Algoritmul lui KIRKPATRICK

Mircea Digulescu, Andrei Matei

Problema localizării punctelor este una dintre problemele pentru care, deși sunt foarte ușor de rezolvat de către ochiul uman, mult timp nu s-a găsit un algoritm optim. În acest articol vom prezenta un algoritm care folosește un spațiu de memorie proporțional cu O(n) și răspunde la interogări în timp de O(n log n).

Enunțul problemei

Enunțul problemei localizării punctelor (în plan) este următorul:

"*Se consideră o împărțire (divizare) a planului în celule poligonale. Dându-se un punct din plan, se cere să se precizeze în care dintre celule se află acesta.*"

Este ciudat că, deși pare o problemă simplă pe care ochiul uman o poate rezolva în mod natural, a durat mult să se ajungă la un algoritm de rezolvare care ocupă un spațiu de memorie $O(n)$ și este capabil să răspundă la o interogare într-un timp $O(\log n)$.

Primul care a dezvoltat un astfel de algoritm a fost *David G. Kirkpatrick*.

Prezentarea algoritmului

Pentru a se putea răspunde eficient la interogări este necesar ca celulele poligonale să fie preprocesate.

Acest algoritm pleacă de la supoziția că divizarea planului este o triangulare. În plus, trebuie ca și exteriorul divizării să fie triunghi. Pentru cazul general, vom începe prin a triangula fiecare celulă a divizării și a marca fiecare triunghi rezultat cu eticheta celulei din care face parte. Pentru a adăuga "exteriorul", vom mai adăuga trei puncte distincte cu coordonate "infinite" (se poate demonstra foarte ușor că orice punct din plan se află în interiorul triunghiului format de cele trei puncte) și vom uni aceste puncte (a, b, c în figura 1) cu puncte de pe înfășurătoarea convexă a diviziunii.

Deși s-ar putea crede că triangulând adăugăm multe muchii noi, se poate demonstra ușor că, în cel mai rău caz, numărul de muchii crește cu un factor constant.

Se observă că, dacă găsim triunghiul care conține punctul, atunci cunoaștem și celula din divizarea inițială.

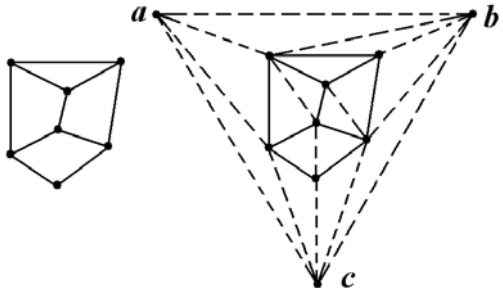


Figura 1: Triangularea unei diviziuni

Fie T_0 triangularea inițială. Un algoritm bazat pe *metoda lui Kirkpatrick* va produce o secvență de triangulări, T_0, T_1, T_2, ..., T_k (cu $k = O(\log n)$) astfel încât T_k constă într-un singur triunghi (cel format de cele trei puncte cu coordonate infinite din T_0) și fiecare triunghi din T_{i+1} se suprapune peste un număr constant de triunghiuri din T_i.

Vom vedea cum folosim o astfel de structură la interogări. În continuare construim această structură.

Presupunem că avem triangularea T_i și vrem să construim T_{i+1}. Pentru a fi siguri că acest proces se oprește după $O(\log n)$ pași ne vom asigura că numărul de triunghiuri din T_{i+1} este mai mic decât numărul de triunghiuri din T_i cu un factor constant. Fiecărei triangulări îi vom atașa un graf, astfel încât fiecare punct este un nod și există muchii între două noduri, dacă există segment între punctele corespunzătoare din triangulare. Din acest punct al rezolvării, prin T_i vom înțelege, în funcție de context, graful asociat triangulării T_i sau chiar triangularea.

La fiecare pas i vom alege o submulțime de noduri din T_i pe care le vom elimina împreună cu muchiile adiacente lor. După ce le eliminăm vom retriangula graful pentru a obține T_{i+1}. Nu se vor elimina niciodată punctele suplimentare cu coordonatele infinite.

Apare întrebarea "*cum alegem nodurile care se elimină din T_i astfel încât fiecare triunghi din T_{i+1} să se suprapună peste un număr constant de triunghiuri din T_i*".

Pentru a alege nodurile care vor fi eliminate vom ține cont de:

- ***grad maxim*** - fiecare nod pe care îl eliminăm să aibă gradul maxim egal cu o constantă d (să fie incident la maxim d muchii) și când eliminăm un astfel de nod, *gaura* rezultată va fi triangulată în T_{i+1} în maxim d - 2 triunghiuri; după retriangulare, fiecare triunghi nou se va suprapune peste maxim d triunghiuri din precedenta triangulare.
- ***set independent*** - între oricare două noduri care sunt eliminate să nu existe muchie (nodurile eliminate formează un set independent în T_i); această restricție ne va folosi la retriangulare; când eliminăm în acest fel m noduri diferite obținem m găuri disjuncte pe care le putem triangula independent; mai mult, deoarece fiecare gaură conține un număr constant de noduri, acestea se vor triangula în $O(1)$.

