Shortest Path Tree in Unit Disc Graph

Opis implementacije projekta u okviru kursa
Geometrijski algoritmi
Matematički fakultet

Anđela Bašić, 1018/2023
basicandjela1999@gmail.com

1. septembar 2024.

Sažetak

Projekat se bavi implementacijom rešenja problema nalaženja najkraćih puteva od izvora do svih temena u beztežinskom grafu jediničnih diskova. Implementacija je slična opisu zadatom pseudo-kodom u radu *Shortest paths in intersection graphs of unit disks* objavljenom 2015. godine u časopisu *Journal of Computational Geometry*, te je vreme izvršavanja $O(n \log n)$ [1].

Sadržaj

1 Uvod **1**

2 Naivni Algoritam **2**

3 Glavni Algoritam **2**
3.1 Delone graf 3
3.2 Kd-stablo i nalaženje najbližeg suseda 4

4 Testovi **4**

Literatura **4**

1 Uvod

Problem (Shortest Path Tree in Unit Disc Graph)**.** *Dato je n tačaka u prostoru i jedna od tih tačaka je izvor S. Graf jediničnih diskova je takav da su svake dve tačke povezane ivicom na rastojanju manjem od 1. Drugim rečima ivica postoji ako se jedinični diskovi seku sa centrima u datim temenima seku. Treba odrediti najkraći put od S do svih temena u tom grafu, uz pretpostavku da je on beztežinski.*

Radi uspešne vizuelizacije u Qt-u, problem je izmenjen tako da posmatramo graf diskova poluprečnika 100.

Glavna klasa je **UnweightedShortestPathTree** koja nasle]uje klasu AlgoritamBaza i implementira njene virtuelne metode.

Polja klase su:

- **vector<Vertex_SPT*> P** - Niz temena, inicijalizuje se nasumično ili iz datoteke u konstruktoru pozivanjem odgovarajućih metoda natklase.
- **Vertex_SPT *s** - Izvor, inicijalizuje se nasumično ili iz datoteke u konstruktoru pozivanjem odgovarajućih metoda natklase..
- **map<Vertex_SPT*, set<Vertex_SPT*» delaunayGraph** - Delone graf koji se inicijalizuje metodom klase **void inicijalizujDeloneGraf()**
- **map<Vertex_SPT*, set<Vertex_SPT*» unitDiscGraph** - Graf jediničnih diskova koji se inicijalizuje metodom klase **void inicijalizujUnitDiscGraf()**
- **map<Vertex_SPT*, pair <int, Vertex_SPT*» resultPathGlavni** - Mapa najkraćih puteva od izvora koja se formira metodom klase **void pokreniAlgoritam() final**
- **map<Vertex_SPT*, pair <int, Vertex_SPT*» resultPathNaivni** - Mapa najkraćih puteva od izvora koja se formira metodom klase **void pokreniNaivniAlgoritam() final**

2 Naivni Algoritam

U okviru naivnog algoritma se prvo određuje graf jediničnih diskova u vremenu $O(n^2)$. Graf se predstavlja preko mape susednih čvorova. Nakon toga se rekurzivno prolazi kroz graf počevši od S i ažuriraju se rastojanja od S i prethodni čvor na putu. Naivni algoritam je složenosti $O(n^2)$.

Metode koje se tiču naivnog algoritma su:

- **void inicijalizujUnitDiscGraf()** - Incijalizacija polja unitDiscGraph neophodnog za sprovođenje naivnog algoritma.
- **void pokreniNaivniAlgoritam()** - Metod iz natklase.
- **void crtajNaivniAlgoritam(QPainter *painter) const** - Metod iz natklase.

3 Glavni Algoritam

U okviru glavnog algoritma opisanog u radu, izbegava se eksplicitno računanje grafa jediničnih diskova, već se za utvrđivanje postojanja određene ivice koristi provera o rastojanju temena (manje od 1 u radu, manje od 100 u implementaciji).

Koraci u rešavanju problema su:

1. Određivanje delone grafa skupa tačaka $DG(n)$
2. Inicijalizujemo $W_0 = \{S\}$, $i = 1$. W_i je skup temena koja su na udaljenosti i od S.

