Problema 1

Exercício. Para cada inteiro $k \geq 2$ determine todos os grafos k-regulares com cintura quatro que têm exatamente $2k$ vértices.

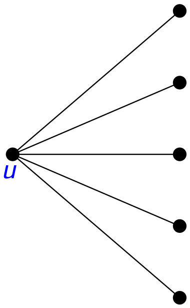
Cintura de um grafo G é o comprimento de um ciclo mais curto de G.

Sugestão. Tome um vértice arbitrário u e olhe sua vizinhança. Com a informação da cintura, deduza uma propriedade de $N(u)$.

Solução

Seja G um grafo k-regular com cintura quatro e com exatamente $2k$ vértices. Como G tem cintura quatro, G é simples.

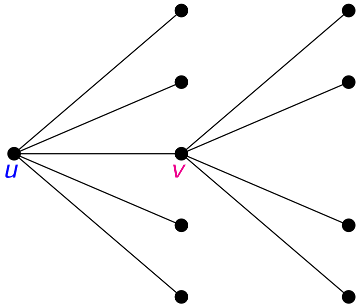
Seja u um vértice qualquer de G.



Como G é k-regular, a vizinhança $N(u)$ tem k vértices. Como G tem cintura quatro, $N(u)$ é independente.

Solução

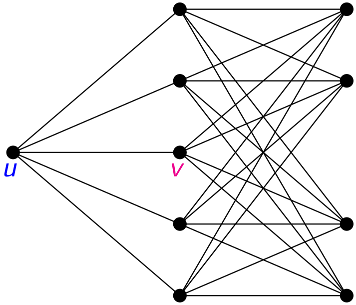
Assim, um vértice de $N(u)$, digamos v, não tem vizinhos em $N(u)$ e possui $k-1$ vizinhos distintos de u.



Como G tem exatamente $2k$ vértices, esses são todos os vértices de G.

Solução

Como todo vértice de $N(u)$ tem grau $k - 1$ e $N(u)$ é independente, todo vértice de $N(u)$ é adjacente a todo vértice de $N(v)$.



Que grafo é este? $K_{k,k}$. ∎

Problema 2

Um grafo é randomicamente Euleriano a partir de um vértice v se **qualquer** trilha maximal começando em v é Euleriana.

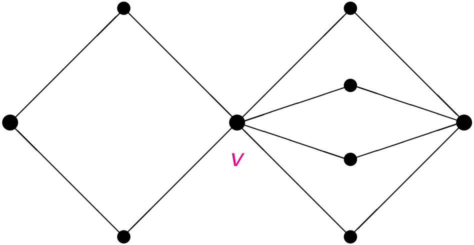


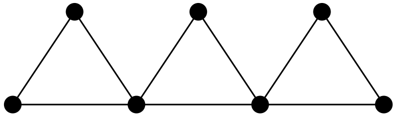
Figura: Grafo randomicamente Euleriano a partir de v.

(a) Exiba um grafo Euleriano que não seja randomicamente Euleriano a partir de nenhum vértice.
(b) Enuncie (e prove) condições necessárias e suficientes para que um grafo Euleriano seja randomicamente Euleriano a partir de um vértice v.

Solução

(a) Alguém tem um exemplo de um grafo Euleriano que **não** é randomicamente Euleriano a partir de nenhum vértice?

Eis um exemplo.



Começando em qualquer vértice, é sempre possível achar uma trilha maximal que evita um dos triângulos dos “cantos”.

Solução

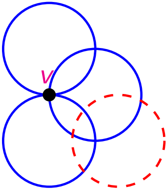
(b) Seja G um grafo Euleriano e seja v um vértice de G. Uma condição necessária e suficiente para que G seja randomicamente Euleriano a partir de um vértice v é a seguinte.

Proposição. Seja G um grafo Euleriano e seja v um vértice de G. Então G é randomicamente Euleriano a partir de v se, e somente se, todo ciclo de G contém v.

Solução

Proposição. Seja G um grafo Euleriano e seja v um vértice de G. Então G é randomicamente Euleriano a partir de v se, e somente se, todo ciclo de G contém v.

($\Rightarrow$) (contra-positiva) Suponha que C é um ciclo que não contém v. Seja $G' = G - E(C)$. Tome uma trilha maximal T em G' começando em v.
Sabemos que T é fechada (visto em aula). Mesmo “devolvendo” as arestas de C, não é possível estender T pois C não contém v.



Proposição. Seja G um grafo Euleriano e seja v um vértice de G. Então G é randomicamente Euleriano a partir de v se, e somente se, todo ciclo de G contém v.

($\Leftarrow$) Seja T uma trilha maximal começando em v. Sabemos que T é fechada e usa todas as arestas de G incidentes a v (visto em aula).

Suponha por contradição que T não seja Euleriana. Então $G - E(T)$ é par e não vazio; assim, ele contém um ciclo C.

Como v tem grau zero neste grafo, C não contém v, uma contradição. Portanto, G é randomicamente Euleriano a partir de v. $\blacksquare$

Problema 3

Exercício (CLRS 22.4-2). Descreva um algoritmo linear que recebe um **grafo orientado acíclico** G e vértices s, t e devolve o **número** de caminhos de s a t em G.

Note que só é preciso contar os caminhos, não exibi-los.

Para simplificar a apresentação, suporemos que o grafo **não possui arestas múltiplas**.

