Técnicas de programação para Games
Aula10
Grafos

Professor Mestre: Adilson Lopes Khouri

4 de novembro de 2018

Sumário

Grafos

Cronograma

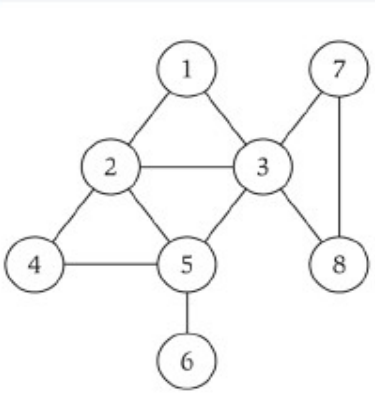
Aula	**Conteúdo**
12/04/2018	XP e banco de dados
17/04/2018	Introdução de estruturas de dados
24/04/2018	Arrays / Matrizes e Ordenação
26/04/2018	Recursão
03/05/2018	Lista Ligada
08/05/2018	Pilha, Fila
10/05/2018	Hash
15/05/2018	Árvore Binária
17/05/2018	Heap
22/05/2018	Grafos
24/05/2018	Prova

Grafos

- Um grafo é definido por $G = (V, E)$ onde:
- V é um conjunto de vértices
- E é um conjunto de arestas que relaciona, de alguma forma, os vértices
- $n = |V|$que é o número de vértices
- $m = |E|$ que é o número de arestas

Grafos

Podemos ver um exemplo de grafo não dirigido:



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

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

$m = 11, n = 8$

Figura: Exemplo de grafo não dirigido [1]

Grafos: Exemplo

Exemplo de uso:

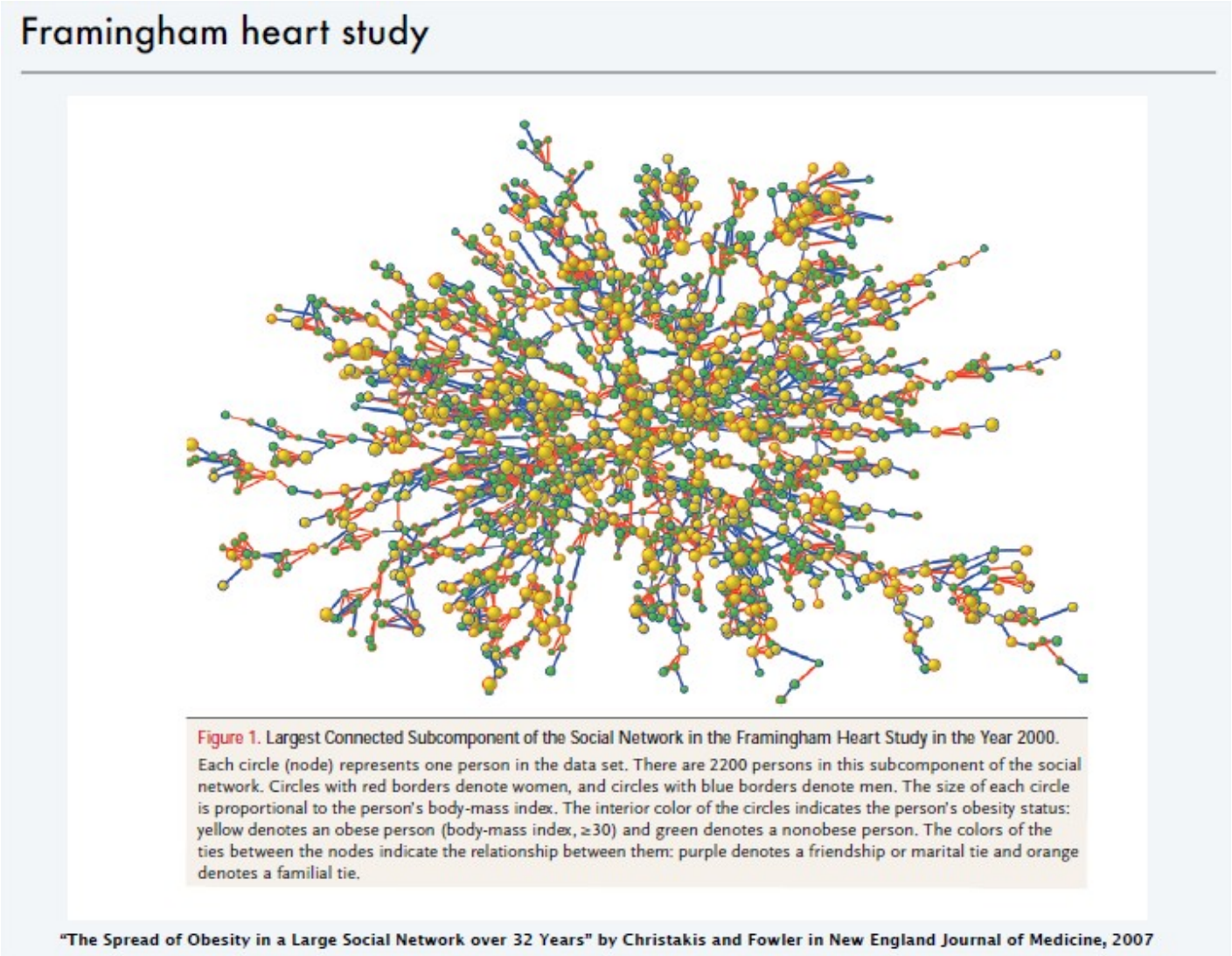


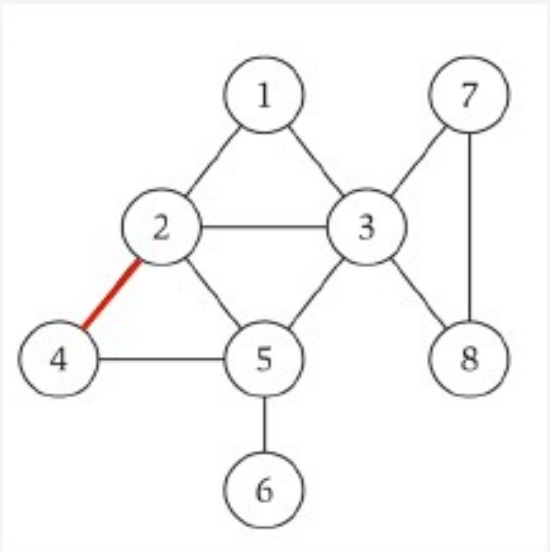
Figura: Exemplo de grafo não dirigido [1]

Grafos

- Qual a relação entre os vértices? Potencialmente qualquer uma, o que torna os grafos uma estrutura de dados com alto poder de modelar o mundo
- Diversos problemas científicos são modelados como grafo
- Em games labirintos podem ser representados como grafos
- Em computação problemas de caminhos mínimos podem ser representados por grafos

Grafos: representação

Podemos representar grafos por matriz de adjacência. Onde cada aresta aparece repetida como na ilustração:



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

Figura: Grafo representação por matriz de adjacência [1]

Grafos

- Quais os pontos positivos e negativos sobre essa representação?
- Vocês já tem o conhecimento sobre como implementar essa solução. Obs: é uma matriz..
- Qual a complexidade de operações nessa estrutura de dados?

Grafos: representação

Uma outra forma para representar grafos é usar uma lista de adjacências. Como na ilustração:

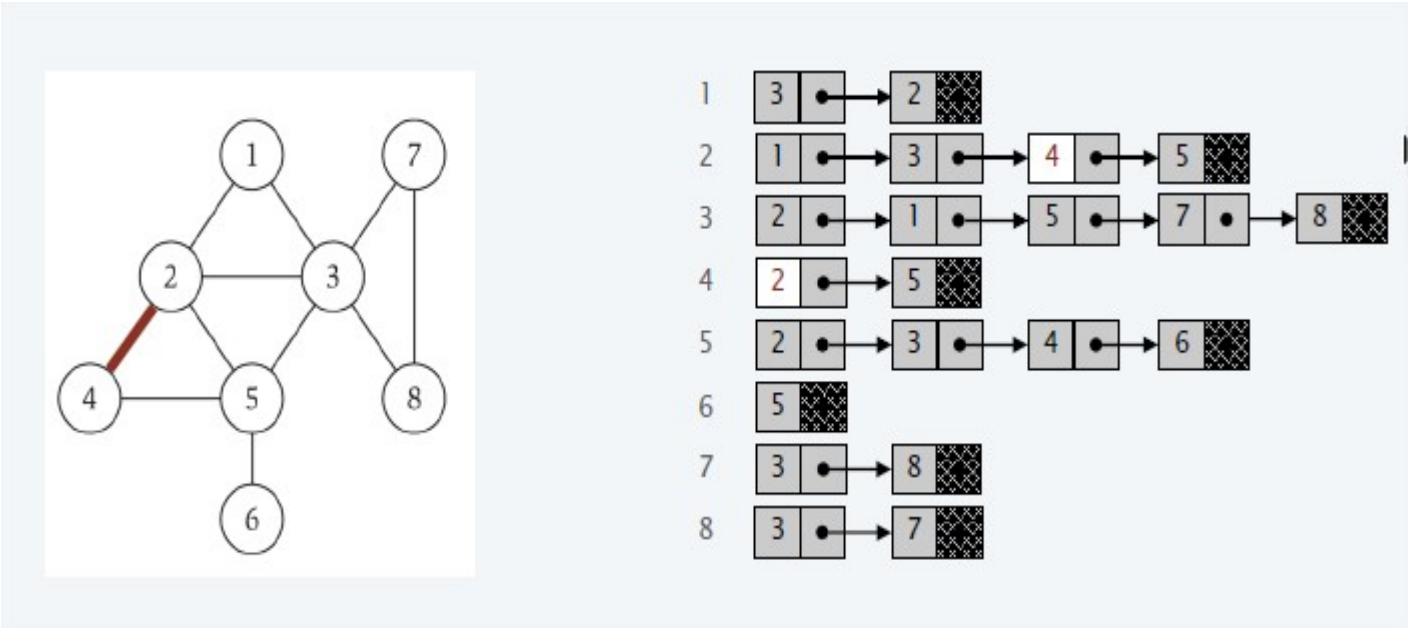


Figura: Grafo representação por lista de adjacências [1]

Grafos

- Quais os pontos positivos e negativos sobre essa representação?
- Vocês já tem o conhecimento sobre como implementar essa solução. Obs: é uma lista..
- Qual a complexidade de operações nessa estrutura de dados?

Grafos

- Programar com os alunos em C#
- Discutir no quadro a comparação com outras estruturas

Grafos: definições

- Caminho (Path) é uma sequência de nós conectados por arestas
- Um grafo não dirigido é conectado se todos os pares de nós, dois a dois, tem uma aresta entre eles
- Um ciclo em grafo não dirigido é um caminho onde o primeiro e último nó são iguais

Grafos: definições

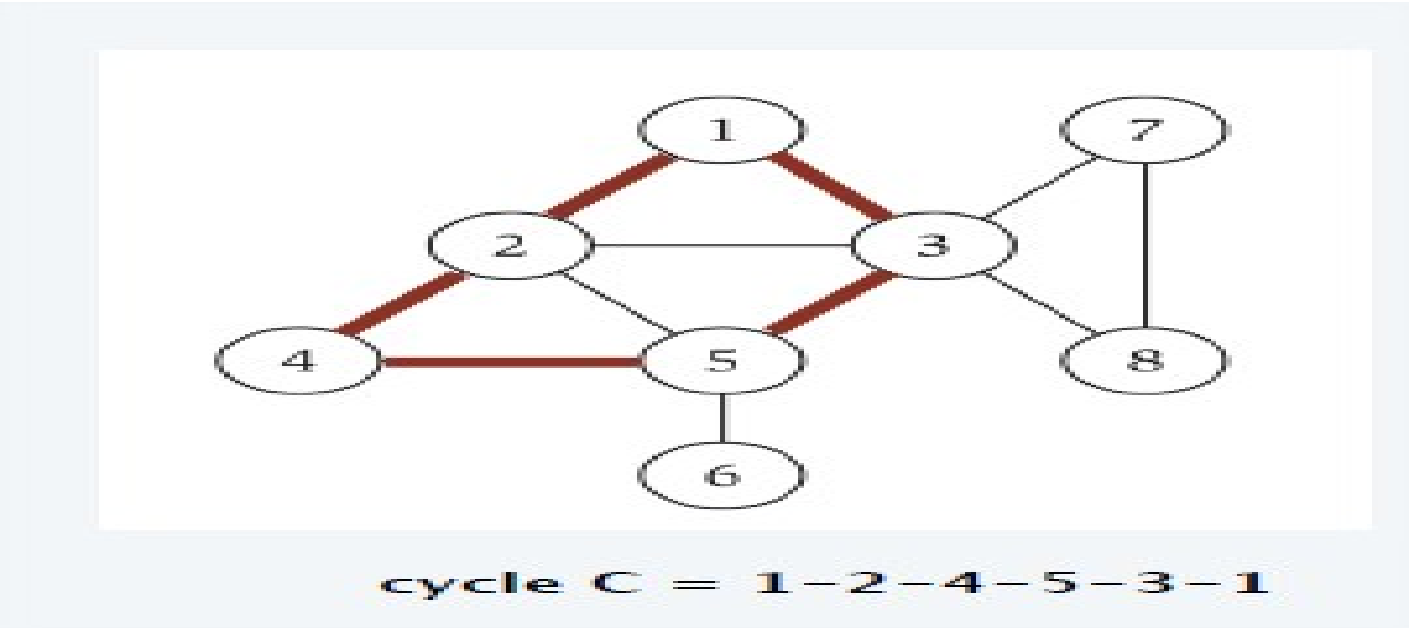


Figura: Grafo com ciclo [1]

Grafos: definições

Um grafo dirigido é considerado um árvore se ele não contém ciclos e estiver conectado.

