



Grafos Orientados

Aluno: **Luiz Carlos da Silva Leão**

Professores: Diana Sasaki e Luerbio Faria

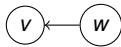
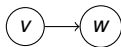
Universidade do Estado do Rio de Janeiro – UERJ

Seminário da disciplina de Grafos
Rio de Janeiro, 17 de Novembro de 2022

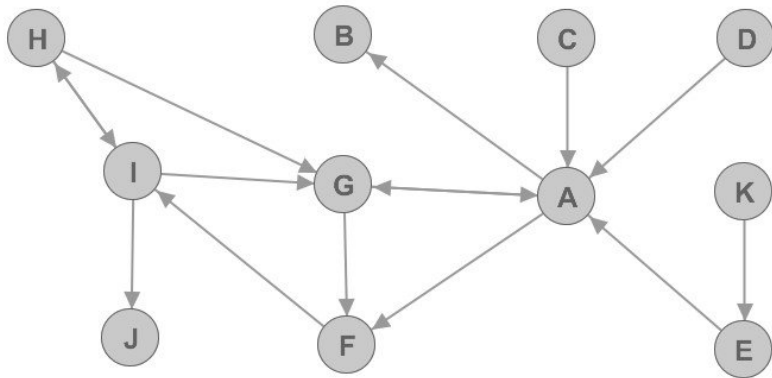
- 1** Definições
- 2** Conceitos
- 3** Teoremas
- 4** Exemplos
- 5** Algoritmos
- 6** Subclasses
- 7** Projeto de Pesquisa
- 8** Referências

Definições

- Um grafo orientado ou direcionado ou digrafo D é um par ordenado (V, E) , denotado por $D = (V, E)$, onde V é um conjunto finito e não vazio de vértices e E é um conjunto de pares ordenados de vértices distintos de V denominados arestas direcionadas ou arcos;
- Assim, em um digrafo, cada aresta direcionada v, w possui uma única direção de v para w ;
- Note que $(v, w) \neq (w, v)$.

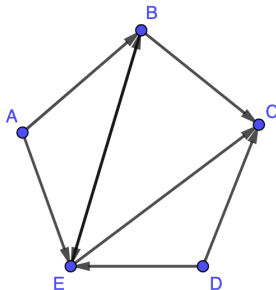


Exemplo de um grafo simples orientado



[1]

- Dada uma aresta direcionada $e = (v, w)$, dizemos que (v, w) é divergente a v e convergente a w
- Além disso, v é dito cauda de (v, w) e w a cabeça de (v, w)
- Em um digrafo simples D , se $(v, w) \in E(D)$, v é dito predecessor de w e w é dito sucessor de v
- Exemplo: $D = (V, E)$ $V = \{a, b, c, d, e\}$
 $E = \{(a, b), (b, c), (a, e), (b, e), (e, b), (e, c), (d, c), (d, e)\}$



O grafo G não direcionado obtido a partir de D pela remoção das direções das arestas é dito grafo subjacente de D .
 D é dito uma orientação de G .

Exemplo:

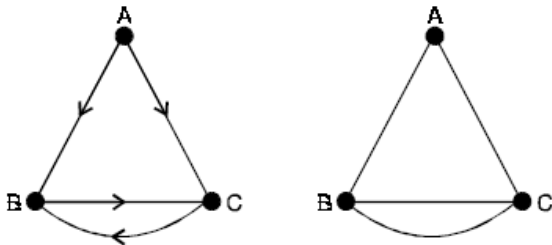


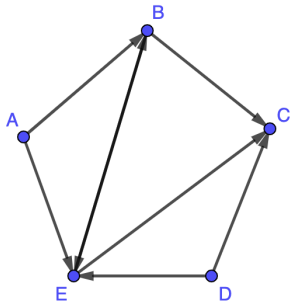
Fig. 8.3.(a), (b).

