Representasi Objek Gambar Dengan *Quad Tree*

Edwin Wijaya - 13513040
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
edwinwijaya94@yahoo.com

***Abstrak*—Struktur pohon merupakan salah satu bentuk khusus dari graf. Jenis suatu pohon dapat dikategorikan berdasarkan jumlah simpul anak yang dimiliki oleh simpul orangtua. Suatu pohon yang setiap simpul orangtuanya memiliki 4 simpul anak dinamakan *quad tree*. Salah satu pemanfaatan *quad tree* untuk merepresentasikan objek gambar beserta manipulasinya. Dalam makalah ini akan dipaparkan bagaimana struktur data *quad tree* dapat dimanfaatkan untuk merepresentasikan objek berupa gambar. Selain itu akan dipaparkan juga apa saja kelebihan dan kekurangan dari struktur data *quad tree* untuk merepresentasikan gambar.**

***Kata kunci*—Graf, Pohon, *Quad tree*, Gambar**

I. PENDAHULUAN

Dalam kehidupan sehari-hari, gambar digunakan untuk mengilustrasikan suatu objek. Hampir semua objek yang ada dapat disajikan dalam bentuk gambar. Suatu gambar pasti memiliki warna. Ada gambar yang hanya memiliki warna hitam – putih, ada pula gambar yang disajikan dengan warna yang beragam. Warna –warna inilah yang menjadikan suatu gambar memiliki makna.

Dalam ilmu komputer, objek berupa gambar dapat direpresentasikan sebagai larik 2-dimensi dengan elemen-elemennya adalah titik berwarna. Titik-titik berwarna inilah yang nantinya dapat membentuk sebuah objek gambar di layar komputer. Penggunaan larik 2-dimensi untuk merepresentasikan gambar bukanlah satu-satunya struktur data yang digunakan untuk merepresentasikan gambar. Suatu struktur data lain, yaitu *quad tree*, dapat pula digunakan untuk merepresentasikan sebuah objek gambar. Sebelum membahas lebih lanjut penggunaan *quad tree* dalam representasi gambar, akan dijelaskan terlebih dahulu pohon (*tree*) secara umum.

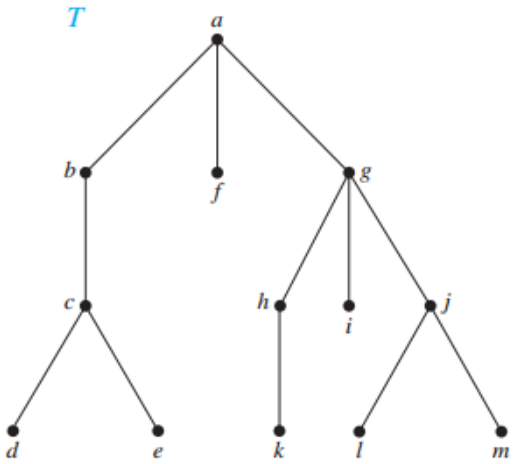
II. POHON

A. Definisi Pohon

Pohon merupakan bentuk khusus dari graf. Suatu pohon dapat didefinisikan sebagai graf tak-berarah terhubung yang tidak memiliki sirkuit. Pohon yang memiliki akar dinamakan sebagai pohon berakar. Suatu pohon berakar memiliki sisi – sisi yang berarah sehingga pohon berakar merupakan suatu graf berarah. Strukur pohon berakar merupakan suatu struktur yang rekursif. Terminologi dari suatu pohon berakar adalah sebagai berikut:

