Penerapan Algoritma Decrease and Conquer pada Search Query dalam Quadtree

Muhammad Rifky Muthahhari – 13519123
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): rifkymuthahhari@gmail.com

***Abstrak*—Data-data dalam bentuk koordinat dapat disimpan menggunakan berbagai jenis struktur data dengan masing-masing pro dan kontra yang dimiliki. Data koordinat yang sering diproses berdasarkan letaknya akan lebih mudah dilakukan jika menggunakan *quadtree* sebagai struktur data penyimpanannya. Proses search *query* pada *quadtree* memberikan keuntungan dengan mengeliminasi data-data yang sudah jelas berada di luar *query* sehingga pencarian dapat dilakukan dengan lebih efisien.**

Kata kunci—quadtree; search; koordinat;

I. PENDAHULUAN

Kumpulan data berbentuk titik koordinat dapat disimpan dengan menggunakan berbagai jenis struktur data. Data tersebut dapat disimpan baik dalam bentuk larik biasa tanpa ada modifikasi maupun menggunakan sebuah struktur data yang lebih kompleks.

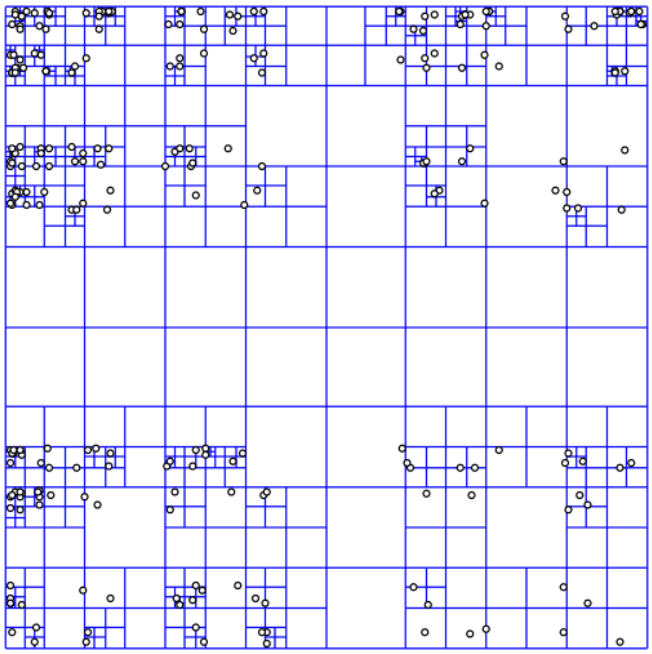
Data titik koordinat akan diproses oleh pemilik data untuk kebutuhannya. Secara alami, data titik koordinat akan lebih sering diproses berdasarkan letak dari tiap titik yang terdefinisi. Data titik koordinat yang berjumlah besar kebanyakan digunakan untuk dilakukan *search query* berdasarkan letak tiap titik terhadap suatu area. Walaupun data tersebut dapat diproses ketika disimpan dalam bentuk paling sederhana – larik, proses yang dilakukan terhadap suatu area dalam data akan menjadi sia-sia ketika semua titik koordinat dalam data harus diproses juga baik titik tersebut berada dalam area atau tidak. Kasus ini mungkin tidak memberikan masalah apa pun dalam hasil yang diberikan, tapi jelas terlihat ada aksi yang bisa dikurangi untuk meningkatkan efektivitas.

Sebagai contoh, diberikan 10.000 titik dalam data yang masing-masing diacak penempatannya. Pemilik data ingin mencari titik-titik yang tersedia dalam suatu area yang dipilih. Jika digunakan algoritma *brute force* didapat minimal dilakukan 10.000 perbandingan untuk didapat hasil yang diinginkan. Meskipun hal ini tidak terlalu buruk, kita dapat melakukan modifikasi dalam struktur data penyimpanan data kita sehingga proses yang dilakukan dapat lebih efisien.

