Pohon *Quad* untuk Merepresentasikan Gambar

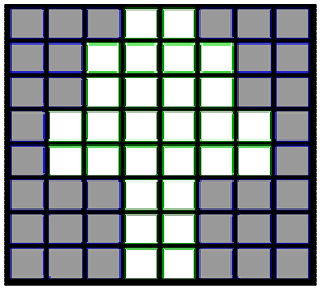
Gressia Melissa – NIM 13506017
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha no.10 Bandung
Email: if16017@if.itb.ac.id

Abstract *– Makalah ini membahas representasi digital untuk gambar. Representasi yang digunakan adalah pohon quad (4-ary atau cabang-empat), yaitu cara perepresentasian yang memiliki kelebihan dari segi struktur dan penyimpanan data. Selain itu, akan dibahas juga kelebihan-kelebihan dan kekurangan-kekurangan pohon quad dari segi struktur grafika komputer. Sebagai tambahan, akan dipaparkan pula mengenai ADT pohon quad dan contoh-contoh kasus untuk memperjelas teori, serta operasi biner untuk pohon quad.*

Kata Kunci *: ADT dan primitif, "divide and conquer", kuadran, manipulasi gambar, operasi biner, pohon quad, penyimpanan data, simpul.*

1. PENDAHULUAN

Gambar adalah larik dua dimensi, dimana tiap elemennya merupakan titik yang berwarna. Sesuai dengan definisi tersebut, pada faktanya, larik dua dimensi merupakan cara yang paling sering digunakan untuk menyimpan gambar dalam komputer. Berikut ini adalah contoh gambar yang direpresentasikan sebagai larik dua dimensi, setiap *pixel* adalah elemen larik.



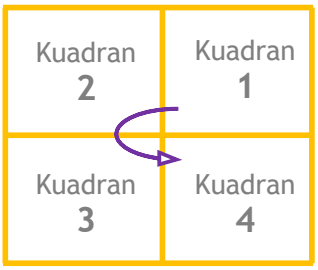
Gambar 1 : Gambar 8 x 8 *pixel* dengan Representasi Larik Dua Dimensi

Larik dua dimensi bukanlah satu-satunya cara untuk merepresentasikan gambar, juga bukan yang paling mangkus. Cara lain yang dapat digunakan dalam perepresentasian gambar ialah dengan pohon *quad*, piramida Gaussian, hirarki pencitraan Laplacian, dan sebagainya. Pohon *quad* termasuk cara yang paling sederhana untuk merepresentasikan gambar, sehingga lebih sering digunakan daripada cara-cara perepresentasian lainnya.

2. STRATEGI POHON *QUAD*

Definisi dari pohon *quad* adalah pohon yang tiap simpulnya memiliki paling banyak 4 anak atau cabang. Anak pohon tersebut diberi nama kuadran 1, kuadran 2, kuadran 3, dan kuadran 4 yang "dibaca" berlawanan arah jarum jam. Tiap kuadran iu sendiri merupakan pointer,. Untuk menggunakan pohon *quad* sebagai struktur data untuk gambar, kuncinya adalah "*Divide and Conquer*".

Simpul pada pohon *quad* sama dengan simpul pohon biner yang digunakan untuk merepresentasikan data dalam dua dimensi. Bentuk pohon bergantung pada proses pengurutan data. Hanya saja, kecepatan proses jauh lebih mangkus daripada perepresentasian data dengan dua dimensi dengan pohon biner. Sedangkan sisi pohon *quad* lebih sering digunakan untuk menyimpan garis daripada titik.