[2]

Conceitos

- Dizemos que um vértice v alcança um vértice u , se existe um caminho direcionado de v para u ;
- A vizinhança de entrada de v , denotada por $N^-(v)$ é o conjunto de todos os vizinhos de entrada de v , ou seja, todos os vértices com arestas direcionadas que convergem para v ;
- A vizinhança de saída de v , denotada por $N^+(v)$ é o conjunto de todos os vizinhos de saída de v , ou seja, todos os vértices com arestas direcionadas que divergem de v ;
- Os conceitos de passeio, trajeto, caminho, ciclo são definidos de maneira análoga à forma como foram definidos para grafos simples. Em digrafos podemos ter ciclos de comprimento 2

- O grau de entrada de v , denotado por $d^-(v)$ é o número de arestas direcionadas que convergem para v
- O grau de saída de v , denotado por $d^+(v)$ é o número de arestas direcionadas que divergem de v
 - Se $d^-(v) = 0$, então v é dito fonte de D ;
 - Se $d^+(v) = 0$, então v é dito sumidouro de D ;



- | | |
|----------------|--------------------------------------|
| • $d^-(a) = 0$ | • $d^+(a) = 2 \rightarrow$ FONTE |
| • $d^-(b) = 2$ | • $d^+(b) = 2$ |
| • $d^-(c) = 3$ | • $d^+(c) = 0 \rightarrow$ SUMIDOURO |
| • $d^-(d) = 0$ | • $d^+(d) = 2 \rightarrow$ FONTE |
| • $d^-(e) = 3$ | • $d^+(e) = 2$ |

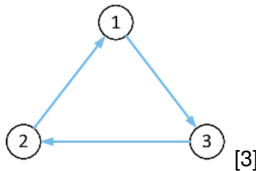
Vocês conseguem visualizar alguma relação entre os graus de saída e os graus de entrada?

$$\sum_{v \in V} d^+(v) = |E| = \sum_{v \in V} d^-(v)$$

Teoremas

Theorem

Se D é um digrafo sem ciclos direcionados, então D tem sumidouro e fonte.



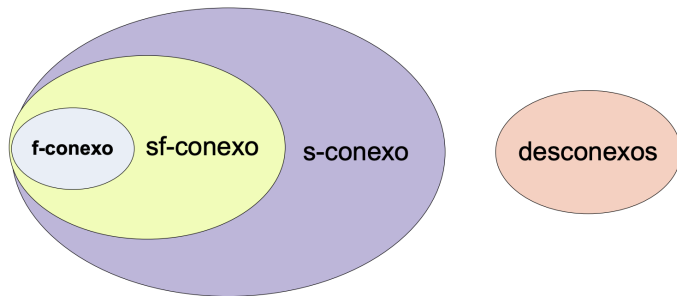
Proof.

Por absurdo, suponha que D não possui sumidouro. Seja $v_1 \in V$. v_1 não é sumidouro, logo existe $v_2 \in V$ tal que $v_1 v_2 \in E$. v_2 também não é sumidouro, logo existe $v_3 \in V$, $v_3 \neq v_1$ (caso contrário, D teria ciclo direcionado) tal que $v_2 v_3 \in E$. Note que esse processo se repetirá indefinidamente. Absurdo, pois D tem um número finito de vértices.

A prova da fonte é análoga.



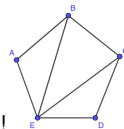
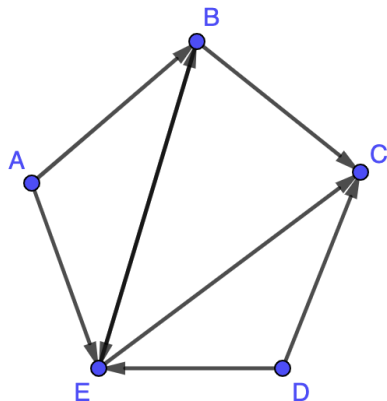
- Um digrafo $D = (V, E)$ é fracamente conexo (ou simplesmente conexo ou s-conexo) se o grafo subjacente de D for conexo.
- Um digrafo $D = (V, E)$ é unilateralmente conexo (ou semi-fortemente conexo ou sf-conexo) se para todo $u, v \in V$ u alcança v ou v alcança u ;
- Um digrafo $D = (V, E)$ é fortemente conexo (ou f-conexo) se para todo $u, v \in V$ u alcança v e v alcança u ;



[4]

Exemplos

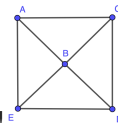
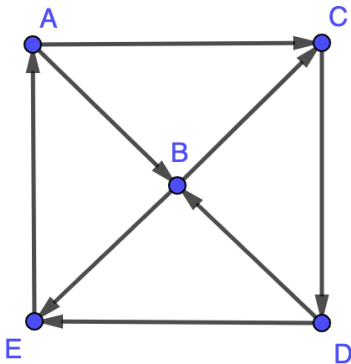
Exemplo 1



- D é fracamente conexo? SIM!
O grafo subjacente é conexo
- D é unilateralmente conexo? NÃO!
 a não alcança d ou d não alcança a
- D é fortemente conexo? NÃO!