1. Akar : satu - satunya simpul pada pohon dengan sisi – sisi yang terhubung memiliki arah keluar dari simpul tersebut.
2. Orangtua dan anak : Jika *v* adalah simpul selain akar, dan simpul u memiliki sisi yang mengarah langsung ke simpul *v*, maka simpul *u* adalah orangtua (*parent*) dari simpul v, sedangkan simpul v sendiri disebut anak (*child*) dari simpul *u*.
3. Lintasan : sisi – sisi yang dilalui dari suatu simpul pada pohon ke simpul lainnya.
4. Saudara kandung : simpul-simpul yang memiliki orangtua yang sama.
5. *Ancestors* : *Ancestor* dari sebuah simpul (bukan akar) adalah simpul-simpul (termasuk akar) yang berada di lintasan dari akar ke simpul tersebut.
6. *Descendants* : Jika *v* adalah suatu simpul, maka *descendant* dari *v* adalah simpul-simpul yang memiliki simpul *v* sebagai salah satu *ancestor* nya.
7. Upapohon : Jika *a* adalah suatu simpul pada pohon, maka upapohon dengan *a* sebagai akar, adalah sebuah upagraf yang terdiri dari *a*, *descendant*, beserta sisi-sisi yang terhubung dengan *descendant*. Seperti yang sudah dijelaskan bahwa pohon berakar merupakan struktur rekursif, maka upapohon sendiri juga merupakan pohon.
8. Derajat : Derajat dari suatu simpul adalah jumlah upapohon atau jumlah anak yang dimiliki simpul tersebut. Derajat dari suatu pohon merupakan derajat maksimum yang dimiliki oleh simpul-simpulnya.
9. Daun : Simpul – simpul pada pohon yang tidak memiliki anak.
10. Simpul dalam : Simpul –simpul yang mempunyai anak.

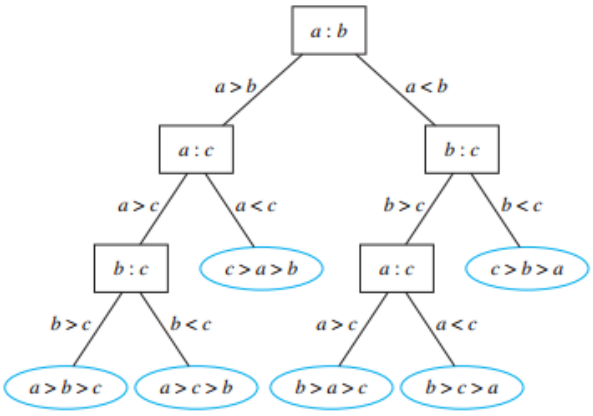
Berikut adalah salah satu contoh pohon berakar:



Gambar 1 – Pohon Berakar
Sumber : Discrete Mathematics and Its Applications, Rosen, 7th edition, halaman 748

B. Jenis Jenis Pohon

Suatu pohon dapat dikategorikan berdasarkan ciri tertentu yang dimiliki pohon tersebut. Misalnya, pohon biner adalah pohon yang memiliki maksimum 2 anak untuk setiap simpul di pohon tersebut. Pohon ternary adalah pohon yang memiliki maksimum 3 anak untuk setiap simpul di pohon tersebut. *Quad tree* (akan dibahas lebih rinci pada bab berikutnya) adalah pohon yang memiliki maksimum 4 anak untuk setiap simpulnya. Secara umum, pohon n-ner (baca: ener) adalah pohon yang memiliki maksimum n buah anak untuk setiap simpul pada pohon tersebut. Selain dikategorikan berdasarkan banyak anak yang dimiliki masing-masing simpul, pohon juga dapat dikategorikan berdasarkan tujuan dari dibentuknya pohon tersebut. Misalnya pohon pencarian biner yang digunakan untuk mencari nilai dari suatu data yang terurut dan pohon keputusan yang digunakan untuk membandingkan sekumpulan kondisi yang kemudian dapat mengarahkan kita untuk bergerak ke satu upapohon.



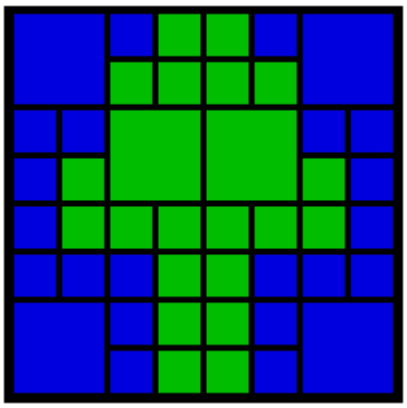
Gambar 2 – Sebuah pohon keputusan (*decision tree*)
Sumber: Discrete Mathematics and Its Applications, Rosen, 7th edition, halaman 761

III. QUAD TREE

A.Definisi

Seperti yang sudah didefinisikan secara singkat pada bab sebelumnya, *quad tree* merupakan struktur pohon berakar yang setiap simpulnya memiliki maksimum 4 buah anak.

