Algoritmos e Estruturas de Dados no Big Data
MLP - ESBD1 - Aula 3

Importância dos breaks a cada 50 min / 1 hora

Já vimos estruturas que organizam dados em função de uma (ou mais chaves).

- Agora veremos como organizar dados que tem relação entre si.

Overview:

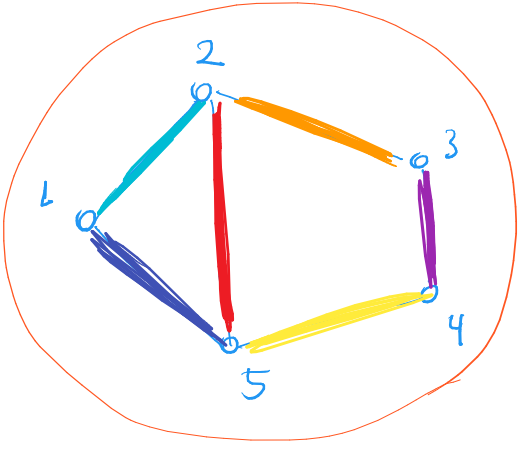
- Grafos
- Filas
- Busca em largura
- Apresentação Atividade 2

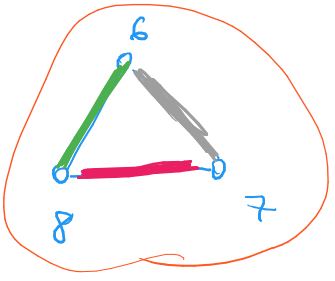
Definição e exemplos: Grafos não orientados

$G = (V, E)$

o nós

— arestas





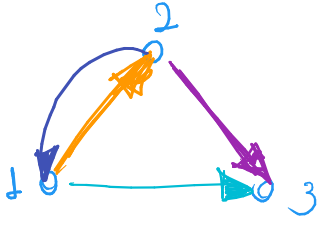
$V = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$E = \{\{1,2\}, \{2,3\}, \{2,5\}, \{1,5\}, \{5,4\}, \{4,3\}, \{6,8\}, \{8,7\}, \{6,7\}\}$

Definição e exemplos: Grafos orientados ou dirigidos

o nós

→ arcos



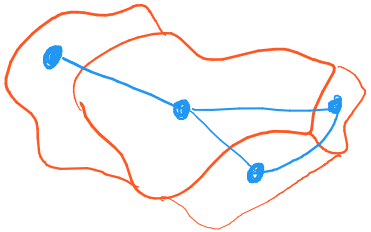
$D = (V, A)$

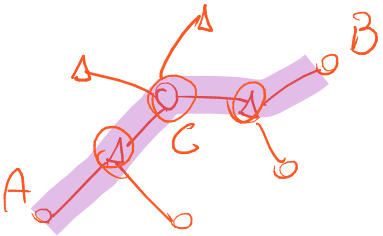
$V = \{1, 2, 3\}$

$A = \{(1,2), (2,3), (1,3), (2,1)\}$

Cenários e aplicações (redes físicas, abstratas e relacionais)

	Vértices?	Arestas?	Orientado?
Redes físicas (transportes, circuitos elétricos/eletrônicos, comunicação, água)	cidades componentes e conexões	rodovias trilhas	não orientado orientado
Redes conceituais (sociais, biológicas, Web)	perfis	conexões	facebook não orientado twitter orientado
Estruturas de dados (árvores, listas ligadas)	nós ou células	apontadores pra, filhos	orientados
Mapas	territórios	fronteira	não orientado
Relações de dependência ou interação (disciplinas, filmes e atores)	disciplinas	pré-requisito	orientado



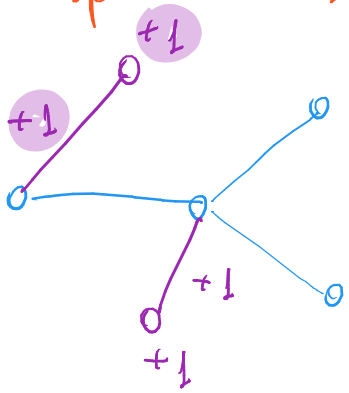


Tipos de grafos:

- dirigidos vs. não-orientados
- esparsos vs. densos
  - Como varia $|E|$ em função de $|V|$?

