Berbagai Variasi Algoritma Huffman

Erry Febrian Pratama[1], Christoforus[2], William Simanjuntak[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if13065@students.if.itb.ac.id[1],
if13094@students.if.itb.ac.id[2],if13097@students.if.itb.ac.id3

Abstrak

Dalam mengatasi berbagai permasalahan optimasi (*optimization problems*), tentunya tidak hanya mencari sekedar solusi, tetapi juga mencari solusi yang terbaik. Solusi yang terbaik adalah solusi yang bernilai minimum atau maksimum dari sekumpulan alternatif solusi yang ada. Untuk memecahkan persoalan optimasi ini dapat menggunakan berbagai algoritma yang ada, teknik yang mungkin paling populer adalah algoritma Greedy. Salah satu contoh dari algoritma Greedy adalah algoritma Huffman. Algoritma ini biasanya digunakan pada permasalahan pemampatan/kompresi data. Algoritma Huffman ternyata memiliki beberapa variasi. Melalui pembahasan pada makalah ini penulis ingin memaparkan tentang variasi dari algoritma Huffman yang ada serta deskripsi maupun kegunaannya.

***Kata kunci**: algoritma Huffman,varias*

1. Pendahuluan

Kode Huffman pada dasarnya merupakan kode prefiks (*prefix code*) yang merupakan himpunan yang berisi sekumpulan kode biner. Kode prefiks direpresentasikan sebagai pohon biner berlabel dimana setiap sisi diberi label 0 (cabang kiri) atau 1 (cabang kanan). Rangkaian bit yang terbentuk pada setiap lintasan dari akar ke daun merupakan kode prefiks untuk karakter yang berpadanan.

Kode ini pun memiliki berbagai macam variasi antara lain :

1. *Adaptive Huffman coding*
2. *Length-Limited Huffman coding*
3. *N-Ary Huffman Template Algorithm*
4. *Huffman With Unequal Letter Costs*

2. Macam-macam Variasi Kode Huffman

A. *Adaptive Huffman Coding*

Metode *Adaptive* digunakan pada saat pembaharuan (*update*) model algoritma baru baik dari proses kompresi maupun dekompresi.

Konsep Dasar :

Encoder :

```
Initialize_model
Repeat for each character
{
    Encode character
    Update model
}
```

Decoder :

```
Initialize_model
Repeat for each character
{
        Decode character
        Update_model

}
```

Masalahnya adalah bagaimana meng-*update* model algoritma yang terdiri dari menambah jumlah dan mengupdate pohon Huffman. Caranya adalah dengan meng-*update* bagian pohon di mana terjadi proses pemampatan/nirmampat.

Pohon Huffman diinisialisasikan dengan simpul *single* yang dikenal dengan *Not-Yet-Transmitted (NYT) Code* yang dikirimkan setiap kali ditemukan karakter baru. Algoritma bekerja dengan penomoran yang unik pada simpul dengan jumlah daun yang berbeda.

Langkah-langkah peng-*update*-an model :

1. Jika kode yang pertama kali ditemukan adalah NYT, maka tambahkan dua simpul pada simpul NYT. Satu simpul sebagai simpul NYT dan simpul yang lain sebagai daun. Tambahkan jumlah daun. Jika bukan NYT, langsung menuju daun.
2. Jika pada blok tidak terdapat angka tertinggi, tukarkan dengan angka tertinggi pada blok.
3. Tambahkan jumlah dari simpul tersebut.
4. Periksa apakah simpul tersebut merupakan simpul akar. Jika bukan pergi ke simpul *parent*.

B. *Length-Limited Huffman Coding*

Variasi Huffman ini digunakan untuk mendapatkan jarak kedalaman terkecil dari suatu simbol, dengan batasan bahwa panjang masing-masing yang dimasukkan tidak kurang dari nilai konstanta yang diberikan. Metode ini biasanya digunakan pada *GNU gzip*.

Langkah-langkah dalam Metode *Length-Limited Huffman* adalah :

1. Memilih dua atau lebih simbol yang ingin dimampatkan
2. Gabungkan simbol-simbol tersebut dan gantikan dengan *pseudo-symbol* beserta frekuensinya.
3. Lakukan langkah di atas secara iteratif sampai semua simpul yang ada menjadi satu simpul akar.
4. Jika simpul-simpul tersebut memiliki frekuensi yang sama, maka pilihlah simpul dengan kedalaman terpendek.

