AiSD2 2020/2021 - Laboratorium 12

Autor: MB

Ważne wskazówki techniczne: wszystkie wielokąty przechowujemy jako listę lub tablicę krotek z wartościami typu `double` reprezentującymi wierzchołki, przy czym wielokąt **zawsze podany jest zgodnie z ruchem wskazówek zegara**.

Tam, gdzie trzeba będzie zwracać trójkąty w rozwiązaniu, kolejność wierzchołków w trójkącie (i tylko w trójkącie) nie ma znaczenia.

Etap 1 (0.5 pkt.)

Wielokąt P nazywamy monotonicznym względem prostej L, jeśli każda prosta ortogonalna do L przecina wielokąt co najwyżej 2 razy.

Wielokąt P nazywamy y-monotonicznym jeśli jest monotoniczny względem prostej tożsamej z osią Y (czyli jest monotoniczny względem dowolnej pionowej prostej).

Spójrzmy na rysunek 1. Zielone linie przecinają wielokąt raz, niebieskie dwa razy, a czerwone - ponad dwa razy. Pierwsze dwa wielokąty są y-monotoniczne, podczas gdy kolejne dwa nie są. Rysunek 2 przedstawia kolejne dwa wielokąty, w tym przypadek wielokąta wklęsłego, który jest y-monotoniczny.

Pierwszy etap polega na sprawdzeniu, czy wielokąt jest y-monotoniczny.

Rozwiązanie powinno być liniowe względem liczby wierzchołków wielokąta.

Etap 2 (1 pkt.)

Mówimy, że dwa punkty w wielokącie P są widoczne, jeśli łączy je odcinek leżący w P. Mówimy, że wielokąt P jest słabo widoczny z krawędzi uv jeśli dla każdego $z \in P$ istnieje takie $w \in uv$ (zależne od z), że z i w są widoczne. Wielokąt jest krawędziowidoczny jeśli istnieje przynajmniej jedna krawędź P (bok wielokąta P) taka że wielokąt P jest z niej słabo widoczny.

Innymi słowy, interesują nas takie wielokąty P, że można wskazać jeden bok P taki że można z niego poprowadzić odcinek do każdego punktu w P.

Przedstawimy teraz algorytm na triangulację (podział na trójkąty) dla wielokątów krawędziowidocznych. Będzie on oparty o niepoprawny algorytm wyznaczania otoczki wypukłej Sklansky'ego.

Na wejściu mamy wielokąt P oraz uv krawędź z której wielokąt P jest widoczny.

Startując z v i idąc zgodnie z ruchem wskazówek zegara wykonaj dla każdych kolejnych 3 punktów p_k, p_{k+1}, p_{k+2} (Niech porządek punktów, po którym idziemy, będzie reprezentowany przez T. Gdy mówimy o przesuwaniu się do przodu lub do tyłu, mamy na myśli przesuwanie się po punktach w T; co to dokładnie znaczy przesuwanie do przodu albo do tyłu, należy prześledzić na rysunku 3 oraz rozróżniać przypadek gdy usuwamy punkt z T lub nie):

1. $S = (y_{k+1} - y_k)(x_{k+2} - x_{k+1}) + (x_k - x_{k+1})(y_{k+2} - y_{k+1})$
2. Jeśli $S <= 0$, to znaczy, że skręt jest wklęsły (w lewo), pomijamy i przesuwamy się naprzód o jeden wierzchołek
3. Jeśli $S > 0$, to znaczy, że skręt jest wypukły (w prawo). Wtedy należy dodać do rozwiązania trójkąt złożony z rozważanych punktów. Usuń w porządku T punkt p_{k+1}. Jeśli p_k to był punkt v, przesuń się do przodu o jeden wierzchołek. W przeciwnym wypadku przesuń się do tyłu o jeden wierzchołek.
4. Jeśli w wyniku działania algorytmu będziemy chcieli dodać trójkąt z krawędzią uv, to to robimy i w tym momencie kończymy algorytm.

