Prirodno-matematički fakultet u Sarajevu

Projekt 2

Kompjuterska geometrija

Studentica:

Emilija Zdilar

Mentori:

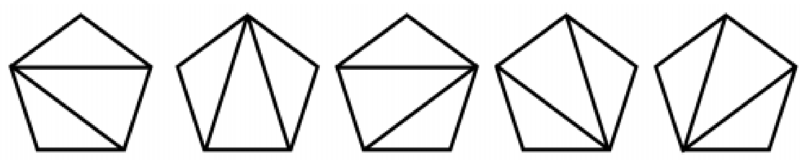
ass. Haris Smajlović

prof. dr. Adis Alihodžić

Sadržaj

Postavka Zadatka

Napisati algoritam koji generira sve moguće triangulacije proizvoljno zadatog konveksnog poligona. Na primjer, za zadati konveksni pravilni petougao algoritam treba da generira sljedeće triangulacije:



Slika 1: izgled rješenja

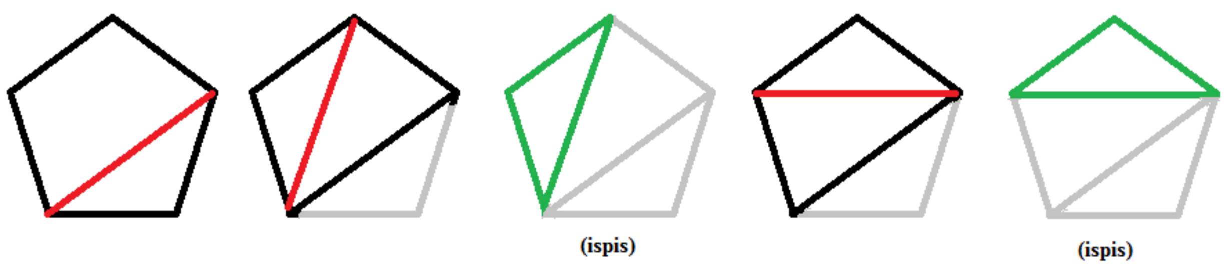
Rješenje

Algoritam za nalaženje svih mogućih triangulacija

Ulaz: skup od n točaka koji definira konveksan poligon.

Izlaz: Sve moguće triangulacije tog poligona.

Ideja se sastoji u tome da se svaka triangulacija poligona može dobiti rekurzivnim *odsjecanjem* rubnih trokuta poligona, sve dok poligon ne svedemo na trokut, a u međuvremenu pamtimo sve dotadašnje dijagonale kojim smo *odsjekli* incidentne trokute.



Slika 1. Prvih nekoliko odsjecanja kod petokuta.

Neka su T_1, T_2 i T_3 prve tri točke tog poligona. Uzmemo za prvu dijagonalu neke triangulacije dijagonalu T_1T_3. Tada sigurno nijedna druga dijagonala tog podskupa triangulacija neće imati T_2 za krajnju točku pa iz tog podskupa mogućih triangulacija izbacujemo T_2. Općenito, označimo te točke sa T_{i-1}, T_i, T_{i+1}. Funkciju pozivamo rekurzivno nad skupom od $n-1$ točke, i to je pozivamo $n-2$ puta, svaki put izbacujući tačku T_i iz skupa tačaka (i dodavajući je nazad nakon što se i-ta rekurzija završi).

Osnova rekurzije: lista se sastoji od samo tri točke.

Nakon što nađemo sve triangulacije nekog podskupa svih triangulacija, vratimo uklonjenu točku i ponavljamo postupak pomjerajući točke za jedno mjesto. Tada:

$$T_{i-1} \text{ -> } T_i, \qquad T_i \text{ -> } T_{i+1}, \qquad T_{i+1} \text{ -> } T_{i+2}$$