Exemplo 2

Verifique se o digrafo D' abaixo é fortemente conexo:



- D' é fracamente conexo? SIM!
Pois o grafo subjacente é claramente conexo.
- D' é unilateralmente conexo? SIM!
Pois para todo $u, v \in D'$ u alcança v ou v alcança u ;
- D' é fortemente conexo? SIM!
Pois para todo $u, v \in D'$ u alcança v e v alcança u

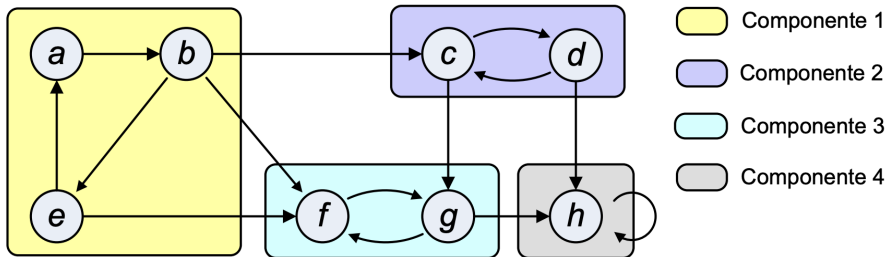
O exemplo acima nos mostra que que um digrafo fortemente conexo é claramente fracamente conexo e unilateralmente conexo

Algoritmos

Seja um grafo $G = (V, E)$ onde V é o conjunto de vértices e E o conjunto de arestas, um subconjunto de vértices C ($C \subseteq V$) é um componente fortemente conectado se respeita as seguintes propriedades:

- Para todo par de vértices (u, v) em C , existe um caminho de u a v e vice-versa;
- C é maximal em relação à propriedade acima.

Quais são os componentes fortemente conectados do grafo abaixo?




[4]

- **Um problema:** Em redes sociais, como detectar grupos de pessoas fortemente conectadas para sugerir algo como amizades, anúncios e jogos para nichos de audiência específicos?;
- Como detectar todos os componentes fortemente conexos de um digrafo?;
- Iremos apresentar para isso o algoritmo de Kosaraju;
- A complexidade desse algoritmo é $O(V + E)$.

- Crie uma pilha vazia e faça uma busca em profundidade (DFS) no grafo;
- Ao final de cada busca em profundidade, insira o vértice na pilha como mostra o algoritmo:

Inicialmente o vetor de visitados é preenchido com falso.

```
void preenche(vertice, visitados[])
{
    visitados[vertice] = verdade
    para cada adjacente do vertice
    {
        se o adjacente não foi visitado
            preenche(adjacente, visitados)
    }
    insere o vertice na pilha
}
```



- Obtenha o grafo transposto;
 - Basta inverter as orientações das arestas do grafo original;
 - Os vértices são os mesmos do grafo original, só inverte as arestas!

```
Grafo obter_transposto()
{
    Grafo grafo(total_vertices);

    para v = 0 até total_vertices - 1
    {
        para cada adjacente "adj" de v
        {
            insira v na lista de adjacentes de adj
        }
    }
    retorna grafo
}
```

- Marque novamente todos os vértices como não visitados para a segunda busca em profundidade;
- Enquanto a pilha não for vazia:
 - Obtenha o elemento do topo;
 - Remova o elemento do topo;
 - Se esse elemento não foi visitado, realize uma busca em profundidade nele;

```
enquanto a pilha não for vazia
{
    v = pilha.topo()
    pilha.pop()

    se visitados[v] == falso
        DFS(grafo_transposto, visitados)
}
```

Algoritmo da função de busca em profundidade:

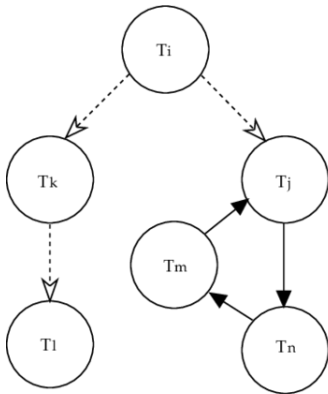
```
void DFS(vertices, visitados)
{
    visitados[vertices] = verdade
    imprime vertices

    para cada adjacente do vértice
    {
        se o adjacente não foi visitado
            DFS(adjacente, visitados)
    }
}
```