W tym etapie należy dla danego wielokąta (krawędź uv podana jest w ten sposób, że zmienna `edge_i` to indeks wierzchołka u, a `edge_j` to indeks wierzchołka v) należy zwrócić listę trójkątów, na jakie dzielimy ten wielokąt.

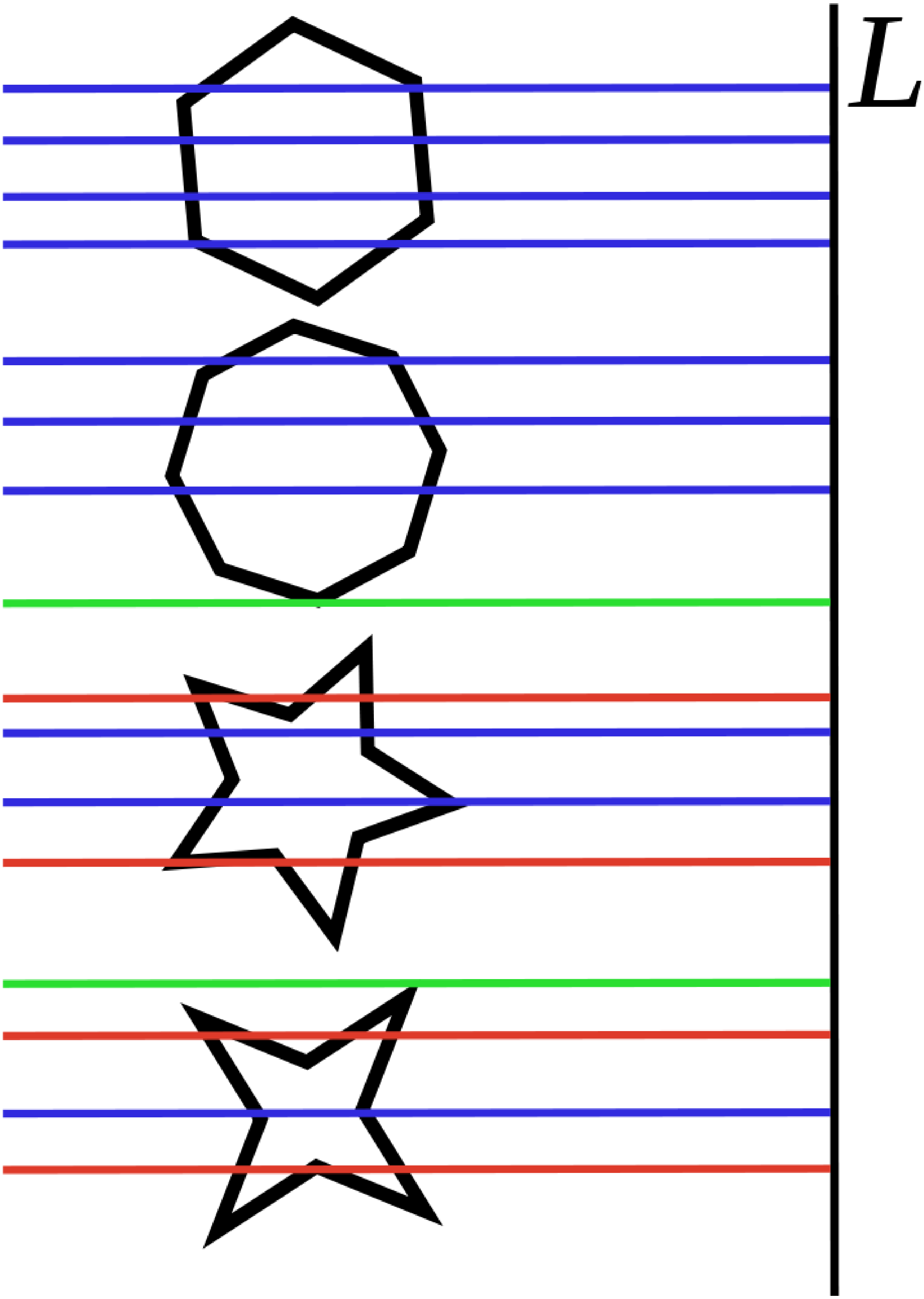


Figure 1: Pierwsze dwa wielokąty są y-monotoniczne

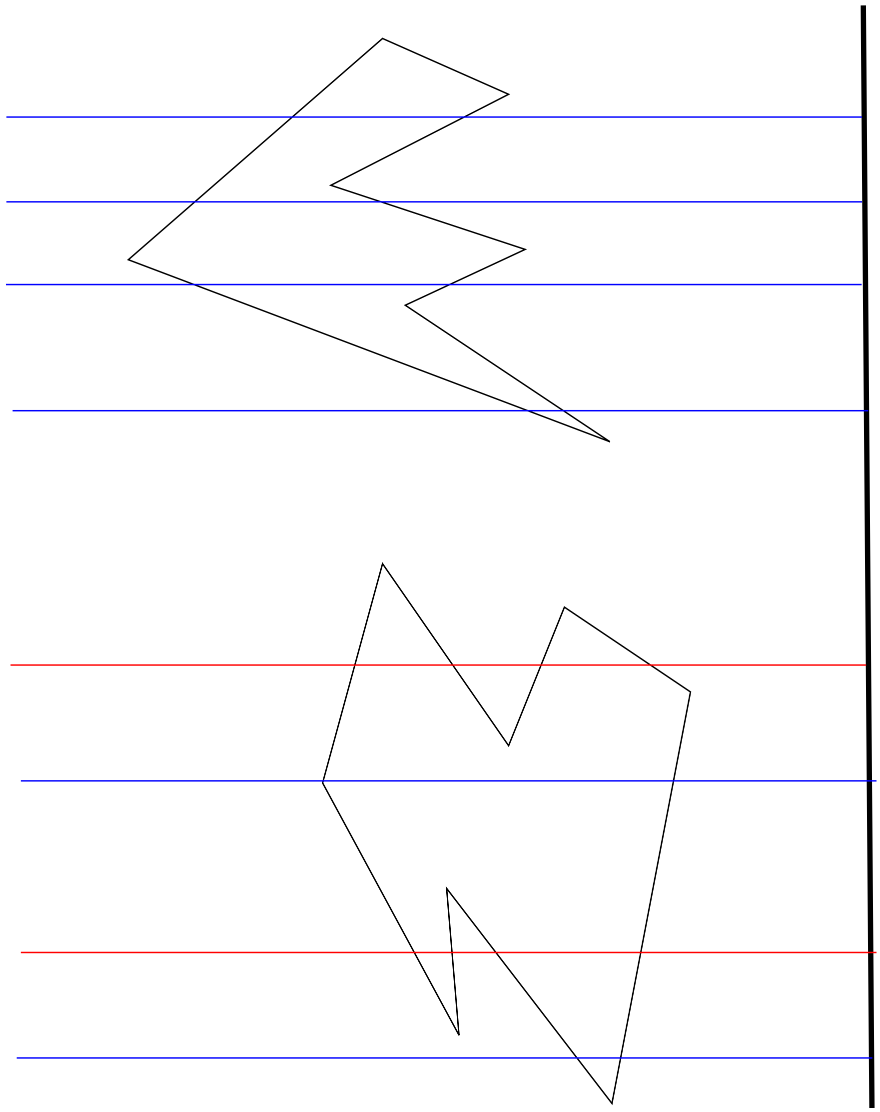


Figure 2: Ciekawsze przypadki. Górny wielokąt jest y-monotoniczny, choć jest wklęsły. Czym różni się od dolnego?

Implementacja powinna być liniowa względem liczby wierzchołków wielokąta na wejściu.

Rysunek 3 przedstawia wizualiację tego algorytmu.

Etap 3 (1 pkt.)

Teraz przedstawimy algorytm na triangulację y-monotonicznego wielokąta P (rysunek pomocniczy 2).