O problemă importantă este dacă vom reuși sau nu întotdeauna să găsim un set independent, suficient de mare cu grad limitat. Vrem ca mărimea acestui set să fie cel puțin o fracțiune constantă din numărul curent de noduri. S-a descoperit că există întotdeauna un astfel de set. Important este să alegem o valoare pentru d suficient de mare (dacă d este prea mic, nu vom găsi suficiente noduri). Vom demonstra că $d = 8$ este suficient de mare.

Lemă: *Dându-se un graf planar (cum este cazul triangulărilor noastre) cu n noduri, există un set independent de noduri cu grad maxim 8, cu cel puțin n/18 noduri. Acest set poate fi determinat într-un timp O(n).*

Vom arăta cum construim acest set și pe urmă vom demonstra mărimea sa:

marcăm toate nodurile care au gradul ≥ 9
cât timp există un nod nemarcat:
  alegem un astfel de nod v
  adăugăm v la set
  îl marcăm pe v și pe toți vecinii lui

Se observă că acest algoritm se termină în $O(n)$.

Demonstrația faptului că setul rezultat va avea cel puțin $n/18$ noduri este următoarea: o formulă fundamentală a lui *Euler* ne spune că într-un graf planar cu n noduri complet triangulat numărul e de muchii satisface relația $e = 3 \cdot n - 6$. Dacă adunăm gradele nodurilor, fiecare muchie este numărată de două ori.

Cum $\sum_v grad(v) = 2e = 6n - 12 < 6n$, o aproximare inferioară a mediei gradelor este 6. Vom demonstra în continuare că există cel puțin $n/2$ noduri cu gradul 8 sau mai mic. Să presupunem contrariul, și anume că există mai mult de $n/2$ noduri de grad 9 sau mai mare. Cum celelalte noduri trebuie să aibă gradul mai mare sau egal cu 3 (cu excepția celor trei noduri cu coordonatele infinite, adăugate artificial), rezultă că suma gradelor în graf este $9 \cdot n/2 + 3 \cdot n/2 + 6$, ceea ce contrazice ecuația de mai sus.

Când algoritmul de mai sus începe, cel puțin $n/2$ noduri sunt inițial nemarcate. De fiecare dată când selectăm un astfel de nod, din cauză că gradul său este 8 sau mai mic, marcăm cel mult 9 noduri noi (pe el însuși și cel mult 8 vecini). Deci acest pas poate fi repetat de cel puțin $(n/2)/9 = n/18$ ori, sau până când rămânem fără noduri nemarcate.

În acest moment demonstrația este completă.

Acum știm că putem construi un set convenabil cu cel puțin $n/18$ noduri. Numărul 18 pare destul de mare, dar în practică va fi mai mic. Totuși această constantă este un motiv pentru care algoritmul lui *Kirkpatrick* nu este folosit foarte des în practică.

Structura lui Kirkpatrick

Acum să vedem efectiv cum este construită structura de localizare (structura lui *Kirkpatrick*) . Începem cu T_0 și la fiecare pas selectăm un subset independent de noduri cu grad minim 8 pe care le eliminăm. Retriangulăm găurile obținute, cu observația că fiecare triunghi nou se suprapune peste maxim 8 triunghiuri din triangularea precedentă. Cu fiecare triangulare, numărul de noduri scade cu un factor de maxim 17/18. Pentru a ajunge la o triangulare alcătuită dintr-un singur triunghi, cel de la infinit, avem $(18 / 17) \cdot k = n$ sau $k = \log_{18/17 \cdot n} n \approx 12 \cdot \lg n$.

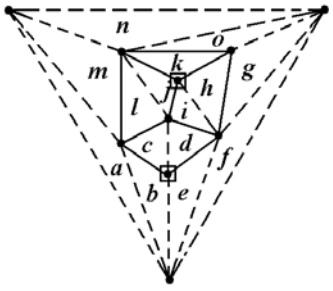
Structura de date se bazează pe aceste triangulări. Rădăcina structurii corespunde singurului triunghi din T_k. Nodurile de la următorul nivel sunt triunghiurile din T_{k-1}, urmate de cele din T_{k-2} și așa mai departe până la frunze, care sunt triunghiurile din T_0, triangularea inițială. Fiecare nod de pe nivelul i reține referințe la triunghiurile peste care se suprapune din nivelul $i + 1$ (sunt cel mult 8).

Pentru a găsi triunghiul în care se află un punct începem de la rădăcină. Dacă punctul nu se află în triunghiul asociat rădăcinii, am terminat; punctul nu se află în interiorul nici unui triunghi din triangularea inițială. Altfel, testăm dacă punctul e inclus în fiecare din cele cel mult 8 triunghiuri de pe nivelul următor care sunt incluse în triunghiul curent. Și așa mai departe până ajungem la frunze.