3. Dok god W_{i-1} nije prazan
   (a) Određuju se temena na rastojanju i od S i dodaju u W_i. Kandidati za koje proveravamo da li ih treba dodati u W_i se čuvaju u skupu Q, koji se inicijalizuje skupom W_{i-1}.
   (b) Dok god Q nije prazan, za svaku tačku kandidata q iz Q, sprovodimo proveru nad svim njegovim susedima p u $DG(n)$, i potom ga brišemo iz Q. Provera se sastoji od nalaženja najbližeg temena w temenu p u W_{i-1} i potom provere udaljenosti temena w i p (treba da bude manja od 1, odnosno 100). Ukoliko je uslov zadovoljen, p se dodaje u rezultat tako da je udaljenost od S postavljena na i, a prethodno teme na tom putu je q. Takođe, doajemo p u Q.
4. Uvećavamo i

Složenost opisanog algoritma zavisi od načina određivanja najbližeg temena iz skupa W_{i-1} kandidatu. U pseudo-kodu nije precizirano kako ova struktura treba tačno da izgleda, ali je dat predlog da prvo možemo odrediti Voronoj dijagram nad skupom tačaka W_{i-1}, a potom odrediti najbliže teme kao teme ćelije u kojoj se tačka upita nalazi. Ova struktura bi, na primer, mogla da bude neka od poznatih struktura za lociranje tačke, npr. trapezoidno razlaganje. Ovako opisan algoritam je složenosti $O(n \log n)$.

U ovom projektu struktura za traženje najbližeg temena iz skupa W_{i-1} je implementirana kao kd-stablo, jer rezultira istom vremenskom složenošću algoritma, a jednostavnija je za implementaciju.

Dakle, uz implemetaciju glavnog algoritma potrebno je implementirati kd-stablo i funkciju za pronalaženje najbližeg suseda, kao i Delone graf. Metode koje se tiču glavnog algoritma su:

- **void inicijalizujDeloneGraf()** - Incijalizacija polja delaunayGraph neophodnog za sprovođenje glavnog algoritma.
- **void pokreniAlgoritam()** - Metod iz natklase.
- **void crtajAlgoritam(QPainter *painter) const** - Metod iz natklase.

3.1 Delone graf

Kod za implementaciju Delone grafa je preuzet sa sajta asistentkinje Jelene Marković. Kod je prepravljen tako da radi korektno nad testiranim podacima. Izmenjne su naredne funkcije:

- **void DCEL_TRIANGLE::add_point_in_triangle(Face* triangle, Vertex* v)** - Prepravljeni su neki pokazivači tako da uvezivanje bude pravilno.
- **void DCEL_TRIANGLE::add_point_on_edge(Face* triangle, Edge* e, Vertex* v)** - prepravljeni su neki pokazivači tako da uvezivanje bude pravilno
- **void DelaunayUtil::legalize_edge(Vertex* v, Edge* e)** - Obrisan uslov za legalizovanje granica.
- **bool Edge::isLeft(Vertex* v) const** - Obrisani specifični uslovi koji se tiču graničnih stranica, i ostavljen je samo opšti uslov.
- **bool Face::circumCircleContains(Vertex_SPT* v)** - Izmenjen postupak tako da se koristi Gausova jednačina za površinu pri izračunavanju poluprečnika i centra opisanog kruga.

Dodate su funkcije:

- **Face* DelaunayUtil::getSuperTriangle(std::vector<Vertex_SPT*> &in_points)** - Pronalaženje početnog trougla u algoritmu za iračunavanje Delone grafa.
- **void UnweightedShortestPathTree::inicijalizujDeloneGraf()** - Rekurzivna funkcija za određivanje delone grafa kao mape susednih temena u Delone triangulaciji radi lakšeg sprovođenja glavnog algoritma.

Kod je organizovan tako da su sve funkcije koje ne pripadaju klasama Vertex_SPT, Edge, Face ili DCEL_TRIANGLE, a tiču se Delone triangualcije, smeštene u pomoćnu klasu DelaunayUtil.

3.2 Kd-stablo i nalaženje najbližeg suseda

Kd-stablo je implementirano pomoću strukture KdNode i klase Kdtree. Klasa Kd-tree objedinjuje dve glavne funkcije u ovom segmentu:

- **KdNode* buildKdTree(vector<Vertex_SPT*>& vertices, int start, int end, int depth)** - Rekurzivno gradi kd-stablo tako što odabira medijanu tačaka po trenutnoj dimenziji deljenja i kreira levo i desno podstablo.
- **Vertex_SPT findNearestNeighbor(KdNode* root, const Vertex_SPT& query, double& minDist)** - Izvršava pretragu najbližeg suseda rekurzivnim pretraživanjem kd-stabla i ažurira najbližeg suseda i minimalnu udaljenost.

4 Testovi

Kod je testiran kroz četiri ulazna fajla input1.txt - input4.txt koja takođe služe za pokretanje algoritma kroz ulaz iz datoteke.

Literatura

[1] Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48(4):360–367, 2015.