1. Wyznacz wierzchołki o najmniejszej i największej współrzędnej y, p_{ymin} i p_{ymax}.
2. Podziel wierzchołki na dwa fragmenty: $C_1 = (p_{ymin}, \ldots, p_{ymax})$ oraz $C_2 = (p_{ymax}, \ldots, p_{ymin})$. Oznaczmy $C_1 = (p_1, p_2, \ldots, p_t)$ i $C_2 = (q_1, q_2, \ldots, q_s)$. Zatem $p_1 = q_s = p_{ymin}$ i $p_t = q_1 = p_{ymax}$.
3. Wyznacz połączenie dwóch posortowanych list C_1 i C_2 aby otrzymać nową posortowaną listę C_m (po współrzędnej y, od dołu do góry).
4. Przejdź przez C_m i kiedykolwiek po wierzchołku p (lub q) następuje q (lub p) dodaj diagonalę pomiędzy nimi.

W ten sposób dokonamy podziału wielokąta y-monotonicznego na podwielokąty krawędziowidoczne. Dla każdego z nich należy uruchomić procedurę z etapu 2.

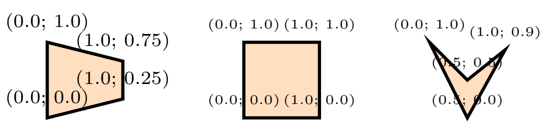
Rysunek 4 przedstawia jak wielokąt y-monotoniczny powinien zostać podzielony. EV_i to kolejne podwielokąty krawędziowidoczne. Krawędzie z których kolejne wielokąty są widoczne to e_1, e_2, e_3, e_4.

Uwaga techniczna: podczas wywoływania funkcji z etapu 2 należy podawać krawędź z której wielokąt jest widoczny tak, aby od `edge_i` dało się przejść przez cały wielokąt (idąc zgodnie ze wskazówkami zegara) aż do `edge_j`.

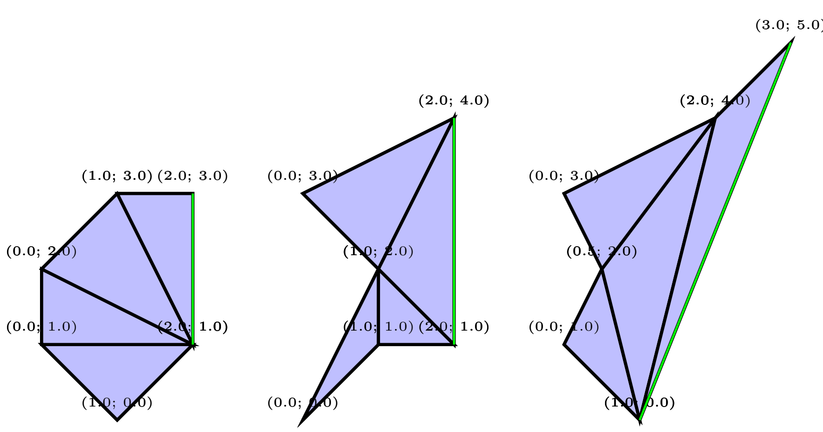
Implementacja powinna być liniowa względem liczby wierzchołków wielokąta na wejściu.

