Notas de Aula - AED2 – Grafos: busca em largura

Grafos são estruturas mais complexas em comparação com listas, vetores e árvores binárias.

É necessário o desenvolvimento de métodos para explorar/percorrer um grafo.

O que é busca em grafos? Processo de seguir sistematicamente pelas arestas a fim de visitar os vértices do grafo.

Grande parte dos algoritmos famosos de processamento de grafos são baseados em métodos de busca:
- busca em largura: caminhos mínimos
- busca em profundidade: teste para verificar se um grafo é acíclico, ordenação topológica, resolução de quebra-cabeças, como labirinto

Informações relevantes sobre a estrutura do grafo podem ser extraídas: podem ser úteis para projetar algoritmos eficientes para determinados problemas.

Notação 1: Para um grafo G (orientado ou não) denotamos por V (ou V [G]) seu conjunto de vértices e por E (ou E[G]) seu conjunto de arestas.

Notação 2: A complexidade de algoritmos para grafos é dada em termos de V e/ou E.

Hoje veremos a busca em largura.

Busca em Largura

“Na teoria dos grafos, busca em largura (ou busca em amplitude, também conhecido em inglês por Breadth-First Search - BFS) é um algoritmo de busca em grafos utilizado para realizar uma busca ou travessia num grafo e estrutura de dados do tipo árvore. Intuitivamente, você começa pelo vértice raiz e explora todos os vértices vizinhos.” - Wikipédia

Busca em largura é um método que expande e examina sistematicamente todos os vértices de um grafo direcionado ou não-direcionado.

Um vértice ***v*** é alcançável a partir de um vértice ***s*** em um grafo G se existe um caminho de ***s*** a ***v*** em G.

Definição: a distância de s a v é o comprimento do caminho mais curto de s a v.

Se v não é alcançável a partir de s, então dizemos que a distância entre ambos vértices é infinita. No início da aplicação do algoritmo de busca em largura, o vértice s começa com o valor de distância igual a zero e os demais com infinito.

Um algoritmo de busca em largura recebe um grafo G = (V, E) e um vértice especificado ***s*** chamado fonte (source).

Percorre todos os vértices alcançáveis a partir de ***s*** em ordem de distância.

O algoritmo constrói uma ***árvore de busca em largura*** com raiz ***s***.
- Cada caminho entre ***s*** e ***v*** nessa árvore corresponde a um caminho mais curto entre ambos vértices.
- O algoritmo descobre todos os vértices a uma distância ***k*** do vértice origem antes de descobrir qualquer vértice a uma distância **k + 1**.

Inicialmente a árvore de busca em largura contém apenas o vértice fonte ***s***.

Para cada vizinho de ***v*** de ***s***, o vértice ***v*** e a aresta (***s, v***) são acrescentadas à árvore.

O processo é repetido para os vizinhos dos vizinhos de ***s***. Isso é feito até que todos os vértices alcançáveis por ***s*** sejam adicionados na árvore. O algoritmo de busca em largura também pode formar uma floresta.

O processo de busca é implementado através de uma fila Q.

Durante a aplicação do algoritmo de busca em largura, cada vértice pode ser colorido por meio das seguintes cores.
- Branca: não visitado (inicialmente, todos os vértices são brancos).
- Cinza: visitado pela primeira vez.
- Preta: todos os seus vizinhos foram visitados.

Vértices de cinza podem ter alguns vértices adjacentes brancos, e eles representam a fronteira entre vértices descobertos e não descobertos.

Ver slides de 11 ao 19

Implementação Busca em Largura

Cores
- Para cada vértice ***v***, a cor atual é guardada no vetor ***cor[v]***, que pode ser branco, cinza ou preto.
- Para o efeito de implementação, o uso da cor não é realmente necessário, mas facilita a compreensão do algoritmo (com o auxílio da fila, conforme apresentado entre os slides 10 e 18, a cor não é necessária).

A raiz da árvore de busca em largura é ***s***.

Cada vértice ***v*** (exceto a raiz) possui um pai ***π[v]***.