Salah satu struktur data yang dapat digunakan adalah *quadtree*. *Quadtree* adalah struktur data yang menyimpan datanya dalam bentuk *tree* yang digolongkan berdasarkan letak data terhadap *quadtree* tersebut. Bentuk struktur data *quadtree* yang dispesialisasi ini memberikan keuntungan ketika akan dilakukan *searching* karena dapat dilakukan algoritma *decrease and conquer* dalam prosesnya untuk mengeliminasi aksi yang tidak dibutukan.

II. LANDASAN TEORI

A. *Quadtree*

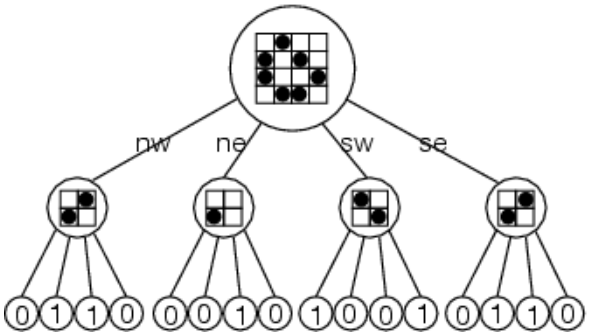


Tabel I. POINT QUADTREE

Quadtree adalah sebuah struktur data yang baik digunakan untuk menyimpan informasi yang akan diproses dalam bentuk kumpulan keys. Kumpulan keys yang disimpan dapat ditata dalam bentuk larik dua dimensi yang akan membentuk struktur seperti sebuah peta. *Quadtree* menggunakan prinsip tersebut dalam penyimpanan informasinya. *Quadtree* tidak mengharuskan penyimpanan dalam ruang dua dimensi, tapi untuk kemudahan akan digunakan ruang dua dimensi dalam pembuatan *quadtree*.

Lokasi informasi yang disimpan berdasarkaan keys dalam bentuk larik dua dimensi terletak pada pohon dengan empat derajat node. Setiap node akan menyimpan satu record dan

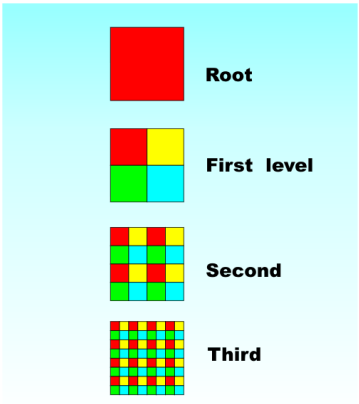
akan mempunyai hingga empat anak. Akar dari pohon yang dibentuk menandai semesta pemetaan menjadi empat bagian: NE (*northeast*), NW (*northwest*), SW (*southwest*), SE (*southeast*). Ide dari pembuatan anak pada suatu node adalah jika pada suatu bagian terdapat informasi yang lebih padat, proses pemecahan menjadi bagian-bagian yang lebih kecil akan dibutuhkan dalam hal ini membuat anak dari node tersebut.



Tabel II. REPRESENTASI QUADTREE DARI CEL BERUKURAN 4X4

Hal yang paling menonjol perbedaannya antara quadtree dengan sebuah pohon biasa adalah proses subdivide dan insert yang dilakukan.

Subdivide adalah proses membagi records pada suatu node yang sudah penuh menjadi empat bagian sama besar dalam bentuk node-node anak. Proses subdivide hanya akan dilakukan ketika terdapat proses insert pada bagian node yang sudah penuh. Records yang dibagi tadi akan ditempatkan pada node anak sesuai dengan lokasinya.



Tabel III. PROSES SUBDIVIDE PADA QUADTREE

Insert pada *quadtree* harus diberikan beberapa kondisi tambahan. Setiap node memiliki kapasitas maksimal sesuai dengan yang sudah didefinisikan pada awal. Ketika proses insert dilakukan pada bagian node yang sudah penuh kapasitasnya, akan dilakukan proses subdivide node tersebut dan records yang tadi dipegang akan dipindahkan ke node anak beserta record baru yang akan ditambahkan. Node yang melakukan subdivide tersebut akan memegang alamat dari setiap anaknya.

B. Algoritma Decrease and Conquer

Algoritma *decrease and conquer* adalah metode algoritma dengan mereduksi persoalan menjadi beberapa sub-persoalan yang lebih kecil. Sub-persoalan yang telah direduksi tadi akan diproses dan membiarkan bagian yang tereduksi tidak diproses.

