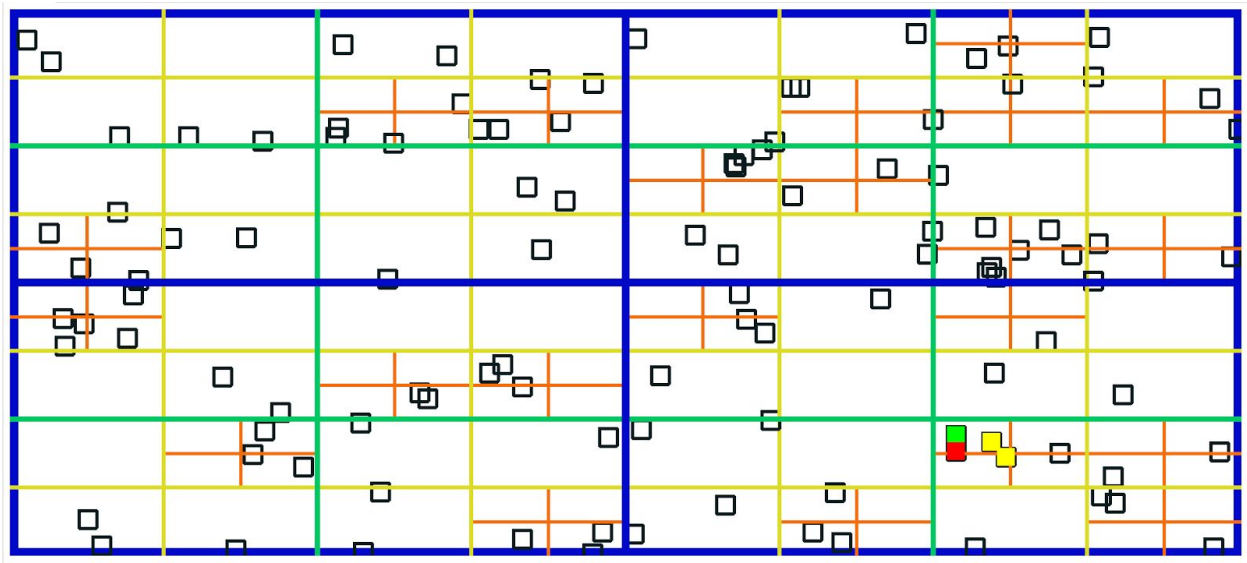
Geometrijski algoritmi @ MATF

Algoritam za određivanje kolizija korišćenjem strukture Quadtree

Nikola Dimitrijević



Opis problema

Za date kvadrate odrediti koji od njih imaju preseke sa nekim od ostalih kvadrata.

Pod presekom ćemo podrazumevati i slučaj kada se samo dodiruju.

Ulaz: *skup od n tačaka u ravni koje predstavljaju gornje levo teme kvadrata, i dužina stranice*

Izlaz: *broj kvadrata koji se seku sa nekim drugim kvadratom*

Naivno rešenje problema

Naivno rešenje je da za svaki element je lako implementirati. Proverimo preklapanje svakog elementa sa svim ostalim elementima. Složenost ovakvog algoritma je $O(n^2)$ gde je n broj elemenata. Sledi iz broja parova koje treba proveriti, a to je n * (n - 1) / 2.

Pseudokod:

```
For i = 0 to numElements
      For j = i + 1  to numElements
              If  element[i] intersects element[j]
                        addIntersection(i, j);
```

Optimalni algoritam

Optimalni algoritam koristi strukturu Quadtree. Quadtree je stablo čiji unutrašnji čvorovi uvek imaju četiri potomka. U listovima se čuvaju elementi.

Rekurzvnom podelom ravni se produbljuje Quadtree.

Kada unosimo element u Quadtree onda pitamo, počevši od korena stabla, u koji kvadrant treba da ubacimo element. Na taj način izaberemo jednog od potomka, i tako rekurzivno dok ne naiđemo na list, tada dodamo element u niz elemenata koji pripadaju tom listu, i proverimo presek sa svim elementima tog lista.

Tako se radi u idealnom slučaju, ali često imamo elemente koji pripadaju više od jednom kvadrantu, pa ćemo ih tada dodeliti svim listovima čije oblasti ga seku.

Definišu se parametri Quadtree-a:

Maksimalna dubina stabla,

Maksimalan broj elemenata koje list može da čuva pre nego što se list pretvori u unutrašnji čvor i njegovi elementi raspodele u 4 nova lista.