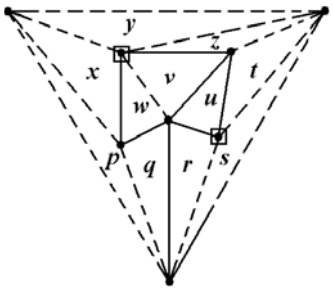
Construirea și analiza

Structura are $\log n$ niveluri și ne ia un timp constant să ne mutăm de la un nivel la următorul (maxim 8 teste de incluziune în triunghi), deci timpul unei interogări este $O(\log n)$. Dimensiunea structurii este egală cu suma mărimilor triangulărilor.

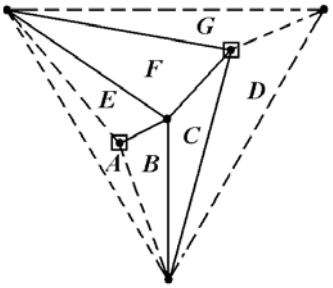
Cum numărul de triunghiuri al unei triangulări depinde liniar de numărul de puncte, rezultă că mărimea structurii este proporțională cu $n \cdot (1 + 17/18 + (17/18)^2 + (17/18)^3 + \ldots) \leq 18 \cdot n$. Deci $O(n)$ înmulțită cu o constantă destul de mare.



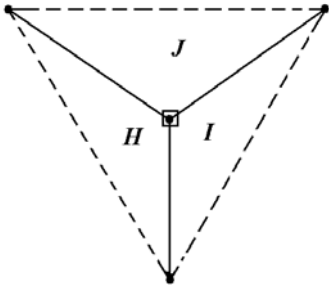
a) Triangularea inițială T_0



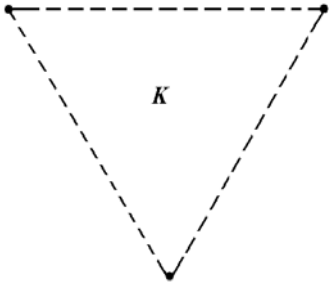
b) Triangularea T_1



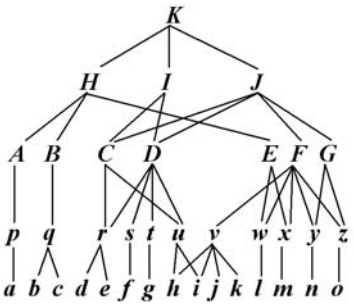
c) Triangularea T_2



d) Triangularea T_3



e) Triangularea T_5



f) Structura folosită pentru interogări

Figura 2: Construirea structurii folosită pentru interogări. Punctele încadrate sunt eliminate la pasul curent.

În figura 2 este prezentat modul în care se realizează construirea structurii care va fi folosită pentru interogări pentru exemplul din figura 1, prin eliminări de puncte și triangulări repetate.

Se poate observa că cea de-a cincea triangulare, și ultima de altfel, este echivalentă cu triunghiul format de punctele suplimentare pe care le-am adăugat.

Dacă dorim să vedem în care celulă se află un anumit punct, atunci vom determina mai întâi triunghiul din T_1 din care face parte și apoi, în funcție de acest triunghi, vom determina celula (poligonul) din care acesta face parte.

În figura 2 f) se află structura pe care am construit-o și care va fi folosită la interogări pentru exemplul din figura 1. Fiecare nod din această structură reprezentând un triunghi, a fost conectat de nodurile, care reprezintă alte triunghiuri, peste care acesta se suprapune.

Bibliografie

1. **David G. Kirkpatrick**, *Optimal Search in Planar Subdivisions*, SIAM Journal on Computing, Vol. 12
2. **Daniel Dominic Sleator, Robert Endre Tarjan**, *A Data Structure for Dynamic Trees*, Symposium on Theory of Computation (1981), pp. 114-122
3. **Franco P. Preparata**, *A New Approach to Planar Point Location*, SIAM Journal on Computing, Vol. 10 (1981), Articolul 3 , pp. 473 - 482
4. **Herbert Edelsbrunner, Lionidas J. Guibas, Jorge Stolfi**, *Optimal point location in a monotone subdivision*, SIAM Journal on Computing, Vol. 15 (1986), Articolul 2, pp. 317 - 340
5. **Michael Ian Shamos**, *Geometric complexity*, In Conference Record of Seventh Annual ACM Symposium on Theory of Computation, pp. 224 - 233
6. **Richard J. Lipton, Robert Endre Tarjan**, *Application of a Planar Separator Theorem*, FOCS 1977, pp. 162-170

Mircea Digulescu și Andrei Matei sunt elevi în clasa a XII-a la Colegiul Național de Informatică Tudor Vianu din București și pot fi contactați prin e-mail la **mircea85@yahoo.com**, *respectiv* **andreimatei@home.ro**.