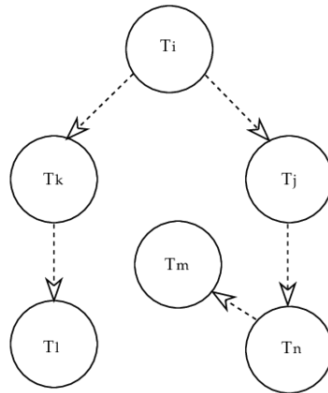
Link da implementação em C++ do algoritmo de Kosaraju:

<https://github.com/lcs188/Grafos20222/blob/main/kosaraju.cpp>

Subclasses

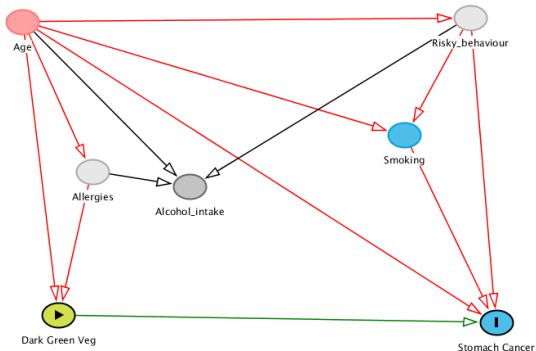


(a) Wait-for graph with a cycle



(b) Wait-for graph with no cycles

[5]



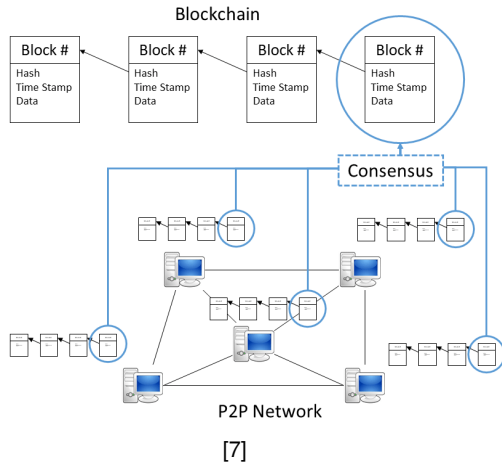
[6]

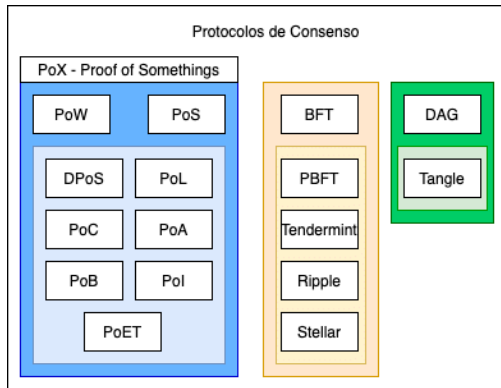
DAGitty: a browser-based environment for creating, editing, and analyzing DAGs

- Um grafo acíclico dirigido (DAG) é um grafo direcionado e sem ciclos;
- Isso significa que não é possível percorrer todo o grafo começando de uma aresta;
- As arestas do grafo acíclico dirigido têm apenas uma direção.

Projeto de Pesquisa

- O que é uma Blockchain?
- Como a Blockchain funciona?
- Blockchain como meio de comunicação

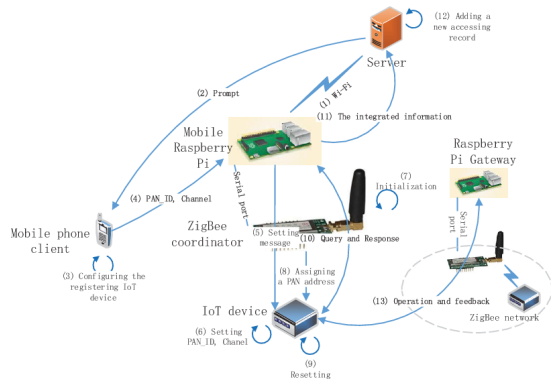




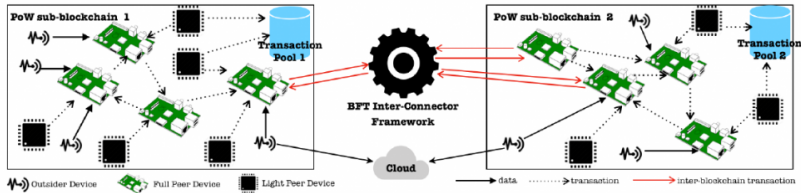
[8]

- O consenso é um problema fundamental em computação distribuída
- Permite com que um conjunto de participantes (ou nós) numa rede chegue a um acordo sobre um conjunto de transações, ou sobre um determinado estado do sistema
- O consenso, portanto, mantém o estado consistente das réplicas e a disponibilidade do sistema

- O que é IoT?
- Como se dá a comunicação entre dispositivos IoT?
- O que são recursos, agentes e dispositivos para IoT?