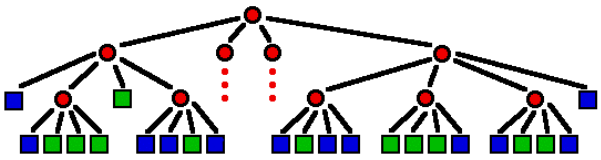
Penggunaan *quad tree* dalam sebuah gambar dapat dijelaskan sebagai berikut. Misalkan ada sebuah gambar yang memiliki ukuran piksel n x n. Simpul akar merepresentasikan keseluruhan area pada gambar yaitu seluas n x n piksel. Empat simpul anak masing-masing merepresentasikan ¼ area yang dimiliki oleh simpul orangtuanya, masing-masing ½ n x ½ n. Pembagian area dapat dilakukan seperti pembagian kuadran dalam koordinat Kartesius. Anak pertama mendapat area pojok kanan atas, anak kedua mendapat area pojok kiri atas, anak ketiga mendapat area pojok kiri bawah, dan anak keempat mendapat area pojok kanan bawah. Kemudian, setiap anak tersebut, memiliki masing-masing 4 anak lagi, dan proses pembagian tersebut dilakukan terus - menerus secara rekursif, hingga setiap simpul daun dapat merepresentasikan 1 buah piksel dalam suatu gambar. Untuk lebih jelasnya, perhatikan sebuah gambar sederhana berikut:



Gambar 3 – Sebuah gambar yang dibagi-bagi ke beberapa area
Sumber :
http://www.cs.ubc.ca/~pcarbo/cs251/welcome.html

Gambar di atas merupakan sebuah gambar yang memiliki 2 warna yaitu hijau dan biru, yang kemudian dibagi-bagi ke dalam beberapa area. Kita anggap persegi terkecil pada gambar tersebut menyatakan 1 piksel. Pembagian area pada gambar tersebut mengikuti aturan pada *quad tree*. Sebuah simpul pada *quad tree* tidak boleh memiliki simpul anak jika keempat simpul anaknya merepresentasikan warna yang sama. Pada gambar tersebut misalnya, area persegi berukuran 4 piksel pada pojok kiri bawah, tidak dibagi ke dalam 4 area karena jika dibagi ke dalam 4 area yang lebih kecil, semuanya merepresentasikan warna yang sama. Jadi proses pembagian kuadran pada *quad tree* berhenti jika area sudah mencapai ukuran 1 piksel atau area yang akan

dibagi hanya memiliki 1 warna saja. Hal ini bertujuan untuk meminimalkan kedalaman dari *quad tree*. Berikut merupakan struktur dari *quad tree* yang digunakan untuk merepresentasikan gambar di atas (Gambar 3) :



Gambar 4 – Struktur sebuah *quad tree*
Sumber :
http://www.cs.ubc.ca/~pcarbo/cs251/welcome.html

Penjelasan :

1. Simpul yang berwarna merah menyatakan bahwa simpul tersebut merupakan simpul dalam (memiliki anak) sehingga tidak merepresentasikan warna apapun.
2. Simpul yang berwarna hijau atau biru merupakan simpul daun dan masing-masing simpul daun merepresentasikan warna hijau atau biru.
3. Kedalaman masing-masing upapohon tidaklah sama. Hal ini dikarenakan aturan pada *quad tree* untuk merepresentasikan sebuah gambar, yaitu suatu simpul tidak boleh memiliki anak jika keempat simpul anaknya merepresentasikan warna yang sama.
4. Aturan pembagian area untuk setiap anak mengikuti pembagian pada sistem koordinat kartesius, anak ke 1 merepresentasikan area pojok kanan atas dari area persegi milik simpul orangtuanya, anak ke 2 merepresentasikan area pojok kiri atas, begitu seterusnya.

Selain direpresentasikan dengan struktur pohon secara eksplisit, *quad tree* juga dapat disimpan dalam format file biner agar penyimpanan menjadi lebih efisien. Setiap simpul yang memiliki anak (simpul dalam) dituliskan dengan angka 1. Simpul daun dituliskan dengan format 0X. Jika simpul daun merepresentasika warna hijau, X=0 sebaliknya jika merepresentasikan warna biru, X=1. Dengan demikian, struktur *quad tree* pada Gambar 4 dapat ditulis dalam kode biner sebagai berikut :

1 1 01 1 01 00 00 00 00 1 01 01 00 01 …
(hanya struktur *quad tree* dari akar dan anak ke 1 yang ditampilkan)

B. ADT Quad Tree

Contoh definisi tipe bentukan untuk *quad tree* :

```
type address : pointer to node
type quadtree : address
type node : <
            info :integer, {informasi warna pada simpul}
            northeast : quadtree, {anak ke 1}
            northwest : quadtree, {anak ke 2}
            southwest : quadtree, {anak ke 3}
            southeast : quadtree {anak ke 4} >
```

Definisi tambahan :

1. Sebuah simpul disebut transparan jika terdiri dari sub – persegi (memiliki 4 simpul anak).
2. Sebuah simpul disebut netral jika simpul tersebut belum memiliki warna.