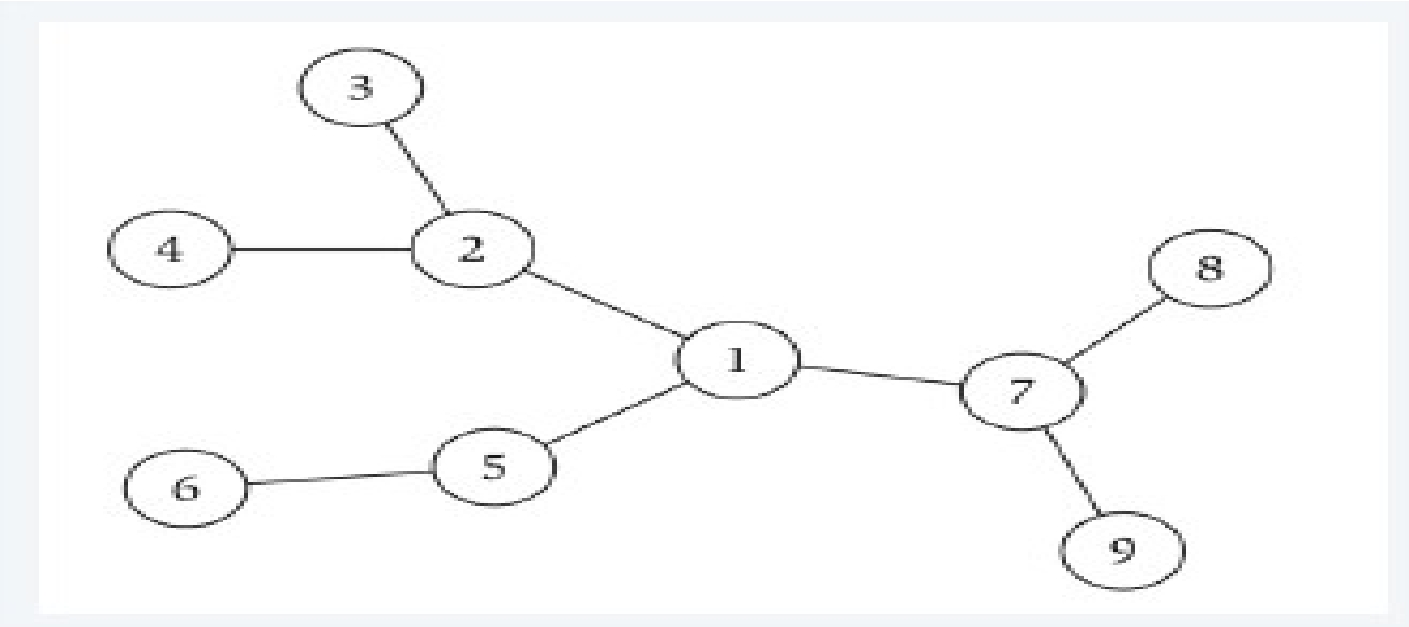


Figura: Grafo com ciclo [1]

Grafos: definições

Ao *criar uma raiz* para o grafo, obtemos:

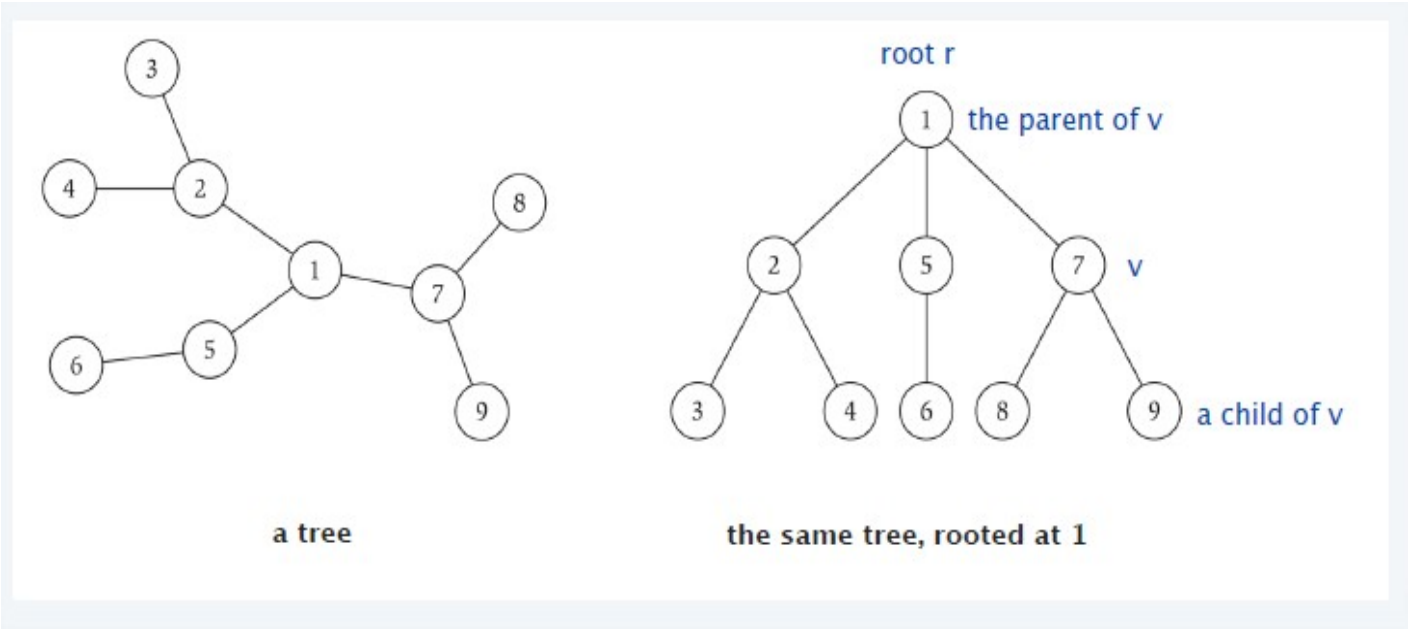


Figura: Grafo com ciclo [1]

Grafos: problemas

- Existe um caminho entre dois nós?
- Dados dois nós qual o caminho mínimo entre eles? Isso é uma forma de perguntar qual o menor caminho para atravessar um labirinto.
- Alguma sugestão de como solucionar esse problema? De forma computacional

Grafos: busca em largura

- Dado um nó inicial, percorre o grafo todo de forma sistemática usando a seguinte estratégia:
- Percorra os vizinhos do nó inicial, marque-os como visitados com um flag. Em seguida, percorra os vizinhos dos vizinhos ... até que não tenham nós sem visitar.
- Dado que grafos possuem ciclos, precisamos marcar os nós já visitados para evitar entrar em loop infinito.

Grafos: busca em largura

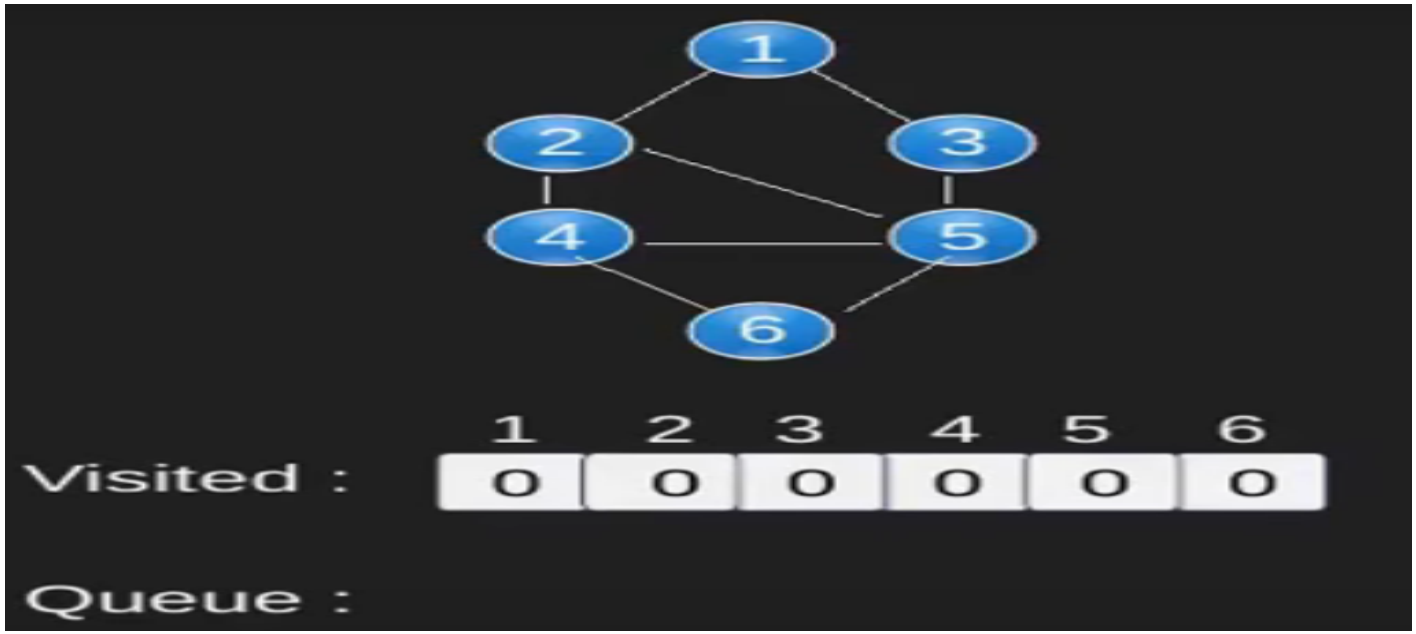


Figura: Grafo busca em largura [1]

Grafos: busca em largura

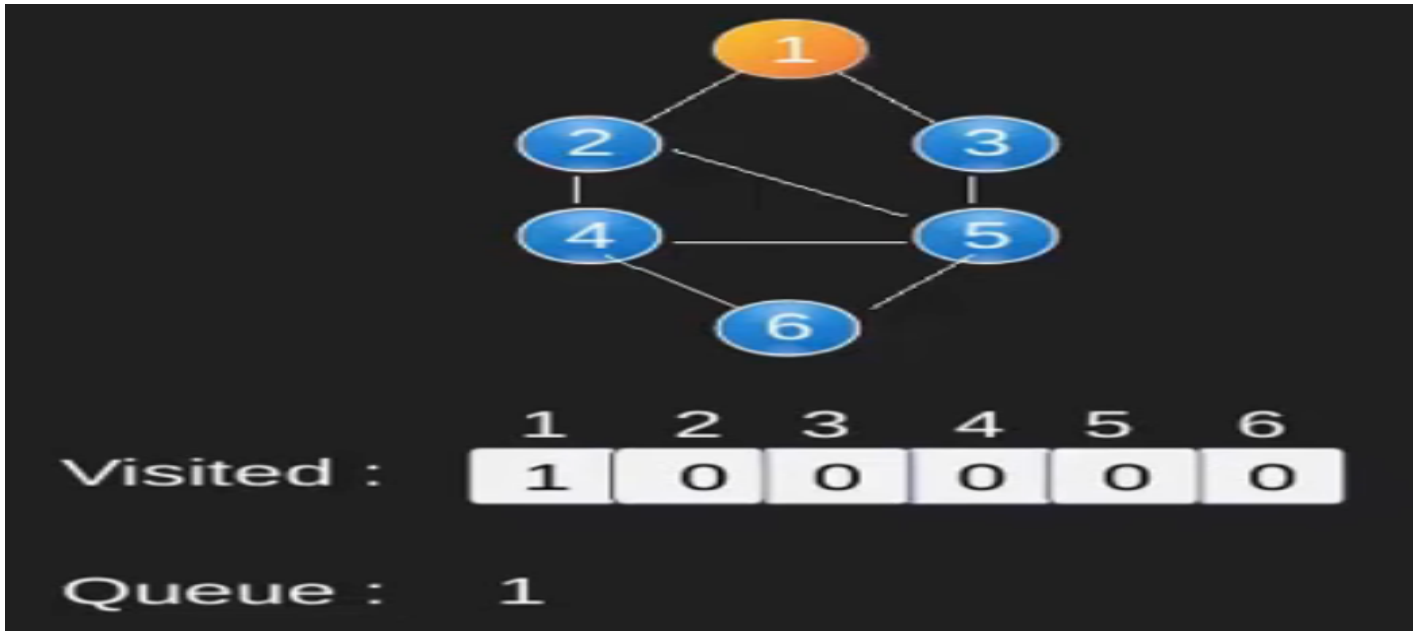


Figura: Grafo busca em largura [1]