[9]



[10]

- Quais os riscos na comunicação entre dispositivos IoT?
- O que a Blockchain resolve e porque é útil para comunicação entre dispositivos?
- Os requisitos de IoT estarão atendidos com a Blockchain?

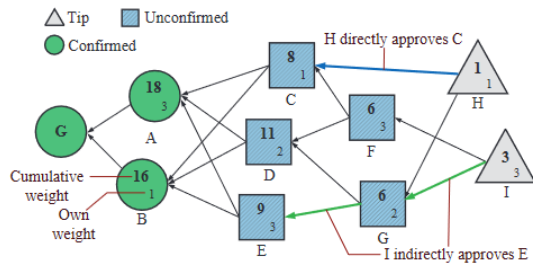
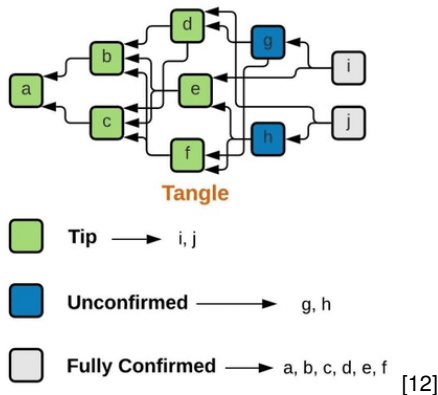


THE BACKBONE OF IOT IS HERE

Scalable, Decentralized, Modular, No Fees

[11]

Iota Tangle Visualization



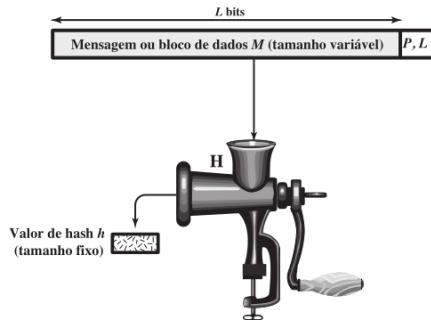
As funções hash são funções determinísticas criptográficas eficientes que transformam qualquer tipo de dado em um número de tamanho fixo, pseudo-aleatório, independentemente do tamanho dos dados de entrada



=

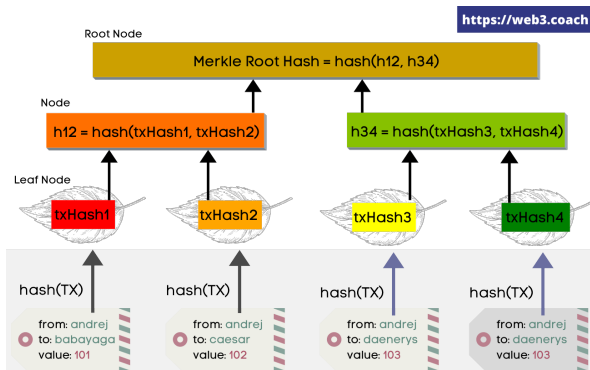
79054025
255fb1a2
6e4bc422
aef54eb4

[14]



P, L = preenchimento mais campo de tamanho

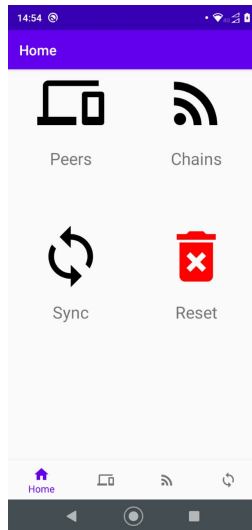
[15]



[16]

- Uma árvore de merkle, é uma estrutura de dados em que cada nó deve ser identificado com um identificador exclusivo (de hash);
- O nó pai terá um identificador exclusivo resultante do hash de seus nós filhos. Essa estrutura se repete até que o nó raiz ou raiz Merkle (raiz Merkle), cuja impressão está associada a todos os nós da árvore;
- Desta forma, a verificação e validação desses dados podem se tornar muito eficientes, tendo que verificar apenas a Raiz Merkle em vez de toda a estrutura.

