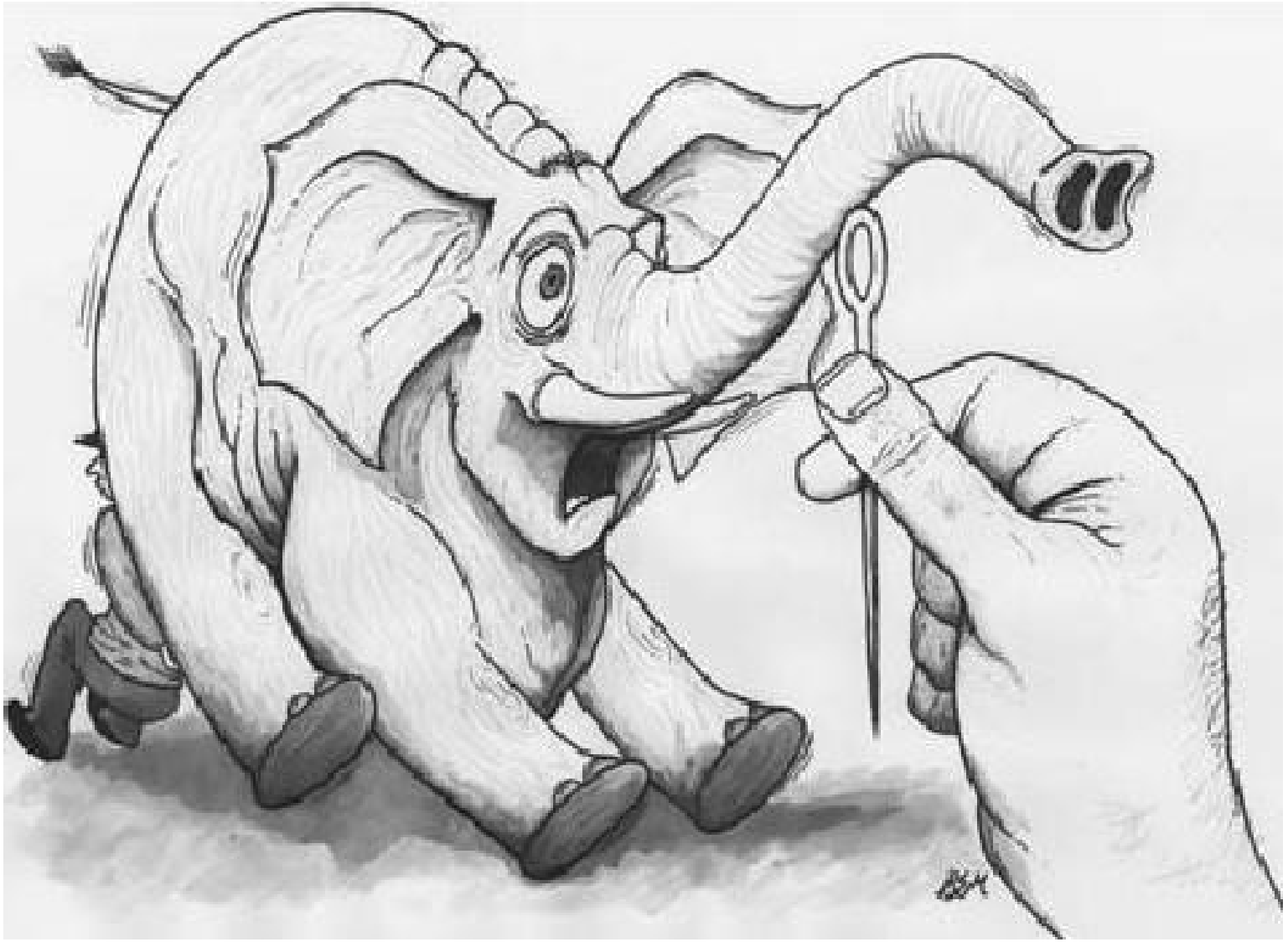
Kompresi Data



Contoh : (1)