Grafos: busca em largura

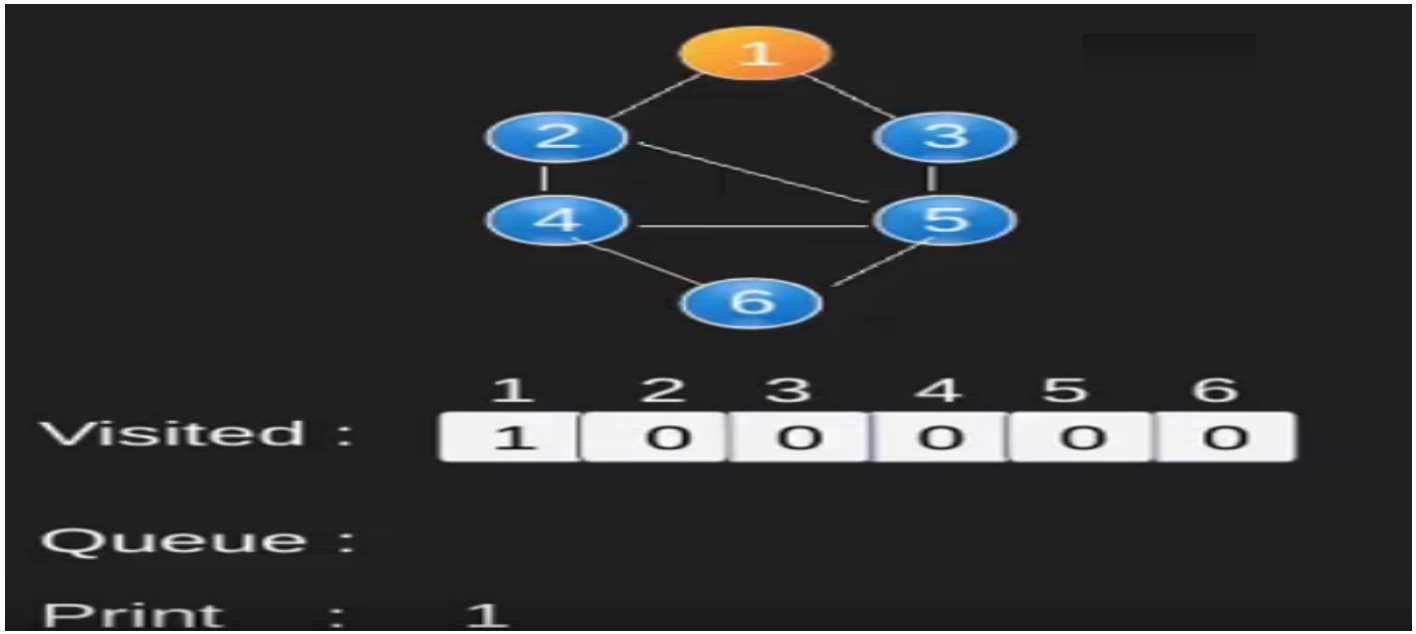


Figura: Grafo busca em largura [1]

Grafos: busca em largura

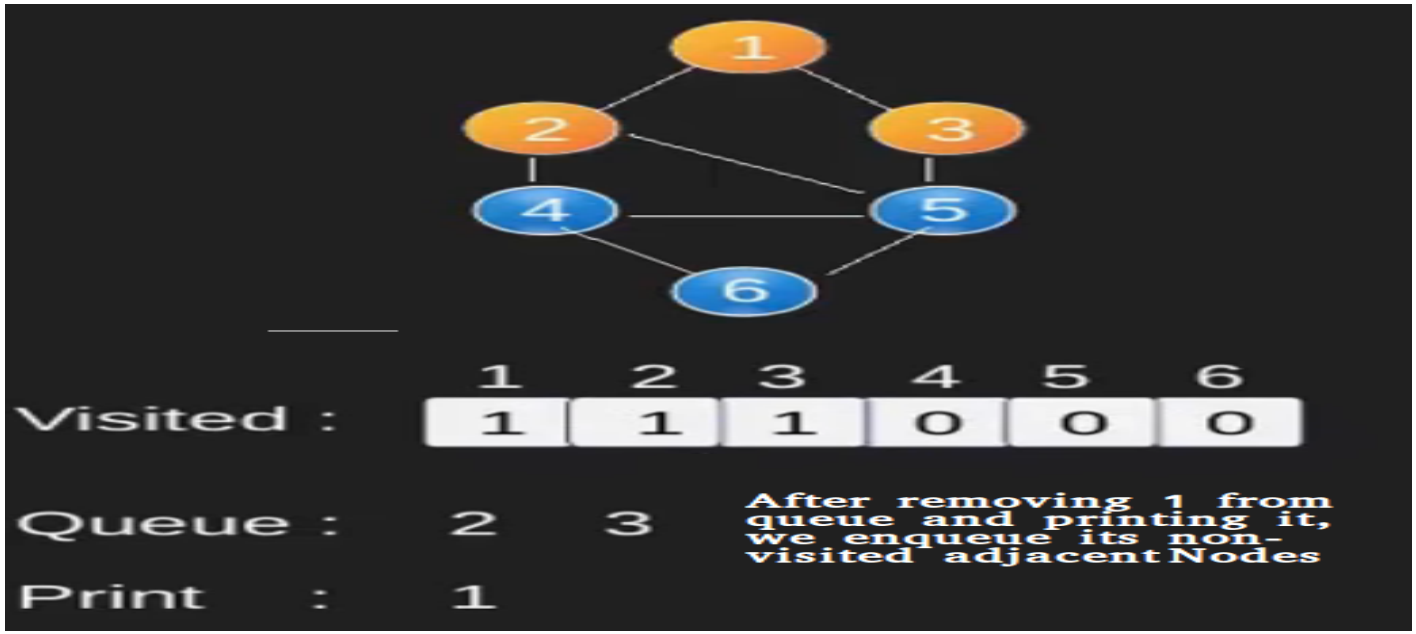


Figura: Grafo busca em largura [1]

Grafos: busca em largura

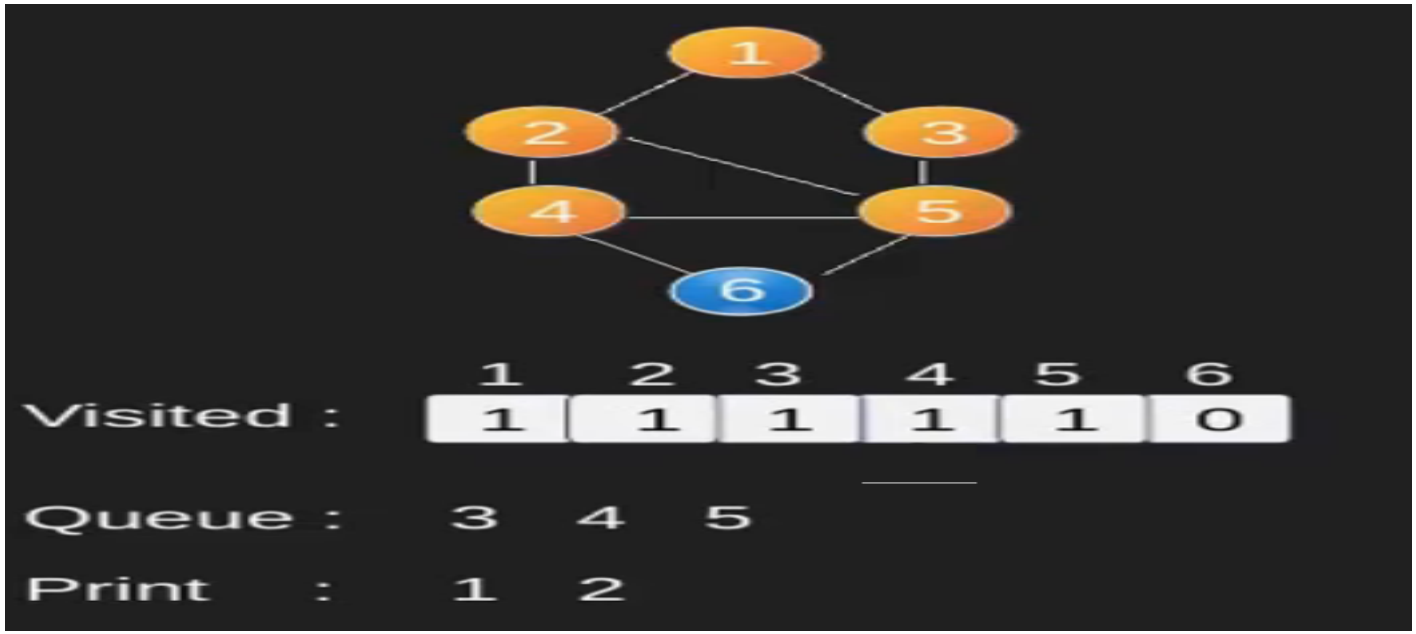


Figura: Grafo busca em largura [1]

Grafos: busca em largura

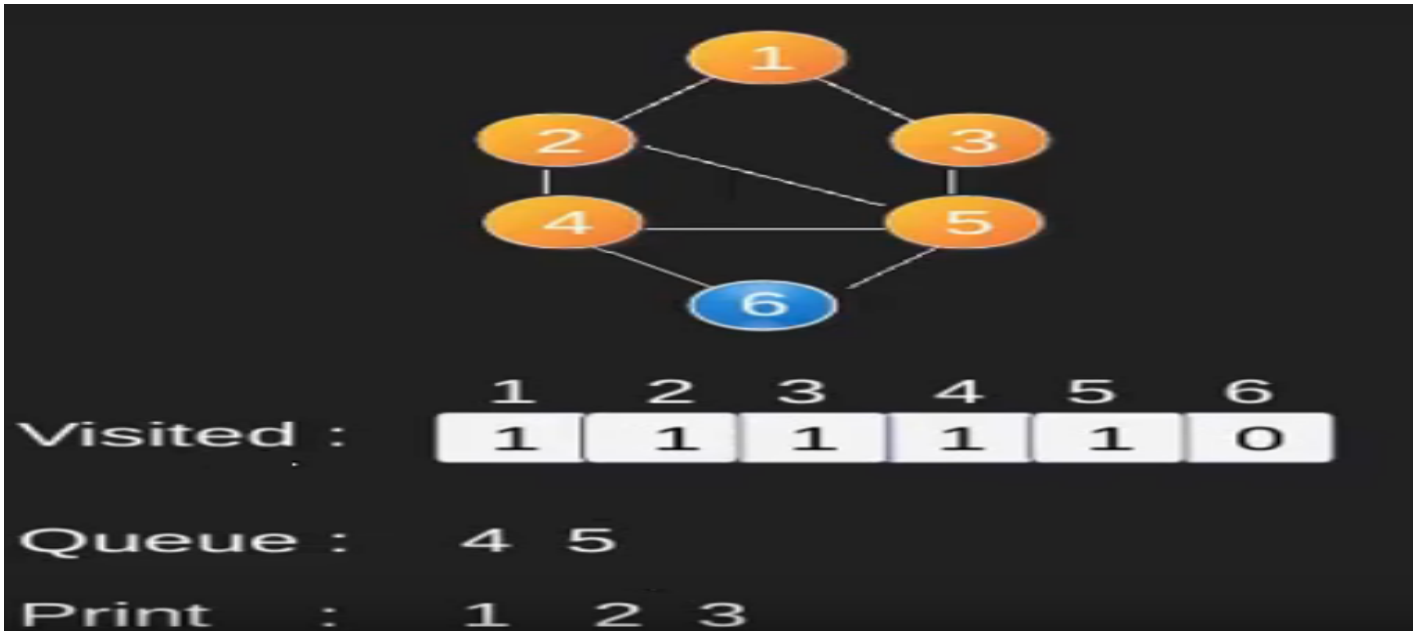


Figura: Grafo busca em largura [1]