Dane testowe

Etap 1



Etap 2



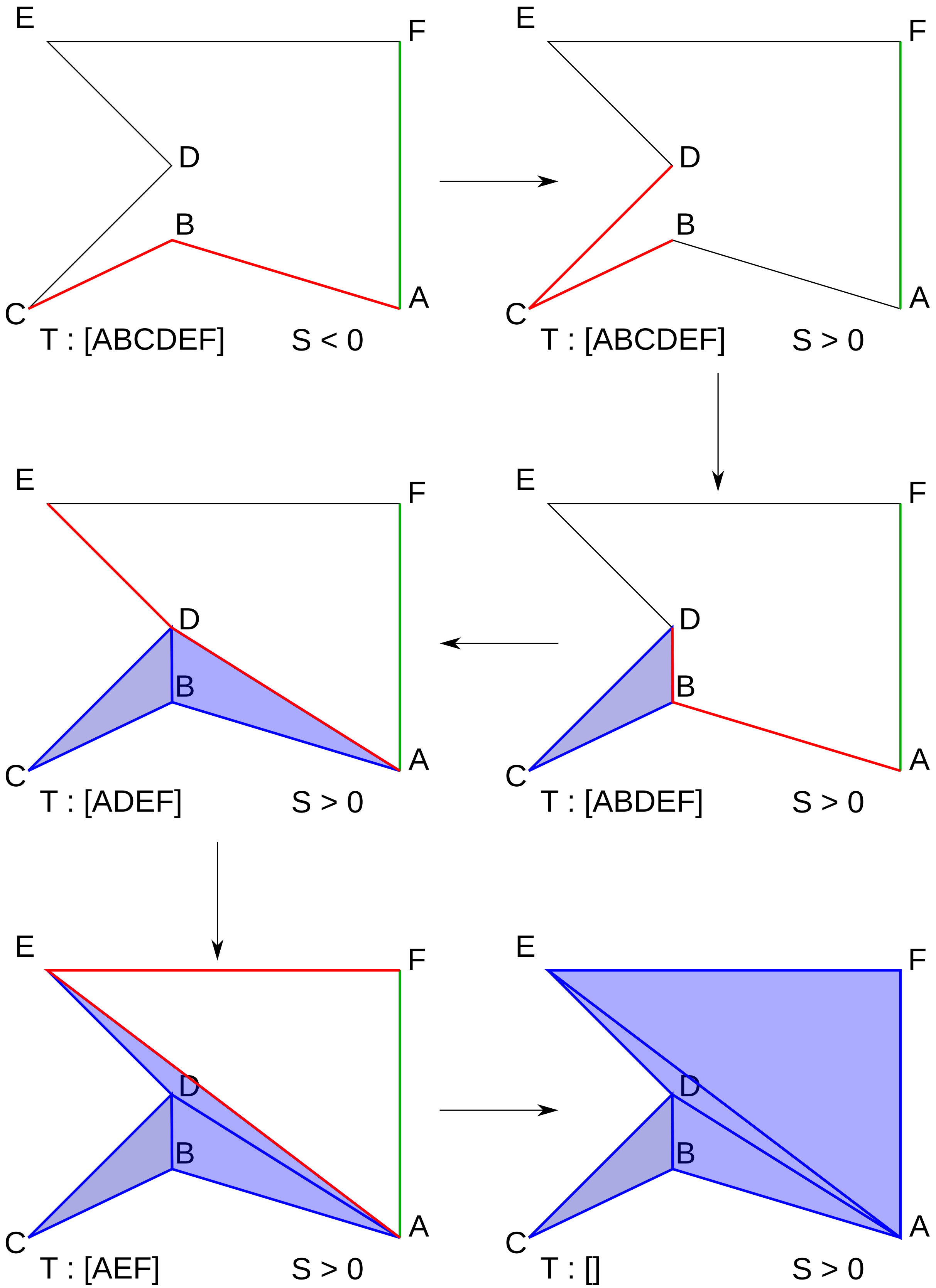


Figure 3: Wizualizacja algorytmu dla triangulacji wielokąta krawędziowidocznego. Zielona krawędź to krawędź uv, z której wielokąt jest widoczny. Czerwona łamana to rozważana trójka punktów p_k, p_{k+1}, p_{k+2}. Niebieskim kolorem zaznaczone są wynikowe trójkąty.