Maksimalan broj elemenata u listu je razumno da bude oko četiri zato što se tada očekuje da ćemo moći da ih razvrstamo u različita podstabla.

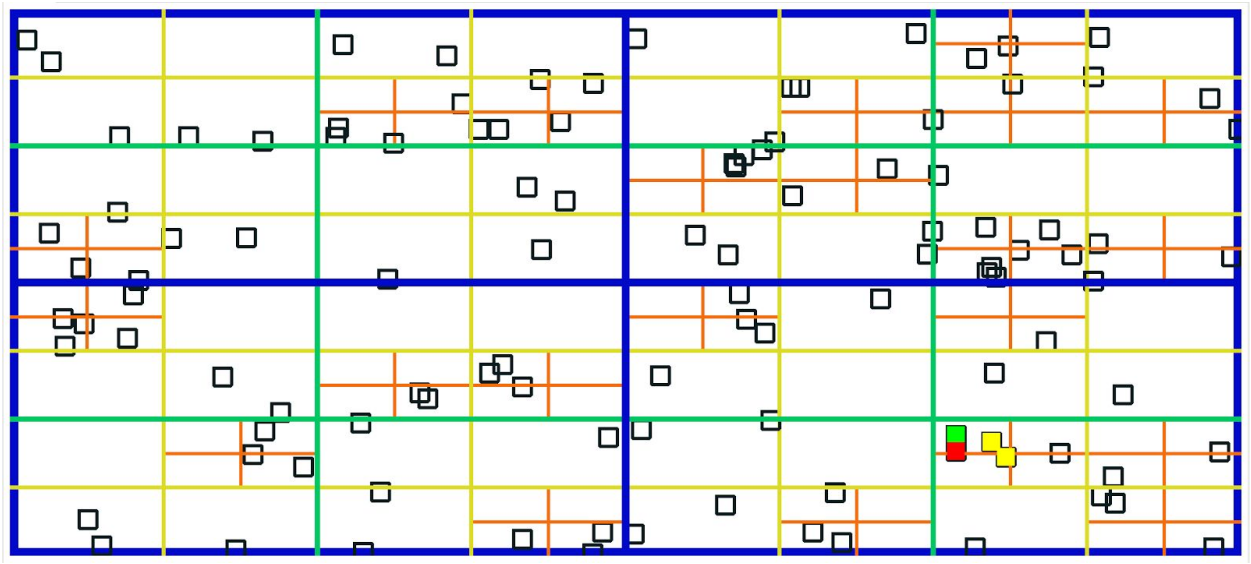
Srž algoritma je funkcija insert, koja dodaje novi element u Quadtree, i usput proverava preseke sa elementima.

Pseudokod funkcije insert:

```
Insert (item)

If isLeafNode

      If depth < maxDepth && numOfElements >= maxElements

            splitNode into four subNodes

            Insert all contained elements into subNodes

            Insert item

      Else

            Check collision with each contained element

            Add item to Nodes' contained elements

Else

      q = findQuadrants of item

      Insert item into subNode which corresponds to q
```

Pirlikom dodavanja novog elementa može se desiti da je prekoračen maksimalni broj elemenata u listu, a da još nismo dostigli maksimalnu dubinu. Tada se list pretvara u podstablo, i nakon toga se ubacuje novi element.

Vizuelizacija algoritma



Prikazan je Quadtree maksimalne dubine 5, koji ima najviše 2 elementa u listu.