Grafos: busca em largura

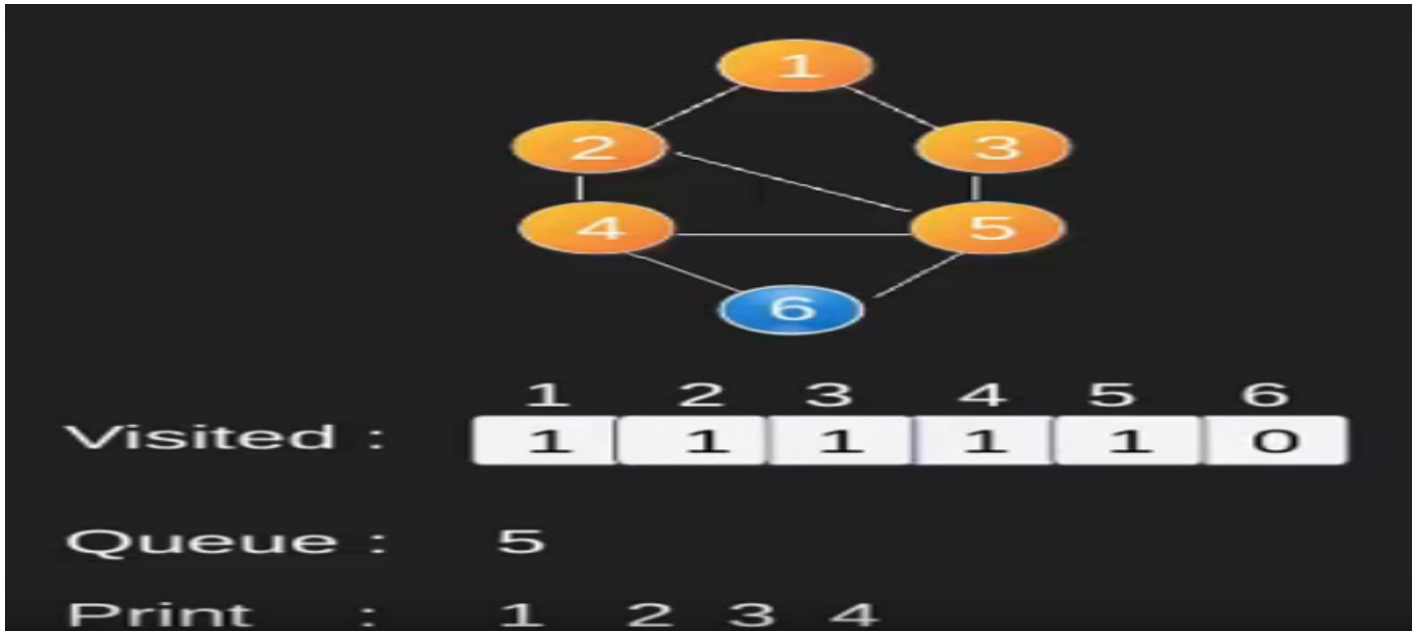


Figura: Grafo busca em largura [1]

Grafos: busca em largura

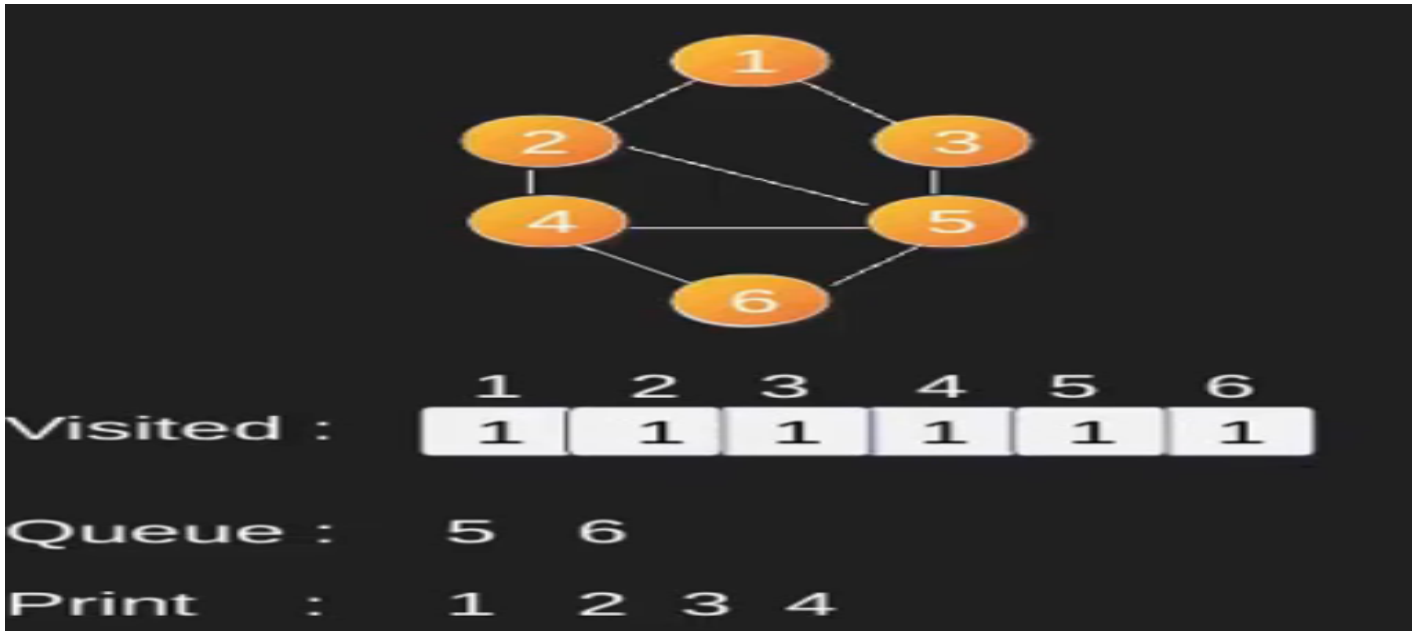


Figura: Grafo busca em largura [1]

Grafos: busca em largura

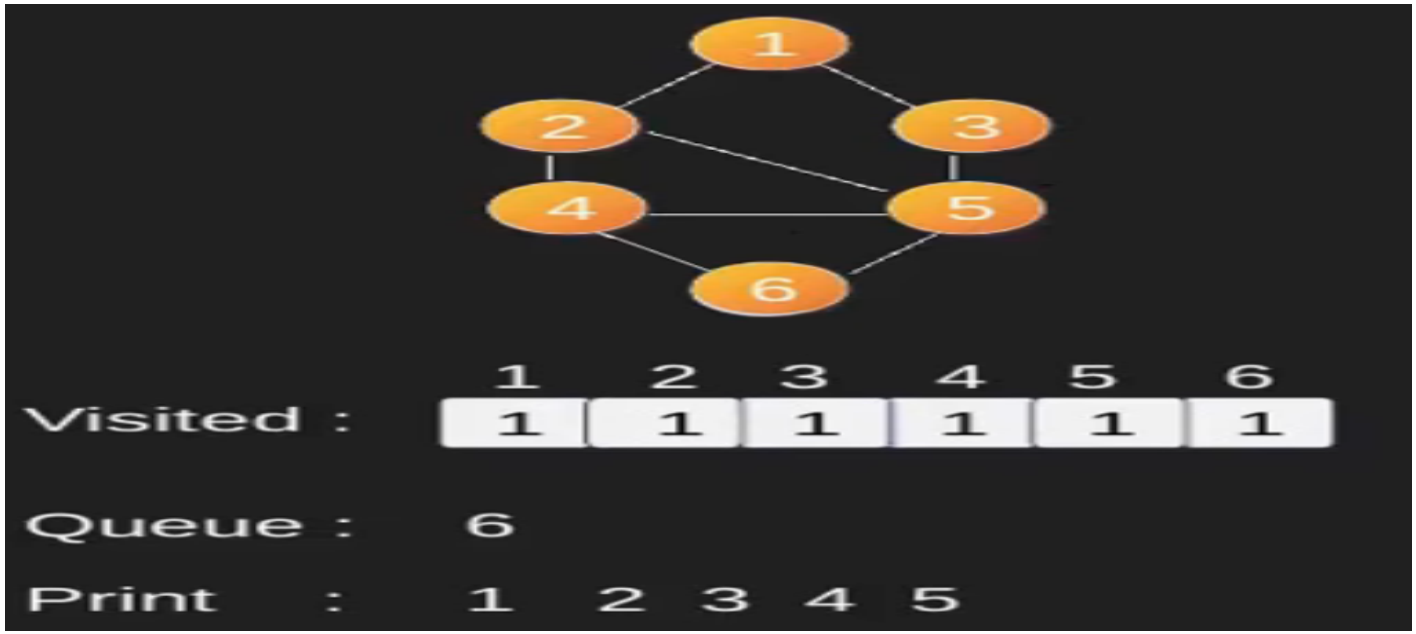


Figura: Grafo busca em largura [1]

Grafos: busca em largura

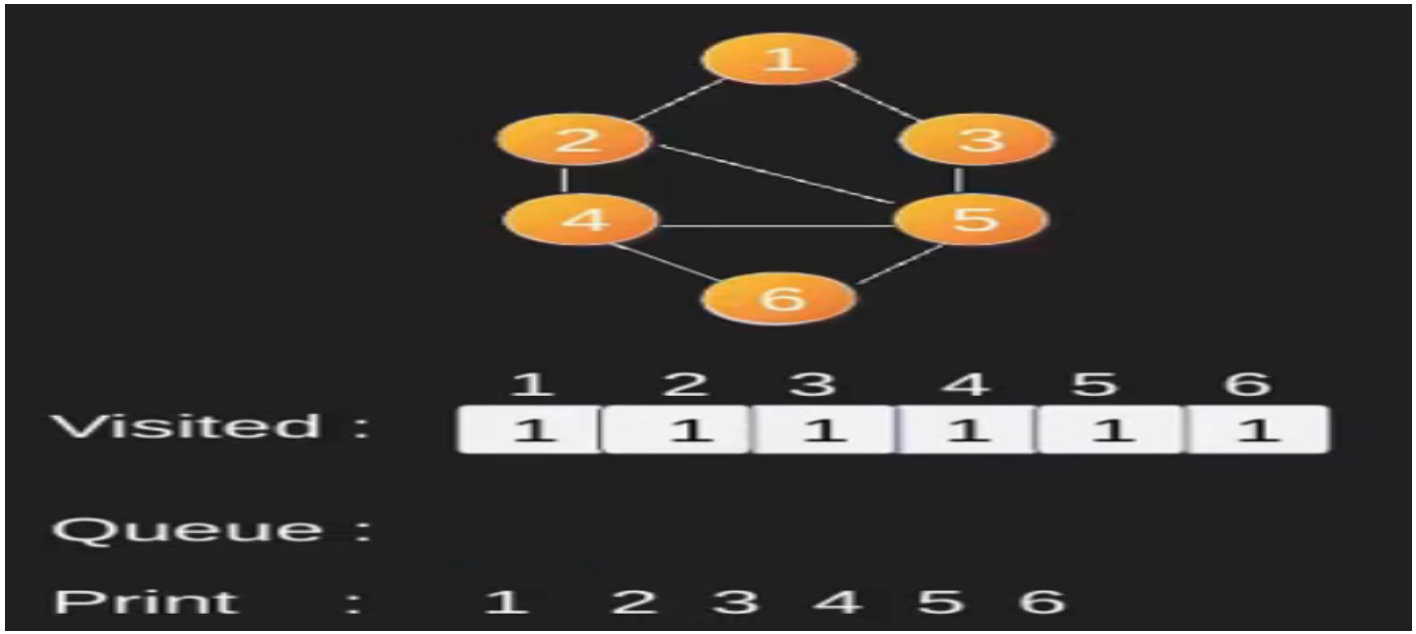


Figura: Grafo busca em largura [1]

Grafos

- Programar com os alunos em C#
- Mostrar complexidade no quadro
- Passar exercício para os alunos

Grafos: busca em profundidade

- Dado um nó inicial, percorre o grafo todo de forma sistemática usando a seguinte estratégia:
- Percorra um próximo vizinho do nó inicial, marque-o como visitado. Em seguida, percorra um vizinho daquele vizinho.
- A estratégia desse algoritmo é como a estratégia para percorrer uma árvore, o desafio adicional é considerar os ciclos possíveis dentro do grafo. Já tratados com a marcação de visitados.

Grafos: busca em profundidade

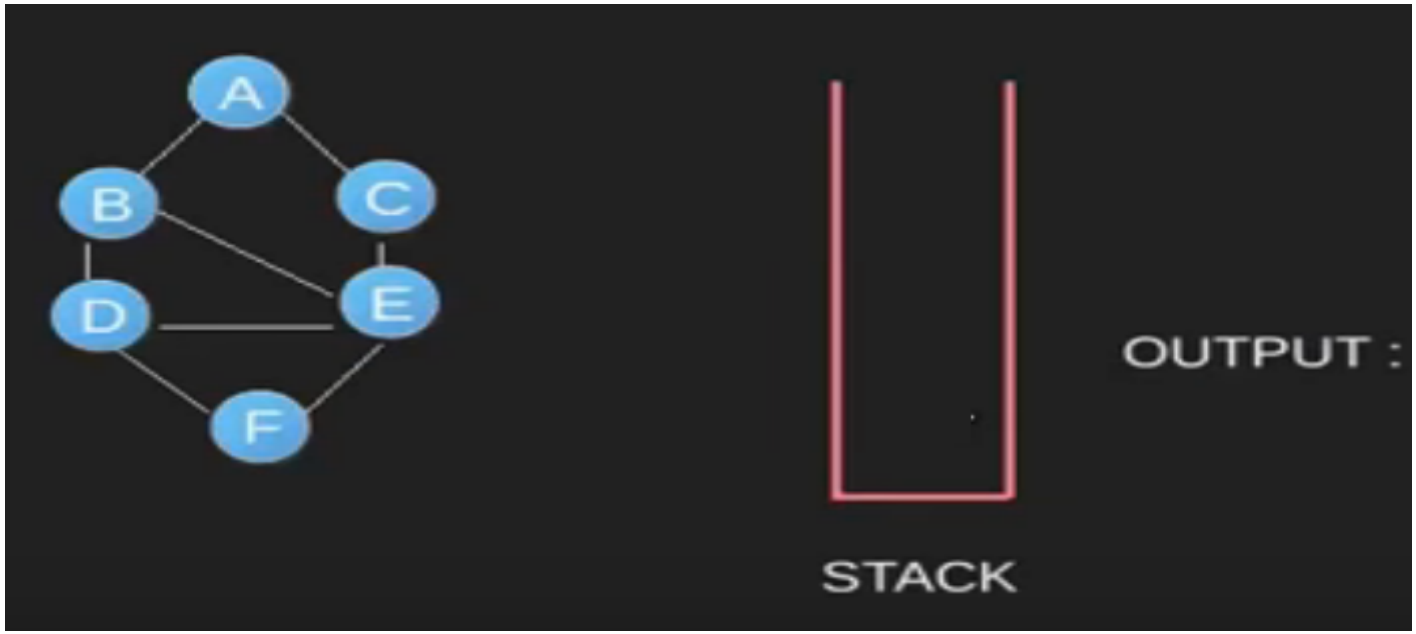


Figura: Grafo busca em profundidade [1]