Grafos densos sem auto laço ou arestas multiplas

Grafos esparsos mas conexos

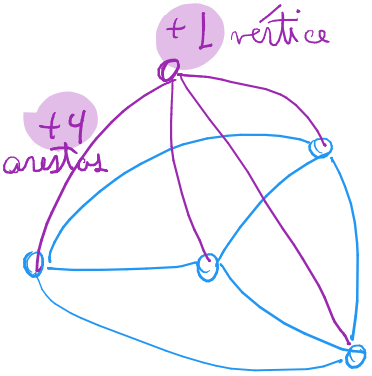


$|V| = n = 4$

$|E| = m = 3$

$n - m = 1$

$m = n - 1 = O(n)$



$|V| = n = 4$

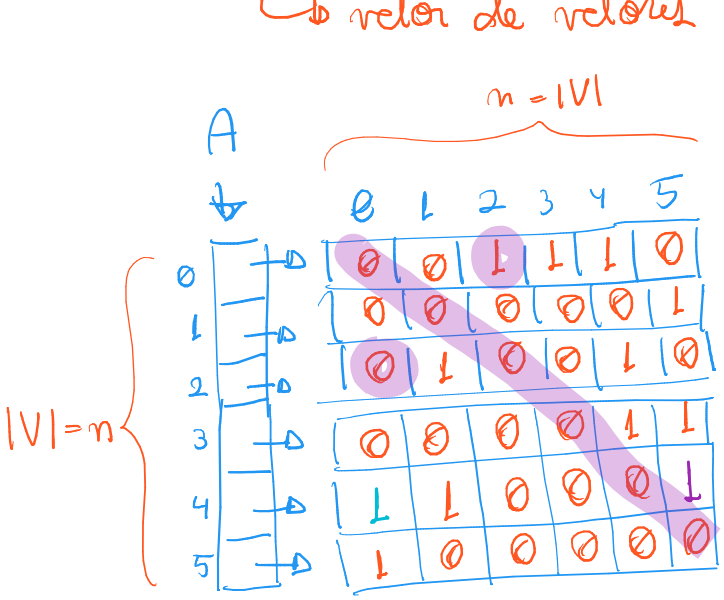
$|E| = m = 6 = \frac{4 \cdot 3}{2}$

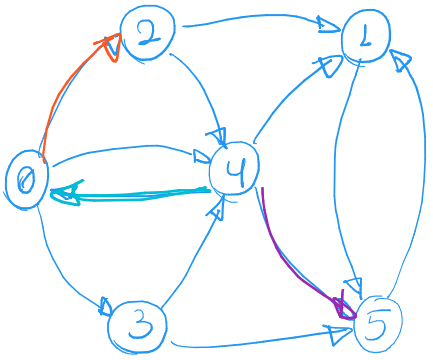
$m = \frac{n(n-1)}{2} = O(n^2)$

- Quais são mais comuns na prática? 01 esparsos

Implementações: matriz de adjacências (revisitando vetores)

