Aplikasi Kode Huffman Sebagai Kompresor Ukuran File Terhadap Pixel Maze di Virupizxel

Biolardi Yoshogi / 13509035
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
vsio@students.itb.ac.id

***Abstrak*—Pixel maze adalah maze yang dibuat dengan pixel melalui tahap pixel art. Pixel maze yang ada di Virupizxel adalah buatan saya sendiri (kecuali fanart[hanya bagian gambar, bukan pixel mazenya]). Pixel maze ini terdiri dari gabungan beberapa warna seperti gambar objek (buah, penghapus, dan objek lainnya) dengan shadingnya. Agar file yang disimpan lebih kecil dari aslinya, diterapkan Kode Huffman untuk merealisasikannya. Pada dasarnya, dimanfaatkan frekuensi banyaknya warna.**

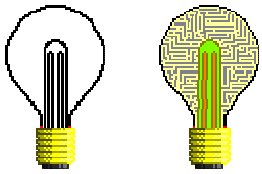
***Kata Kunci*—pixel maze, kode huffman, pohon, struktur diskrit**

I. PENDAHULUAN

Pixel maze adalah maze yang dibuat dengan pixel melalui tahap pixel art. Pixel maze buatan saya yang ada di blog saya Virupizxel (www. virupizxel.blogspot.com/) ini mulai dari yang sederhana hingga yang kompleks berdasarkan banyaknya warna, shading, frame animasi, besar ukuran, dan hal lain yang mungkin ditambahkan ke depannya.
Pixel maze yang saya buat dan berada di Virupizxel itu juga memiliki nama berdasarkan tanggal yang mana duadigit hari berdasarkan potongan kata angka dari bahasa Indonesia, dua huruf dari depan untuk puluhan dan tiga huruf dari depan untuk satuan, bulan berdasarkan tiga potongan huruf dari bahasa Inggris, dan empat digit tahun berdasarkan rumus penamaan pada tabel periodik pada kimia. Contohnya disertai gambar berdasarkan yang sudah diposting:

Gambar 1.1 Binilununaugsanol

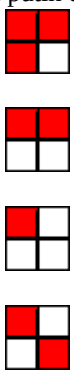
Gambar **1.**2 Binilununocttisat



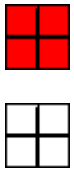
Gambar **1.**3 Binilunundecnoena: a. frame 1; b.frame 2

Pada Gambar **1.**1 (pixel maze pertama yang diposting di blog Virupizxel), Binilununaugsanol menunjukkan “Bi-nil-un-un-” merupakan tahun 2011, “-aug-“ merupakan bulan Agustus, “-sa-nol” merupakan hari satu pada puluhan dan nol pada satuan. Maze ini merupakan maze yang sederhana.
Pada Gambar **1.**2, Binilununocttisat menunjukkan “Bi-nil-un-un-” merupakan tahun 2011, “-oct” merupakan bulan Oktober, “-ti-sat” merupakan hari 3 pada puluhan dan satu pada satuan. Maze ini mulai lebih kompleks karena melibatkan warna lebih dari yang dimiliki maze pada gambar 1 dan mulai menyerupai suatu bentuk yaitu sebuah labu.
Pada Gambar **1.**3, Binilunundecnoena menunjukkan “Bi-nil-un-un-” merupakan tahun 2011, “-dec” merupakan bulan Desember, “-no-ena” merupakan hari 0 pada puluhan dan enam pada satuan. Maze ini mulai lebih kompleks juga karena diimplementasikannya animasi pada pixel maze ini.
Pixel maze di dalam blog saya juga menerapkan peraturan khusus bahwa path maze dalam ruang terpisah 4 pixel hanya boleh diisi 3 pixel path atau wall. Ilustrasi seperti di bawah ini (dengan asumsi bahwa 1 kotak sama dengan 1

pixel, yang path berwarna putih dan wall berwarna merah:



Gambar **1.**4 Yang diperbolehkan

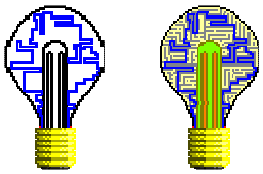


Gambar **1.**5 Yang tidak diperbolehkan

Saya biasanya menyediakan juga solusi pixel maze yang telah saya bikin. Berikut ini solusi pixel maze dari yang tadi telah diperlihatkan di makalah ini:

