HUFFMAN CODING

Anhar
anhar19@gmail.com

Huffman Coding

- Optimal code pertama dikembangkan oleh David Huffman
- Utk sumber $S = \{x_1, \ldots, x_n\}$; Probabilitas $P = \{p_1, \ldots, p_n\}$; Codewords $\{c_1, \ldots, c_n\}$; dan Panjang $\{l_1, \ldots, l_n\}$. Terdapat optimal binary prefix code dengan karakteristik:

  Teorema:
  (a) Jika $p_j > p_i$, *maka* $l_j \leq l_i$
  (b) Dua codeword dari dua simbol dg probabilitas terendah mempunyai panjang yg sama
  (c) Dua codeword terpanjang identik kecuali pada digit terakhir

Huffman Coding

HuffmanAlgorithm()

for *each letter create a tree with a single root node and order all trees according to the probability of letter occurrence;*

while *more than one tree is left*

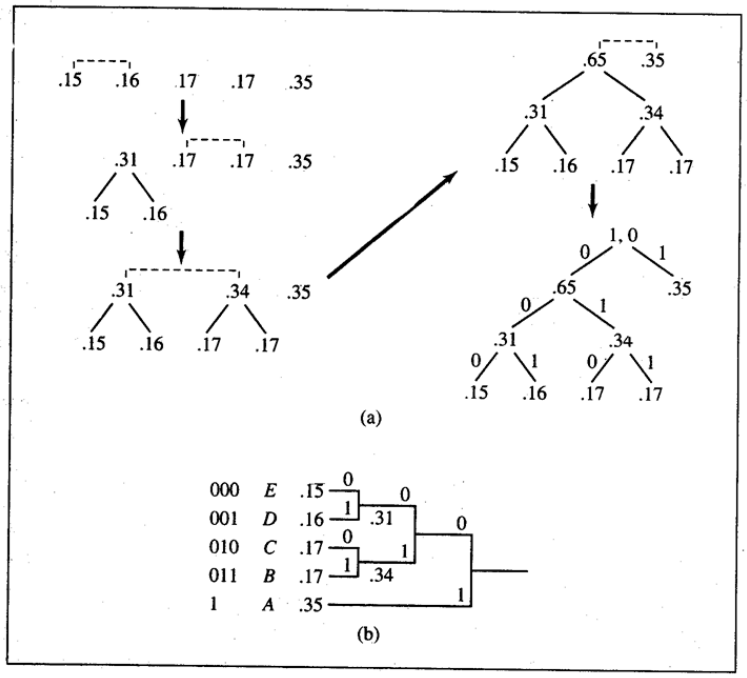
take the two trees t_1, t_2 *with the lowest probabilities* p_1, p_2 and *create a tree with probability in its root equal* to $p_1 + p_2$ *and with* t_1 *and* t_2 *as its subtrees;*

associate 0 *with each left branch and* 1 *with each right branch;*

create a unique codeword for each letter by traversing the tree from the root to the leaf containing the probability corresponding to this letter and putting all encountered 0*s and* 1*s together;*

Contoh:

x_i	p_i
A	0,35
B	0,17
C	0,17
D	0,16
E	0,15



Huffman Coding

- Dari Huffman tree dapat dibuat tabel codeword:

A	1
B	011
C	010
D	001
E	000

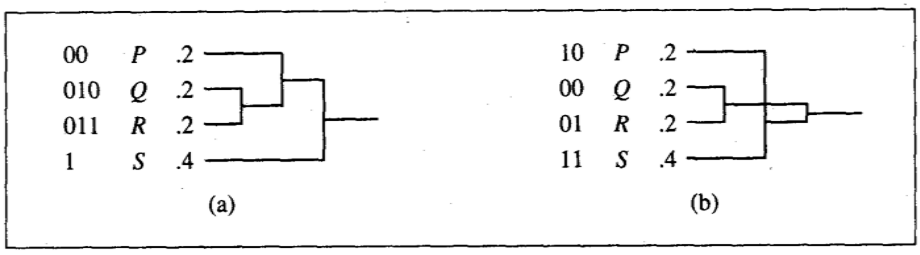
L_{Huff} = 0,35*1 + 0,17*3 + 0,17*3 + 0,16*3 + 0,15*3 = 2,3

H(S) = 2,23284

Efisiensi = (2,23284/2,3) x 100 % = 97,08%

Huffman Coding

- Tergantung pada bagaimana memilih probabilitas terendah saat membangun Huffman tree
  → Huffman tree tidak unik
- Namun, panjang rata-rata codeword selalu sama utk tree yang berbeda

