Zadanie 10 lista 4

Jakub Kuciński, prowadzący Szymon Dudycz

26 czerwca 2020

Spis treści

1 Idea rozwiązania

Oznaczmy kolejne wierzchołki wejściowego wielokąta wypukłego $W_{1,n}$ jako $A_i = (x_i, y_i)$. Niech
$T[i][j]$ - możliwie najmniejsza długość najdłuższej przekątnej w triangulacji wielokąta $W_{i,j}$ składającego się z wierzchołków $A_i, \dots, A_j$ (oczywiście taki wielokąt też jest wypukły).
Zastosujemy programowanie dynamiczne. Rozważmy krawędź (A_i, A_j). Wiemy, że w triangulacji wielokąta $W_{i,j}$ krawędź (A_i, A_j) musi być bokiem pewnego trójkąta. Wiemy, że wierzchołki A_i, A_j są wierzchołkami tego trójkąta, zatem trzeci wierzchołek $A_k \in \{A_{i+1}, \dots, A_{j-1}\}$. Wybranie pewnego wierzchołka A_k dla naszego trójkąta dzieli nam wielokąt $W_{i,j}$ na mniejsze wielokąty $W_{i,k}$ i $W_{k,j}$, których minimalną triangulację już znamy. Zatem dla danych i oraz j szukamy takiego k, że największa spośród wartości $T[i][k]$, $T[k][j]$, $d(A_i, A_k)$, $d(A_k, A_j)$ jest możliwie najmniejsza. Możemy zatem zapisać zależność na wartości tabeli T:

$$T[i][j] = \begin{cases} 0, & \text{jeśli } j \leqslant i+2 \\ \min\limits_{k \in \{i+1, \dots, j-1\}} (\max(T[i][k], T[k][j], d(A_i, A_k), d(A_k, A_j))), & \text{w przeciwnym wypadku} \end{cases}$$

Naszym wynikiem będzie oczywiście wartość w elemencie $T[1][n]$, bo odpowiada ona całemu wejściowemu wielokątowi. Rozważamy wartości $T[i][j]$ tylko dla $i \leqslant j$. Przyjmujemy również, że odległość d sąsiednich wierzchołków jest równa 0 - jest to potrzebne dla przypadku, gdy jako 3. wierzchołek trójkąta rozważamy wierzchołek będący sąsiadem A_i lub A_j, bo wówczas powstaje tylko jedna przekątna.

2 Algorytm

Zmienna i w algorytmie odpowiada początkowi rozważanego wielokąta, wartość $p + 1$ odpowiada liczbie jego wierzchołków, a zmienna k numerowi wierzchołka, który rozważamy jako trzeci wierzchołek trójkąta o boku (A_i, A_{i+p}). W tablicy $Trace$ zapamiętujemy numer trzeciego wierzchołka k dla którego otrzymaliśmy najmniejszą triangulację dla danego wielokąta. Przechodzimy rosnąco po liczbie wierzchołków wielokątów, bo w zależności elementów T widzimy, że wartość T dla danego wielokąta zależy tylko od wartości T dla mniejszych wielokątów.

Algorithm 1 *Minimalna_triangulacja*

```
for i = 1, ..., n do
    for p = 0, 1, 2 do
        if i + p ⩽ n then
            T[i][i + p] = 0
for p = 3, ..., n do
    for i = 1, ..., n do
        if i + p ⩽ n then
            min_length = ∞
            for k = i + 1, ..., i + p − 1 do
                if max(T[i][k], T[k][i + p], d(A_i, A_k), d(A_k, A_{i+p})) < min_length then
                    Trace[i][j] = k
                    min_length = max(T[i][k], T[k][i + p], d(A_i, A_k), d(A_k, A_{i+p}))
            T[i][i + p] = min_length
        else
            break
return T[1][n]
```

Algorithm 2 *Zbior_przekatnych*(i, j)

```
if j ⩽ i + 2 then
    return ∅
if Trace[i][j] == i + 1 then
    return (i + 1, j) ∪ Zbior_przekatnych(i + 1, j)
if Trace[i][j] == j − 1 then
    return (i, j − 1) ∪ Zbior_przekatnych(i, j − 1)
k = Trace[i][j]
return (i, k) ∪ (k, j) ∪ Zbior_przekatnych(i, k) ∪ Zbior_przekatnych(k, j)
```

Do odtworzenia zbioru przekątnych w minimalnej triangulacji wystarczy wywołać drugą funkcję na parametrach 1 oraz n.

3 Złożoność obliczeniowa i pamięciowa

3.1 Złożoność pamięciowa

Potrzebna pamięć jest tego samego rzędu co rozmiar tabeli T oraz $Trace$, czyli $\mathcal{O}(n^2)$. Zbiór przekątnych najmniejszej triangulacji wymaga $\mathcal{O}(n)$ pamięci, bo dla n-kąta dowolna triangulacja składa się z $n-3$ przekątnych.

3.2 Złożoność obliczeniowa

Uzupełnianie odpowiednich elementów T zerami wykonuje się w czasie $\mathcal{O}(n)$. Dominująca będzie druga "większa" pętla.

Warunki w środku pętli wymagają stałej liczby operacji. Zatem liczba wykonanych operacji będzie wprost proporcjonalna do liczby obrotów wszystkich pętli. Wewnętrzna pętla wykonuje się $p-2$ razy. Środkowa $n-p+1$. Możemy zatem zapiać, że łączna liczba obrotów pętli wynosi:

$\sum_{p=3}^{n}(n-p+1)(p-2) = \frac{1}{6}n(n^2-3n+2) = \mathcal{O}(n^3)$

Rozważmy złożoność odtwarzania przekątnych w minimalnej triangulacji. Mamy jeden przypadek, który nie dodaje żadnej przekątnej i nie wywołuje rekurencyjnie tej samej procedury oraz 3 przypadki, w których powstają rekurencyjne wywołania i zostaje dodana przynajmniej jedna przekątna. Jeśli rozważymy drzewko wywołań rekurencyjnych, to liczba liści jest w oczywisty sposób mniejsza równa podwojonej liczbie węzłów niebędących liściami. Zastanówmy się teraz ile może być maksymalnie węzłów niebędących liśćmi. Odpowiadają one przypadkom, w którym dodajemy do zbioru jakąś przekątną. Wiemy z kolei, że w dowolnej triangulacji n-kąta mamy dokładnie n-3 przekątne. Zatem mamy co najwyżej $n-3$ węzły niebędące liśćmi, a stąd łącznie co najwyżej $\mathcal{O}(n)$ wywołań rekurencyjnych.
Czyli ostatecznie złożoność obliczeniowa algorytmu to $\mathcal{O}(n^3)$.