Gambar 1.6 Solusi Binilununaugsanol

Gambar **1.**7 Solusi Binilununocttisat

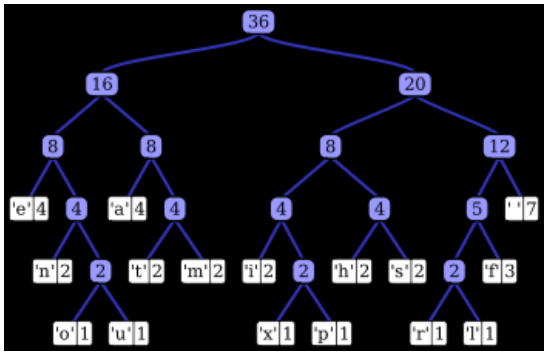


Gambar **1.**8 Solusi Binilunundecnoena: a. frame 1; b.frame 2

Solusi yang diperlihatkan di Gambar 1.6 sampai 1.8 bukanlah solusi optimal sehingga jika ingin menyelesaikannya dengan solusi optimal, hal itu dipersilakan. Akan tetapi, dalam makalah ini tidak akan dibahas bagian itu melainkan kompresi ukuran file menjadi lebih kecil dengan memanfaatkan Kode Huffman dengan memanfaatkan frekuensi banyaknya warna yang terdapat pada pixel maze tersebut. Mengapa kompresi ini diperlukan? Menurut saya agar bisa lebih mengecilkan ukuran file size tanpa merusak detailnya dan mempercepat load gambar ketika di buka di internet melalui browser. Asumsi lain juga, diasumsikan bahwa penyimpanan warna per pixel dan per posisi adalah dengan nilai hex code yang mana disimpan sebagai ASCII code dalam bentuk bit sehingga akan boros penggunaan memori.

II. TEORI DASAR KODE HUFFMAN

Huffman coding adalah sebuah pengkodean entropialgoritmayang digunakan untuk kompresi data lossless. Istilah ini mengacu pada penggunaankode variabel-panjang meja untuk pengkodean simbol sumber (seperti karakter dalam sebuah file) di mana tabel kode variabel-panjang telah diturunkan dalam cara tertentu berdasarkan estimasi probabilitas kejadian untuk setiap kemungkinan nilai simbol sumber.



Gambar 2.1 Pohon Huffman yang dihasilkan dari frekuensi yang tepat dari teks "ini adalah contoh dari sebuah pohon huffman". Frekuensi dan kode masing-masing karakter di bawah. Encoding kalimat dengan kode ini membutuhkan 135 bit, sebagai lawan dari 288 bit jika 36 karakter 8 bit yang digunakan. (Ini mengasumsikan bahwa struktur pohon kode diketahui decoder sehingga tidak perlu dihitung sebagai bagian dari informasi yang dikirimkan.)

Arang	Frek	Kode
ruang	7	111
suatu	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

Gambar 2.2

Huffman coding menggunakan metode tertentu untuk memilih perwakilan untuk setiap simbol, menghasilkan kode awalan (kadang-kadang disebut "awalan kode bebas", yaitu, bit string yang mewakili beberapa simbol tertentu tidak pernah merupakan awalan dari bit string yang mewakili yang lain simbol) yang mengekspresikan simbol sumber yang paling umum menggunakan string bit yang lebih pendek daripada yang digunakan untuk simbol sumber yang lebih umum. Huffman mampu merancang metode kompresi yang paling efisien jenis ini: tidak ada pemetaan lain simbol sumber individu untuk string unik dari bit akan menghasilkan output yang lebih kecil ukuran rata-rata ketika frekuensi simbol yang sebenarnya setuju dengan yang digunakan untuk membuat kode

Untuk satu set simbol dengan distribusi probabilitas seragam dan sejumlah anggota yang merupakan kekuatan dua , Huffman coding adalah setara dengan biner sederhana blok pengkodean , misalnya, ASCII coding. Huffman coding adalah suatu metode yang luas untuk menciptakan kode awalan bahwa istilah "kode Huffman" secara luas digunakan sebagai sinonim untuk "kode awalan" bahkan ketika kode tersebut tidak diproduksi oleh algoritma Huffman itu.

.

Walaupun algoritma asli Huffman adalah optimal untuk simbol-oleh-simbol pengkodean (yaitu aliran simbol yang tidak berhubungan) dengan distribusi masukan dikenal probabilitas, tidak optimal ketika pembatasan simbol-oleh-simbol dijatuhkan, atau ketika fungsi probabilitas massa tidak diketahui, tidakterdistribusi secara identik , atau tidak independen (misalnya, "kucing" adalah lebih umum daripada "CTA").