↳ vetor de vetores





A[i][j] = 1 sse existe arco de i p/ j

$(i,j) \in E$

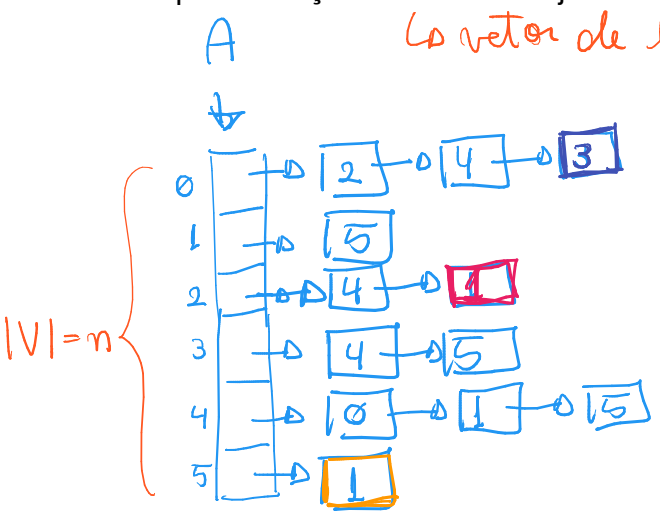
✓ O(1) p/ acessar uma posição

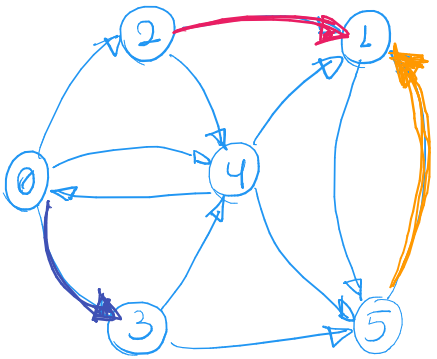
× tamanho $O(n^2)$ mesmo que o grafo seja esparso

× percorrer todos os vizinhos leva tempo O(n)

} ineficiente se esparso

Implementações: listas de adjacências (revisitando listas ligadas)





✓ econômica em memória p/ grafos esparços $O(n+m)$

✓ mais eficiente p/ percorrer em sequência os vizinhos

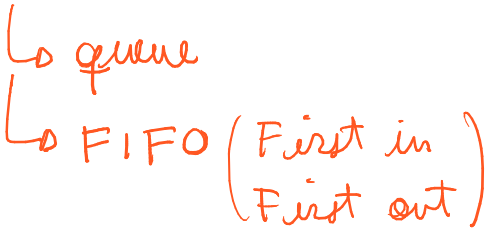
✗ não é eficiente p/ acessos diretos

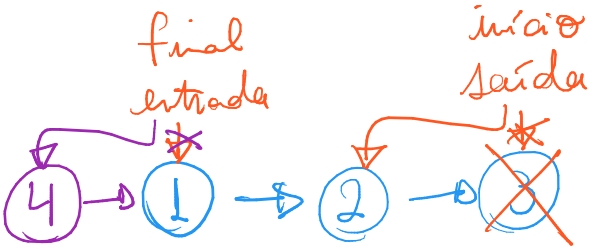
Matriz de adjacência vs. listas de adjacência:

- eficiência de espaço das implementações (analisar densos vs. esparsos),
- eficiência de tempo das operações.

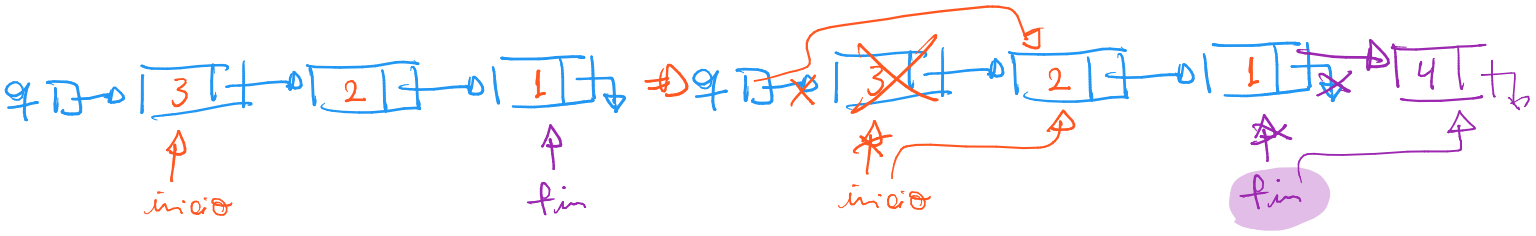
https://colab.research.google.com/drive/1eN5RybekBsGzutbOrY975piFFTxUUitn

Filas: Comportamento e exemplo de fila,





- Implementação em vetor, vetor circular e lista ligada.