O caminho de s até v é dado por: ***v, π[v], π[π[v]], π[π[π[v]]], …, s***

A variável ***d[v]*** é usada para armazenar a **distância** de ***s*** a ***v***. Determinada durante o processo de busca.

O algoritmo de busca em largura recebe um grafo G (na forma de listas de adjacências), e um vértice "***s que pertence a V[G]***" e devolve:
1 - Para cada vértice ***v***, a distância de ***s*** a ***v*** em G
2 - Árvore ou floresta de busca em largura

Nos slides 22 e 23 é apresentada um pseudocódigo para a busca em largura:
- Entre as linhas 1 e 8, os vetores ***cor, d e pi*** são inicializados.
- as cores, exceto o referente ao vértice ***s*** (inciado com a cor cinza), são inicializadas com branco.
- as distâncias entre ***s*** e os demais vértices são inciadas com infinito. Também, a distância ***d[s]*** é iniciada, mas com 0.
- o pai de cada vértice, incluindo ***s***, é iniciado com nulo.
- Na linha 8 é iniciada uma fila vazia, na qual o vértice ***s*** é enfileirada (linha 9).
- Entre as linhas 10 e 18, o grafo é explorado a partir do vértice ***s***.
- A exploração do grafo é feita enquanto a fila não estiver vazia.
- Na linha 11, um vértice ***u*** é desenfileirado

- Entre as linhas 12 e 17, os vértices adjacentes a ***u*** são explorados. Caso o vértice ***Adj[u]*** explorado seja branco:
→ a cor do vértice é mudada para cinza (linha 14).
→ a distância ***d[v]*** é a soma da distância computada no vértice ***u*** (pai de ***v***) e um (linha 15).
→ na árvore de busca pi[v] é adicionado o vértice ***u*** (pai de ***v***) (linha 16).
→ Por fim, o vértice v é adicionado na fila para ter os seus adjacentes explorados futuramente.
- Após a execução do loop da linha 12, o vértice ***u*** já teve todos os seus adjacentes explorados, ou seja, a sua cor é mudada para preto (linha 18).

ver slides de 25 a 34.

No exemplo, parece estranho a ordem como os vértice são adicionados na fila, mas isso ocorre de acordo com a ordem em que o mesmo parece na lista de adjacência (no exemplo apresentado nesses slides, não é considerada a ordenação).

É importante ressaltar que há uma outra forma de implementar busca em largura, cuja forma de exploração é feita enquanto todos os vértices não serem coloridos na cor preta. Nessa abordagem (apresentada no livro de Ziviani), a busca não é iniciada de um vértice qualquer, mas sim do primeiro (o que tiver menor valor).

Análise de Complexidade da Busca em Largura

A inicialização consome tempo O(|V|).

Depois que um vértice deixa de ser branco, ele não volta a ser branco novamente.
- Cada vértice é adicionado na fila apenas uma vez.
- Cada operação sobre a fila consome tempo O(1), resultando em um total de O(|V|).

Em uma lista de adjacência, cada vértice é percorrido apenas uma vez
- A soma dos comprimentos das listas de adjacência é O(|E|)
- Logo, o tempo gasto para percorrer as listas é O(|E|)

Conclusão: a complexidade da busca em largura é, no pior caso, na ordem de O(|V| + |E|)

Caminho Mais Curto

No slide 39 é apresentado um algoritmo recursivo para imprimir o caminho mais curto entre dois vértices (***s*** e ***v***).

No algoritmo há dois casos bases:
1 – se ***s*** == ***v***, o vértice s é impresso
2 – se pai de ***v*** é nulo.

No caso iterativo, primeiro é feita uma chamada recursiva considerando o pai de ***v***. Após a chamada recursiva ser executada, o vértice v é impresso.

Exemplo no slide 39

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. Introduction to Algorithms. Third edition, The MIT Press, 2009.

Marin, L. O. Grafos: Busca em Largura. AE23CP - Algoritmos e Estrutura de Dados II. Slides. Engenharia de Computação. Dainf/UTFPR/Pato Branco, 2017.

Tenenbaum, A.; Langsam, Y. Estruturas de Dados usando C. Pearson, 1995.

Ziviani, N. Projeto de Algoritmos - com implementações em Java e C++. Thomson, 2007.