Teknik Dasar:

- Kompresi

  Teknik ini bekerja dengan menciptakan sebuah pohon biner node. Ini dapat disimpan dalam reguler array yang , ukuran yang tergantung pada jumlah simbol, n. Sebuah node dapat berupa simpul daun atau simpul internal .Awalnya, semua node node daun, yang mengandungsimbol itu sendiri, berat(frekuensi penampilan) dari simbol dan opsional, sebuah link ke sebuah simpulorangtua yang membuatnya mudah untuk membaca kode (secara terbalik) mulai dari simpul daun . Node internalberat badan mengandung simbol, link ke dua node anak dan link opsional ke node induk. Sebagai konvensi umum, sedikit '0 'mewakili mengikuti anak kiri dan bit '1' merepresentasikan anak kanan berikut. Sebuah pohon selesai telah sampai node ndan n daun - 1 node internal.Sebuah pohon Huffman yang menghilangkan simbol yang tidak terpakai menghasilkan panjang kode yang paling optimal.
  Proses ini pada dasarnya dimulai dengan node daun yang mengandung probabilitas simbol yang mereka wakili, maka node baru yang anak-anaknya adalah 2 node dengan probabilitas terkecil dibuat, sehingga probabilitas simpul baru adalah sama dengan jumlah probabilitas anak-anak. Dengan 2 node sebelumnya digabung menjadi satu node (dengan demikian tidak mempertimbangkan mereka lagi), dan dengan node baru yang sekarang dianggap, prosedur ini diulang sampai hanya tinggal satu simpul, pohon Huffman.

- Dekompresi

  Secara umum, proses dekompresi hanya masalah menerjemahkan aliran kode awalan untuk nilai byte individu, biasanya dengan melintasi simpul pohon Huffman dengan simpul sebagai setiap bit dibaca dari input stream (mencapai simpul daun selalu berakhir pencarian untuk itu nilai byte tertentu). Sebelum ini dapat terjadi, bagaimanapun, pohon Huffman harus entah bagaimana direkonstruksi. Dalam kasus yang paling sederhana, dimana frekuensi karakter yang cukup diprediksi, pohon dapat preconstructed (dan bahkan statistik disesuaikan pada setiap siklus kompresi) dan dengan demikian kembali setiap kali, dengan mengorbankan setidaknya beberapa ukuran efisiensi kompresi. Jika tidak, informasi untuk merekonstruksi pohon harus dikirim apriori. Sebuah pendekatan naif mungkin

untuk prepend hitungan frekuensi masing-masing karakter ke aliran kompresi. Sayangnya, overhead dalam kasus seperti itu bisa mencapai beberapa kilobyte, jadi metode ini memiliki sedikit penggunaan praktis. Jika data yang dikompresi menggunakanpengkodean kanonik , model kompresi dapat tepat direkonstruksi dengan hanya B 2 B bit informasi (dimana B adalah jumlah bit per simbol). Metode lain adalah dengan hanya prepend pohon Huffman, sedikit demi sedikit, ke output stream. Sebagai contoh, dengan asumsi bahwa nilai 0 merupakan node induk dan 1 simpul daun, setiap kali yang terakhir ditemui rutin membangun pohon hanya membaca 8 bit berikutnya untuk menentukan nilai karakter yang daun tertentu. Proses ini berlanjut secara rekursif sampai node daun terakhir dicapai; pada saat itu, pohon Huffman sehingga akan setia direkonstruksi.Overhead menggunakan metode tersebut berkisar dari kira-kira 2-320 byte (asumsi alfabet 8-bit). Banyak teknik lain yang mungkin juga. Dalam kasus apapun, karena data terkompresi dapat mencakup tidak terpakai "bit trailing" decompressor harus mampu untuk menentukan kapan untuk berhenti memproduksi output.

III. METODE PEMECAHAN

Dengan metode Kode Huffman, asumsi dibuat suatu program yang mana terdiri dari algoritma berikut ini. Dibuat asumsi terlebih dahulu :

- Masukan adalah file gambar seperti .bmp, .png, .jpg, gif, dan sejenisnya (Akan tetapi ketika dibuka di programnya, kelebihan kompres masing-masing ekstensi file diasumsikan hilang dan tidak terikat)
- Yang diperhitungkan adalah pixel per pixel ketika file telah dimasukan ke program, bukan text terenkripsi yang terdapat pada file gambar (Dengan kata lain, yang diperiksa adalah warnanya langsung)
- Outputnya sebagai file yang terkompresi disimpan dengan ekstensi file .pmc (Pixel Maze Compressed)
- Pixel maze berupa animasi tetap diterima
- Informasi warna berupa hex code (seperti #4422FF, #110044) dan merupakan tipe string, akan tetapi penyimpanannya tetap berupa ASCII code masing-masing per karakter (kecuali simbol pagar [#]) sehingga "4422FF" = "00110100001101000011001000110010010000110010 110" dan 110044 = 00110001001100010011000000110000001101000011010 00 yang mana masing-masing panjangnya menjadi 48 bit.
- Lokasi penempatan pixel per pixel diabaikan dalam kompresi ini
- Pixel Maze didefinisikan dengan tipe bentukan PixelMaze
- Terdapat tipe bentukan FrekuensiTable yang terdiri dari Frekuensi bernilai Integer dan Info bernilai String sebagai kode hex warna
- Terdapat tipe bentukan HuffTable yang terdiri dari Huff bernilai Bit hasil kompresi nilai Info dari FrekuensiTable dan InfoHex bernilai String (akan tetapi, posisi penempatan pixel tidak menggunakan InfoHex melainkan Huff hasil kompresi)

```
Function ColorCount (input _pixelmaze : PixelMaze) → FrekuensiTable
{ Mengembalikan nilai bertipe FrekuensiTable }

Function ToHuff (input _frek : FrekuensiTable) → HuffTable
{ Mengembalikan nilai bertipe HuffTable }

Function Compress (input _huff : HuffTable) → FileGambar
{ Mengembalikan nilai bertipe FileGambar }
```

