Декларативно програмиране в процедурните езици

Упражнение по топологично сортиране

Владислав Ненчев

Софийски Университет “Св. Климент Охридски”
Факултет по Математика и Информатика
Катедра по Математическа Логика и Приложенията ѝ

18 Ноември 2016

Топологично сортиране - алгоритъм

$$front = \{ v \in vertices \mid v.incomming = \emptyset \}$$

while $front \neq \emptyset$

$x = pop(front)$

print x

for $\langle x, y \rangle \in edges$

$u.incomming.remove(\langle x, y \rangle)$

$$front = front \cup \{ v \in vertices \mid v.incomming = \emptyset \}$$

Топологично сортиране - реализация

```
public Graph(params GraphEdge[] graphEdges)
{
  edges = new HashSet<GraphEdge>(graphEdges);

  vertices = new HashSet<GraphVertex>(
    edges.Select(edge => edge.From));
  vertices.UnionWith(edges.Select(edge => edge.To));
}

private HashSet<GraphVertex> vertices;
private HashSet<GraphEdge> edges;
```

Топологично сортиране - реализация (продължение)

```
public IEnumerable<GraphVertex> TopologicalSort()
{
  Dictionary<GraphVertex, HashSet<GraphEdge>> in =
    edges.GroupBy(edge => edge.To).
      ToDictionary(
        grouping => grouping.Key,
        grouping =>
          new HashSet<GraphEdge>(grouping.ToList()));
  Dictionary<GraphVertex, HashSet<GraphEdge>> out =
    edges.GroupBy(edge => edge.From).
      ToDictionary(
        grouping => grouping.Key,
        grouping =>
          new HashSet<GraphEdge>(grouping.ToList()));
```

Топологично сортиране - реализация (продължение)

```
HashSet<GraphVertex> front =
  new HashSet<GraphVertex>(vertices.Where(
    vertex => in.GetOrDefault(vertex).Count == 0));
while(front.Count > 0)
{
  GraphVertex current = front.First();
  yield return current;

  front.Remove(current);
  out.GetOrDefault(current).
    Select(edge => edge.To).
      ForEach(vertex => in.GetOrDefault(vertex).
        Remove(Edge(current, vertex)));
  front.UnionWith(
    out.GetOrDefault(current).
      Select(edge => edge.To).
        Where(vertex =>
          in.GetOrDefault(vertex).Count == 0));
}
```