Ordenação topológica

Outro algoritmo

TOPOLOGICAL-SORT'(G)

```
for each vertex u in G.V do
    u.i <- 0
for each edge (u,v) in G.E do
    v.i <- v.i + 1              // arcos com destino v
L <- EMPTY                      // lista
S <- EMPTY                      // conjunto
for each vertex u in G.V do
    if u.i = 0 then
        SET-INSERT(S, u)
while S != EMPTY do
    u <- SET-DELETE(S)          // retira um nó de S
    for each vertex v in G.adj[u] do
        v.i <- v.i - 1
        if v.i = 0 then
            SET-INSERT(S, v)
    LIST-INSERT-TAIL(L, u)
return L
```

Conectividade (1)

Seja $G = (V, E)$ um grafo não orientado

G é conexo se existe algum caminho entre quaisquer dois nós

$V' \subseteq V$ é uma componente conexa de G se

- existe algum caminho entre quaisquer dois nós de V' e
- não existe qualquer caminho entre algum nó de V' e algum nó de $V \setminus V'$

Conectividade (2)

Seja $G = (V, E)$ um grafo orientado

G é fortemente conexo se existe algum caminho de qualquer nó para qualquer outro nó

$V' \subseteq V$ é uma componente fortemente conexa de G se

- existe algum caminho de qualquer nó de V' para qualquer outro nó de V' e
- se, qualquer que seja o nó $u \in V \setminus V'$
  - não existe qualquer caminho de algum nó de V' para u ou
  - não existe qualquer caminho de u para algum nó de V'

Grafo transposto

O grafo transposto do grafo orientado $G = (V, E)$ é o grafo

$$G^T = (V, E^T)$$

tal que

$$E^T = \{(v, u) \mid (u, v) \in E\}$$

Componentes fortemente conexas

Strongly connected components

G – grafo orientado

SCC(G)

1. Aplicar DFS(G) para calcular o instante `u.f` em que termina o processamento de cada vértice `u`
2. Calcular G^T
3. Aplicar DFS(G^T), processando os vértices por ordem decrescente de `u.f` (calculado em 1), no ciclo principal de DFS (linha 5)
4. Devolver os vértices de cada árvore da floresta da pesquisa em profundidade (construída em 3) como uma componente fortemente conexa distinta

Árvore de cobertura mínima (1)

Minimum(-weight) spanning tree

Seja $G = (V, E)$ um grafo pesado não orientado conexo

Uma árvore é um grafo não orientado conexo acíclico

(Retirando qualquer arco de uma árvore, obtém-se um grafo não conexo)

Uma árvore de cobertura de G é um subgrafo $G' = (V', E')$ de G tal que

- $V' = V$
- $E' \subseteq E$
- G' é uma árvore

Árvore de cobertura mínima (2)

Minimum(-weight) spanning tree

Seja o peso de um grafo $w(G)$ a soma dos pesos dos arcos de G

$$w(G) = \sum_{e \in E} w(e)$$

Uma árvore de cobertura mínima de G é uma árvore de cobertura G' de peso mínimo:

Para qualquer árvore de cobertura G'' de G tem-se

$$w(G') \leq w(G'')$$

Árvore de cobertura mínima

Algoritmo de Prim

$G = (V, E)$ – grafo pesado não orientado conexo

MST-PRIM(G, w, s)

```
for each vertex u in G.V do
    u.key <- INFINITY                // cost of adding u
    u.p <- NIL
s.key <- 0
Q <- G.V                  // priority queue, with key u.key
while Q != EMPTY do
    u <- EXTRACT-MIN(Q)
    for each vertex v in G.adj[u] do
        if v in Q and w(u,v) < v.key then
            v.p <- u
            v.key <- w(u,v)          // decrease key in Q
```