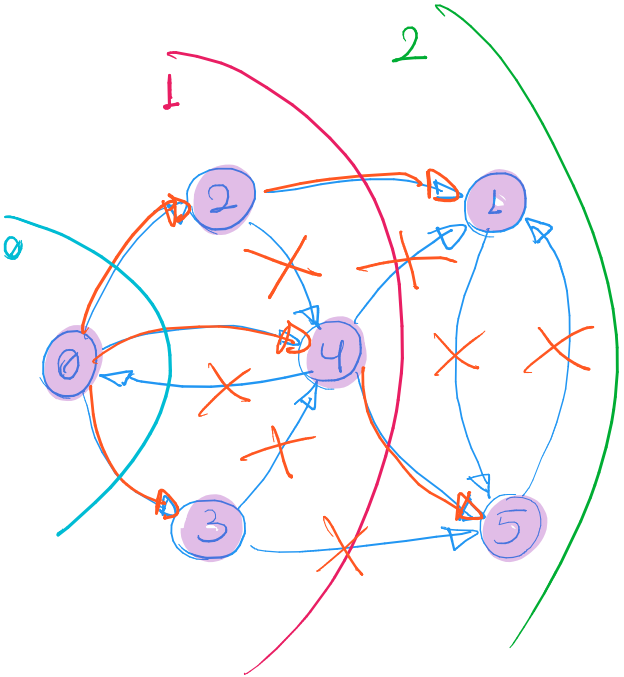


- Exemplo/aplicação?

https://colab.research.google.com/drive/1eN5RybekBsGzutbOrY975piFFTxUUitn

Busca em largura:

- ideia de visitar vértices por camadas centradas na origem,
- exemplo do cálculo de distância.



origem 0

encontrados mas ainda não visitados

fila | 0 | 2 | 1 | 4 | 3 | 5 |

vértices	0	1	2	3	4	5
dist	0	2	1	1	1	2

dist em # de arestas

Pseudocódigo da busca em largura:

Vs c/ calculo de distancias

```
buscaLargDist(G=(V,E), s):
    p/ todo v ∈ V: dist[v] = +∞        ——— O(n)
    dist[s] = 0                        ——— O(1)
    coloque s na fila Q                — O(1)
    enquanto Q ≠ ∅:                    ——— no total, ao longo das n iterações do enquanto, leva tempo O(m)
        v = Q.pop
        para cada u vizinho de v:      — O(|δ+(v)|)        O(n)
            se dist[u] = +∞:
                dist[u] = dist[v] + 1  — O(1)
                coloque u em Q
```

⊗ adicionar vetor p/ guardar o predecessor de cada vértice no caminho

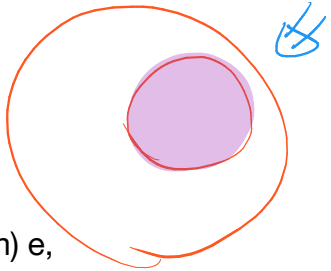
- análise de eficiência. ↳ $O(n+m)$

Extra da terceira aula: amostragem uniforme de grandes volumes de dados.

Sortear um elemento de um conjunto

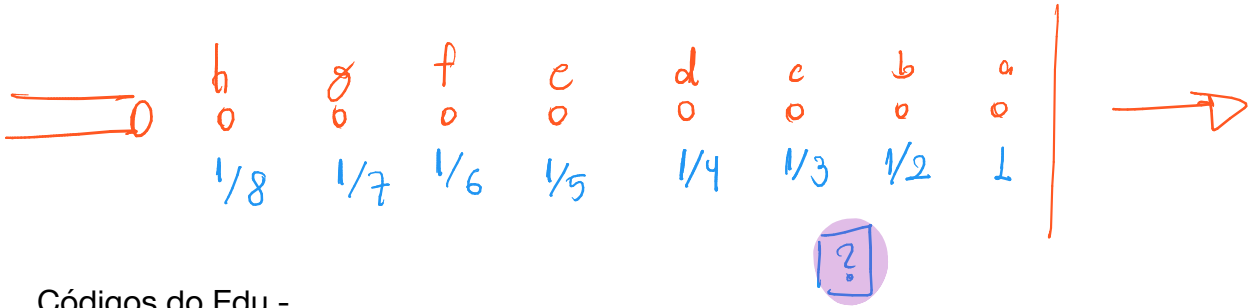
- de modo aleatório com probabilidade uniforme não costuma ser difícil.

- Mas, e se o tamanho do conjunto for tão grande
  - que você não consegue mantê-lo armazenado?
- Esse é o diferencial desse algoritmo simples e elegante,
  - que armazena apenas um elemento por vez.



