

1

Algoritmos em Grafos: Pesquisa e Ordenação

R. Rossetti, L. Ferreira, H. L. Cardoso, F. Andrade
CAL, MIEIC, FEUP

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

2

Índice

- ◆ Pesquisa em profundidade
- ◆ Pesquisa em largura
- ◆ Ordenação topológica

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

3

Pesquisa em profundidade (*depth-first search*)

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

4

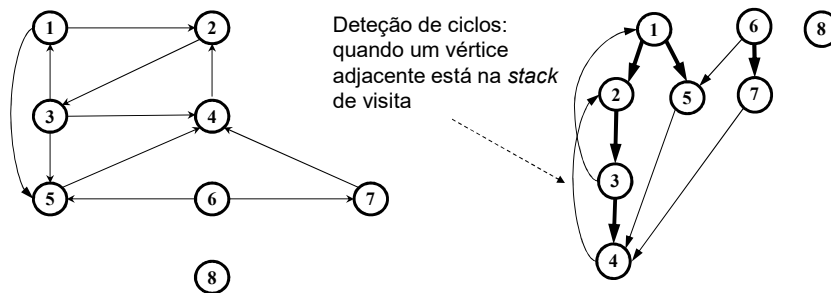
Pesquisa em profundidade

- ◆ Arestas são exploradas a partir do vértice v mais recentemente descoberto que ainda tenha arestas a sair dele.
- ◆ Quando todas as arestas de v forem exploradas, retorna para explorar arestas que saíram do vértice a partir do qual v foi descoberto.
- ◆ Se se mantiverem vértices por descobrir, um deles é seleccionado como a nova fonte e o processo de pesquisa continua a partir daí.
- ◆ Todo o processo é repetido até todos os vértices serem descobertos.

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

Exemplo

Vértices numerados por (pré)ordem de visita e dispostos por profundidade de recursão:



Arestas a traço forte (que acederam a vértices ainda não visitados): floresta DFS constituída por uma ou mais árvores DFS (de expansão em profundidade).

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

Pseudo-código

```
G = (V, E)
Adj(v) = {w | (v, w) ∈ E} (∀ v ∈ V)
```

DFS(G) :

```
1. for each v ∈ V
2.   visited(v) ← false
3. for each v ∈ V
4.   if not visited(v)
5.     DFS-VISIT(G, v)
```

DFS-VISIT(G, v) :

```
1. visited(v) ← true
2. pre-process(v)
3. for each w ∈ Adj(v)
4.   if not visited(w)
5.     DFS-VISIT(G, w)
6. post-process(v)
```

Numa aplicação concreta de DFS, há processamento a fazer num destes pontos

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

7

Pesquisa em largura (*breadth-first search*)

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

8

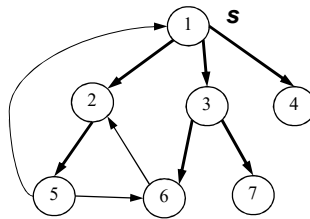
Ideia base

- ◆ Dado um vértice fonte s , explora-se sistematicamente o grafo descobrindo todos os vértices a que se pode chegar a partir de s (vértices adjacentes).
- ◆ Só depois é que se passa para outro vértice.

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

Exemplo

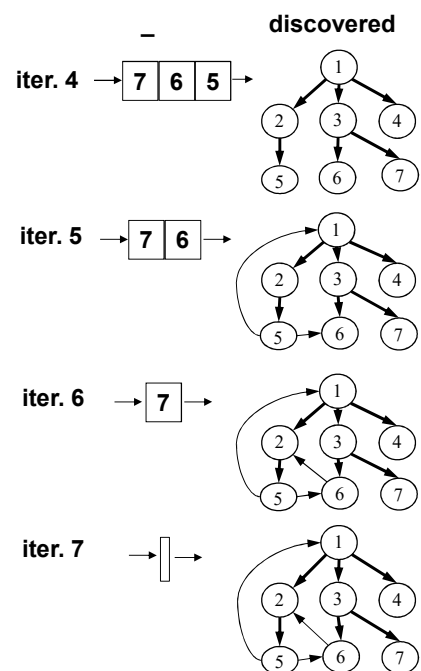
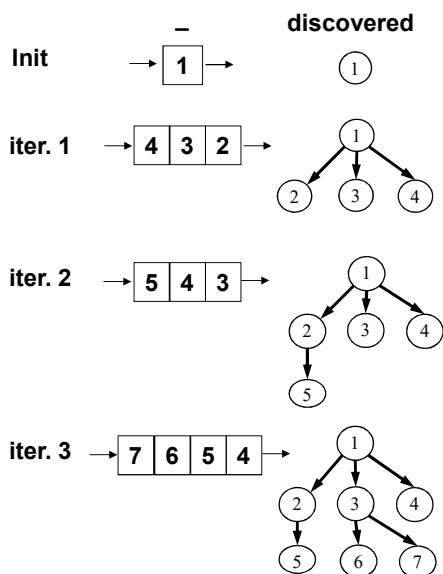
Exemplo em grafo dirigido, com vértices numerados por (pré)ordem de visita e dispostos por níveis de distância ao vértice inicial (s).