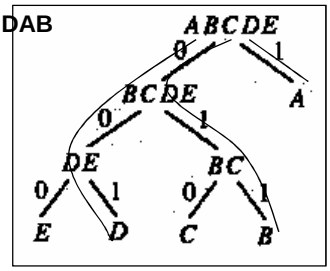


Huffman Coding

- Proses coding: mentransmisikan codeword sesuai dg simbol-simbol yg akan dikirim, mis ABAAD → 101111001
- Untuk decode message, konversi tabel harus diketahui penerima → dp dibangun Huffman tree

0011011 → DAB

A 1
B 011
C 010
D 001
E 000



- Masalah: pengirim (encoder) dan penerima (decoder) harus menggunakan coding (Huffman tree) yang sama

Huffman Coding

Bagaimana encoder memberi tahu decoder utk menggunakan code yang mana:

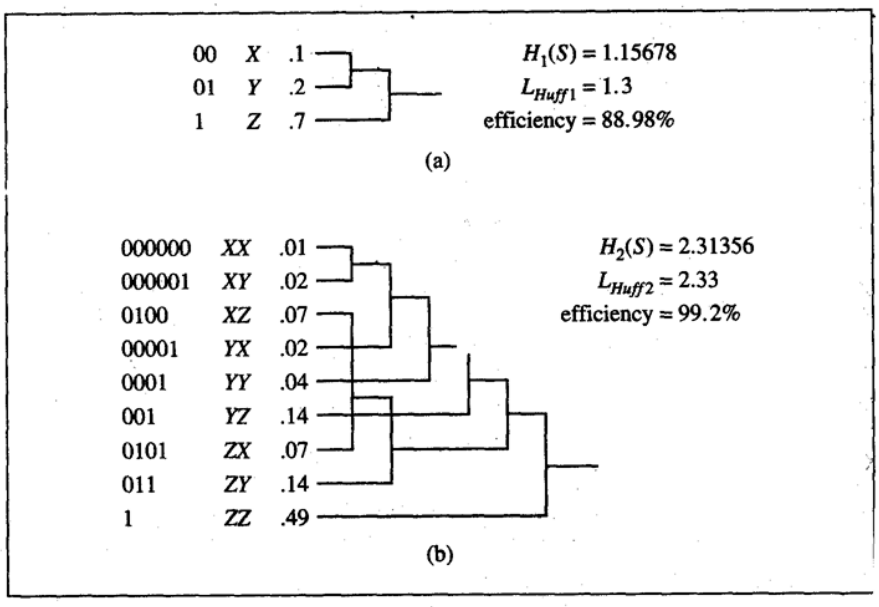
- Baik encoder dan decoder sudah sepakat sebelumnya utk menggunakan Huffman tree tertentu sebelum terjadi pengiriman message
- Encoder membangun Huffman tree yang baru (*fresh*) setiap message baru akan dikirimkan, dan mengirimkan tabel konversi bersama-sama dg message
  → Keuntungannya terasa jika digunakan utk message yang besar

Huffman Coding

Apakah masih ada ruang perbaikan utk Huffman Coding?

- Semua kemungkinan Huffman tree akan memberikan panjang rata-rata yang sama
- Namun ingat Shannon's Fundamental Theorem:
  - Semua kemungkinan pasangan simbol dp digunakan untuk membangun Huffman tree
    → kompresi data dp ditingkatkan
  - Namun perbaikan harus 'dibayar' dg 'Tabel Konversi' yang lebih besar

Contoh:



Huffman Coding

- Menggunakan pasangan simbol
  - Jika message besar dan simbol sedikit → penambahan ukuran tabel tidak signifikan
  - Namun utk jumlah simbol yang banyak → ukuran tabel akan sangat besar → umumnya masih berguna (memberikan keuntungan)
    - 26 huruf Inggris → 676 pasangan
    - 96 karakter (dari 126 kode ASCII) → 9216 pasangan
    - Tidak semua pasangan mungkin muncul (mis. XQ, ZK) → pasangan dp dieliminir
- Percobaan menggunakan: (I) single character, (II) Single character + 100 group yg paling sering muncul, (III) Single character + 512 group yg paling sering muncul (Rubin, 1976)

Compression Rate	I	II	III
English Text	40%	49%	55%
PL/I Program	60%	73%	71%
Photograpic Image	50%	52%	62%

Huffman Coding dengan Persyaratan Memori Kecil

- Ukuran Huffman tree proporsional dg jumlah simbol yg dikodekan
- Ukuran tumbuh Jika pasangan, triples atau yg lebih besar *n-tuples* digunakan
- Panjang code juga tumbuh sesuai dg jumlah simbol
- Makin panjang codeword, makin jarang digunakan (tapi tetap hrs disimpan dlm tree)
- Usulan C.C Weaver dikembangkan lebih jauh oleh Michael Hankamer: simbol yg jarang digunakan disatukan dalam satu set disebut ELSE (simbol-simbol dlm ELSE tdk dikodekan dg Huffman)
- Jika simbol dari ELSE akan dikodekan, Huffman codeword utk ELSE dikirimkan diikuti dg simbol itu sendiri (mis. ASCII)