Problema 3

Problema: calcular o número de caminhos de s a t.

Ideia: combinar ordenação topológica e programação dinâmica!

Seja G um grafo orientado acíclico. Seja

$$p(v) = \text{número de caminhos de } v \text{ a } t.$$

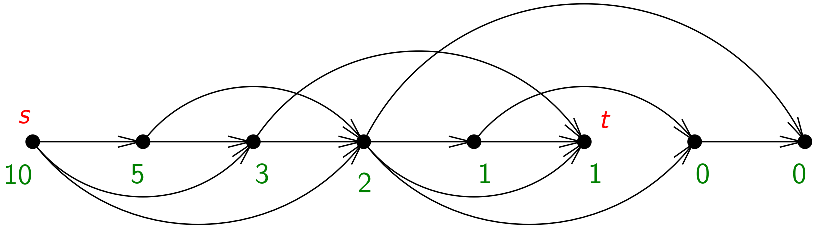
Queremos determinar $p(s)$.

Para isto, queremos achar uma recorrência para $p(v)$.

Problema 3

Seja G um grafo orientado acíclico. Seja

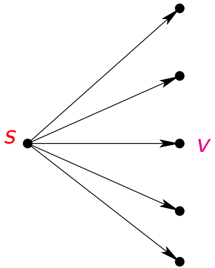
$$p(v) = \text{número de caminhos de } v \text{ a } t.$$



Queremos achar uma recorrência para $p(v)$.

Problema 3

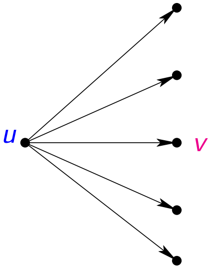
Considere $p(v)$ para cada $v \in \mathrm{Adj}[s]$.



Qual é a relação entre $p(s)$ e s valores $p(v)$ para $v \in \mathrm{Adj}[s]$?

Problema 3

De modo mais geral, considere um vértice u (em vez de s).

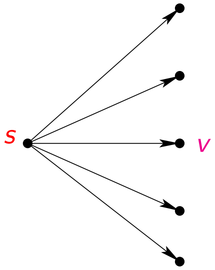


Qual é a relação entre $p(u)$ e os valores $p(v)$ para $v \in \mathrm{Adj}[u]$?

Deduza esta relação/recorrência e projete um algoritmo linear para calcular os valores $p(u)$. Escreva um **pseudocódigo** (sim, precisa).

Problema 3

Considere $p(v)$ para cada $v \in \mathrm{Adj}[s]$.

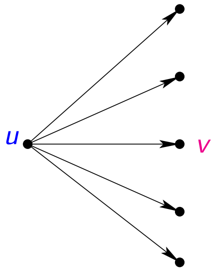


Qual é a relação entre $p(s)$ e estes valores?

$$p(s) = \sum_{v \in \mathrm{Adj}[s]} p(v).$$

Solução

O mesmo vale para qualquer vértice u (em vez de s).



$$p(u) = \sum_{v \in \mathrm{Adj}[u]} p(v).$$

Solução

Fixe uma ordenação topológica $v_1, v_2, \ldots, v_n$ de G.

O valor $p(v_i)$ depende apenas de valores $p(v)$ onde v sucede v_i.

$$p(v_i) = \begin{cases} 1 & \text{se } v_n = t, \\ 0 & \text{se } v_n \neq t, \\ \sum_{v \in \mathrm{Adj}[v_i]} p(v) & \text{caso contrário.} \end{cases}$$

Note que se v_i é um sorvedouro distinto de t, então a terceira regra diz que $p(v_i) = 0$.

Solução

$\textsc{Conta-Caminhos}(G, s, t)$ $\triangleright$ G é acíclico

1. **para cada** $v \in V[G] - \{t\}$ **faça**
2. $\quad p[v] \leftarrow 0$
3. $p[t] \leftarrow 1$
4. compute uma ordenação topológica $v_1, v_2, \ldots, v_n$ de G
5. **para** $i \leftarrow n-1$ **até** 1 **faça** $\triangleright$ em ordem inversa
6. $\quad$ **para cada** $v \in \mathrm{Adj}[v_i]$ **faça**
7. $\qquad p[v_i] \leftarrow p[v_i] + p[v]$
8. **devolva** p

Linhas 1–3	$O(V)$
Linha 4	$O(V+E)$
Linhas 5–8	$O(V+E)$
Total	$O(V+E)$

Solução

CONTA-CAMINHOS(G, s, t) $\quad \triangleright$ G é acíclico

1. **para cada** $v \in V[G] - \{t\}$ **faça**
2. $\quad p[v] \leftarrow 0$
3. $p[t] \leftarrow 1$
4. compute uma ordenação topológica $v_1, v_2, \ldots, v_n$ de G
5. **para** $i \leftarrow n - 1$ **até** 1 **faça** $\triangleright$ em ordem inversa
6. $\quad$ **para cada** $v \in \mathrm{Adj}[v_i]$ **faça**
7. $\quad\quad p[v_i] \leftarrow p[v_i] + p[v]$
8. **devolva** p

Para ver a corretude basta provar que no início da linha 5 vale o seguinte invariante: $p[v_i] = p(v_i)$. O algoritmo simplesmente implementa a recorrência do slide anterior.