Sortowanie topologiczne skierowanych grafów acyklicznych

Metody boolowskie w informatyce

Robert Sulkowski

http://robert.brainusers.net

23 stycznia 2010

Definicja 1 (Cykl skierowany)**.** Niech $C = (V, A)$ będzie grafem skierowanym, gdzie $V = \{v_1, \ldots, v_n\}$. C nazywamy cyklem skierowanym, jeżeli

$$A = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n), (v_n, v_1)\}$$

Mówimy, że graf skierowany D posiada cykl, jeżeli istnieje podgraf C grafu D będący cyklem skierowanym.

Definicja 2 (Skierowany graf acykliczny)**.** Graf skierowany D nazywamy acyklicznym, jeżeli nie posiada cykli.

Przykład 1. Niech $(V, \leq)$ będzie zbiorem częściowo uporządkowanym przez zwrotną, asymetryczną i przechodnią relację $\leq$. Zauważmy, że zbiór ten można przedstawić za pomocą grafu skierowanego: V będzie zbiorem wierzchołków a relacja $\leq$ zbiorem łuków grafu (relacja jest zbiorem par uporządkowanych (x, y)). Jeśli nie liczyć pętli (x, x) (zwrotność relacji), powstały w ten sposób graf jest acykliczny ze względu na asymetryczność relacji $\leq$.

Definicja 3 (Sortowanie topologiczne)**.** Niech $D = (V, A)$ będzie grafem skierowanym. Sortowaniem topologicznym nazywamy uporządkowanie (ponumerowanie) wierzchołków grafu D

$$v_1, v_2, \ldots, v_n$$

takie że dla każdego łuku $(v_i, v_j) \in A$ zachodzi $i \leq j$.

Definicja 4 (Stopień wejścia wierzchołka)**.** Niech $D = (V, A)$ będzie grafem skierowanym. Stopniem wejścia wierzchołka $v \in V$ nazywamy ilość łuków postaci $(u, v) \in A$ (v jest następnikiem pary) i oznaczamy $d_{in}(v)$.

Lemat 1. Jeśli $D = (V, A)$ jest skierowanym grafem acyklicznym, to zawiera co najmniej jeden wierzchołek o stopniu wejścia $d_{in}(v) = 0$ (wierzchołek do którego nie prowadzi żaden łuk).

Dowód. Wybierzmy wierzchołek $v_0 \in V$. Jeśli $d_{in}(v_0) = 0$ to kończy nasz dowód. W przeciwnym wypadku istnieje łuk $(v_1, v_0) \in A$. Jeśli $d_{in}(v_1) = 0$ to kończy dowód, w przeciwnym razie istnieje łuk $(v_2, v_1) \in A$ gdzie $v_2 \neq v_0$ (bo powstałby cykl, ale graf jest acykliczny). W ten sposób możemy wybierać wierzchołki grafu D mając pewność, że żaden się nie powtórzy, ponieważ D jest grafem acyklicznym. Z uwagi na to, że ilość wierzchołków jest skończona, ostatni wybrany wierzchołek będzie miał zerowy stopień wejścia — gdyby istniał łuk z któregoś z poprzednich wierzchołków, powstałby cykl (zaczynaliśmy od dowolnego wierzchołka). Jeśli G nie jest spójny to nasze rozumowanie jest nadal prawidłowe, gdyż ograniczamy się do jednej komponenty spójności, gdyż każda komponenta jest acykliczna.

□

Twierdzenie 2. Każdy skierowany graf acykliczny posiada sortowanie topologiczne.