Huffman Coding: WeavenHankamer Algorithm

WeaverHankamerAlgorithm()

divide source letters $S = \{x_1, \ldots, x_n\}$, *where each* x_i *is L bits long, into two sets,*

$S_1 = \{x: p(x) > 1/2^L\}$ and $S_2 = \{x: p(x) \leq 1/2^L\}$;

$p(\text{ELSE}) = \sum_{x \in S_2} p(x)$;

create Huffman code for the set $S_0 = S_1 \cup \{\text{ELSE}\}$;

the codeword for an element $x \in S_2$ *is the Huffman codeword of* ELSE *concatenated with* x;

Huffman Coding: WeavenHankamer Algorithm

Contoh:

ASCII: 128 simbol panjang 7-bit. Dua dari simbol *x* dan *y* muncul dg prob. ¼, dan 126 simbol lainnya dg prob. 1/252. Huffman coding memberikan dua codeword dg panjang 2, dua codeword dg panjang 7 dan 124 codeword dg panjang 8

- L_{Huff} = 2.(1/4).2 + 2.(1/252).7 + 124.(1/252).8 = 4,992 bit/simbol
- Entropi: H(S) = -2.(1/4)lg(1/4) – 126.(1/252).lg(1/252) = 4,989 bit/simbol
- Jumlah bit dari semua codeword yg dibangkitkan algortima Huffman = 2*2 + 2*7 + 124*8 = 1010 bit

Dg WeaverHankamerAlgorithm(), S_0 = {x,y,ELSE}.

Codeword Huffman: 00 utk *x*, 01 utk *y* dan 1 utk ELSE

- Hanya diperlukan 5 bit, bandingkan dg 1010 bit sebelumnya
- Namun,

  L_{WH} = 2.(1/4).2 + 126.(1/252).(1+7) = 5 bit/simbol
  (lebih besar dari 4,992)

  → Modifikasi Hankamer pada Huffman coding adalah tidak optimal (cukup dekat mendekati optimal)

Adaptive Huffman Coding

- Pada pembahasan sebelumnya diasumsikan probabilias simbol sudah diketahui sebelumnya.
- Bagaimana mengetahui probabilitas simbol?
  - Di dapat dari sampel yg besar (mis. utk message bhs Inggris, ambil 10 juta karakter: Encyclopedia + novel + artikel majalah + koran + …)
    → tidak akan efektif utk mengkodekan file tertentu (walaupun dlm bhs Inggris), mis, LISP atau Java code vs Puisi, dll.
    → Sebelum coding 'baca' dulu file/message sebelum membuat tabel konversi → perlu preprocessing (waktu/delay)
    → Dalam kasus tertentu tdk keseluruhan file/message utuh bisa didapat saat pengkodean utk transmisi (mis. pengiriman line-by-line mode)
- Adaptive Huffman coding dikembangkan oleh Newton Faller dan Robert G Gallager dan diperbaiki oleh Donald Knuth dan Jeffrey S. Vitter
- Adaptive coding adalah metoda yang secara dinamis meng-update dua Huffman tree identik di Encoder dan Decoder (didasarkan pada informasi text yg sudah dikirim sampai saat ini)

Adaptive Huffman Coding

- Definisi:

  Suatu binary tree dimana node-nya mempunyai counter frekuensi (kemunculan) memiliki *sibling property* jika tiap node (kecuali *root*) saat scanning (*breadth-first-right-to-left* tree) membangkitkan *list* dari node-node yg mempunyai harga counter frekuensi yg tidak meningkat

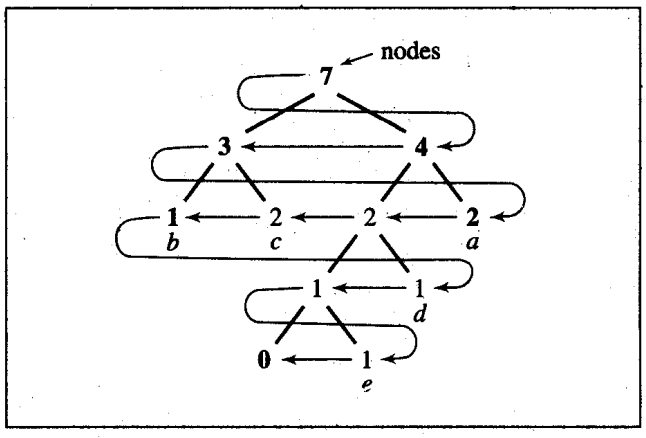
- Teorema (Faller-Gallager Theorem)