Algoritma *decrease and conquer* berbeda dengan algoritma *divide and conquer* yang memproses semua sub-persoalan dan menggabungkan semua solusi setiap sub-persoalan pada akhir. Hal ini didasari sebuah pengetahuan bahwa terdapat bagian dari persoalan yang tidak relevan atau penting untuk diproses dalam menentukan jawaban akhir. Bagian yang tidak penting tersebut direduksi dari persoalan penuh sehingga persoalan menjadi lebih kecil. Proses ini dilakukan terus-menerus secara rekursif hingga mendapatkan jawaban dari solusi.

Algoritma *decrease and conquer* memiliki dua tahapan utama: *decreas* dan *conquer*. Tahap *decrease* merupakan tahap mereduksi persoalan menjadi beberapa persoalan yang lebih kecil (biasanya dia sub-persoalan). Sedangkan, proses *conquer* adalah tahap memproses satu sub-persoalan secara rekursif. Pada *decrease and conquer* tidak terdapat proses *combine*.

Terdapat tiga varian *decrease and conquer:*

1. *Decrease by constant*
2. *Decrease by a constant factor*
3. *Decrease by a variable size*

Terdapat banyak aplikasi dari algortima *decrease and conquer*. Berikut merupakan penerapan algoritma *decrease and conquer* yang paling sering ditemukan:

- Selection sort
- Interpolation search
- Binary search

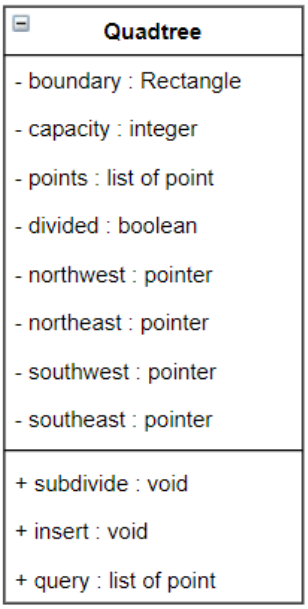
Sebagai contoh, algoritma *decrease and conquer* dapat digunakan untuk memecahkan permasalahan mencari median dari suatu larik tidak terurut. Proses yang dilakukan mirip dengan proses pada quick sort. Lakukan partisi pada senarai seperti proses partisi pada algoritma Quick Sort (varian 2). Partisi menghasilkan setengah elemen senarai lebih kecil atau sama dengan pivot p dan setengah bagian lagi lebih besar dari pivot p. 2. Misalkan s adalah posisi pem-partisian. Jika s = $\lceil n/2 \rceil$, maka pivot p adalah nilai median yang dicari Jika s > $\lceil n/2 \rceil$, maka median terdapat pada setengah bagian kiri Jika s < $\lceil n/2 \rceil$, maka median terdapat pada setengah bagian kanan. Dari algoritma di atas didapat kompleksitas waktu untuk setiap kalkulasi menjadi:

$$T(n) = T(n/2) + cn = \ldots = O(n)$$

III. METODOLOGI

A. *Kelas Quadtree*

Kelas *quadtree* yang digunakan pada makalah ini memiliki struktur sebagai berikut:

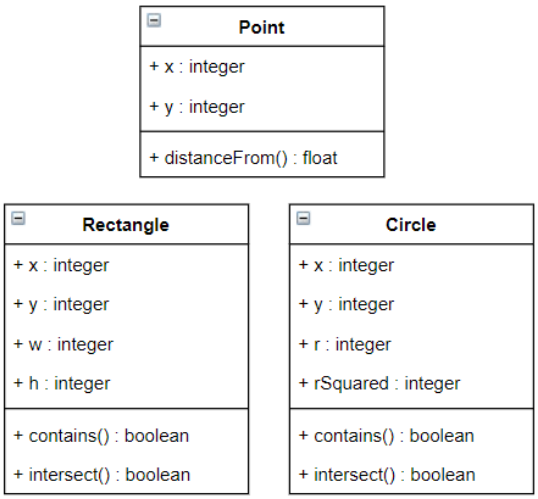


Tabel IV. KELAS QUADTREE

Pada makalah ini, fokus yang akan dikaji adalah penerapan algoritma *decrease and conquer* pada method query yang terdapat dalam kelas *quadtree* di atas.