A ideia do algoritmo é considerar os elementos em um fluxo (stream) e,

- ao considerar o k-ésimo elemento que chegou,
  - escolher mantê-lo selecionado com probabilidade 1/k.
- Por que ele funciona?



Códigos do Edu - https://colab.research.google.com/drive/1rU_Ee5jaEhyagPGPtLz-mwCDm2yHK6ka?usp=sharing (http://bit.ly/EduardoMolinaAula3) MLP ESBD1 Aula 3 Códigos

Atividade 2

Graus de Separação em Redes Sociais: considere o grafo de uma rede social, em que cada perfil corresponde a um vértice, e existe uma aresta entre dois vértices apenas se os perfis correspondentes são amigos. Queremos saber qual o grau de separação médio entre dois perfis da rede. Isto é, saber quantas arestas tem, em média, um caminho mais curto que conecta dois perfis. Para tanto, elaboramos o seguinte experimento.

Temos uma **função de construção** que recebe como entrada um # (número) de vértices, um # de arestas, e que usa esses parâmetros para gerar aleatoriamente o grafo de uma rede. Também temos uma **função de testes** que recebe o grafo gerado anteriormente, realiza uma bateria de 100 testes, calcula a média dos resultados dos testes, e devolve o grau de separação médio encontrado. Em cada uma das 100 iterações a **função de testes** sorteia dois perfis (vértices) da rede e chama uma **função de busca** para calcular o grau de separação dos vértices sorteados.

No entanto, essa **função de busca** não está implementada. *Sua missão é implementá-la*, realizar os testes para cada par (# de vértices, # de arestas) indicados na tabela a seguir, e *preencher cada célula da tabela* com o valor da média dos graus de separação encontrado. Atente que, na tabela estão indicados

os # de arestas por vértice, mas a **função de construção** recebe o número total. Com a tabela preenchida, *analise brevemente como cresce o grau de separação médio* em função do # de vértices e de arestas do grafo.

	# Médio de Arestas por Vértice		
# Vértices (n)	5	raiz(n)	n / 5
100			
1000			
10000			
100000			

Realizados os testes anteriores, temos mais um desafio. Considere agora a versão alternante do grau de separação (distância) entre dois perfis da rede. Na versão alternante só são considerados caminhos em que sucedendo um perfil masculino vem um perfil feminino e vice-versa. *Faça uma nova versão da sua **função de busca*** para considerar apenas caminhos alternantes, refaça os testes usando essa

nova função e *preencha a próxima tabela* com os graus médios de separação encontrados. *Como essa versão do problema afetou os graus de separação?*

	# Médio de Arestas por Vértice		
# Vértices (n)	5	raiz(n)	n / 5
100			
1000			
10000			
100000			

Para ajudar vocês a realizar os testes, preparamos o seguinte material:

- https://colab.research.google.com/drive/1IlQyIG6DchtF1Saqpnb6hQJ-R9iPYC9R?usp=sharing MLP ESBD1 Atividade 2

Vídeo sobre o assunto “The Science of Six Degrees of Separation” - https://youtu.be/TcxZSmzPw8k

Material complementar

Grafos e suas implementações:

- [Playlist] Grafos: tipos, implementação e construção aleatória - http://bit.ly/MarioSanFeliceGrafosVideo
- [PDF] Grafos: tipos, implementação e construção aleatória - http://bit.ly/MarioSanFeliceCompGrafosPDF

Filas e Busca em largura:

- [PDF] Fila implementada em vetor, interfaces, cálculo de distâncias - http://bit.ly/MarioSanFeliceCompFilaVetorPDF
- [PDF] Filas em vetor circular e em lista ligada, interfaces, listas de adjacência e ortogonais - http://bit.ly/MarioSanFeliceCompFilaCircularListaPDF
- [Playlist] Busca em largura, caminhos mínimos em grafos não ponderados - http://bit.ly/MarioSanFeliceBuscaLargVideo
- [PDF] Busca em largura, caminhos mínimos em grafos não ponderados - http://bit.ly/MarioSanFeliceCompBuscaLargPDF