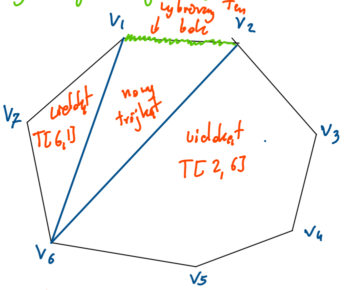
AiSD Zadanie 4.9-- Krystian Jasionek

środa, 5 maja 2021 10:04

9. (2pkt) Ułóż algorytm rozwiązujący poniższy problem triangulacji wielokąta wypukłego:

PROBLEM:
Dane: ciąg par liczb rzeczywistych $(x_1, y_1), \ldots, (x_n, y_n)$, określających kolejne wierzchołki n-kąta wypukłego P
ZAŁOŻENIE: dane są określone poprawnie.
Zadanie: Znaleźć zbiór S nieprzecinających się przekątnych, które dzielą P na trójkąty, taki, że długość najdłuższej przekątnej w S jest możliwie najmniejsza.

Wyznaczmy długość najdłuższej przekątnej podziału.



Oznaczmy $T[i,j]$ jako minimum z najdłuższych przekątnych podziału wielokąta $\langle v_i, \ldots, v_j \rangle$.

Obserwacje:

1) Bok wielokąta musi należeć do jakiegoś trójkąta.

2) Z każdego wierzchołka wychodzi $n-3$ przekątnych.

Wybierając bok (v_1, v_2) możemy poprowadzić przekątną od v_1 do wierzchołków $v_3 - v_6$. Weźmy np. v_6 i poprowadźmy też przekątną $v_6 - v_2$.

Otrzymamy w ten sposób trójkąt (v_1, v_2, v_6) oraz dwa mniejsze wielokąty. Możemy szukać w nich rekurencyjnie najdłuższych przekątnych.

Przekątne, którymi wydzieliliśmy mniejsze wielokąty mogą być szukanymi najdłuższymi, ale równie dobrze najdłuższa przekątna może znaleźć się w podziale któregoś z mniejszych wielokątów. Będziemy zatem wybierać maksimum z tych czterech wartości - dwóch przekątnych i dwóch wywołań rekurencyjnych ($\max\{P[6,1], P[2,6], |v_1 v_6|, |v_2 v_6|\}$).

Zauważmy, że w ten sposób rozważymy też zdegenerowane przypadki dwukątów, tzn. odcinków. Dla takich wywołań rekurencyjnych zwracamy po prostu 0, bo odcinek nie zawiera przekątnej. Podobnie zwrócimy dla trójkątów, dla których obserwacja, że każdy wierzchołek wyprowadza $n-3$ przekątnych daje nam 0 przekątnych (n to liczba wierzchołków). Nas interesuje minimum z takich najdłuższych przekątnych, więc musimy wyznaczyć z nich minimum

Powyższe rozważania pozwalają zapisać następującą zależność rekurencyjną:

$$P[i,j] = \begin{cases} 0, \\ \min\limits_{i<k<j} \left\{ \max\{P[i,k], P[k,j], \underbrace{|v_i v_k|, |v_k v_j|}_{\text{tylko jeśli odpowiednio } k-i>1 \text{ i } k-j>1,\ \text{inaczej jest to punkt.}}\} \right\} \end{cases}$$

Korzystając z niej możemy dynamicznie skonstruować tablicę $P[n][n]$, zaczynając od najbliższych sobie wierzchołków i, j i zwiększając dystans między nimi. Wynik rekurencji odczytamy z komórki $P[1][n]$, która odpowiada najdłuższej przekątnej dla wielokąta $\langle v_1 \ldots v_n \rangle$. Do obliczenia każdego pola tablicy musimy obliczyć minimum z $O(n)$ wcześniej policzonych pól. Mamy $O(n^2)$ pól. Ustalmy zatem złożoności to i

Czasowa: $O(n^3)$
pamięciowa: $O(n^2)$.

Jak wyznaczyć szukany w treści zadania zbiór, zawierający tą przekątną $P[1][n]$?

Oprócz wartości najdłuższej przekątnej możemy w każdym polu tablicy P przechowywać też wartość o wartości k, dla której otrzymaliśmy tą przekątną. Dla przypadków zdegenerowanych zapiszmy wartość $k=-1$, która wyznacza przypadek podstawowy. Teraz wystarczy przejść się po tablicy od tyłu i przeskakiwać pod wskazane pole k, wyznaczając tak kolejne przekątne z szukanego zbioru. Możemy to zrobić w czasie $\leq O(n^2)$, bo tablica ma rozmiar $n \times n$, zatem łączny koszt algorytmu to $O(n^3)$.