B. *Kelas Point, Circle, dan Rectangle*

Kelas Point, Circle, dan Rectangle dibutuhkan dalam pembuatan *quadtree*. Point merupakan sebuah data yang disimpan – records, yang dipetakan pada *quadtree*. Berikut merupakan struktur dari kelas Point, Circle, dan Rectangle yang digunakan:



Tabel V. KELAS POINT, CIRCLE, DAN RECTANGLE

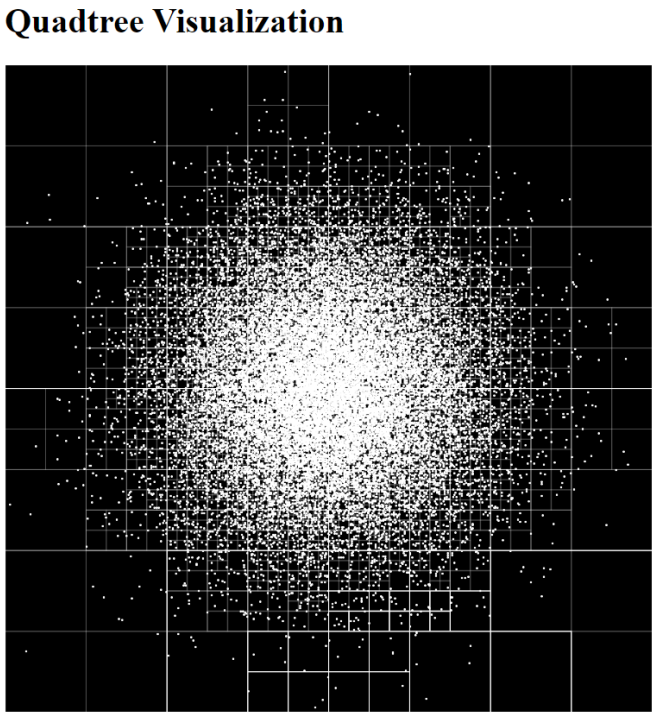
C. *Pengembangan Aplikasi*

Platform yang digunakan untuk melakukan visualisasi adalah web dengan logikanya ditulis dalam file Javascript. Visualisasi pada Javascript dibantu dengan menggunakan *library* p5.

D. *Setup dan Batasan Pengujian*

Setup yang dilakukan adalah membuat sebuah kanvas kosong yang nantinya akan visualisasikan *quadtree* yang terbentuk. Sebuah point akan divisualisasi sebagai titik sesuai dengan informasi koordinat yang disimpan. Query yang dilakukan akan divisualisasikan sebagai sebuah Circle disekitar *mouse* diklik pada saat pengujian. Hal yang akan di

Dalam pengujian kali ini akan dibuat 25.000 titik secara acak pada kanvas untuk ditambahkan ke dalam *quadtree*. Terdapat juga sebuah struktur data larik satu dimensi yang juga akan menyimpan 25.000 titik yang diberikan. Pengujian akan dilakukan dengan melakukan query pada suatu titik dalam kanvas dan menghitung waktu yang dibutuhkan dalam kalkulasi setiap struktur data untuk mengembalikan list of point dari daerah query yang dilakukan.