ADT *quad tree* untuk representasi gambar setidaknya memiliki 4 primitif utama berikut :

1. *Ancestor Check* : mengecek apakah sebuah simpul diakses untuk menampilkan gambar (mengecek apakah node merupakan simpul daun). Jika ya, mengembalikan TRUE, sebaliknya, jika ada ancestor yang tidak transparan, mengembalikan FALSE.
2. *Division to a Quad* : membagi sebuah persegi menjadi 4 sub-persegi. Misalkan diberikan sebuah simpul anak, algoritmanya sebagai berikut :
   1. Jika simpul orangtua tidak transparan, maka simpan warna yang ada pada simpul orangtua.
   2. Ubah simpul orangtua menjadi transparan.
   3. Bentuk 3 simpul anak lainnya, dan warna pada simpul orangtua diberikan ke anak.
3. *Tree Traversal* : diberikan sebuh simpul, Tree Traversal menghasilkan semua *descendants* dari simpul tersebut.
4. *Reassembly* : Mengatur ulang struktur dari *quad tree*. Setelah warna dari setiap simpul daun diberikan, ada kemungkinan ketiga saudara kandung memiliki warna yang samadengan simpul daun tersebut, sehingga warna dari simpul daun tersebut dipindahkan ke simpul orangtua. Kemudian keempat simpul daun dihapus. Hal ini bertujuan untuk meminimalkan kedalaman dari *quad tree*.

C. Operasi Lanjutan

Setelah keempat primitif di atas terbentuk, kita dapat melakukan operasi – operasi yang lebih kompleks,

1. *Zeroing* : digunakan untuk menetralkan sebuah *quad tree*. Algoritmanya dengan membuat netral simpul akar.
2. *Superimpose* : salah satu metode untuk menghilangkan bagian gambar yang seharusnya tidak terlihat oleh pengamat adalah dengan mengurutkan permukaan (*surface*) pada gambar mulai dari yang paling jauh ke yang paling dekat dengan pengamat. Kemudian hanya permukaan yang bisa dilihat pengamat saja yang ditampilkan. Untuk meletakkan sebuah permukaan (*surface*) baru di atas gambar yang ada (*superimpose*), dilakukan pengecekan terhadap ancestor (*ancestor check*). Jika ada ancestor yang memiliki warna berbeda atau netral, maka simpul tersebut akan mempunyai simpul anak. Setelah proses tersebut, *reassembly* akan dilakukan jika diperlukan.

3. Penyisipan gambar baru : untuk menyisipkan gambar baru di belakang gambar yang ada, perlu dilakukan *ancestor check* terlebih dahulu. Jika ada simpul yang sudah memiliki warna, maka proses berhenti. Jika ada simpul yang netral, maka akan dilakukan pembagian ke dalam sub-persegi sehingga warna dari gambar yang akan disisipkan bisa diatur. Jika ada simpul yang transparan, maka akan dilakukan *ancestor check* terhadap simpul anak-anaknya secara rekursif. Proses tersebut berlangsung hingga semua simpul berhasil dicek. Pada setiap akhir pewarnaan gambar baru yang disisipkan, akan dilakukan proses *reassembly* untuk meminimalkan kedalaman *quad tree*..
4. Mengecek status dari *quad tree* : untuk mengecek apakah area dari sebuah *quad tree* tranparan, netral atau berwarna perlu dilakukan *ancestor check*. Jika sebuah simpul transparan ditemukan, maka akan dilakukan pengecekan secara rekursif terhadap simpul anak nya. Pengecekan akan berhenti jika sebuah simpul netral ditemukan yang berarti area dari quad tersebut tidak semuanya berwarna, sebaliknya jika tidak ditemukan simpul yang netral, berarti area dari quad tersebut seluruhnya sudah berwarna.
5. *Output* : Setelah pengaturan warna pada seluruh simpul *quad tree* selesai, *quad tree* siap untuk ditampilkan ke layar sebagai objek gambar. Proses ini dapat dilakukan dengan menggunakan prosedur *Tree Traversal* yang sudah ada dan dimulai dari akar.