Sekarang, proses teknisnya:

- Dilakukan input file Pixel Maze ke dalam fungsi ColorCount. Dalam fungsi ini, tiap warna dihitung satu per satu dan warna dicatat. Jumlah warna untuk warna yang sama akan bertambah jika warna tersebut ada sebelumnya.

Frekuensi Warna	**Warna dalam Hex**
Jumlah Warna 1	Warna 1
Jumlah Warna 2	Warna 2
…	…
…	…
Jumlah Warna N - 1	Warna N - 1
Jumlah Warna N	Warna N

Table 3.1 Ilustrasi

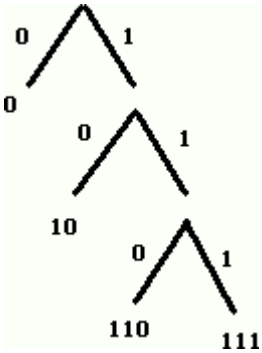
Tabel informasi tersebut akan disimpan dalam bentuk tipe FrekuensiTable yang akan dipakai oleh fungsi lain.

- FrekuensiTable yang telah dihasilkan tadi, dimasukkan lagi ke dalam fungsi ToHuff. Dalam fungsi ini, Informasi warna yang telah dihasilkan akan dikonversikan dalam bentuk bit yang lebih pendek (dimulai dari 0 basis 2) berdasarkan nilai frekuensi yang tercantum dalam FrekuensiTable.

Warna dalam Hex	Kode Huffman
Warna 1	110
Warna 2	0
Warna 3	10
Warna 4	111

Table 3.2 Ilustrasi dengan asumsi terdapat 4 warna yang berbeda dan (jumlah warna 2) > (jumlah warna 3) >

(jumlah warna 1) > (jumlah warna 4)



Gambar 3.1 Pohon Huffman

Hasil konversi ke Kode Huffman akan disimpan dalam tipe HuffTable yang mana isi informasinya berdasarkan Table 3.2 tadi. Hasil ini juga akan digunakan untuk fungsi lainnya.

- Dilakukan tahap kompresi olek fungsi Compress. Satu pixel berisi informasi bukan dalam warna hex lagi melainkan dalam bentuk bit Kode Huffman hasil konversi tadi. Ketika didekompres, maka pewarnaan akan didasarkan pada indeks kode huffman.

IV. ANALISIS KASUS

Dengan sebuah contoh kasus:

Gambar 4.1

Panjang bit : 5756 bit

Hasil fungsi ColorCount :

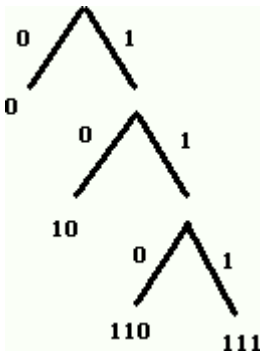
Frekuensi Warna	**Warna dalam Hex**
522	000000
437	FFFFFF
1	FF0000
1	00FF00

Tabel 4.1 ColorCount

Hasil Fungsi ToHuff :

Warna dalam Hex	**Kode Huffman**
000000	0
FFFFFF	10
FF0000	110
00FF00	111

Tabel 4.2 ToHuff



Gambar 4.1 Pohon Huffman

:Jika dikompress, maka panjang bit : 1402 bit

Terjadi perubahan panjang bit dari sebelum dikompresi (5756 bit) ke sesudah dikompresi (1402 bit)

V. KESIMPULAN

Dengan metode kode Huffman, maka jumlah bit yang harus disimpan dapat dikurangi dengan relatif signifikan. Dengan begitu, dapat disimpulkan bahwa kode Huffman berguna untuk mengkompresi ukuran file pixel maze menjadi lebih kecil.

REFERENSI

[1] http://virupizxel.blogspot.com. Waktu Akses: 11:29, 12 Desember 2011

[2] http://en.wikipedia.org/wiki/Huffman_coding. Waktu Akses: 11:29, 12 Desember 2011. Diterjemahkan oleh Google Translate

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, 12 Desember 2011

Biolardi Yoshogi / 13509035