Tabel VI. VISUALISASI QUADTREE TERHADAP 25.000 POINTS

Algoritma dalam penentuan list of point hasil query untuk struktur data larik satu dimensi yang digunakan adalah iterasi sederhana untuk setiap point yang ada dalam larik tersebut. Pada kasus ini dapat dilihat bahwa terdapat 25.000 pengecekan untuk setiap kali dilakukan query. Algoritma yang digunakan adalah sebagai berikut:

```
function bruteforcesearch(range) {
  let points = [];
  for (let p of array) {
    if (range.contains(p)) {
      points.push(p);
    }
  }
  return points;
}
```

GAMBAR I. IMPLEMENTASI METHOD QUERY BRUTE FORCE

E. Algoritma Decrease and Conquer

Penerapan algoritma *decrease and conquer* digunakan dalam method query yang dimiliki *quadtree*. Implementasi dari method query sebagai berikut:

```
query(range, found) {
  // decrease
  if (!range.intersects(this.boundary)) {
    return found;
  }

  if (!found) {
    found = [];
  }
  if (this.divided) {
    this.northwest.query(range, found);
    this.northeast.query(range, found);
    this.southwest.query(range, found);
    this.southeast.query(range, found);
  }

  // conquer
  for (let p of this.points) {
    if (range.contains(p)) {
      found.push(p);
    }
  }

  return found;
}
```

Tabel VII. IMPLEMENTASI METHOD QUERY QUADTREE

Tahap *decrease* yang dilakukan adalah seperti berikut: (1) sebuah range diberikan kepada method query, (2) Method akan mengecek apakah range tersebut ada dalam node pada *quadtree*, proses pertama akan dilakukan dengan mengecek apakah range berada pada akar pohon, (3) Jika range berada dalam node, cek setiap anak dari node juga masuk dalam range, (4) Proses *decrease* terjadi ketika suatu anak dari node (NE, NW, SE, SW) tidak masuk dalam range yang diberikan, (5) jika suatu anak tidak masuk dalam range, akan diabaikan bagian anak tersebut.

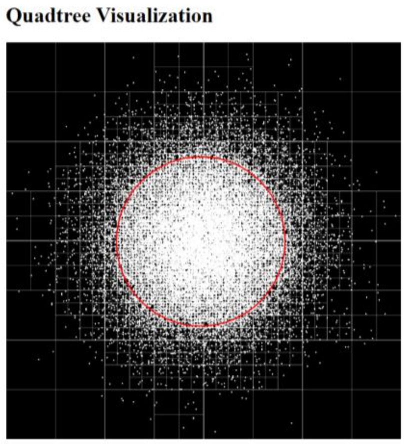
Tahap *conquer* yang dilakukan adalah mengecek untuk setiap point dalam listofpoints termasuk ke dalam range. Larik found adalah larik hasil pencarian yang akan ditambahkan hasil pencarian dari anak-anak node secara rekursif.

IV. DATA DAN ANALISIS

A. Hasil Pengujian

Pengujian dilakukan dengan 5 kali percobaan masing-masing diatur agar posisi query yang dipilih berbeda dan memiliki karakteristik yang berbeda.

1. Pengujian #1 (Tepat pada tengah kanvas)



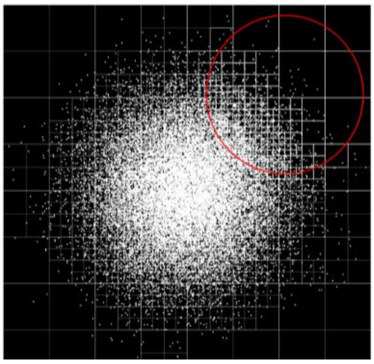

```
QuadTree Time                                   sketch.js:71
begin at   : 1620718039863                      sketch.js:72
end at     : 1620718039870                      sketch.js:73
total time : 7 ms                               sketch.js:74
Brute Force Time                                sketch.js:88
begin at   : 1620718039870                      sketch.js:89
end at     : 1620718039874                      sketch.js:90
total time : 4 ms                               sketch.js:91
                                                sketch.js:61
(19207) [Point, Point, Point, Point, Point, Point, Point, Point, Point, P
oint, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poin
t, Point, Point, Point, Point, Point, Point, Point, Point, Point, Point,
Point, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poi
nt, Point, Point, Point, Point, Point, Point, Point, Point, Point, Point,
Point, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poi
nt, Point, Point, Point, Point, Point, Point, Point, Point, Point, Point,
Point, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poi
nt, Point, Point, Point, Point, Point, Point, Point, Point, Point, Point,
Point, Point, Point, Point, Point, Point, Point, …]
                                                sketch.js:62
```

