Dawid Pawliczek

Lista 1, Zadanie 3

Treść zadania. *Porządkiem topologicznym* wierzchołków acyklicznego digrafu $G = (V, E)$ nazywamy taki liniowy porządek jego wierzchołków, w którym początek każdej krawędzi występuje przed jej końcem. Jeśli wierzchołki z V utożsamimy z kolejnymi liczbami naturalnymi $1, 2, \ldots, |V|$, to każdy porządek liniowy możemy opisać permutacją tych liczb, co pozwala również na *leksykograficzne* porównywanie porządków. Ułóż algorytm, który dla zadanego acyklicznego digrafu znajduje **pierwszy leksykograficznie** porządek topologiczny.

Pomysł (modyfikacja algorytmu Kahna). Klasyczne sortowanie topologiczne Kahna usuwa kolejno wierzchołki o zerowym stopniu wejściowym (*indegree*). Aby otrzymać pierwszą permutację leksykograficznie, zawsze wybieramy *najmniejszy etykietą* spośród wierzchołków o *indegree* $= 0$. Do realizacji wystarczy kolejka priorytetowa (np. kopiec).

Algorithm 1 LexicoKahn($G = (V, E)$)

```
1:  Q ← pusta min-kolejka priorytetowa
2:  order ← [ ]                                          ▷ wynikowa permutacja
3:  for all v ∈ V do
4:      oblicz indeg[v]
5:      if indeg[v] = 0 then
6:          Q.push(v)
7:
8:          while Q ≠ ∅ do
9:              v ← Q.popMin()                           ▷ najmniejsza etykieta
10:             order.append(v)
11:             for all następnik u wierzchołka v do
12:                 indeg[u] ← indeg[u] − 1
13:                 if indeg[u] = 0 then
14:                     Q.push(u)
15:
16:
17:                     return order
```

Algorytm.

Złożoność. Każdy wierzchołek jest dokładnie raz wstawiany i zdejmowany z kopca; operacje te kosztują $O(\log |V|)$. Łączny czas: $O\big((|V| + |E|) \log |V|\big)$, pamięć pomocnicza: $O(|V|)$.