Gambar 2 : Pengurutan Kuadran untuk Pohon *Quad*

Langkah awal, yaitu dengan membagi area gambar menjadi empat bagian. Keempat bagian tersebut selanjutnya dibagi lagi menjadi 4 subbagian. Kemudian, keempat subbagian tersebut dibagi lagi menjadi 4 bagian yang lebih kecil. Langkah tersebut dilakukan secara rekursif , yaitu pengulangan pembagian tiap bagian gambar menjadi empat bagian yang lebih kecil. Jumlah perulangan harus ditentukan batasannya atau pembagian akan berlangsung hingga tak berhingga kali. Secara umum, batasan perulangan ini ditentukan dengan mempertimbangkan batas penyimpanan data, waktu pemrosesan, ataupun resolusi dari perangkat *output*. Subbagian terkecil dari pohon *quad* ini dikenal dengan nama *pixel*, atau dikenal juga dengan istilah daun (pada pohon *quad*). iap *pixel* bernilai 0 atau 1 (biner, akan dibahas lebih lanjut).

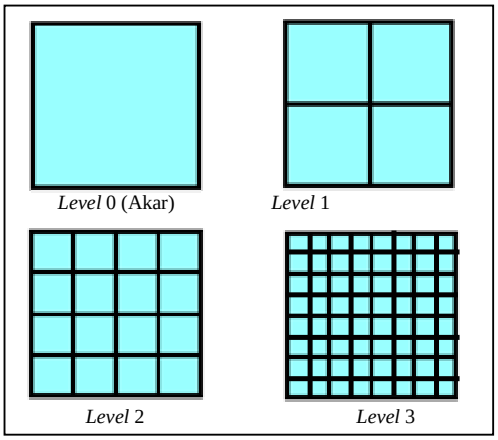
Dalam pohon *quad*, anak-anak simpul merepresentasikan empat (4) kuadran atau area dan akar pohon merepresentasikan gambar itu sendiri (keseluruhan gambar). Sedangkan jumlah perulangan dinamakan kedalaman atau *level* pohon.

Atau, untuk lebih singkatnya, keberlangsungan proses adalah sebagai berikut :

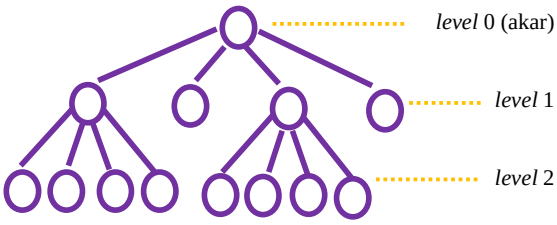
[1] Gambar utuh sebagai akar (kedalaman 0).
[2] Pembagian area gambar atau akar menjadi 4 kuadran atau anak (kedalaman 1).
[3] Tiap kuadran dibagi lagi menjadi 4 subkuadran (kedalaman 2).
[4] Tiap subkuadran akan dibagi menjadi 4 bagian area yang lebih kecil (kedalaman 3).
[5] Proses pembagian kuadran menjadi 4 subkuadran yang lebih kecil diulangi sampai kedalaman yang ditentukan.

Kuadran atau area pada gambar memiliki sifat berikut:

[1] Memiliki tepat satu warna.
[2] Terbentuk dari empat (4) subkuadran yang lebih kecil.



Gambar 3 : *Level* Pohon *Quad*



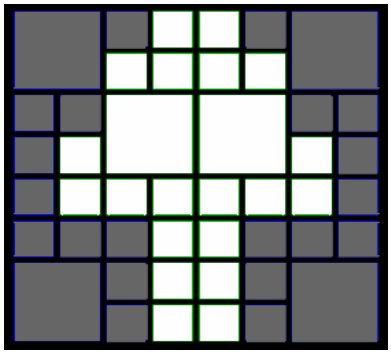
Gambar 4 : Penggambaran *Level* Pohon

Untuk merepresentasikan gambar dengan pohon *quad*, setiap daunnya harus merepresentasikan besar area yang sama pada gambar. Selain itu, perlu diingat bahwa tidak ada satu simpulpun yang diperbolehkan memiliki anak yang semuanya berwarna sama. Kedalaman harus diusahakan seminimal mungkin.