Tabel I. DATA PENGUJUAN #1

	Algoritma	
	QuadTree	***Brute Force***
Total time (ms)	7	4
Total points	19207	

2. Pengujian #2 (Kanan atas kanvas)

Quadtree Visualization



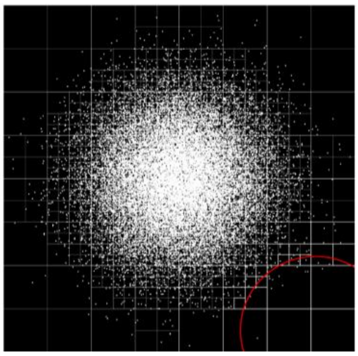
```
QuadTree Time                                           sketch.js:71
begin at   : 1620718051449                              sketch.js:72
end at     : 1620718051450                              sketch.js:73
total time : 1 ms                                       sketch.js:74
Brute Force Time                                        sketch.js:88
begin at   : 1620718051450                              sketch.js:89
end at     : 1620718051455                              sketch.js:90
total time : 5 ms                                       sketch.js:91
                                                        sketch.js:61
  (1681) [Point, Point, Point, Point, Point, Point, Point, Point, Point,
  Point, Point, Point, Point, Point, Point, Point, Point, Point, Point, P
  oint, Point, Point, Point, Point, Point, Point, Point, Point, Point, Po
  int, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poi
  nt, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poin
▸ t, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poin
  t, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poin
  t, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poin
  t, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poin
  t, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poin
  t, …]
                                                        sketch.js:62
```

Tabel II. DATA PENGUJUAN #2

	Algoritma	
	QuadTree	***Brute Force***
Total time (ms)	1	5
Total points	1681	

3. Pengujian #3 (Kanan bawah kanvas)

Quadtree Visualization



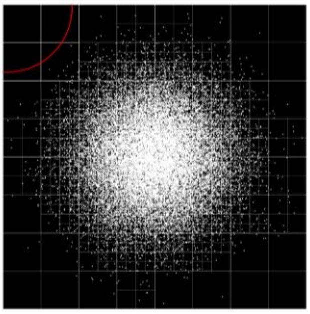
```
QuadTree Time                                           sketch.js:71
begin at   : 1620718063591                              sketch.js:72
end at     : 1620718063591                              sketch.js:73
total time : 0 ms                                       sketch.js:74
Brute Force Time                                        sketch.js:88
begin at   : 1620718063591                              sketch.js:89
end at     : 1620718063595                              sketch.js:90
total time : 4 ms                                       sketch.js:91
                                                        sketch.js:61
  (20) [Point, Point, Point, Point, Point, Point, Point, Point, Point, Po
▸ int, Point, Point, Point, Point, Point, Point, Point, Point, Point, Poi
  nt]
                                                        sketch.js:62
```

Tabel III. DATA PENGUJUAN #3

	Algoritma	
	QuadTree	***Brute Force***
Total time (ms)	0	4
Total points	20	

4. Pengujian #4 (Kiri atas kanvas)

Quadtree Visualization



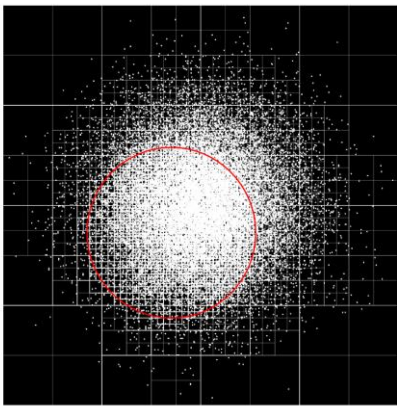
```
                                                        sketch.js:62
QuadTree Time                                           sketch.js:71
begin at   : 1620718071490                              sketch.js:72
end at     : 1620718071490                              sketch.js:73
total time : 0 ms                                       sketch.js:74
Brute Force Time                                        sketch.js:88
begin at   : 1620718071490                              sketch.js:89
end at     : 1620718071494                              sketch.js:90
total time : 4 ms                                       sketch.js:91
▸ []                                                    sketch.js:61
                                                        sketch.js:62
```