C. *n-ary Huffman Template Algorithm*

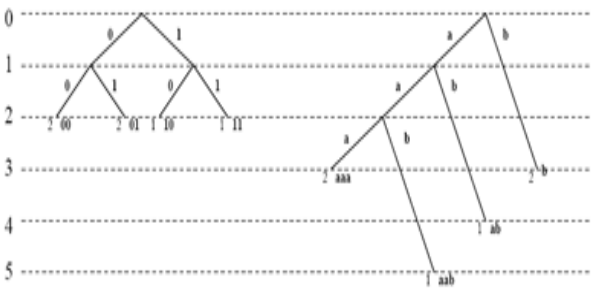
Algoritma ini sebenarnya mirip dengan algoritma Huffman biasa. Bedanya, pohon Huffman yang digunakan dalam algoritma ini memiliki lebih dari dua akar (0 dan 1). Sementara dengan *Huffman template algorithm*, memungkinkan untuk menggunakan ukuran non-numerik (ukuran selain biaya dan frekuensi).

D. *Huffman With Unequal Letter Costs*

Pada metode ini terdapat suatu permasalahan dimana suatu set kode yang terdiri dari beberapa huruf dengan frekuensi kemunculan dan biaya (*cost*) yang berbeda. Metode ini ditujukan untuk mencari kode prefiks (*prefix code)* dan menghitung biaya minimumnya (*minimum cost*). *Prefix Code* merupakan sekumpulan kode yang *prefix-free.* Biaya (*cost*) dari ini merupakan jumlah biaya dari masing-masing huruf pada kode tersebut.

Contoh :

Terdapat dua buah kode di mana masing-masing kode memiliki biaya minimum untuk frekuensi (f1,f2,f3,f4) = (2,2,1,1) tapi pada biaya huruf yang berbeda. Kode {00,01,10,11} memiliki biaya minimum untuk kasus Huffman yang standar **Σ** = {0,1} dimana panjang tiap huruf adalah 1. Kode {aaa, aab, ab, b} memiliki biaya minimum untuk alphabet **Σ =** {a,b} dimana panjang a adalah 1 dan panjang b adalah 3.



Biaya Kode Kiri = 2 · 2 + 2 · 2 + 1 · 2 + 1 · 2 = 12
Biaya Kode Kanan = 2 · 3 + 2 · 3 + 1 · 4 + 1 · 5 = 21.

Langkah-langkah umum metode *Huffman Coding with Unequal Letter Cost* :

1. Mencari kode *K-Prefix* yang optimal.
2. Mengubah kode *K-Prefix* menjadi kode prefiks yang optimal.
3. Setelah didapat kode hasil kemudian hitung biaya (*cost*)-nya.

3. Kesimpulan

Kode Huffman ternyata memiliki banyak variasi, di antaranya adalah *Adaptive Huffman coding, Length-limited Huffman coding, n-ary Huffman template algorithm,* dan *Huffman coding with unequal letter costs*. Berbagai teknik ini dapat digunakan pada aplikasi yang berbeda-beda, tetapi umumnya digunakan untuk pemampatan data. Hal ini wajar saja terjadi karena inti dari seluruh variasi teknik ini adalah sama dengan kode Huffman, yaitu memecahkan permasalahan optimasi data.

4. Daftar Pustaka

1. Golin, Mordecai J., Claire Kenyon, Neal E. Young. *Huffman Coding with Unequal Letter Costs Paper*.
2. http://en.wikipedia.org/wiki/Huffman_coding, diakses pada hari Rabu, 17 Mei 2006, pukul 22.15 WIB.
3. http://www.nist.gov/dads/HTML/karyHuffman.html, diakses pada hari Jumat, 19 Mei 2006, pukul 15.00 WIB.
4. http://www.nist.gov/dads/HTML/karyTree.html, diakses pada hari Jumat, 19 Mei 2006, pukul 15.05 WIB.
5. Munir, Rinaldi. *Diktat Kuliah IF2252 Strategi Algoritmik*. 2006. Bandung: Instituk Teknologi Bandung