Grafos: busca em profundidade

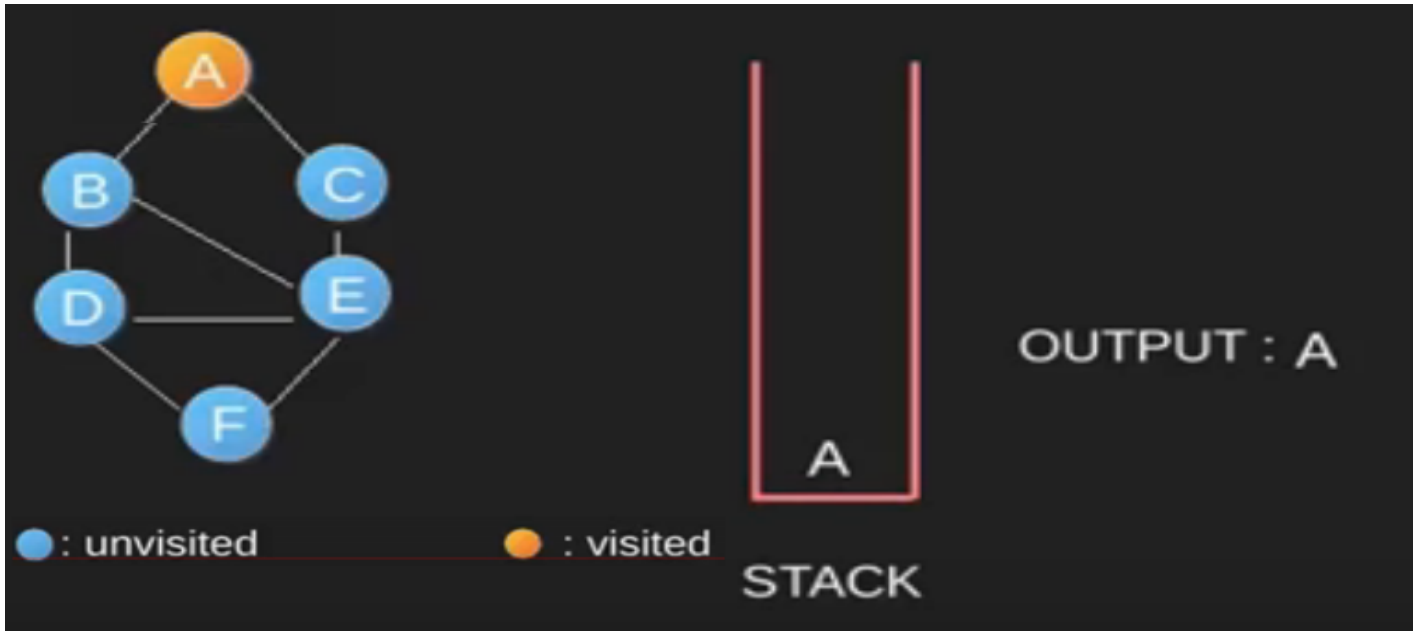


Figura: Grafo busca em profundidade [1]

Grafos: busca em profundidade

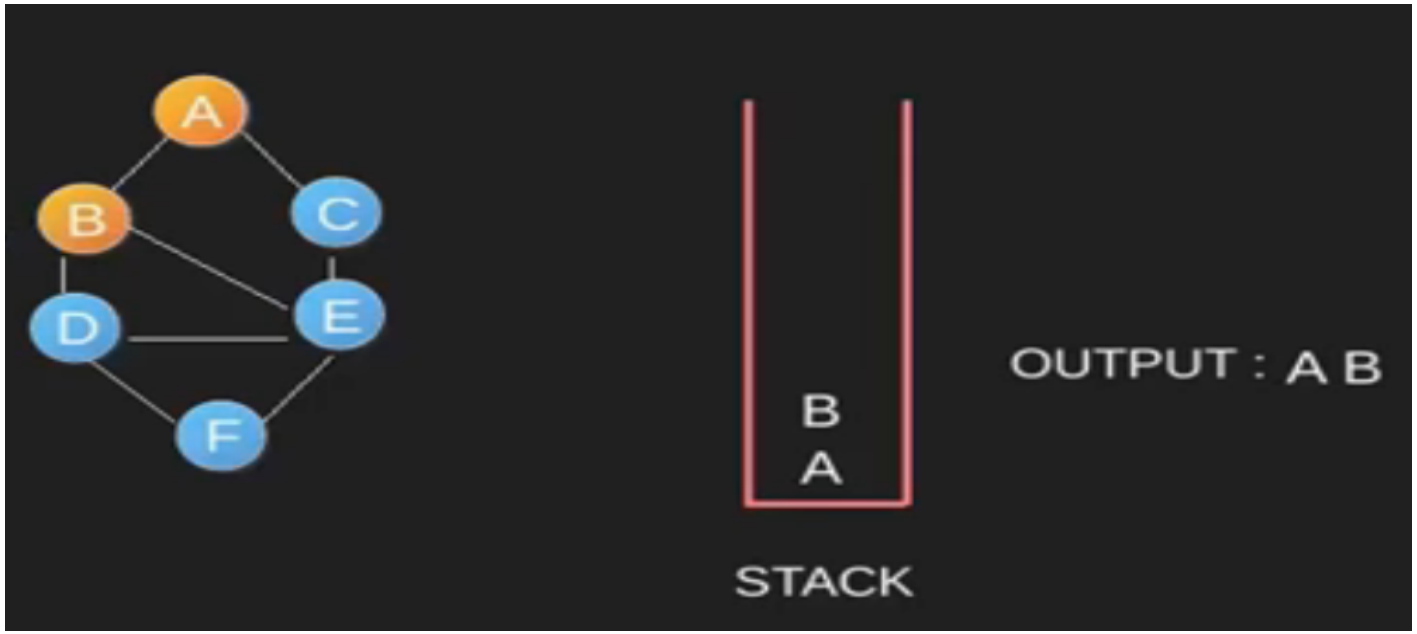


Figura: Grafo busca em profundidade [1]

Grafos: busca em profundidade

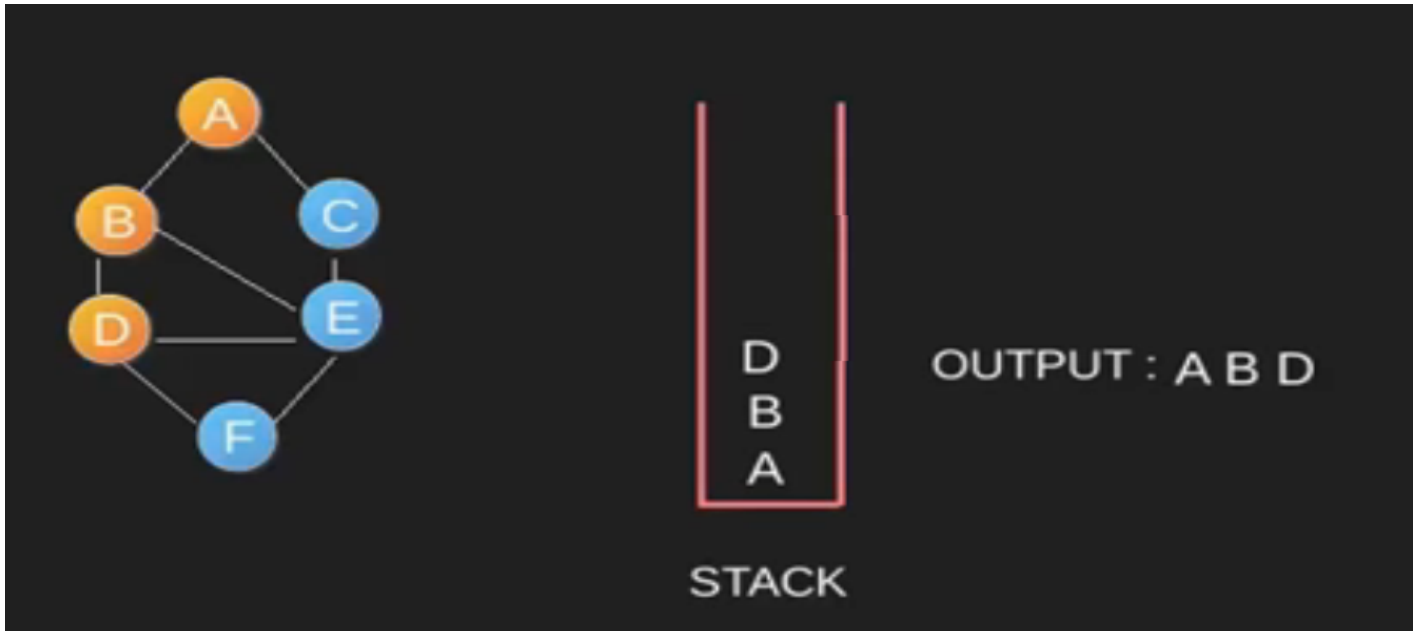


Figura: Grafo busca em profundidade [1]

Grafos: busca em profundidade

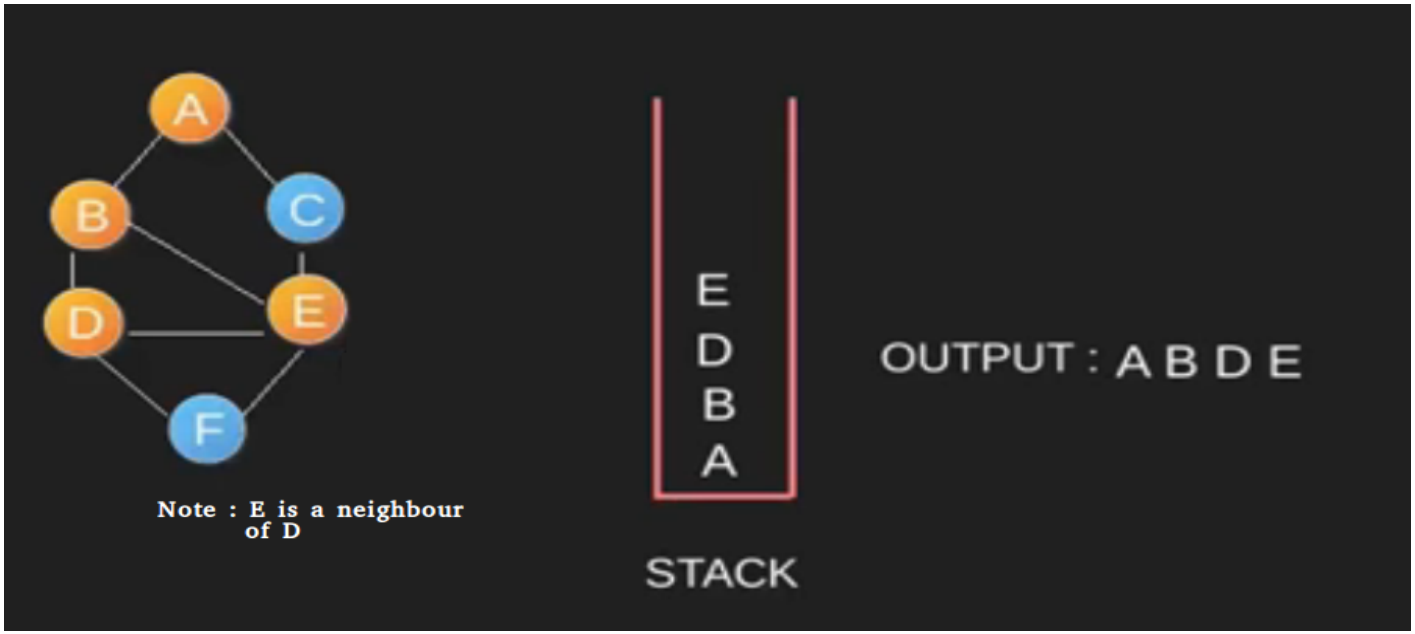


Figura: Grafo busca em profundidade [1]

