GEOMETRIA OBLICZENIOWA - ĆWICZENIA

Adrian Mucha 236526, Politechnika Wrocławska WPPT 19.05.2020

Zadanie 11

Podaj algorytm liniowy obliczający 3-kolorowanie striangulowanego wielokąta prostego.

Rozwiązanie

Niech T_P będzie traingulacją wielokąta P. Wybierzemy podzbiór wierzchołków P taki, że każdy trójkąt w T_P ma co najmniej jeden wybrany wierzchołek i umieścimy w tych wierzchołkach kamery. Każdemu wierzchołkowi nadamy jeden z kolorów `czarny`, `szary` lub `biały` w taki sposób by dowolne dwa wierzchołki połączone krawędzią lub przekątną miały różne kolory. Każdy trójkąt wielokąta ma wierzchołek `czarny`, `szary` oraz `biały`. Kamery umieścimy we wszystkich wierzchołkach tego samego koloru, który jest najmniej liczny dzięki czemu możemy chronić P, używając co najwyżej $\lfloor \frac{n}{3} \rfloor$ kamer.

Aby dokonać 3-kolorowania wielokąta, wykorzystamy graf dualny T_P, oznaczmy go jako $G(T_P)$. Wierzchołkami grafu będą trójkąty w T_P. Trójkąt odpowiadający węzłowi v oznaczamy jako $t(v)$. Krawędź między węzłami v i μ istnieje gdy $t(v)$ i $t(\mu)$ rozdziela przekątna. Krawędź w $G(T_P)$ odpowiada przekątnej w T_P. $G(T_P)$ jest drzewem. Kolorowanie grafu możemy zatem znaleźć za pomocą przeszukiwania w głąb. Podczas przeszukiwania w głąb zachowujemy następujący niezmiennik: **wszystkie wierzchołki napotkanych dotąd trójkątów zostały pomalowane na biało, szaro, lub czarno i żadne dwa połączone wierzchołki nie otrzymały tego samego koloru**. Przechodząc z wierzchołka μ do wierzchołka v, $t(v)$ oraz $t(\mu)$ rozdziela przekątna. Ponieważ wierzchołki $t(\mu)$ zostały już pomalowane, więc tylko jeden wierzchołek $t(v)$ został do pokolorowania (można to zrobić na jeden sposób).

Algorithm 1 3-kolorowanie striangulowanego wielokąta

1: **function** VISITNODE(*Wierzchołek* μ)
2: oznacz μ jako odwiedzony
3: **for** dla każdego wierzchołka $v \in$ `neighbours`(μ) **do**
4: **if** v jest nieodwiedzony **then**
5: pokoloruj przeciwległy wierzchołek na pozostały kolor niekolidujący z wierzchołkami na krawędzi dzielącej $t(v)$ oraz $t(\mu)$
6: VisitNode(v)
7: **end if**
8: **end for**
9: **end function**
10: **function** DEPTHFIRSTSEARCH(*Graf* G)
11: **for** dla każdego wierzchołka $\mu \in G$ **do**
12: oznacz μ jako nieodwiedzony
13: **end for**
14: wybierz dowolny $\mu \in G$
15: pokoloruj wierzchołki trójkąta $t(\mu)$
16: VisitNode(μ)
17: **end function**
18:
19: DEPTHFIRSTSEARCH($G(T_P)$)
20: **return** ilość wierzchołków koloru najmniej licznego

Procedura `DepthFirstSearch` ma złożoność $\mathcal{O}(|V|)$ gdyż wraz z podprocedurą `VisitNode` odwiedzają każdy wierzchołek jeden raz. Obliczenie *trzeciego brakującego koloru* odbywa się w czasie $\mathcal{O}(1)$.