Notas de Aula - AED2 – Grafos: busca em profundidade
Prof. Jefferson T. Oliva

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: achar componentes conectados, achar todos os nós conectados a apenas um componente, testar bipartição em grafos, Dijkstra (usa fila de prioridades)
- busca em profundidade: teste para verificar se um grafo é acíclico, verificar se um grafo é conexo, verificar se existe um caminho entre dois vértices

Busca em Profundidade

Objetivo: visitar todos os vértices e numerá-los na ordem em que são descobertos e em que o processamento é finalizado.

Estratégia: buscar, sempre que possível, o mais profundo no grafo.

Aplicável tanto a grafos orientados quanto não-orientados.

Possui um número enorme de aplicações:
- Determinar os componentes de um grafo
- Ordenação topológica
- Determinar componentes fortemente conexos
- Sub-rotina para outros algoritmos

Um algoritmo de busca em profundidade recebe um grafo G = (V, E). Diferentemente da busca em largura, na em profundidade não é fornecido um vértice fonte e todos os vértices são explorados.

A busca em profundidade explora arestas a partir do vértice v mais recentemente descoberto
- Esse processo é repetido, recursivamente a partir do vértice v, até acabar os vértices alcançáveis
- Caso não é possível explorar mais vértices, a busca é continuada a partir de um vizinho próximo ao vértice explorado

A busca em profundidade é executada até todos os vértices serem visitados.

A busca em profundidade utiliza o princípio da técnica *backtracking*, uma vez que é feita uma busca exaustiva e após chegar mais “longe possível”, retorna para uma solução parcial (vértice) mais próxima e continua a busca.

Enquanto a busca em largura utiliza fila para a exploração do grafo, a busca em profundidade utiliza pilha, mesmo de forma indireta (chamadas recursivas).

Diferentemente da busca em largura (que gera um sub-grafo em forma árvore), a busca em profundidade pode gerar várias árvores a partir de um único grafo.

Em outras palavras, a busca em profundidade pode formar uma floresta de busca em profundidade.

A busca em profundidade possui duas principais semelhanças com a busca em largura.
- Descobre vértices a partir da varredura do grafo
- Há atribuição de cores aos vértices
→ Branca: "vértice ainda não visitado" (Inicialmente todos os vértices são brancos)
→ Cinza: "vértice visitado, mas ainda não finalizado"
→ Preta: "vértice visitado e finalizado"

Na busca em largura é registrada a distância entre o vértices e outros vértices.

Na busca em profundidade cada vértice, ao final do processamento, possui um "carimbo de tempo"
- Momento da descoberta: mudança da cor branca para cinza
- Momento da finalização do processamento do vértice: mudança da cor cinza para preta

Outra forma de entender busca em profundidade: imagine que os vértices são armazenados em uma pilha à medida que são visitados
- Suponha que a busca atingiu um vértice u
- Escolhe-se um vizinho não visitado v de u para prosseguir a busca
- Empilhe v e repete-se o passo anterior com v
- Se nenhum vértice não visitado foi encontrado, então desempilhe um vértice da pilha e volte ao primeiro passo

Ver slides de 11 a 20.

Implementação da Busca em Profundidade

Para cada vértice v, a cor atual é guardada no vetor cor[v], que pode ser branco, cinza ou preto
- d[u]: momento da descoberta de u
- Vetor f[u]: momento de término da exploração de u
- Vetor pi[u]: pai de u

O algoritmo de busca em profundidade recebe um grafo G (na forma de listas de adjacências) e devolve
- Para cada vértice v, os instantes de descoberta e de finalização
- Floresta de busca em profundidade

Ver slides 24 e 25.

Análise de Complexidade da Busca em Profundidade

Complexidade
- Procedimento DFS
→ Inicialização dos vetores cor e pi: O(|V|)
→ Exploração dos vértices: O(|V|)
- Procedimento DFS_visit
→ Exploração dos vértices adjacentes: O(|E|)
- Custo total: O(|V| + |A|)

Classificação de Arestas

Arestas de árvore: são arestas de uma árvore de busca em profundidade. A aresta (u, v) é uma aresta de árvore se v foi descoberto pela primeira vez ao percorrer a aresta (u, v).

Arestas de retorno: conectam um vértice u com um antecessor v em uma árvore de busca em profundidade (inclui self-loops). (de descendente para ancestral)

Arestas de avanço: não pertencem à árvore de busca em profundidade mas conectam um vértice a um descendente que pertence à árvore de busca em profundidade. (de ancestral para descendente)

Arestas de cruzamento: são as demais arestas. Podem conectar vértices na mesma árvore de busca em profundidade, ou em duas árvores diferentes. Entre uma árvore ou sub-árvores.

Na busca em profundidade cada aresta pode ser classificada pela cor do vértice que é alcançado pela primeira vez
- Branco indica uma aresta de árvore
- Cinza indica uma aresta de retorno
- Preto indica uma aresta de avanço quando u é descoberto antes de v: d[u] < d[v]
- Preto indica uma aresta de cruzamento quando v é descoberto depois de u: d[u] > d[v]

Ver slide 32

Teste para verificar se um grafo é acíclico
- Basta verificar se não há arestas de retorno
- Caso uma aresta de retorno seja encontrada, então o grafo tem ciclo
- Caso contrário, o grafo é acíclico

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 Profundidade. 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.