Grafos: busca em profundidade

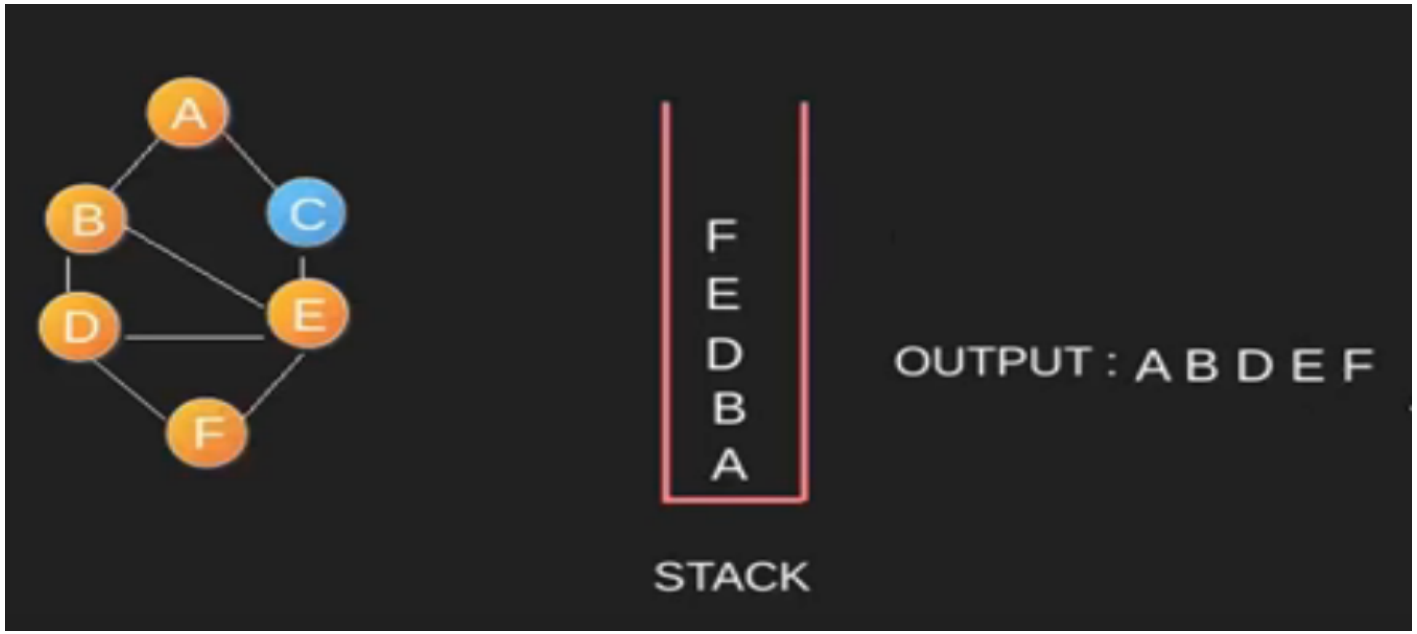


Figura: Grafo busca em profundidade [1]

Grafos: busca em profundidade

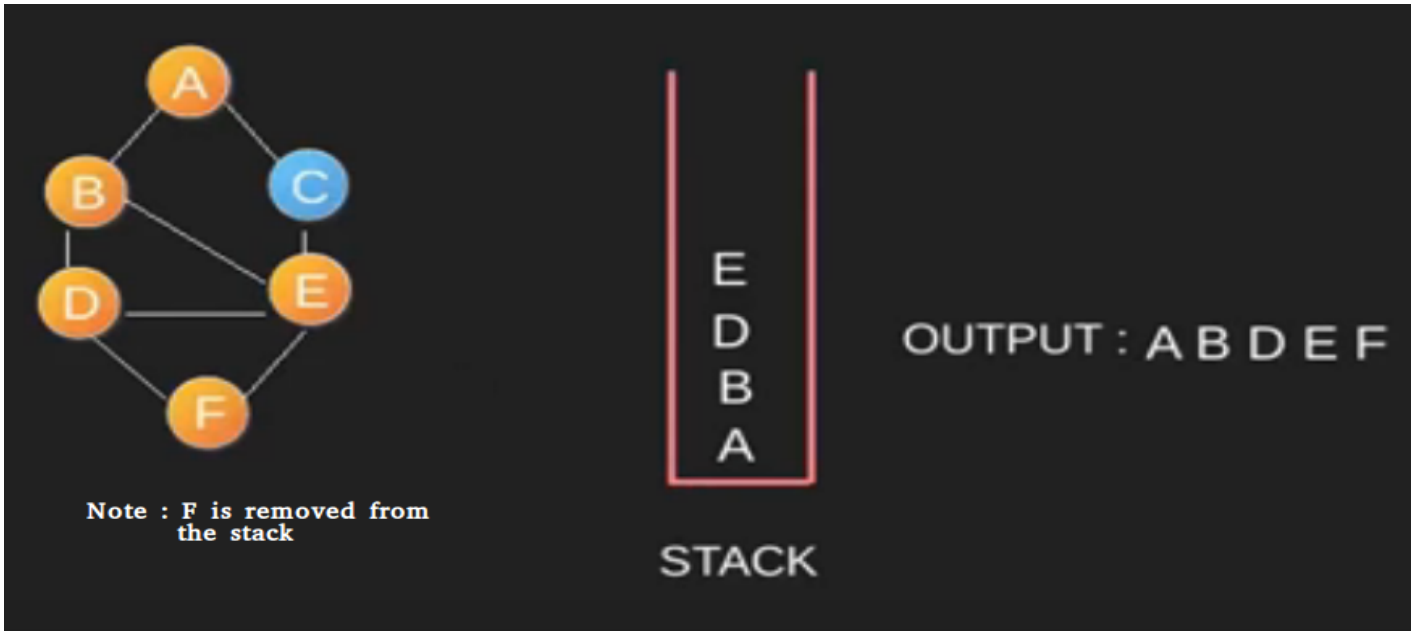


Figura: Grafo busca em profundidade [1]

Grafos: busca em profundidade

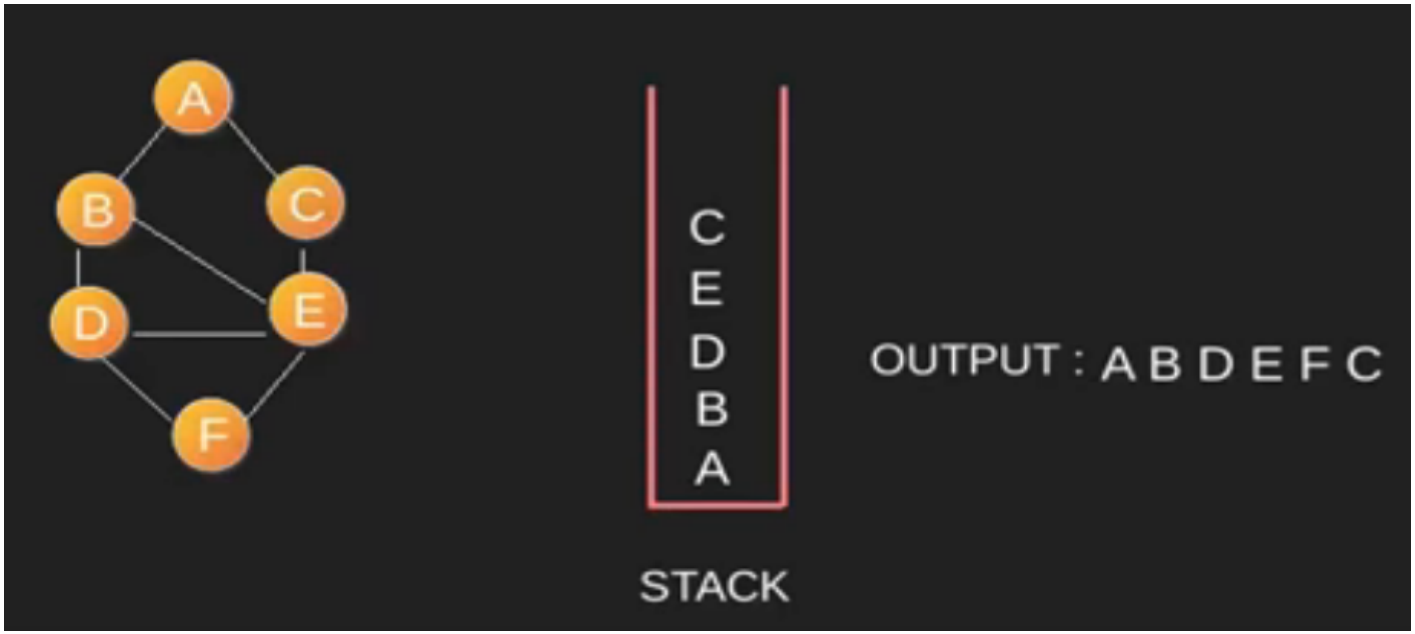


Figura: Grafo busca em profundidade [1]

Grafos: busca em profundidade

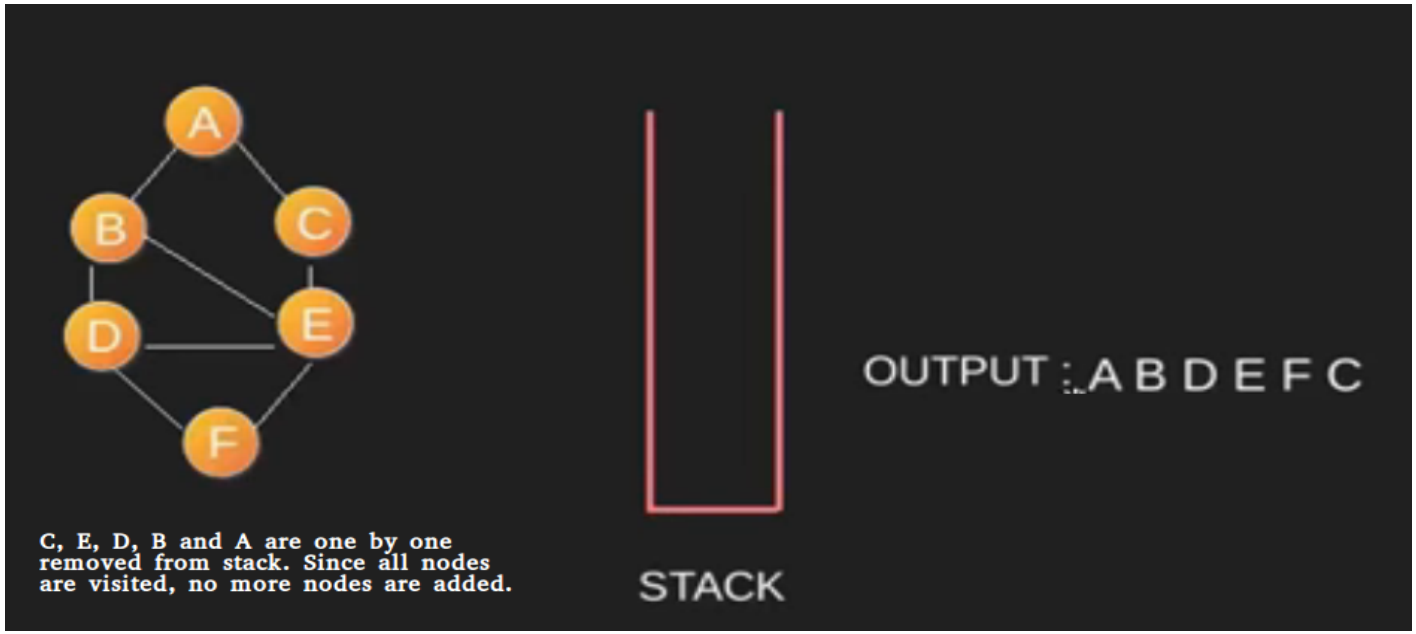


Figura: Grafo busca em profundidade [1]

Grafos

- Programar com os alunos em C#
- Mostrar complexidade no quadro
- Passar exercício para os alunos

Grafos: mais definições

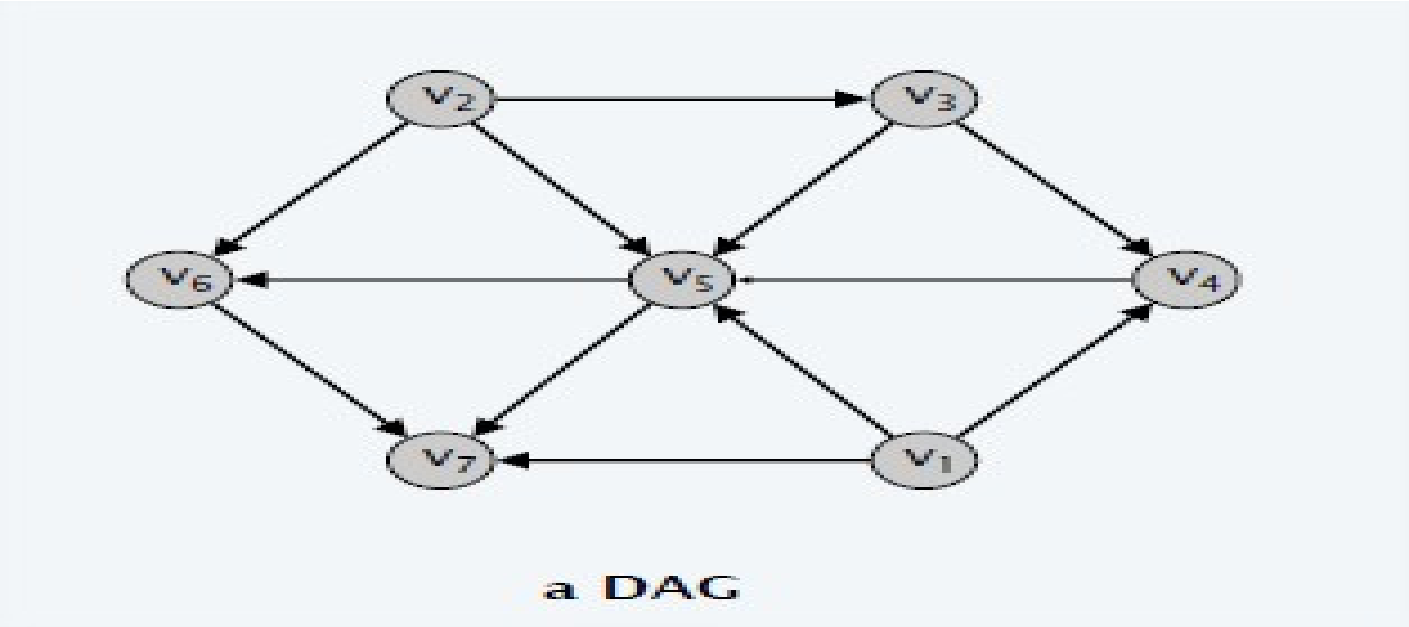


Figura: Grafos mais definições [1]

Grafos: mais definições

O grafo abaixo apresenta ciclos? Se sim, quais?

