Ela é iniciada em um vértice arbitrário, o marcando como visitado. Em cada iteração, o algoritmo segue para um vértice não visitado que é adjacente ao que ele está no momento

Esse processo continua até, um vértice sem vértices adjacentes não visitados ser encontrado. O algoritmo volta para o vértice inicial e tenta continuar visitando vértices não visitados.

Eventualmente ao voltar para o vértice inicial, ele não terá saída

Depth-first Search

Ela é realizada de maneira concêntrica, visitando primeiro, todos os vértices adjacentes ao vértice inicial, então, todos os vértices não visitados com duas arestas de distância, e assim por diante, até que todos os vértices nos mesmos componentes conectados ao vértice inicial forem visitados

Se ainda houverem vértices não visitados, o algoritmo deve ser reiniciado em um vértice arbitrário de outro componente conectado do grafo

Breadth-first Search

Transversais

Grafos

Um grafo é informalmente pensado como uma coleção de pontos no plano chamados de vértices. Grafos para algoritmos computacionais são geralmente representados de duas formas: matriz de adjacência e listas de adjacência.

A matriz de adjacência de um grafo com n vértices é uma matriz booleana n × n com uma linha e uma coluna para cada vértice do grafo
A lista de adjacência de um grafo ou digrafo é uma coleção de listas ligadas, uma para cada vértice, que contém todos os vértices adjacentes ao vétice da lista.

Topological Sorting

Para o topological sorting ser possível, o digrafo em questão deve ser um DAG(Grafo Acíclico Dirigido). Ser um DAG não é somente necessário, mas é suficiente para o topological sorting ser possível;

Se um digrafo não possui ciclos direcionados, o problema de topological sorting para ele tem uma solução.

Existem dois algoritmos eficientes que verificam se um digrafo é um DAG e, caso seja, produzem uma ordenação dos vértices que resolve o problema de topological sorting.

O primeiro algoritmo é uma aplicação simples da Depth-first Search: fazer uma travessia DFS e observa a ordem em que os vértices se tornam pontos sem saída.

O segundo algoritmo é baseado em uma implementação direta da técnica decrease-(by one)-and-conquer: repetidamente, identifica em um digrafo restante uma fonte, que é um vértice sem arestas de entrada, e o exclui juntamente com todas as arestas que saem dele.