Arestas a traço forte formam uma árvore de expansão em largura (árvore BFS), com raiz s .

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

Passo a passo



Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

Pseudo-código

11

```

BFS(G, s) :
1.   for each v ∈ V do discovered(v) ← false
2.   Q ← ∅

3.   ENQUEUE(Q, s)
4.   discovered(s) ← true

5.   while Q ≠ ∅ do
6.       v ← DEQUEUE(Q)
7.       pre-process(v)
8.       for each w ∈ Adj(v) do
9.           if not discovered(w) then
10.              ENQUEUE(Q, w)
11.              discovered(w) ← true
12.      post-process(v)
  
```

Numa aplicação concreta de BFS, há processamento a fazer num destes pontos

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

Notas

12

- ◆ Para qualquer vértice *v* atingível a partir de *s*, o caminho na árvore BFS é o caminho mais curto no grafo (com menor número de arestas).
- ◆ BFS é um dos métodos mais simples e é o arquétipo para muitos algoritmos importantes de grafos.
 - Prim's Minimum-Spanning Tree
 - Dijkstra Single-Source Shortest-paths
- ◆ Se em vez de uma fila for usada uma pilha, obtém-se um algoritmo iterativo de visita em profundidade!

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

13

Ordenação topológica

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

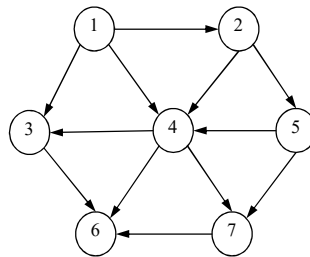
14

Problema

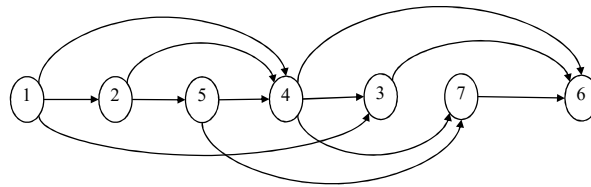
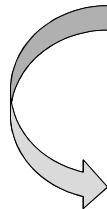
- ◆ Ordenar os vértices de um DAG (grafo acíclico dirigido) tal que, se existe uma aresta (v, w) no grafo, então v aparece antes de w
 - Intuitivamente, dispor as setas todas no mesmo sentido
 - Impossível se o grafo for cíclico
 - Pode existir mais do que uma ordenação (solução)

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

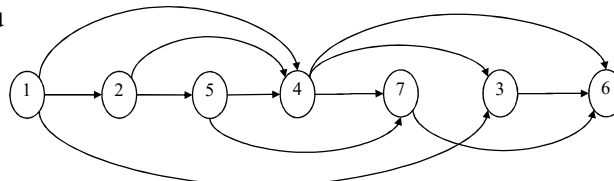
Exemplo



15



ou



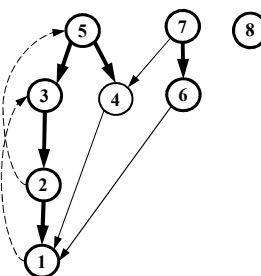
Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

Método baseado em DFS

16

- Na visita em profundidade (DFS) de um DAG, a pós-ordem de visita dá uma ordenação topológica inversa