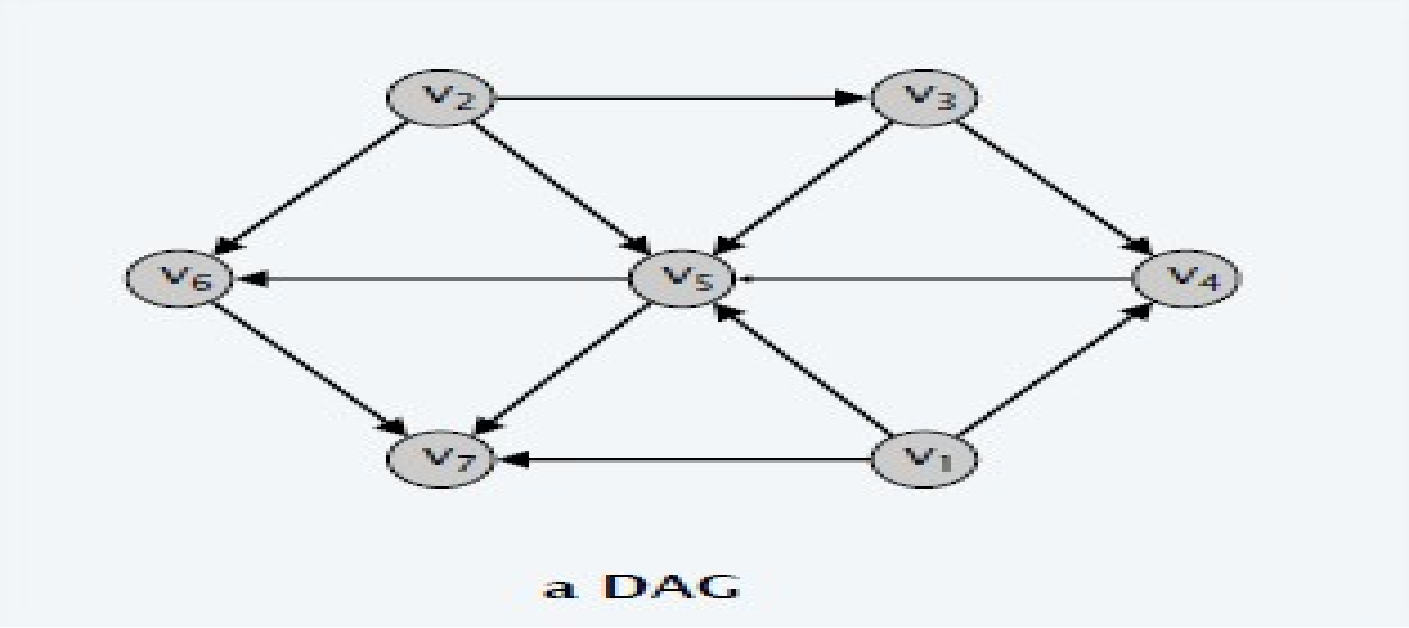


Figura: Grafos mais definições [1]

Grafos: mais definições

Algoritmos de grafo não dirigido se aplicam aqui?

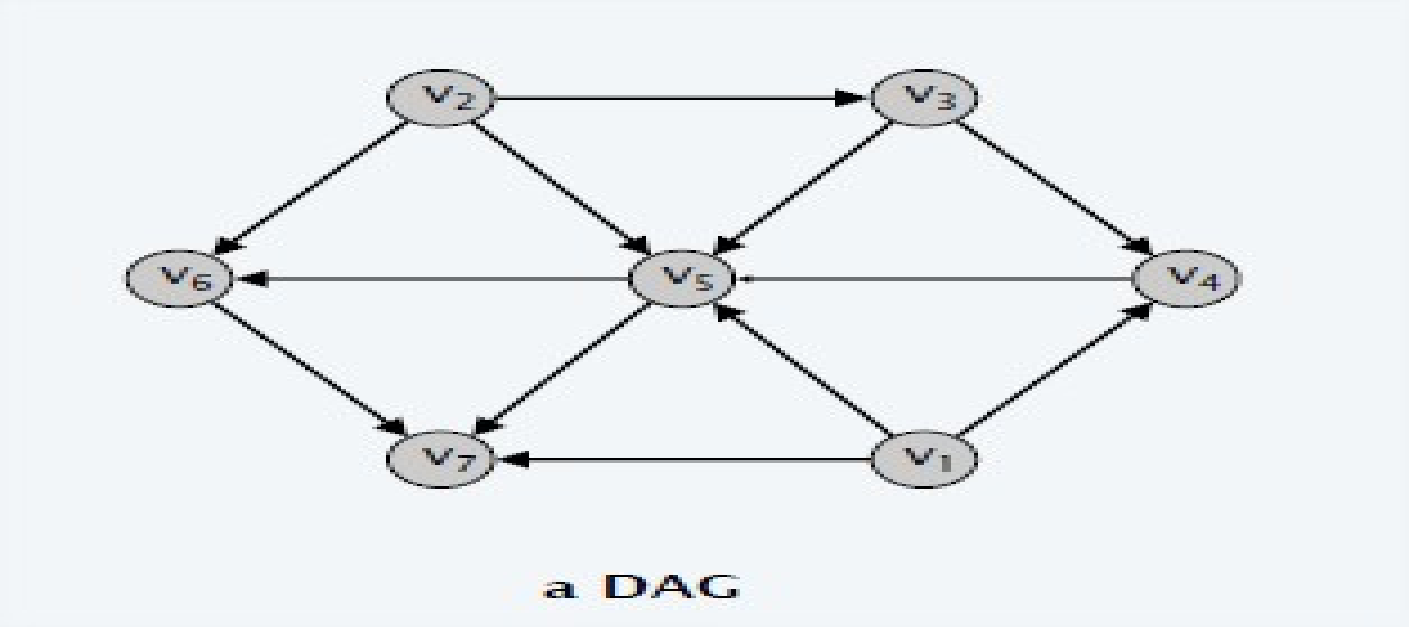


Figura: Grafos mais definições [1]

Grafos

- Dado um grafo e um nó “fonte”, encontre o caminho mínimo para todos os outros nós desse grafo.
- Existe o algoritmo de Dijkstra para resolver esse problema com os seguintes passos:
- Cria uma árvore de caminho mínimo, atribuí um valor de distância infinito para cada nó menos o “fonte” que terá distância zero
- Enquanto a árvore de caminho mínimo não contiver todos os nós:
- Pegue um vértice fora da árvore
- Adicione na árvore
- Atualize todos os nós vizinhos desse novo vértice.

Grafos: Caminhos mínimos

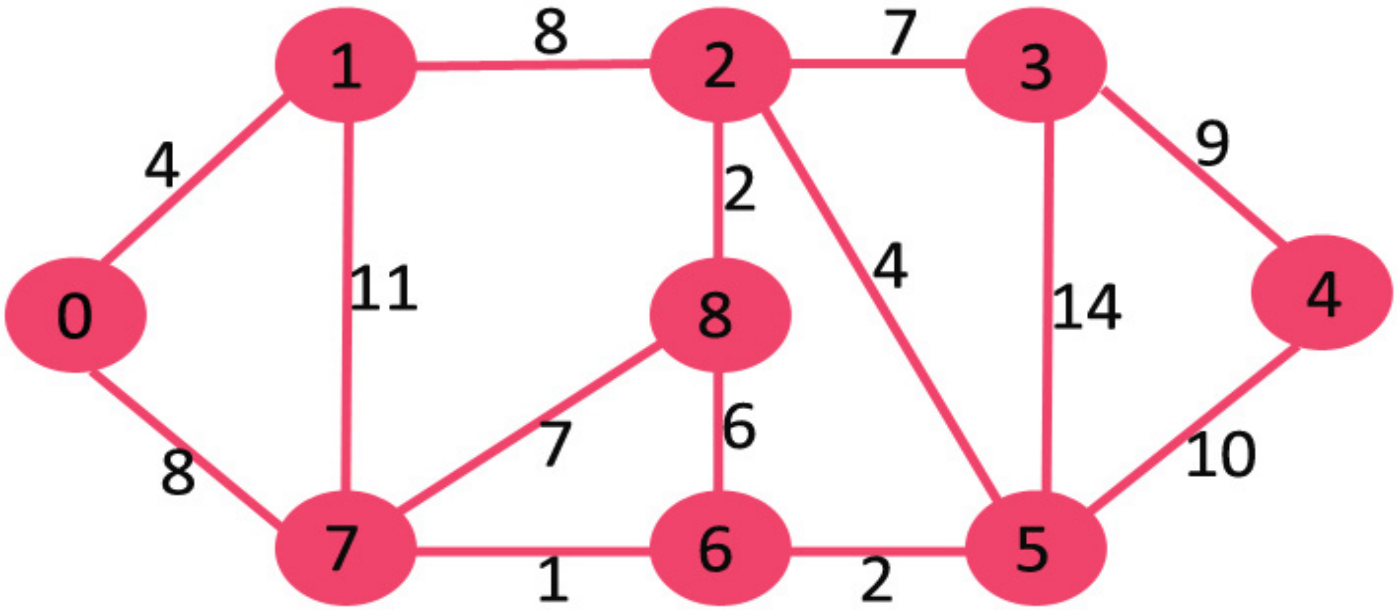


Figura: Grafos caminhos mínimos [1]

Grafos: Dijsktra

Inicializa o vetor de distâncias para cada vértice com infinito menos o vértice escolhido como *fonte*. Em seguida, actualize a distância dos vertices adjacentes

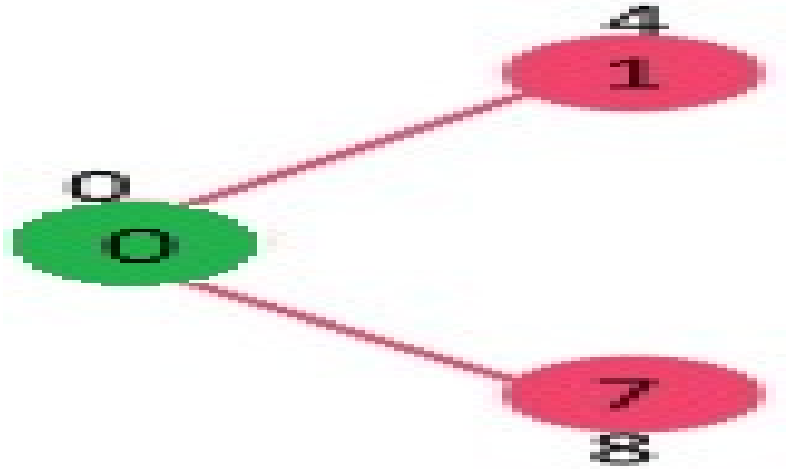


Figura: Grafos Dijsktra [1]

Grafos: Dijsktra

Do conjunto de vertices não incluídos pegue aquele com menor distância e inclua ele na árvore de caminho mínimo. Atualize o peso dos nós adjacentes.

