também é um corte mínimo na nova rede (D,c',s,t).

Resposta. Falso. Considere a seguinte rede:



Nessa rede o valor do fluxo máximo é 2 (saturando os arcos de capacidade 1), assim um corte mínimo tem capacidade 2. Um exemplo de corte mínimo nessa rede é $(\{s\},\{u,v,w,t\})$. Se somamos $\lambda = 3 > 0$ às capacidades, temos:



Nessa nova rede, um corte mínimo tem capacidade 6 e é $(\{s,u,v,w\},\{t\})$. Note que o corte mínimo da rede anterior $(\{s\},\{u,v,w,t\})$, nesta rede tem capacidade $8 > 6$, portanto não é mais mínimo. De fato, nesse exemplo, nenhum corte mínimo da rede anterior é mínimo na nova rede e o corte mínimo na nova rede não era mínimo na anterior.

11. Seja (G,c,s,t) uma rede de fluxo. Um fluxo inteiro é *par* (*ímpar*, respectivamente) se $f(a)$ é par (ímpar, respectivamente) para toda aresta $a \in E$. Prove ou mostre um contra-exemplo para cada uma das afirmações seguintes.

(a) Se todas as capacidades são inteiros pares, então existe um fluxo máximo que é par.

(b) Se todas as capacidades são inteiros ímpares, então existe um fluxo máximo que é ímpar.

12. Dados vértices s e t em um *grafo direcionado* G, dizemos que uma coleção $P_1, P_2, \ldots, P_k$ de caminhos com início em s e final em t é *aresta-disjunta* se para quaisquer caminhos P_i e P_j, P_i e P_j não têm arestas em comum. Responda as seguintes questões:

(a) Demonstre que, dado um grafo G e vértices s e t, o número máximo de caminhos disjuntos nas arestas de s a t é igual ao número mínimo de arestas necessárias para desconectar s de t.

(b) O problema dos caminhos aresta-disjuntos (CAD) consiste em, dados um grafo G e vértices s e t em G, encontrar uma coleção aresta-disjunta máxima de caminhos de s a t. Mostre como solucionar CAD, usando fluxo máximo.

13. Um emparelhamento perfeito em um grafo bipartido $G = (V, E)$, é um conjunto de arestas $M \subseteq E$ tal que cada vértice em V incide em exatamente uma aresta de M. Seja $G = (V, E)$ um grafo bipartido não direcionado, proponha um algoritmo eficiente para saber se G possui um emparelhamento perfeito.

14. Um certo dia o Rei Artur caprichosamente decidiu que era tempo das donzelas da corte de Camelot se casarem. Na corte havia n donzelas e n cavaleiros. Apesar de Artur ser conhecido como um regente impiedoso, ele não queria casar nenhuma donzela com algum cavaleiro de quem ela não gostasse. Assim, ele pediu a Merlin que arranjasse o casamento de todas as donzelas de modo que isto não ocorresse. Suponha que cada donzela fornece a Merlin uma lista dos cavaleiros com quem ela aceitaria se casar. Como se vê, além de autoritário, o rei Artur era machista e preconceituoso. Mostre como Merlin pode determinar se é possível realizar estes n casamentos forçados.

15. Várias famílias saem juntas para jantar. Para aumentar a interação social*, cada pessoa gostaria de se sentar em uma mesa em que não houvesse outro membro da sua família. Suponha que no total existam p famílias e a família i tem a_i membros. Suponha que há q mesas disponíveis e que a mesa j tem capacidade para acomodar b_j pessoas. Mostre como formular este problema como um problema de fluxo máximo.

16. Suponha que Neo quer enviar uma informação de um ponto (vértice) s a outro ponto t em uma rede com capacidades unitárias (isto é, cada aresta tem capacidade igual a 1). A Matrix pode impedir isto destruindo um conjunto de *arestas* da rede (o que seria equivalente a remover as arestas do grafo orientado). Mostre como a Matrix pode determinar um conjunto mínimo de arestas cuja remoção destrói todos os caminhos de s a t. Justifique.

17. Considere novamente o exercício anterior, mas agora suponha que a Matrix quer *desligar* um conjunto de *vértices* da rede (o que seria equivalente a remover tais vértices do grafo orientado). Mostre como a Matrix pode determinar um conjunto mínimo de vértices cuja remoção destrói todos os caminhos de s a t. Justifique.

18. Exercícios dos capítulos 23, 24, 25 e 26 do livro de texto.

---

*Suponha por absurdo que todos vão desligar seus smartphones.