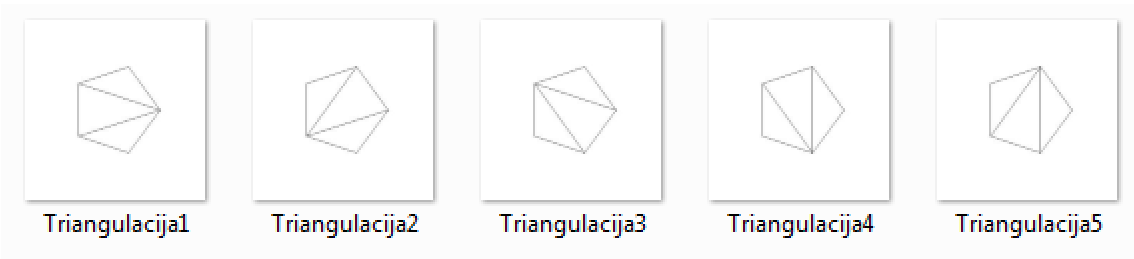
Provjeravamo da li dijagonala $T_{i-1}\,T_{i+1}$ već u skupu. Na taj način nema ponavljanja triangulacija.

Postupak završava nakon što je T_{i+1} bila zadnja zadana točka poligona.

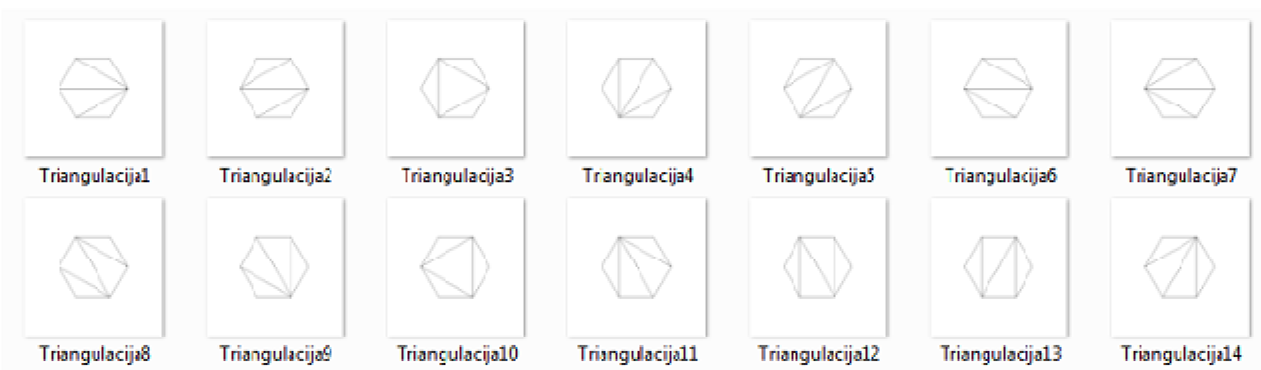
Rezultat se smješta u listu listi dijagonala, koja predstavlja sve moguće triangulacije nekog poligona.

Iscrtavanje svih mogućih triangulacija

Funkcija koja iscrtava sve moguće triangulacije koristi *PIL* – Python Imaging Library. Za svaku moguću triangulaciju kreira sliku tog poligona i odgovarajućih dijagonala te sliku sprema u izlazni direktorij pod rednim brojem kako je nađena.