Contohnya, jika gambar yang ingin direpresentasikan ialah gambar hitam-putih, maka hanya diperlukan satu bit untuk mewakili warna pada tiap daun, misalnya 0 untuk hitam dan 1 untuk putih.

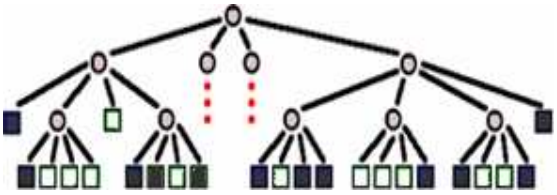
Area atau kuadran pada pohon *quad* dengan kedalaman n mewakili gambar yang berukuran 2^n x 2^n *pixel*. Area pohon *quad* juga dapat digunakan sebagai resolusi variabel yang mewakili wilayah data. Misalnya, suhu di suatu daerah disimpan dalam pohon *quad*, dengan simpul daun menyimpan suhu masing-masing area yang termasuk dalam area yang dimaksud dan akar merupakan suhu rata-rata. Jadi, semakin dalam *level*-nya, suhu yang diwakilkan akan semakin detail.

Contoh kasus 1 :
Gambar yang ingin direpresentasikan :



Gambar 5 : Gambar 8 x 8 *pixel*

Gambar di atas direpresentasikan sebagai pohon *quad*, dengan pemosisian kuadran berlawanan arah jarum jam sesuai dengan yang telah didefinisikan pada bagian pendahuluan, menjadi sebagai berikut :

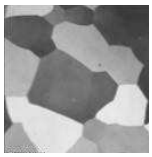
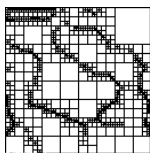
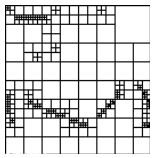
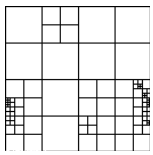


Gambar 6 : Representasi Pohon *Quad*

Representasi pohon *quad* di atas ditampilkan berlawanan arah jarum jam diurutkan dari atas-kanan kuadran. Simpul teratas merupakan akar pohon. Pada kedalaman 1, kuadran 2 sama dengan pencerminan dari kuadran 1 dan kuadran 3 sama dengan pencerminan dari kuadran 4 (tidak ditampilkan karena keterbatasan ruang).

Contoh kasus 2 :
Di bawah ini merupakan contoh gambar sesuai dengan tingkat ke-“kasar”-an, atau minimalisasi detail gambar dengan mengurangi kedalaman pohon *quad* (*pixel* semakin besar, resolusi semakin kecil).

Tabel 1. Minimalisasi Detail Gambar

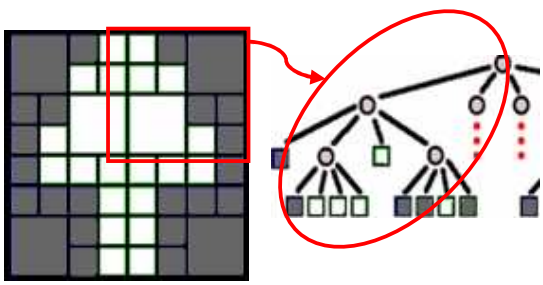
Ukuran *Pixel*	Tampilan Output	Representasi
Gambar *Original* (mendekati 0)		
0, 20		
0,40		
0,55		

3. POHON *QUAD* DALAM BINER

3.1 Skema Penyimpanan dalam Biner

Salah satu cara yang mangkus untuk menyimpan pohon *quad* adalah dengan skema struktur data biner. Untuk simpul selain simpul tanpa anak (atau dikenal juga dengan daun), direpresentasikan dengan 1. Sedangkan untuk simpul-simpul dalam, 00 merupakan representasi untuk simpul berwarna putih dan 01 untuk simpul berwarna hitam.

Sebagai contoh, ambil kuadran 1 pada contoh kasus-1 di samping (lihat gambar di bawah).