IV. TRANSLASI, DILATASI, DAN ROTASI

Sebelum membahas bagaimana proses translasi , dilatasi (perbesaran gambar), dan rotasi dilakukan pada *quad tree*, kita akan sedikit mengubah struktur dari *quad tree* yang ada. Translasi, dilatasi, dan rotasi yang akan dilakukan pada gambar merupakan operasi pada setiap sub-persegi terkecil (piksel) atau pada setiap simpul daun. Oleh karena itu setiap simpul pada *quad tree* harus dapat menyimpan informasi berupa warna, ukuran, dan koordinat salah satu sudut persegi yang direpresentasikannya.

A. Translasi

Translasi merupakan pergeseran objek searah sumbu x, sumbu y atau keduanya. Untuk melakukan translasi, koordinat awal dari gambar ditranslasi kemudian koordinat yang berseberangan dengan kooordinat awal dihitung untuk kemudian dilakukakn translasi juga. Pada akhir proses translasi akan dicek apakah ada bagian gambar hasil translasi yang berada di luar area gambar. Jika ada, maka gambar akan dipotong dengan terlebih dahulu membagi – bagi *quad tree* yang ada menjadi simpul-simpul daun, kemudian membangun ulang *quad tree*.

B. Dilatasi

Proses dilatasi dilakukan dengan mengubah setiap titik xO dan yO sebagai berikut :

xO=(xO — xcentre) * scale_factor / image_size – xcentre

Proses yang serupa dilakukan untuk yO. xO dan yO menyatakan koordinat salah satu sudut persegi yang direpresentasikan oleh suatu simpul. Xcentre dan ycentre merupakan koordinat pusat gambar. Scalefactor merupakan perbesaran gambar yang diinginkan, sedangkan imagesize merupakan ukuran dari gambar aslinya. Jikaada bagian gambar hasil dilatasi yang berada di luar area gambar, maka gambar akan dipotong, proses yang sama pada translasi.

C. Rotasi

Proses rotasi dilakukan dengan koordinat pusat gambar sebagai sumbunya. Setiap persegi yang direpresentasikan oleh daun dirotasi sehingga koordinatnya berubah dari (x,y) ke (xR,yR) dengan fungsi berikut :

rotateX(x, y) = half_image_size + ((x-half_image_size) * cos(angle) - (y-half_image_size) * sin(angle)) / image_size

Proses yang serupa dilakukan dengan rotateY untuk merotasi koordinat y.

V. KOMPLEKSITAS

Pada dasarnya, dalam merepresentasikan objek gambar, struktur data *quad tree* tidak lebih hemat jika dibandingkan dengan struktur data larik 2-dimensi, ditinjau dari segi kompleksitas ruang. Akan tetapi *quad tree* memiliki kelebihan pada kompleksitas waktu algoritmanya. Kebanyakan algoritma untuk memproses *quad tree* (tidak dibahas dalam makalah ini) melakukan *tree traversal* secara *preorder* sehingga waktu eksekusi berbanding linear dengan jumlah simpulnya. Menurut *Quad Tree Complexity Theorem,*

"*Jika sebuah quad tree dengan kedalaman q yang merepresentasikan gambar dengan ukuran 2^q x 2^q piksel, serta setiap piksel merepresentasikan area dengan keliling p, maka banyak simpul pada quad tree tidak melebihi 16.q -11 + 16.p.*" [2]

Pembuktian:

1. Setiap persegi hanya bisa dibagi maksimum ke dalam 4 buah sub-persegi. Jika sebuah garis yang masuk ke dalam persegi dan menyentuh setiap sisi persegi (*curve*) memiliki panjang minimal *n* dan keliling dari persegi tidak lebih dari *p*, maka

persegi tersebut hanya bisa terbagi ke dalam sub-persegi yang jumlahnya tidak lebih dari $4 * \left\lceil \frac{p}{n} \right\rceil$