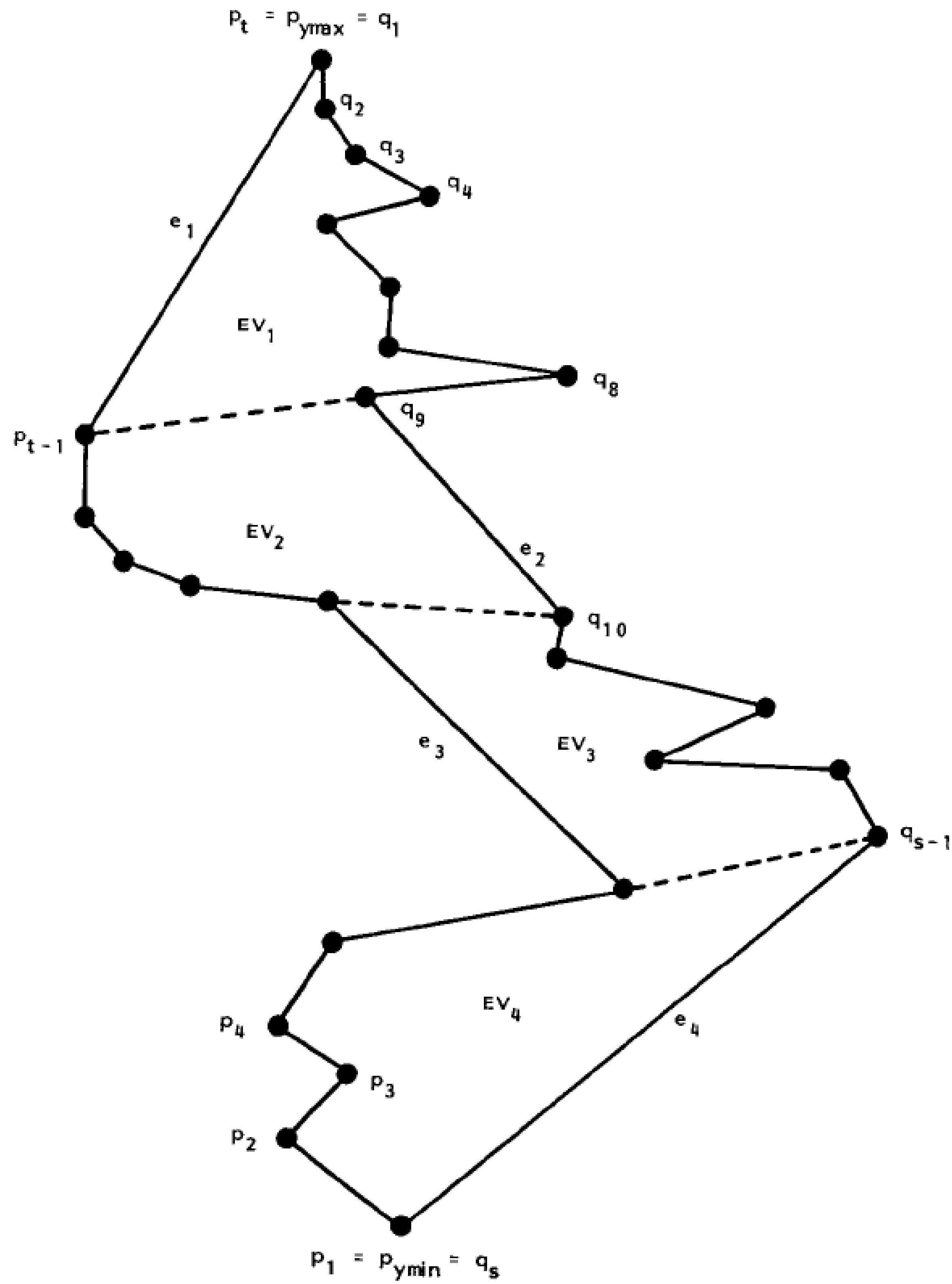
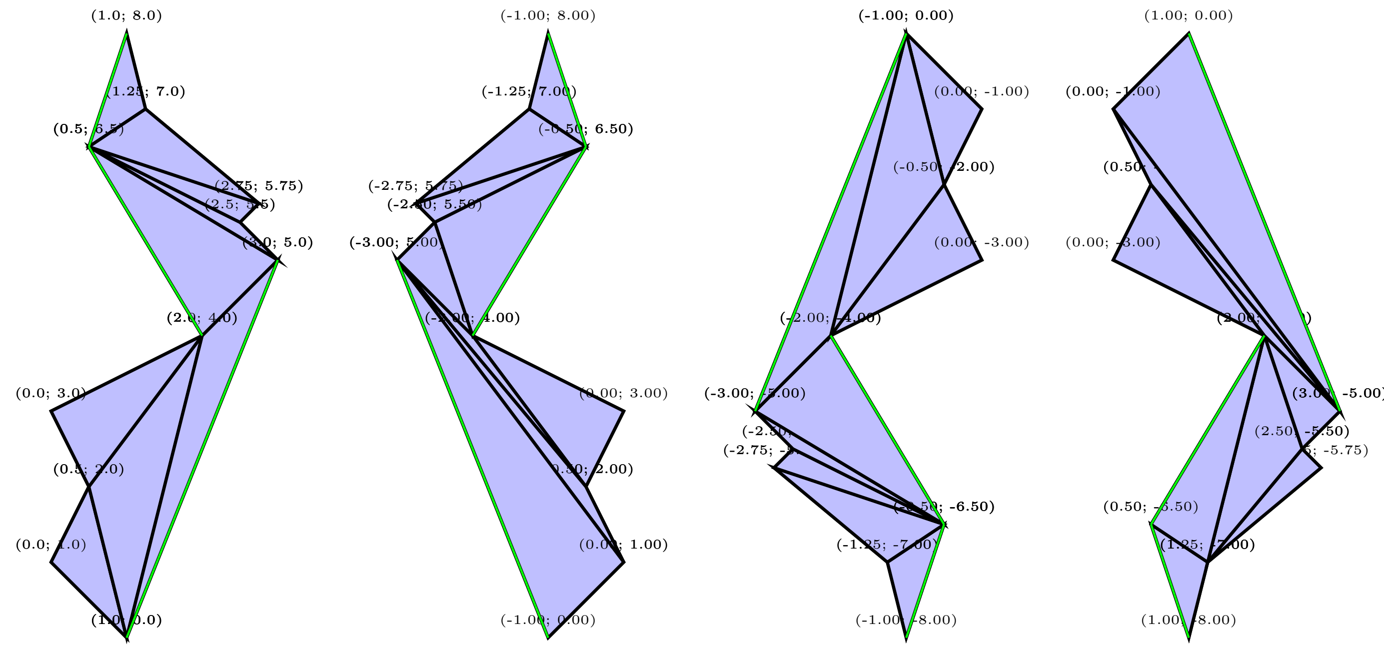


Figure 4: Podział wielokąta y-monotonicznego na wielokąty krawędziowidoczne

Etap 3



(1.0; 8.0)
(1.25; 7.0)
(0.5; 6.5)
(2.75; 5.75)
(2.5; 5.5)
(3.0; 5.0)
(2.0; 4.0)
(0.0; 3.0)
(0.5; 2.0)
(0.0; 1.0)
(1.0; 0.0)
(-1.00; 8.00)
(-1.25; 7.00)
(-0.50; 6.50)
(-2.75; 5.75)
(-2.50; 5.50)
(-3.00; 5.00)
(-2.00; 4.00)
(0.00; 3.00)
(0.50; 2.00)
(0.00; 1.00)
(-1.00; 0.00)
(-1.00; 0.00)
(0.00; -1.00)
(-0.50; -2.00)
(0.00; -3.00)
(-2.00; -4.00)
(-3.00; -5.00)
(-2.75; -5.75)
(-0.50; -6.50)
(-1.25; -7.00)
(-1.00; -8.00)
(1.00; 0.00)
(0.00; -1.00)
(0.00; -3.00)
(3.00; -5.00)
(2.50; -5.50)
(0.50; -6.50)
(1.25; -7.00)
(1.00; -8.00)