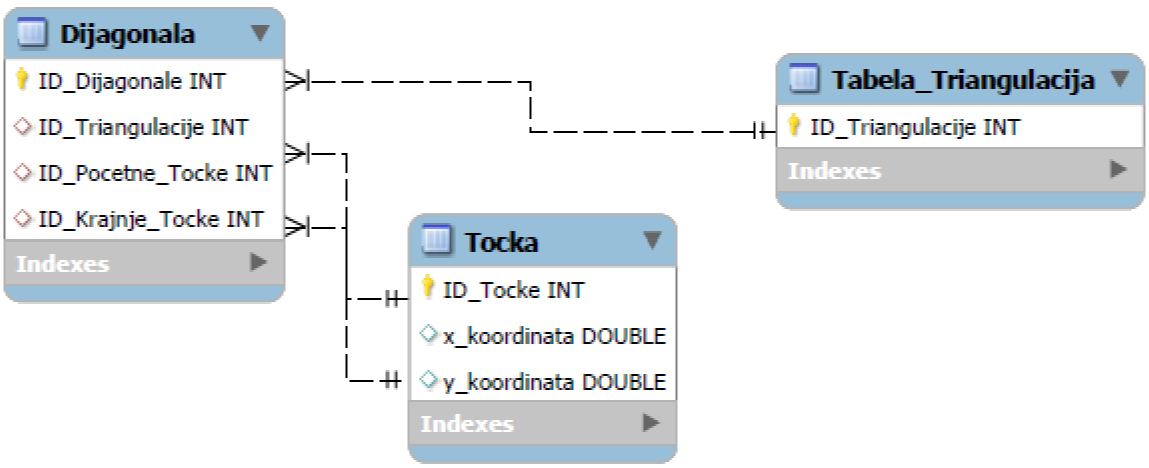
Slika 2: Triangulacije petokuta



Slika 3: Triangulacije šestokuta

Veza sa bazom podataka

Radi preglednosti, baza je implementirana da pohranjuje sve moguće triangulacije jednog nasumičnog poligona, i da se očisti pri svakom novom pozivu. Bazu je lako proširiti da pohranjuje triangulacije različitih poligona dodavanjem tabele poligon, i povezivanjem s tabelom triangulacija vezom *1: n.* Dijagram je sljedeći:



Slika 4: EER dijagram

Baza je kreirana ranije i eksportovana korištenjem *mysqldump*. Korištena je Python biblioteka *peewee* za povezivanje s bazom, kreiranje te brisanje redova. Tabele baze opisane su *peewee* modelima u datoteci *baza.py.*

```
mysql> use Triangulacije;
Database changed
mysql> show tables;
+-------------------------+
| Tables_in_triangulacije |
+-------------------------+
| dijagonala              |
| tocka                   |
| triangulacija           |
+-------------------------+
3 rows in set (0.00 sec)

mysql> select * from Tocka;
+----------+------------------+------------------+
| ID_Tocke | x_koordinata     | y_koordinata     |
+----------+------------------+------------------+
|       15 |              300 |              200 |
|       16 | 230.901699437495 | 295.105651629515 |
|       17 | 119.098300562505 | 258.778525229247 |
|       18 | 119.098300562505 | 141.221474770753 |
|       19 | 230.901699437495 | 104.894348370485 |
+----------+------------------+------------------+
5 rows in set (0.00 sec)

mysql> select * from triangulacija;
+------------------+
| ID_Triangulacije |
+------------------+
|                1 |
|                2 |
|                3 |
|                4 |
|                5 |
+------------------+
5 rows in set (0.00 sec)

mysql> select * from dijagonala;
+---------------+------------------+------------------+------------------+
| ID_Dijagonale | ID_Triangulacije | ID_Pocetne_Tocke | ID_Krajnje_Tocke |
+---------------+------------------+------------------+------------------+
|            50 |                1 |               15 |               17 |
|            51 |                1 |               15 |               18 |
|            84 |                2 |               15 |               17 |
|            85 |                2 |               17 |               19 |
|           118 |                3 |               16 |               18 |
|           119 |                3 |               15 |               18 |
|           152 |                4 |               16 |               18 |
|           153 |                4 |               16 |               19 |
|           186 |                5 |               17 |               19 |
|           187 |                5 |               16 |               19 |
+---------------+------------------+------------------+------------------+
10 rows in set (0.00 sec)
```

Slika 5: DB *triangulacije* nakon poziva funkcije na petokutu.

Vremenska složenost

Označimo li sa C_n Katalanov broj, broj različitih triangulacija za poligon određen s n točaka je C_{n-2}:

$$T_n = C_{n-2} = \frac{(2n-4)!}{(n-1)!\,(n-2)!}$$

Dubina rekurzije je *n-2*. Budući da će se neke grane rekurzije poklapati, a neke će biti prekinute prije nego se dođe do lista, možemo uzeti da imamo onoliko zasebnih grana koliko je triangulacija. Posmatramo li jednu granu rekurzije od korijena do lista, treba nam $O(n^2)$ vremena za izbacivanje elemenata iz niza, a Katalanov broj se može Stirlingovom formulom aproksimirati na:

$$\frac{4^n}{n^2}$$

To znači da je očekivano vrijeme izvršavanja:

$$T(n) = \frac{4^n}{n^2} \cdot O(n^2) = O(4^n)$$

U odnosu na to, *brute-force* metoda bi imala kompleksnost:

$$T(n) = n \cdot T(n-1) = n \cdot (n-1) \cdot T(n-2) = [\dots] = O(n!)$$