  Suatu tree dengan *sibling property* adalah suatu Huffman tree

Adaptive Huffman Coding

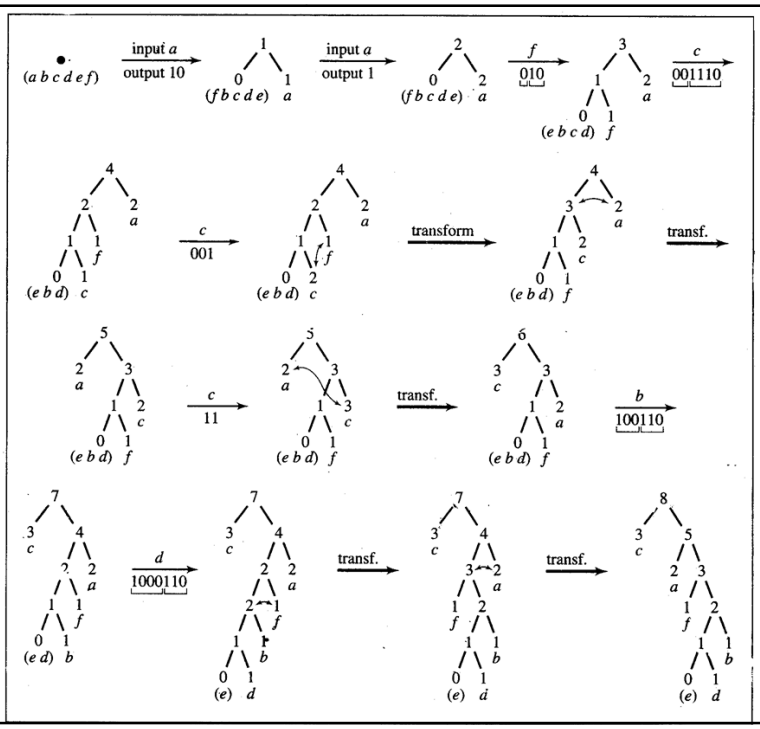
- Pada adaptive Huffman coding, tree mencakup counter utk tiap simbol, dan suatu counter di-update setiap input simbol yg sesuai dikodekan
- Memeriksa apakah *sibling property* dipertahankan menjamin Huffman tree yg sedang dibangun tetap Huffman tree
- Jika *sibling property* dilanggar, tree harus direstrukturisasi utk mengembalikan property ini
- Algoritma memp. link list nodes berisi node-node dari tree diurut berdasarkan scanning *breadth-first-right-to-left* tree
- $Block_i$ adalah bagian dari list dimana tiap node memp. frekuensi *i*, dan node pertama dari tiap block disebut *leader*

Adaptive Huffman Coding



- List nodes = (**7 4 3 2** 2 2 **1** 1 1 1 **0**)
- Memp 6 block – $block_7$, $block_4$, $block_3$, $block_2$, $block_1$ dan $block_0$ dg *leader* terlihat **boldface**
- Simbol-simbol yg belum digunakan disimpan pd node dg frek. 0 → 0-*node*

```
FGK Dynamic Huffman Encoding(symbol s)
    p = leaf that contains symbol s;
    c = the Huffman codeword for s;
    if p is the 0-node
        c = c concatenated with the number of 1s representing position of s in 0-node
            and with 0;
        write the last symbol in the 0-node over s in this node;
        create a new node q for symbol s and set its counter to 1;
        p = a new node to become the parent of both 0-node and node q;
        counter(p) = 1;
        include the two new nodes to nodes;
    else increment counter(p);
    while p is not the root
        if p violates the sibling property
            if the leader of the block; that still includes p is not parent(p)
                swap p with the leader;
        p = parent(p);
        increment counter(p);
    return codeword c;
```



PR (Tugas-2):
kumpulkan minggu depan (waktu kuliah)

1. Cari entropy H(S) untuk sumber S = {X, Y, Z} dg masing-masing probabilias P = {0.05, 0.05, 0.9} dan bandingkan dg rata-rata panjang codeword Huffman L_{Huff} dihitung dg single letter dan pasangan letter. Apakah L_{Huff} cukup memuaskan mendekati H(S)? Kalau tidak adakah ruang utk perbaikan?

2. Untuk sumber 128 simbol tujuh-bit, empat mempunyai probabilitas 1/16, dua belas 1/48, empat puluh delapan 1/192 dan sisanya enampuluh empat 1/256. Cari panjang rata-rata codeword dan jumlah bit digunakan oleh semua codeword menggunakan Algoritma Huffman (statis) dan modifikasi Hankamer dari Huffman coding.

3. Tentukan deretan bit yang dikirimkan untuk mengkodekan deretan becaaccdef dari sumber (a,b,c,d,e,f) menggunakan Adaptive Huffman Coding. Tunjukan (gambarkan) langkah-langkah proses encoding-nya.