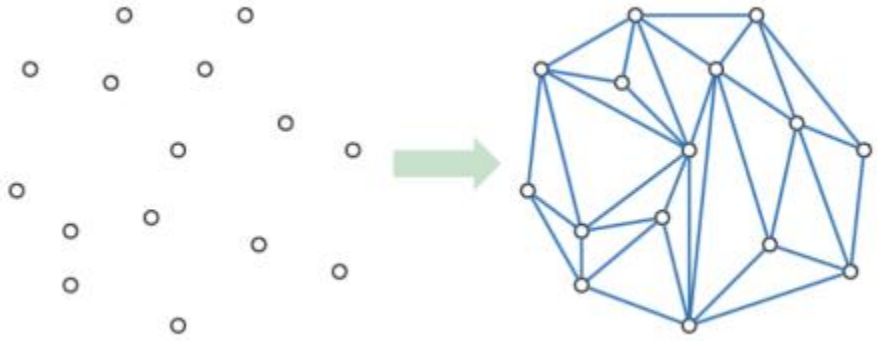
Dokumentacija za projekat iz predmeta Uvod u kompjutersku geometriju

Studentica: Iman Mahmutović
Broj indexa: 5848/M
Tema: Rekurzivna triangulacija skupa tačaka (Tema 2)

Problem

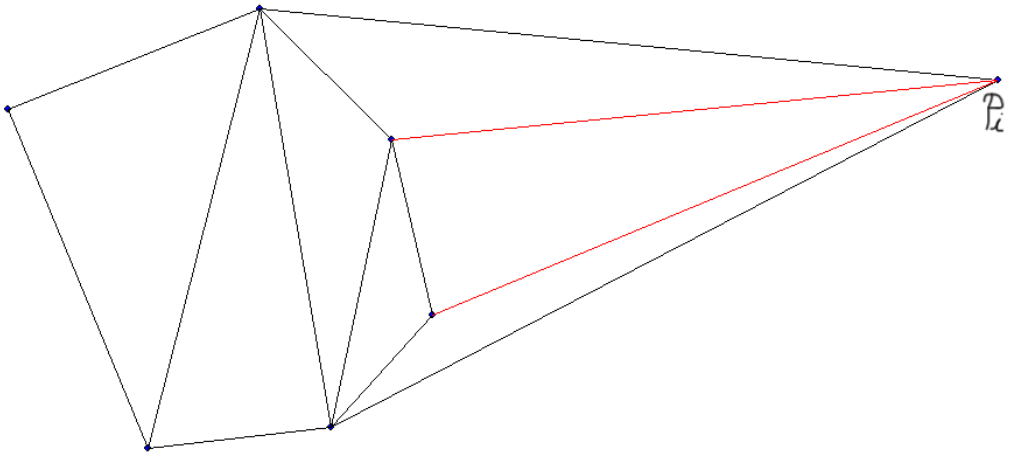
Neka je P skup od n tačaka u ravni. Triangulacija P podrazumijeva pronalazak konveksnog omotača tačka iz P, a onda triangulaciju tog konveksnog poligona koristeći sve tačke iz P kao vrhove trouglova. Implementirati algoritam za triangulaciju skupa tačaka P koji radi u vremenu $O(\mathrm{n}^2)$. Algoritam treba raditi rekurzivno, dodavajući jednu po jednu tačku u strukturu (struktura predstavlja triangulaciju prethodno dodatih tačaka). Kod dodavanja nove tačke razlikujemo slučajeve kada se nova tačka nalazi u konveksnom omotaču postojećih tačaka i kada se ne nalazi. Obavezno uraditi i vizualizaciju. Bonus 3 boda se može osvojiti ako se uradi implementacija u vremenu $O(\mathrm{n} \log n)$.



Rješenje

U ovom projektu sam implementirala funkciju *triangulacijaTacakaRekurzivno (vector<Tacka>, vector<Duz> &).* Ova funkcija koristi sljedeće pomoćne funkcije koje su već implementirane na vježbama: *Orijentacija(Tacka, Tacka, Tacka)* i *nadjiTangente(Tacka, vector<Tacka>&).* Funkciju *nadjiTangente(Tacka, vector<Tacka>&)* sam izmijenila da koristi strukturu deque za konveksni omotač umjesto vektora. Razlog ovome je što će se u glavnoj funkciji često dodavati i brisati tačke sa konveksnog omotača i vrijeme izvršavanja tih operacija je brže kod deque-a nego kod vekotra. Na samom početku sortiramo tačke na osnovu x-koordinate i konstruišemo prvi konveksni omotač: trougao sastavljen od prve tri tačke. Onda iteriramo kroz ostale tačke, pri čemu smo sigurni da je svaka nova tačka van dosadašnjeg konveksnog omotača jer smo na početku sortirali tačke. Kada dodajemo novu tačku p_i, konstruišemo gornju i donju tangentu iz te tačke na dosadašnji omotač. U triangulaciju dodajemo te tangente i duž/i koje dobijemo tako što

spojimo tačku p_i sa svim tačkama između gornje i donje tangente (ako takve tačke postoje). Na sljedećoj slici vidimo da smo p_i spojili sa konveksnim omotačem pomoću dvije tangente i još dvije duži između tih tangenti koje su označene crvenom bojom.



Potrebno je još i obrisati tačke između dviju tangenti iz konveksnog omotača i dodati tačku p_i u konveksni omotač. Vidimo da u svakom koraku čuvamo konveksni omotač dosad obrađenih tačaka kao i njihovu triangulaciju. Algoritam je sličan algoritmu koji induktivno konstruiše konveksni omotač samo što ovdje imamo sortirane tačke i dodatno još čuvamo triangulaciju tačaka.

Vremenska i prostorna složenost algoritma

Sortiranje tačaka ima složenost $O(\mathrm{n}\log n)$. Za $n - 3 \approx n$ tačaka, određujemo tangente na omotač. Za svaku tačku nam treba $O(\log n)$ vremena pa nam treba $O(\mathrm{n}\log n)$ vremena za određivanje svih tangenti. Dodavanje i brisanje elemenata iz deque-a je linearno u odnosu na broj elemenata koje dodajemo/brišemo a u algoritmu se izvodi ukupno $O(\mathrm{n})$ dodavanja i ukupno $O(\mathrm{n} - \mathrm{h})$ brisanja (h je broj tačaka na omotaču), što ne utiče na ukupnu vremensku složenost. Dakle ukupno vrijeme izvršavanja algoritma je $O(\mathrm{n}\log n)$. Da smo koristili vektor za konveksni omotač, radi skupih operacija brisanja i dodavanja, ukupno vrijeme izvršavanja bi bilo $O(\mathrm{n}^2)$.

Prostorna složenost algoritma je $O(n)$. Razlog tome je što se u algoritmu koriste sljedeće strukture podataka:

- Vektor *tacke*: Ova struktura podataka pohranjuje sve tačke u skupu. Njena veličina je $O(n)$.

- Deque *konveksni_omotac*: Ova struktura podataka pohranjuje tačke konveksnog omotača. Njena veličina varira tokom izvršavanja algoritma, ali u najgorem slučaju može biti $O(n)$.
- Vektor *stranice*: Ova struktura podataka pohranjuje sve stranice triangulacije. Njena veličina je $O(3n - h - 3) \approx O(n)$ gdje je h broj tačaka na konveksnom omotaču.

Ostale varijable ne utiču na ukupnu prostornu složenost.