Gambar 7 : Area yang Diberi Tanda adalah Kuadran 1

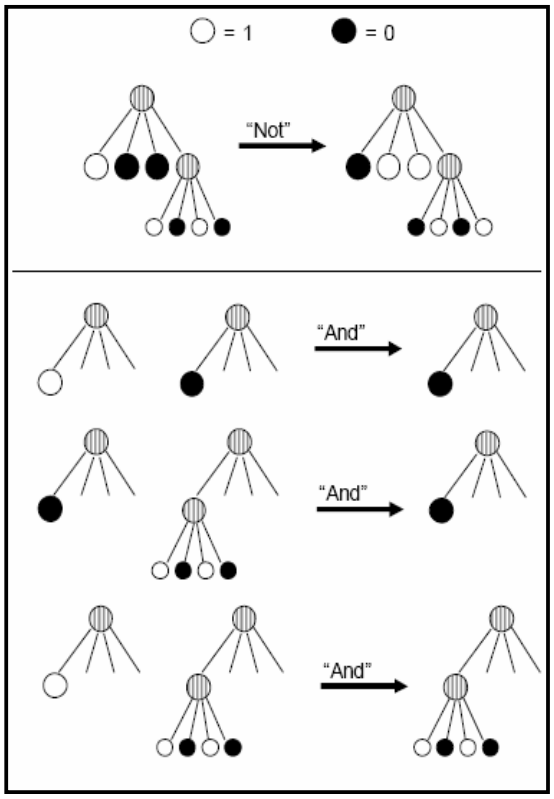
Representasi wilayah yang ditandai pada Gambar 6 dalam biner :

1 1 01 1 01 00 00 00 00 1 01 01 00 01

Angka 1 pertama merupakan akar, angka 1 berikutnya merepresentasikan simpul kuadran 1, sedangkan angka 1 lainnya (tanpa 0) merupakan simpul yang memiliki anak (simpul dalam), serta 01 dan 00 merepresentasikan daun. Proses pembacaan data dilakukan secara *prefix*.

3.2 Operasi Biner dengan Pohon *Quad*

Beberapa operasi biner yang dapat diimplementasikan pada pohon *quad* ialah "NOT" atau negasi, "AND" atau perkalian, dan "OR" atau penjumlahan. Selain itu, masih ada operasi-operasi biner lainnya, seperti "XOR", "XAND", dan sebagainya. Namun, yang akan dibahas lebih lanjut hanya 3 operasi dasar "AND", "OR", dan "NOT".



Gambar 8 : Contoh Operasi Biner pada Pohon *Quad*

3.2.1 Negasi (NOT)

Operasi ini akan menghasilkan nilai kebalikan dari nilai masukan. Pada pohon *quad* misalnya, simpul yang bernilai 1 akan dikembalikan dengan nilai 0. Simpul yang diproses hanyalah simpul-simpul daun, karena simpul dalam pada pohon tidak memiliki warna untuk merepresentasikan nilai.

3.2.2 Penjumlahan (OR) dan Perkalian (AND)
Diberikan dua buah pohon sebagai masukan, operasi ini memproses keluaran sama seperti pemrosesan aljabar boolean. Begitu pula dengan operasi-operasi lainnya (misalnya XOR) , prinsipnya sama dengan prinsip pada aljabar boolean.

4. ADT POHON *QUAD*

Suatu simpul tidak memiliki warna (transparan) jika memiliki subbagian atau simpul anak. Suatu simpul bersifat netral jika tidak ada warna yang diberikan pada simpul tersebut.

ADT yang akan dipaparkan di sini merupakan pohon *quad* eksplisit, yaitu setiap simpulnya ditempatkan tanpa memperhatikan apakah ia akan ditampilkan dalam gambar atau tidak. Dengan demikian, akan ada simpul yang ditempatkan pada pohon, namun pada kenyataannya tidak diakses pada saat gambar ditampilkan pada alat keluaran (output device). Sebuah simpul tidak memiliki fungsi jika ada satu atau lebih anaknya yang tidak berwarna (transparan).