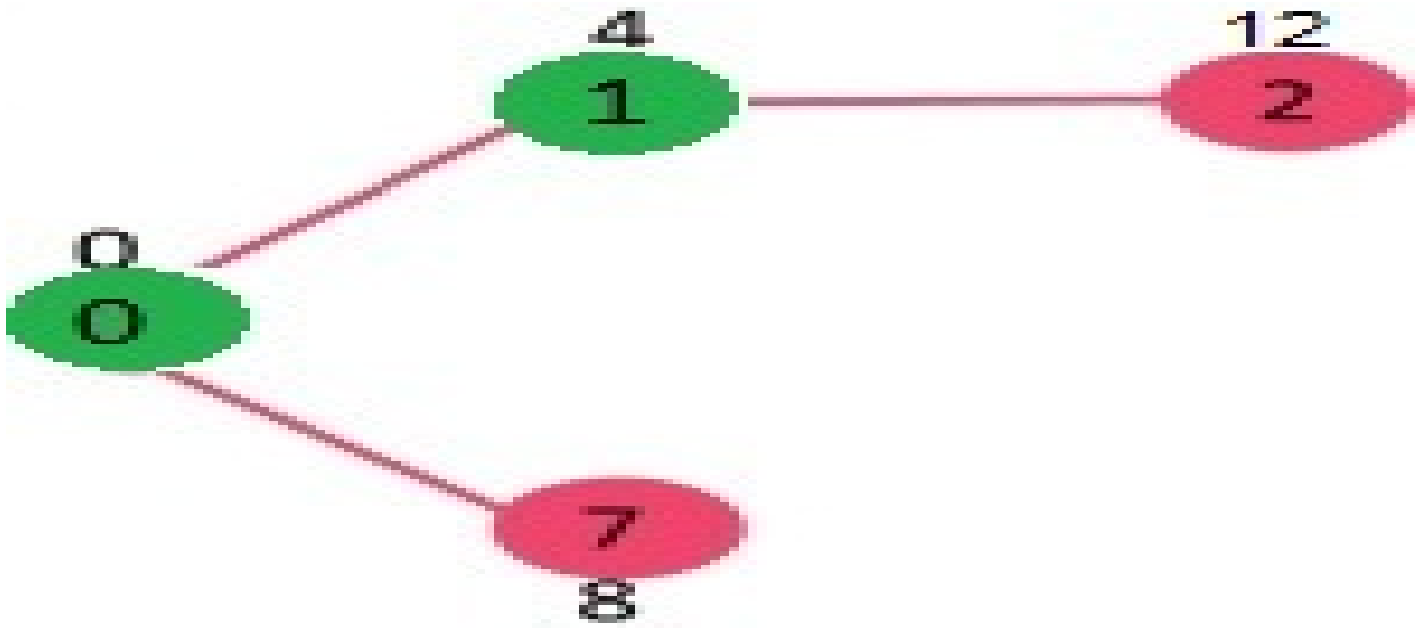


Figura: Grafos Dijsktra [1]

Grafos: Dijsktra

Do conjunto de vertices não incluídos pegue aquele com menor distância e inclua ele na árvore de caminho mínimo. Atualize o peso dos nós adjacentes.

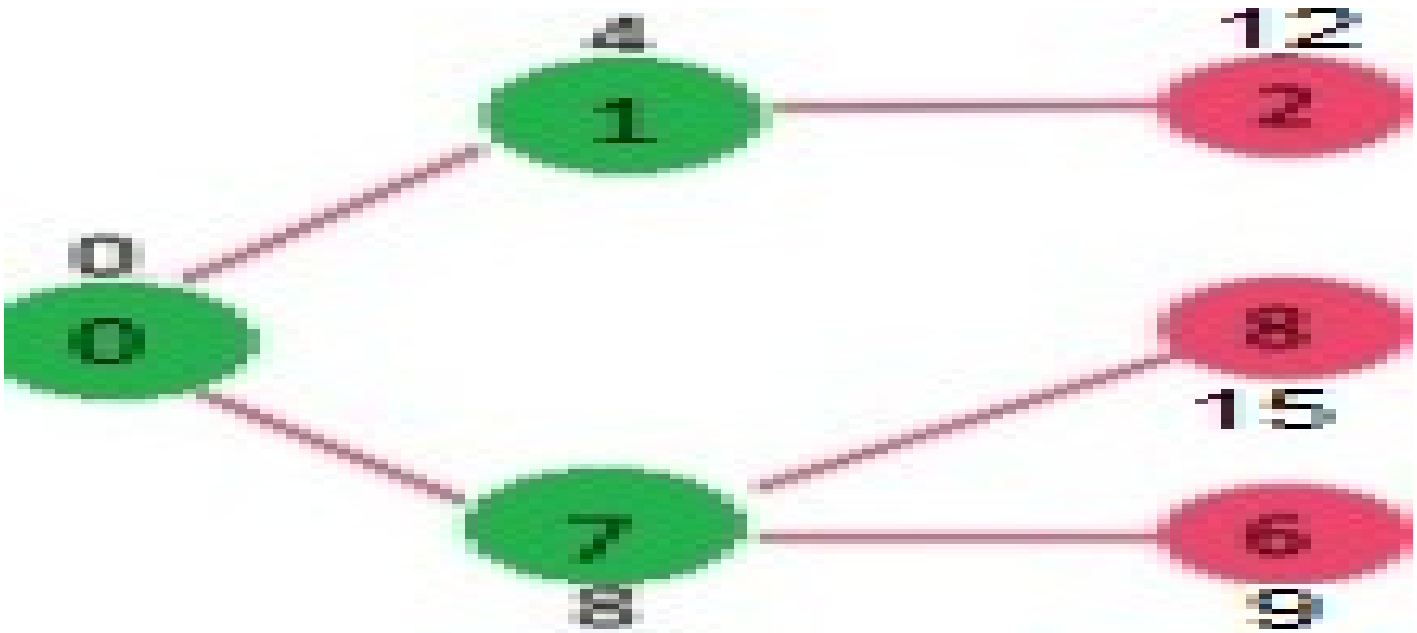


Figura: Grafos Dijsktra [1]

Grafos: Dijsktra

Do conjunto de vertices não incluídos pegue aquele com menor distância e inclua ele na árvore de caminho mínimo. Atualize o peso dos nós adjacentes.

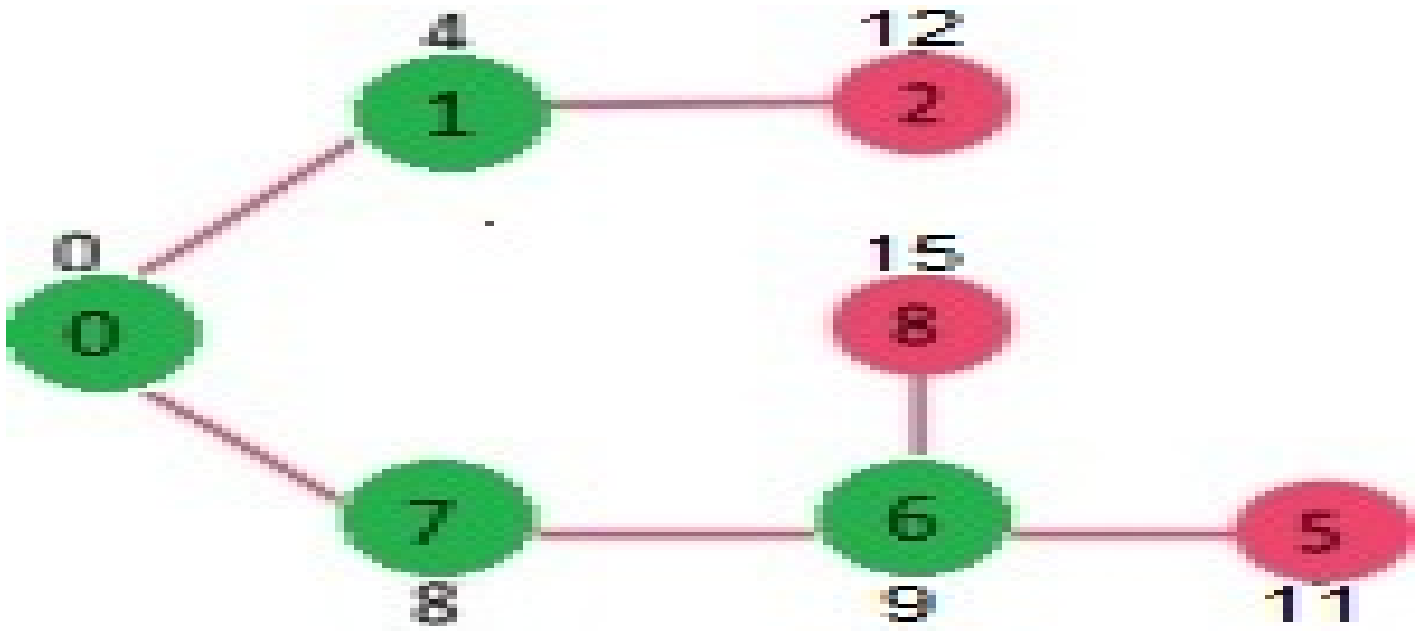


Figura: Grafos Dijsktra [1]

Grafos: Dijsktra

Repita esse processo até que todos os nós sejam incluídos

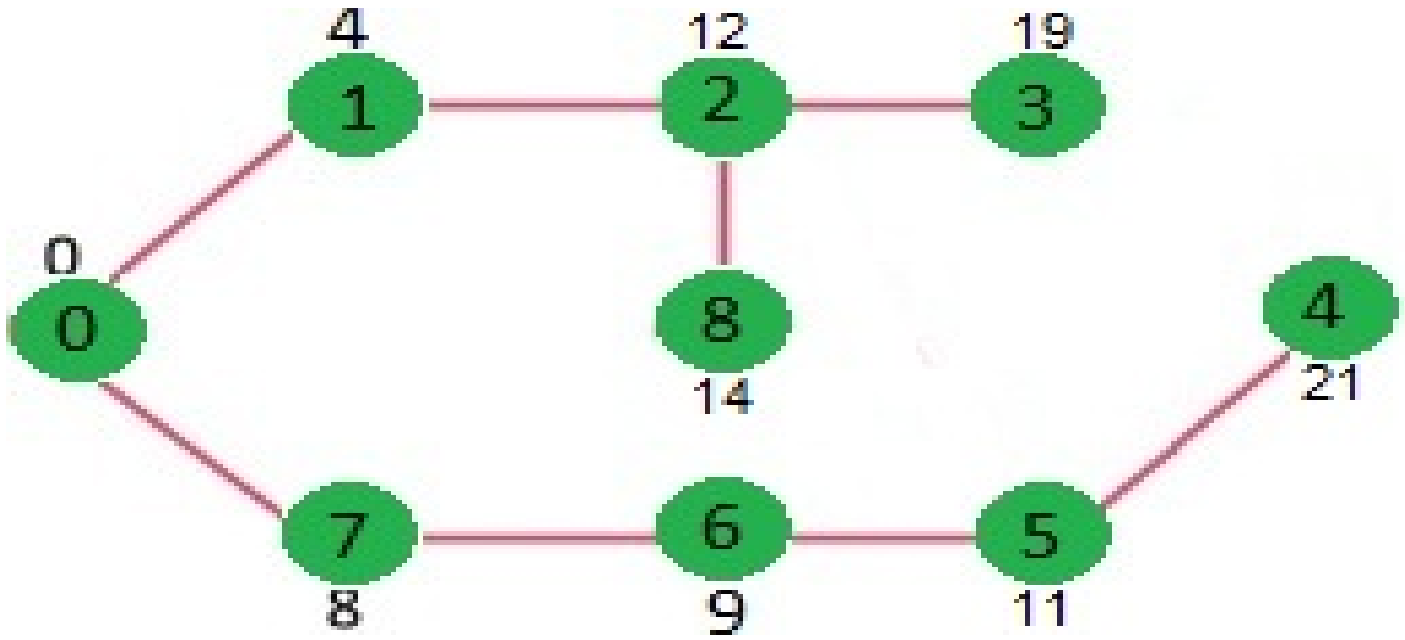


Figura: Grafos Dijsktra [1]

Grafos: Dijsktra x Prim

- Algumas pessoas confundem os algoritmos de Dijkstra e Prim (não dirigidos). Vale ressaltar que a árvore geradora minima (PRIM) garante que todos os nós sejam conectados com o custo mínimo para conectar todos os nós.
- A árvore de caminho mínimo garante o caminho mínimo de todos os nós para um determinado nó denominado: *fonte*.
- O algoritmo de Dijkstra tem problemas em arestas com peso negative, o algoritmo de Prim lida com eles. O algoritmo de Prim lida com apenas grafos não dirigidos.

Grafos

- Programar com os alunos em C#
- Mostrar complexidade no quadro
- Passar exercício para os alunos

Dúvidas...

Alguma dúvida?

Contato

- E-mail: 0800*dirso@gmail.com* (alunos SENAC)
- E-mail: *adilson.khouri.usp@gmail.com*
- Phone: $+55119444-26191$
- Linkedin
- Lattes
- GitHub

Referências I

[1] Geeks. (2018) A computer science portal for geeks. [Online]. Available: https://www.geeksforgeeks.org

[2] A. V. Aho, J. E. Hopcroft, and J. Ullman, *Data Structures and Algorithms*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1983.

[3] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change (2Nd Edition)*. Addison-Wesley Professional, 2004.

[4] Beck, *Test Driven Development: By Example*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[5] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley, 1999.

[6] Fernanda. (2014) Sql join: Entenda como funciona o retorno dos dados. [Online]. Available: https://www.devmedia.com.br/sql-join-entenda-como-funciona-o-retorno-dos-dados/31006

[7] S. team. (2018) Sqlite sample database. [Online]. Available: http://www.sqlitetutorial.net/sqlite-sample-database/