2. Misalkan akar berada pada kedalaman 1, maka pada kedalaman *k*, sebuah persegi yang direpresentasikan oleh suatu simpul akan memiliki sisi sepanjang $2^{(q-k+1)}$.
3. Dari aturan no.1 dan no.2, maka setiap persegi tidak akan memiliki lebih dari $4 * \left\lceil \frac{p}{2^{q-k+1}} \right\rceil$ sub-persegi pada kedalaman *k*. Misalkan ekspresi di atas sebagai B(k).
4. Karena hanya simpul dalam saja yang memiliki anak, maka pada kedalaman *k*, jumlah node yang memiliki anak tidak lebih dari B(k). Selanjutnya, karena pada level *k+1*, setiap simpul adalah anak dari simpul yang ada pada level *k*, maka pada level *k+1*, jumlah simpul tidak melebihi 4*B(k).
5. Pada 2 level teratas (simpul akar dan anaknya) hanya terdapat maksimum 5 simpul, maka jumlah simpul pada suatu *quad tree* memenuhi pertidaksamaan berikut (sumber : Referensi [1] halaman 148) :

$$5 + \sum_{k=2}^{q} 4B(k) \leqslant 5 + \sum_{k=2}^{q} 16 \text{ CEILING } (p/2^{q-k+1})$$

$$\leqslant 5 + \sum_{k=2}^{q} 16\,(1 + p/2^{q-k+1})$$

$$\leqslant 5 + \left(16 \sum_{k=2}^{q} 1\right) + 8 \sum_{k=2}^{q} p/2^{q-k}$$

$$\leqslant 5 + 16(q-1) + 8 \sum_{i=0}^{q-2} p/2^{i}$$

$$\leqslant 16q - 11 + 8 \sum_{i>0} p/2^{i}$$

$$\leqslant 16q - 11 + 16p.$$

*) CEILING(x) menyatakan bilangan bulat terkecil yang lebih besar dari x.

VI. Simpulan

Quad tree sebagai salah satu jenis struktur data pohon dapat digunakan untuk merepresentasikan objek berupa gambar. Setiap daun pada *quad tree* menyatakan area pada gambar yang memiliki warna yang sama. Quad tree membagi gambar ke dalam 4 buah persegi yang berukuran sama secara rekursif, hingga ukuran 1 piksel tercapai atau simpul sudah merepresentasikan warna yang sama. *Quad tree* dapat digunakan untuk melakukan berbagai operasi terhadap gambar seperti translasi, dilatasi, dan rotasi, Secara kompleksitas ruang, quad tree tidak lebih hemat jika dibandingkan dengan larik 2-dimensi, akan tetapi *quad tree* memiliki keunggulan dalam kompleksitas waktu karena hampir semua kompleksitas waktu algoritmanya berbanding lurus dengan jumlah simpul.

VII. Ucapan Terima Kasih

Saya mengucapkan syukur kepada Tuhan Yang Maha Esa karena dapat menyelesaikan makalah ini tepat waktu. Saya juga mengucapkan terima kasih kepada Pak Rinaldi Munir dan Bu Harlili atas bimbingan dan pengajaran yang telah diberikan pada mata kuliah Matematika Diskrit. Terima kasih saya sampaikan juga kepada keluarga dan teman-teman yang telah mendukung saya dalam kuliah ini.

Referensi

[1] Hunter, Gregory M., Operations on Images Using Quad Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No.2, April 1979. p.145-153

[2] Samet, Hanan, *Hierarchical Spatial Data Structures*. P.194-212.

[3] Woodwark, J.R.,The Explicit Quad Tree as a Structure for Computer Graphics. In The Computer Journal, Vol. 25, No.2, 1982. p.235-238.

[4] Rosen, Kenneth, *Discrete Mathematics and its Applications, Seventh Edition.* New York: McGraw-Hill, 2012, ch.11.

[5] Oliver, M.A.; Wiseman, N.E., Operations on Quadtree Leaves and Related Image Areas. In The Computer Journal, Vol.26, No.4, 1983, p.375-380.

[6] de Berg, Mark, et.al, *Computational Geometry : Algorithms and Applications , Third Edition*. Springer, 2008 , ch.14.

[7]. http://www.cs.ubc.ca/~pcarbo/cs251/welcome.html Diakses pada tanggal 9 Desember 2014 pukul 17.00.

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, 10 Desember 2014

Edwin Wijaya
13513040