São detetados ciclos quando vértice adjacente está na *stack* de visita



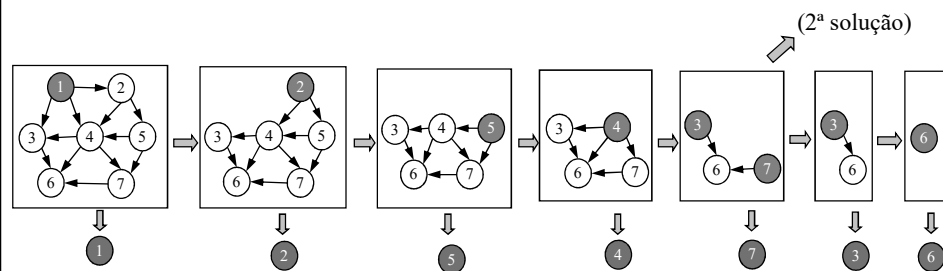
- Mas o método não é genérico, pois algumas ordenações topológicas válidas não podem ser obtidas desta forma
 - e.g., vértice 4 nunca fica entre vértices 3 e 2

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

17

Método geral

1. Descobrir um vértice sem arestas de chegada (*indegree*=0)
2. Imprimir o vértice
3. Eliminá-lo, assim como as arestas que dele saem
4. Repetir o processo no grafo restante



Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

18

Algoritmo de ordenação topológica

- ◆ Refinamento do método geral:
 - simular eliminação atualizando *indegree* dos vértices adjacentes
 - memorizar numa estrutura auxiliar vértices por imprimir *c/ indegree*=0
- ◆ Dados de entrada:
 - *V* - conjunto de vértices
 - *adj(v)* - conjunto (ou lista) de vértices adjacentes a cada vértice *v*
 - ou conj. de arestas que saem de *v*, que por sua vez indicam vértices adj.
- ◆ Dados de saída:
 - *T* - sequência (ou lista) dos vértices por ordem topológica
 - ou *numTop(v)* - número atribuído a cada vértice *v* por ordem topológica
- ◆ Dados temporários:
 - *indegree(v)* - nº de arestas que chegam a *v*, partindo de vértices por visitar
 - *C* - conjunto de vértices por visitar cujo *indegree* é 0 (candidatos)

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

19

Algoritmo de ordenação topológica

TOP-SORT(in $G=(V,E)$, out T):

1. for each $v \in V$ do $\text{indegree}(v) \leftarrow 0$
2. for each $v \in V$ do for each $w \in \text{adj}(v)$ do $\text{indegree}(w) \leftarrow \text{indegree}(w) + 1$
3. $C \leftarrow \{ \}$ // Pode ser uma fila (Queue), pilha (Stack), etc.
4. for each $v \in V$ do if $\text{indegree}(v) = 0$ then $C \leftarrow C \cup \{v\}$
5. $T \leftarrow []$ // Pode ser uma lista (LinkedList)
6. while $C \neq \{ \}$ do
7. $v \leftarrow \text{remove-one}(C)$
8. $T \leftarrow T \text{ concatenado-com } [v]$
9. for each $w \in \text{adj}(v)$ do
10. $\text{indegree}(w) \leftarrow \text{indegree}(w) - 1$
11. if $\text{indegree}(w) = 0$ then $C \leftarrow C \cup \{w\}$
12. if $|T| \neq |V|$ then Fail("O grafo tem ciclos")

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

20

Análise do algoritmo

- ◆ As diferentes escolhas do próximo vértice no ponto 7 dão as diferentes soluções possíveis
- ◆ Se as inserções e eliminações em C forem efectuadas em tempo constante (usando por exemplo uma fila FIFO), algoritmo pode ser executado em tempo $O(|V| + |E|)$
 - o corpo do ciclo de atualização do indegree (passos 9, 10, 11) é executado no máximo uma vez por aresta
 - as operações de inserção e remoção na fila (nos passos 4, 7 e 11) são executadas no máximo uma vez por vértice
 - a inicialização leva um tempo proporcional ao tamanho do grafo

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

21

Aplicações

- ◆ Grafos Acíclicos Dirigidos (DAG) são usados em aplicações onde é necessário indicar a precedência entre eventos.
- ◆ Exemplo: escalonamento de sequências de tarefas.
- ◆ A ordenação topológica de um DAG dá-nos uma ordem (sequência) dos eventos (tarefas, trabalhos, etc.) representados nesse DAG.
- ◆ No caso de existirem múltiplas soluções, podem-se usar critérios adicionais

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP

22

Referências e mais informação

- ◆ T.H. Cormen, C. E. Leiserson, R. L. Rivest , C. Stein. Introduction to Algorithms, 3rd Edition. MIT Press, 2009
 - Capítulo 22
- ◆ “Data Structures and Algorithm Analysis in Java”, Second Edition, Mark Allen Weiss, Addison Wesley, 2006

Algoritmos em Grafos: Introdução • CAL - MIEIC/FEUP