- O Freechains é um sistema peer-to-peer de disseminação de mensagens por tópicos através do padrão publish-subscribe;
- As postagens são estruturadas em um grafo acíclico direcionado criptográfico que é imune a modificações (Merkle DAG);
- Integrantes do Projeto de Pesquisa: Prof. Dr. Francisco Sant'Anna, Fabio Bosisio, Lucas Pires, Alexis Pinheiro e Luiz Leão



Referências



Stefan Kostić, Mirjana Simic, and Miroljub Kostić.

Social network analysis and churn prediction in telecommunications using graph theory.

Entropy, 22:753, 07 2020.



skedsoft.

Types of digraphs.

<https://www.skedsoft.com/books/graph-theory/types-of-digraphs>.



Huan Gao, Yongqiang Wang, and Angelia Nedić.

Dynamics based privacy preservation in decentralized optimization, 2022.



Humberto César Brandão de Oliveira.

Aula 11 - conectividade.

https://www.bcc.unifal-mg.edu.br/~humberto/disciplinas/2010_2_grafos/pdf_aulas/aula_11.pdf.



Elisa Bertino, Benjamin Catania, and A. Vinai.

Transaction models and architectures.

08 2000.



Chaochen Wang.

Directed acyclic graphs (dags).

URL: <https://wangcc.me/DAG-CSS/#33>.



Wei Cai, Zehua Wang, Jason Ernst, Zhen Hong, and Chen Feng.

Decentralized applications: The blockchain-empowered software system.


IEEE Access, 6:53019–53033, 10 2018.



Jauberth Abijaude, Fabíola Greve, and Péricles Sobreira.

Blockchain e Contratos Inteligentes para Aplicações em IoT, Uma Abordagem Prática, pages 149–197.

Sociedade Brasileira de Computação, 07 2021.

 Ming Tao, Xiaoyu Hong, Chao Qu, Jie Zhang, and Wenhong Wei.

Fast access for zigbee-enabled iot devices using raspberry pi.

2018 Chinese Control And Decision Conference (CCDC), pages 4281–4285, 2018.

 Emanuele Ragnoli.

A blockchain architecture for the internet of things.

[https:](https://www.ibm.com/blogs/research/2018/10/blockchain-internet-of-things/m)

[//www.ibm.com/blogs/research/2018/10/blockchain-internet-of-things/m](https://www.ibm.com/blogs/research/2018/10/blockchain-internet-of-things/m).

Acessado: 2022-06-01.

 Valerio Vaccaro.

Iota - the backbone of iot is here.

https://miro.medium.com/max/1400/1*IkWDoX1XWc1mM1re2DcLXQ.png.

Acessado: 2022-06-14.



Mohd Akhtar, Danish Rizvi, Mohd Ahad, Salil Kanhere, Mohammad Amjad, and Giuseppe Coviello.

Efficient data communication using distributed ledger technology and iota-enabled internet of things for a future machine-to-machine economy.

Sensors, 21:4354, 06 2021.



Bin Cao, Yixin Li, Lei Zhang, Long Zhang, Shahid Mumtaz, Zhenyu Zhou, and Mugen Peng.

When internet of things meets blockchain: Challenges in distributed consensus.

IEEE Network, PP:1, 03 2019.



Jeff Atwood.

Speed hashing.

<https://blog.codinghorror.com/speed-hashing/>.

Acesso em: 19 out. 2019.



W. Stallings.

Cryptography and network security.

Prentice Hall, 2003.



web3coach.

Exploring blockchain filesystem and merkle trees in go.

<https://pt.slideshare.net/mcastrosouza/algoritmo-de-kosaraju>.



Marcos Castro.

Algoritmo de kosaraju.

<https://pt.slideshare.net/mcastrosouza/algoritmo-de-kosaraju>.



Francisco Sant'Anna, Fabio Bosisio, and Lucas Pires.

Freechains: Disseminação de conteúdo peer-to-peer.

In Anais Estendidos do XX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, pages 101–108, Porto Alegre, RS, Brasil, 2020. SBC.

Grafos Orientados

Aluno: **Luiz Carlos da Silva Leão**

Professores: Diana Sasaki e Luerbio Faria

Universidade do Estado do Rio de Janeiro – UERJ