Dowód. Niech $D = (V, A)$ będzie skierowanym grafem acyklicznym. Korzystając z poprzedniego lematu, w grafie D istnieje co najmniej jeden wierzchołek $v_1 \in V$ o zerowym stopniu wejścia ($d_{in}(v) = 0$). Zauważmy, że zdejmując ten wierzchołek z grafu D otrzymamy graf H, który jest również acykliczny (zdejmujemy wszystkie łuki incydentne z v_1 więc w wyniku tej operacji nie może powstać cykl). Zatem H posiada wierzchołek v_2 o zerowym stopniu wejścia. Powtarzając tę operację aż do wyczerpania wierzchołków otrzymamy topologiczne sortowanie zgodne z kolejnością ich zdejmowania:

$$v_1, v_2, \ldots, v_n$$

Zauważmy, że niespójność grafu G nie psuje rozumowania. Kolejność zdejmowania wierzchołków z różnych komponent spójności nie ma znaczenia dla sortowania topologicznego, ponieważ nie istnieje łuk, który by je łączył. □

Wniosek 1. Sortowanie topologiczne nie jest wyznaczane jednoznacznie.

Wniosek 2. Dane sortowanie topologicznie generuje porządek liniowy wierzchołków.

Zaprezentujemy teraz algorytm, który sprawdza czy dany skierowany graf D jest acykliczny i, jeśli jest, znajduje sortowanie topologiczne. Algorytm wykorzystuje metodę pokazaną w dowodzie poprzedniego twierdzenia. Szukamy wierzchołków o zerowym stopniu wejścia i usuwamy je z grafu D. Korzystamy z listy sąsiedztw wyjścia i utrzymujemy listę wierzchołków, które aktualnie posiadają zerowy stopień wejścia.

Algorytm 1 (Kahn, 1962)**.** Wymagane struktury danych:

1. Listy sąsiedztw wyjścia $A[v]$ gdzie $v \in V$ jest wierzchołkiem grafu $D = (V, A)$.
2. Tablica *ind* która przechowuje stopień wejścia $ind[v]$ wierzchołka v.
3. Tablica *topnr* która przechowuje numer $topnr[v]$ nadany wierzchołkowi v w sortowaniu topologicznym.
4. Lista L wierzchołków o zerowym stopniu wejścia.
5. Zmienna logiczna *acyclic* (odpowiada na pytanie, czy graf jest acykliczny) i zmienna typu całkowitego m (licznik ściągniętych wierzchołków).

```
PROCEDURE topsort;
BEGIN
  m:=0; wyczyść listę L; for v:=1 to n do ind[v]:=0; {Inicjalizacja}
  for v:=1 to n do {Uzupełniamy stopień wejścia na podstawie listy sąsiedztw}
     for w in A[v] do ind[w]:=ind[w]+1;
  for i:=v to n do {Uzupełniamy listę wierzchołków o zerowym stopniu wejścia}
     if ind[v]:=0 then połóż i na końcu listy L;
  while (L jest niepusta) do begin
     v:=ściągnij pierwszy wierzchołek z~listy L;
     m:=m+1; {Powiększamy licznik}
     topnr[v]:=m; {Nadajemy numer wierzchołka w sortowaniu topologicznym}
     for w in A[v] do begin {Przeglądamy sąsiadów}
        ind[w]:=ind[w]-1; {Pomniejszamy stopień wejścia sąsiada}
        {Jeśli stopień wejścia spadł do zera, dodajemy go do listy L}
        if ind[w]=0 then połóż w na końcu listy L;
     end;
  end;
  {Graf jest acykliczny, gdy algorytm ściągnął tyle wierzchołków, ile graf posiadał}
  acyclic:=(m=n);
END;
```

Twierdzenie 3. Algorytm sprawdza, czy graf skierowany D jest acykliczny i jeśli jest, konstruuje topologiczne sortowanie. Jego złożoność wynosi $O(|E|)$.

Dowód. Algorytm jest poprawny, gdyż bazuje na dowodzie twierdzenia 2. Zauważmy, że $|E| = \Omega(|V|)$, stąd inicjalizacja danych potrzebuje nie więcej niż $O(|E|)$ kroków. Każdy łuk przetwarzany jest dokładnie raz podczas inicjalizacji i co najwyżej raz w pętli głównej. Zatem złożoność algorytmu wynosi $O(|E|)$.

□

Wniosek 3. Problem sprawdzania, czy dany graf jest acykliczny posiada złożoność rzędu $\Theta(|E|)$. Zatem należy do klasy problemów P.