Perhatikan bahwa jika gambar memiliki n *pixel* (atau n daun), maka pada pohon eksplisit, rata-rata terdapat 1,3 n simpul.

Prosedur-prosedur primitif pada pohon *quad* :

4.1 Pengecekan Simpul Anak
Simpul diberikan sebagai variable masukan, prosedur ini akan mengembalikan TRUE jika simpul diakses untuk ditampilkan pada gambar. Jika salah satu anak tidak transparan (tersembunyi), maka kuadran tersebut tidak akan tampil di layar. Dimulai dari akar, proses algoritma harus mengecek semua simpul dan anaknya. Jika salah satu anak tersembunyi atau tidak transparan, maka algoritma akan berhenti dan mengembalikan FALSE.

4.2 Pembagian Wilayah
Untuk pemanipulasian gambar, diperlukan pembagian wilayah atau area gambar menjadi subwilayah. Prosedur ini menerima masukan sebuah simpul dan membagi simpul tersebut menjadi kuadran. Algoritma ini tergolong sederhana. Prosesnya ialah mengecek *parent* dari simpul yang diberikan. Jika *parent* tersembunyi atau tidak transparan, maka proses selanjutnya ialah :
[1] Menyimpan warna *parent.*
[2] Menetralkan warna *parent.*
[3] Memindahkan warna asal *parent* kepada anak-anak simpul.

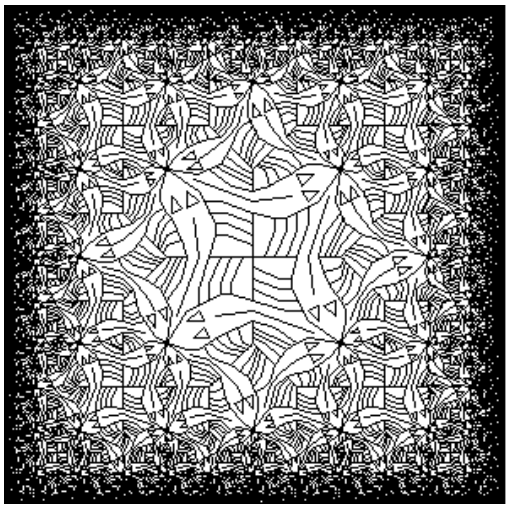
4.3 *Traversal* Pohon
Menerima masukan sebuah simpul, prosedur Tree Traversal menghasilkan anak simpul, kemudian proses diulangi untuk tiap anak simpul.

4.4 Pemosisian
Setelah warna suatu area berubah, sangat mungkin bahwa area tersebut memiliki nilai (*value*) yang sama dengan "saudara"-nya. Dalam kasus ini, warna dipindahkan ke *parent* dari kedalaman (atau *level*) minimum pohon. Prosedur ini menerima sebuah simpul sebagai masukan, kemudian menghasilkan "saudara" dari simpul tersebut. Jika semua simpul memiliki warna yang sama, warna tersebut dipindahkan ke *parent* dan proses berlangsung secara rekursif. Selain pada kondisi itu, proses akan berhenti dilakukan.

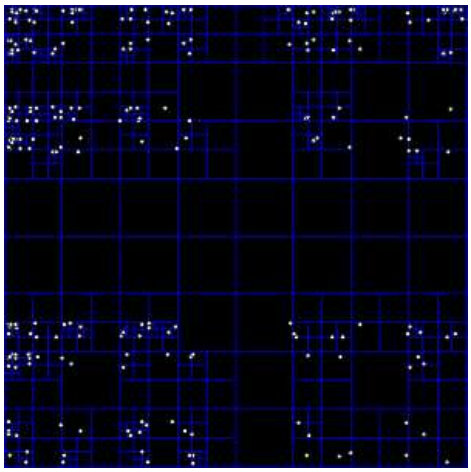
5. KELEBIHAN DAN KEKURANGAN REPRESENTASI GAMBAR DENGAN POHON *QUAD*

Pohon *quad* dapat diimplementasikan dengan luas pada ilmu grafika komputer. Secara umum, pohon *quad* dapat dimanipulasi dan pengaksesan gambar dengan pohon *quad* jauh lebih cepat daripada dengan cara lainnya. Karena alasan itulah, pohon *quad* kerap digunakan tidak hanya untuk representasi gambar, tetapi juga untuk segmentasi data, kompresi, dan lain-lain.