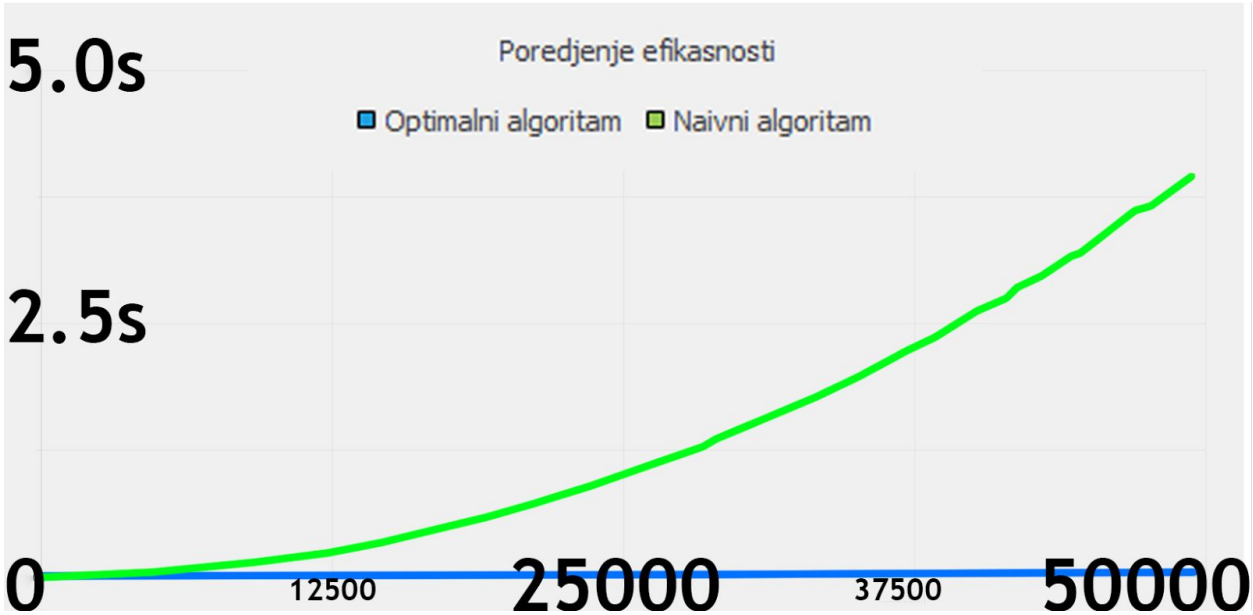
Kvadrat za kog smo tražili preseke je obojen zelenom bojom

Crvenom su objeni kvadrati koji seku zeleni.

Žutom bojom su obojeni kvadrati koji su proveravani da li se seku sa zelenim, a ne seku se.

Poredjenje efikasnosti naivnog i optimalnog algoritma

Za manje ulaze, odnosno do oko 1100 oba algoritma se jako brzo izvrše. Ali nakon toga naivni algoritam krene naglo da zaostaje naspram optimalnog.



Testiranje ispravnosti algoritma

Izlaz je broj elemenata koji se seku sa nekim drugim elementom.

Naziv testa	Opis testa	Ulaz	Očekivani izlaz
Efficient noPoints	Zadavanje praznog niza tačaka.	[]	0
Efficient lonelyPoint	Zadavanje samo jedne tačke.	[{1, 1}]	0
Efficient twoSamePoints	Zadate dve iste tačke. Obe moraju imati presek.	[{1, 1}, {1, 1}]	2
Naive noPoint	Zadavanje praznog niza tačaka.	[]	0
Naive lonelyPoint	Zadavanje samo jedne tačke.	[{1, 1}]	0
Naive twoSamePoints	Zadate dve iste tačke. Obe moraju imati presek.	[{1, 1}, {1, 1}]	2

Compare randomInput_1	Zadavanje nasumićnog niza tačaka veličine 100	Niz dimenzije 100	Poklapanje rezultata naivnog i optimalnog algoritma
Compare randomInput_2	Zadavanje nasumićnog niza tačaka veličine 1000	Niz dimenzije 1000	Poklapanje rezultata naivnog i optimalnog algoritma
Compare randomInput_3	Zadavanje nasumićnog niza tačaka veličine 1111	Niz dimenzije 1111	Poklapanje rezultata naivnog i optimalnog algoritma
Compare inputWithCollision	Zadat niz u kome znamo da su dva elementa u koliziji	[{1, 1}, {100, 50}, {300, 300}, {40, 60}, {60, 40}, {200, 200}, {100, 50}]	2 za oba algoritma
Compare threeSamePoints	Zadate tri iste tačke. Svaki element mora da ima koliziju.	[{1, 1}, {1, 1}, {1, 1}]	3 za oba algoritma
Compare barelyTouching	Zadate tačke i veličina kvadrata tako da se jedva dodiruju.	[{100, 100}, {100, 110}, {100, 90}] Size = 10	3 za oba algoritma
Compare almostTouching	Zadate tačke i veličina kvadrata tako da nemaju preseka, ali malo fali.	[{100, 100}, {100, 111}, {100, 89}] Size = 10	0 za oba algoritma
Compare allInCenter	Zadate tačke i veličina tako da svi elementi sadrže centar Quadtree-a	[{590, 240}, {600, 250}, {610, 260}, {590, 240}, {599, 249}]	5 za oba algoritma