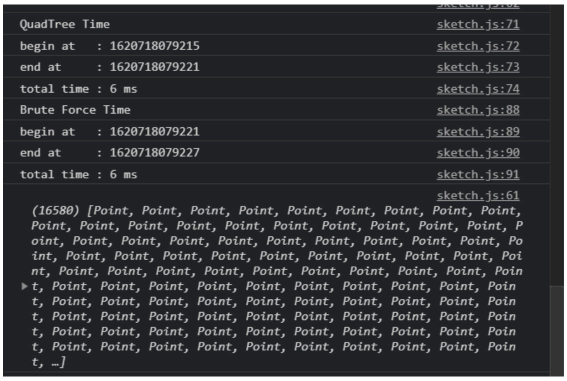
Tabel IV. DATA PENGUJUAN #4

	Algoritma	
	QuadTree	***Brute Force***
Total time (ms)	0	4
Total points	0	

5. Pengujian #5

Quadtree Visualization





Tabel V. DATA PENGUJUAN #5

	Algoritma	
	QuadTree	***Brute Force***
Total time (ms)	6	6
Total points	16580	

B. *Analisis*

Dari data hasil pengujian didapatkan berbagai nilai untuk setiap query yang dilakukan. Hal yang dapat ditonjolkan pada hasil pengujian adalah terlihat terdapat perbedaan waktu yang signifikan ketika query yang dilakukan mencakup dearah dengan densitas points tinggi. Hal ini disebabkan sifat alami dari algoritma *decrease and conquer* yang mengeliminasi bagian yang tidak dibutuhkan sehingga query pada daerah dengan densitas points rendah jauh lebih cepat dilakukan. Sebaliknya, query pada daerah dengan densitas points yang sangat tinggi akan membuat aksi tambahan pada method query untuk setiap kali rekursif dipanggil menjadi berat sehingga waktu total algoritma *brute force* dapat dan telah mendahului waktu total algoritma *decrease and conquer* pada method query *quadtree*.

V. PENUTUP

A. *Kesimpulan*

Struktur data data *quadtree* dapat digunakan sebagai alternatif penyimpanan informasi terutama informasi berupa titik koordinat. *Quadtree* berhasil melakukan kerja yang baik dalam mengategorikan titik koordinat menjadi bagian-bagian dalam susunan pohon terstruktur. Algoritma *decrease and conquer* pada search query dalam *quadtree* berhasil membuat pencarian menjadi lebih efisien ketika query yang dipilih mencakup daerah dengan densitas informasi lebih rendah. Biaya tambahan dalam operasi query pada *quadtree* membuat algoritma *decrease and conquer* pada search query *quadtree* memiliki performa sama atau lebih rendah dari algoritma *brute force*. Kedua struktur data dan algoritma berhasil melakukan pekerjaannya dengan baik.

UCAPAN TERIMA KASIH

Pertama-tama penulis panjatkan syukur kepada Tuhan Yang Maha Esa karena dengan rahmat dan hidayah-Nya makalah ini dapat selesai tepat waktu. Tak lupa juga penulis ucapkan terima kasih kepada Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D. yang telah membimbing selama satu semester dalam mata kuliah IF2211 Strategi Algoritma pada kelas K03 sehingga penulis dapat memahami materi-materi yang diberikan hingga memiliki cukup pengetahuan untuk menulis makalah ini. Penulis juga mengucapkan terima kasih kepada teman dan keluarga yang senantiasa membantu dalam kegiatan perkuliahan baik secara material maupun mental selama satu semester ini sehingga penulis dapat mengikuti perkuliahan dengan lancar.

REFERENSI

Finkel, Raphael & Bentley, Jon. (1974). Quad Trees: A Data Structure for Retrieval on Composite Keys.. Acta Inf.. 4. 1-9. 10.1007/BF00288933.

Munir, Renaldi. (2021). Decrease and Conquer. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Decrease-and-Conquer-(2018).pdf . Diakses pada tanggal 10 Mei 2021.

Kode diambil dan diedit dari https://github.com/CodingTrain/QuadTree.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Mei 2021

Muhammad Rifky Muthahhari
13519123