Gambar rekursif (gambar yang memasukkan elemen yang sama yang ditampilkan berulang-ulang) dapat diimplementasikan dengan mudah jika menggunakan pohon *quad*. Pada kasus ini, akar pohon *quad* memiliki 4 anak, yang salah satu dari anak tersebut adalah gambar utama, sedangkan tiga anak lainnya merujuk ke akar.



Gambar 9 : Contoh Gambar Rekursif, *á la Escher*



Gambar 10 : Contoh Representasi dengan Pohon *Quad*

Selain yang telah dipaparkan di atas, kelebihan-kelebihan penggunaan pohon *quad* antara lain :

[1] Penghapusan gambar hanya memerlukan satu langkah, yaitu hanya dengan membuat simpul akar sebagai simpul netral.

[2] Pembesaran (*zoom*) untuk melihat detail gambar pada pohon hanya memerlukan satu operasi, yaitu dengan berpindah ke kedalaman atau *level* berikutnya.

[3] Untuk mengurangi kompleksitas dari gambar, cukup dengan menghilangkan simpul-simpul *level* terdalam pada pohon (daun).

[4] Pengaksesan lebih dalam pada suatu area hanya memerlukan proses yang singkat. Hal ini sangat berguna untuk memperbarui area gambar tertentu.

[5] Kompleksitas waktu asimptotik rata-rata yang diperlukan adalah O (log n).

Satu-satunya kekurangan dari pohon *quad* adalah penggunaan memori yang cukup besar. Jika pohon *quad* diimplementasikan menggunakan pranala, sebagian besar memori dipakai oleh pranala tersebut. Walaupun demikian, masalah tersebut telah teratasi karena terdapat banyak cara untuk memadatkan pohon *quad*, sehingga proses pemindahan data pun menjadi lebih mudah dan mangkus.

6. KESIMPULAN

Pohon *quad* adalah pohon cabang-4, yang memiliki kemampuan untuk membuat pemanipulasian gambar dengan proses yang sangat *powerful*, sederhana, dan prosesnya berlangsung lebih cepat daripada cara perepresentasian lain, misalnya representasi dengan larik dua dimensi atau pohon biner. Pohon *quad* dapat mengubah struktur data lanjar menjadi struktur data rekursif. Oleh karena itu, gambar yang direpresentasikan dengan pohon *quad* lebih dinamis, itulah yang menyebabkan pohon *quad* sangat cocok untuk pemanipulasian atau perepresentasian gambar, karena selain sederhana, pohon *quad* juga memiliki lebih banyak kelebihan dibandingkan kelemahannya.

DAFTAR REFERENSI

[1] http://www.cs.berkeley.edu/~demmel/cs267/lectu re26/lecture26.html

[2] http://en.wikipedia.org/*Quad*tree

[3] Munir, Rinaldi. 2006. *Dikat Kuliah Matematika Diskrit edisi ke-4*. Bandung : ITB.

[4] Ranade, S. and Shneier M. 1995. *Using quadtrees to smooth images*. PWS Publishing Co. hal.376-379

[5] Samet, H. 1990. *Region Representation: Quadtrees from Binary Arrays*. Computer Graphics & Image Processing, vol. 13, no. 1. hal. 90-93

[6] Woodwark, J. R. 1982. *The Explicit Quad Tree as a Structure for Computer Graphics*. The Computer Journal, vol. 25. hal.383-386.