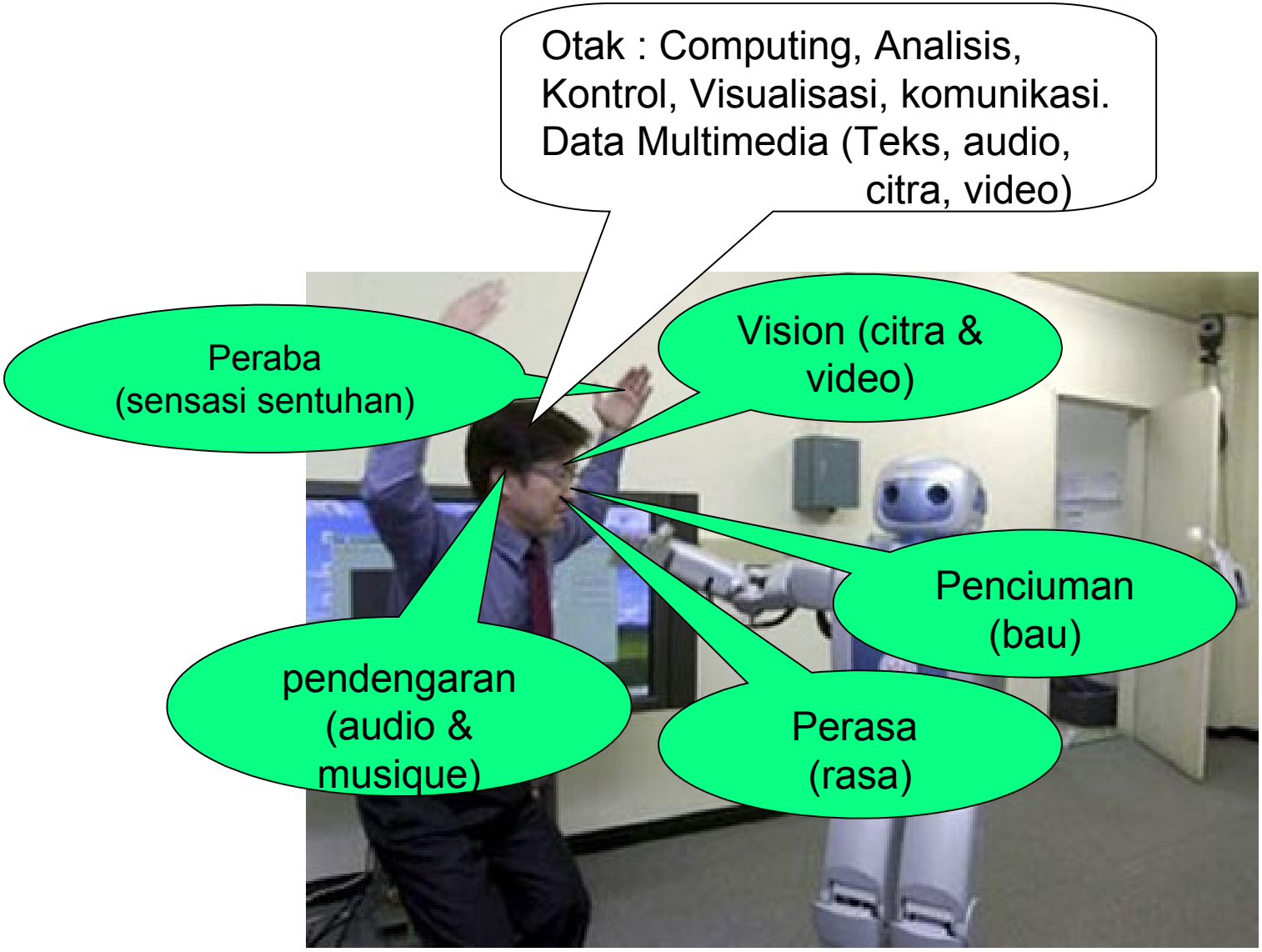
- Contoh kebutuhan data selama 1 detik pada layar resolusi 640 x 480 :
  - Data Teks
    - 1 karakter = 2 bytes (termasuk karakter ASCII Extended)
    - Setiap karakter ditampilkan dalam 8 x 8 pixels
    - Jumlah karakter yang dapat ditampilkan per halaman =

$$\frac{640 \times 480}{8 \times 8} = 4800 \text{ karakter}$$

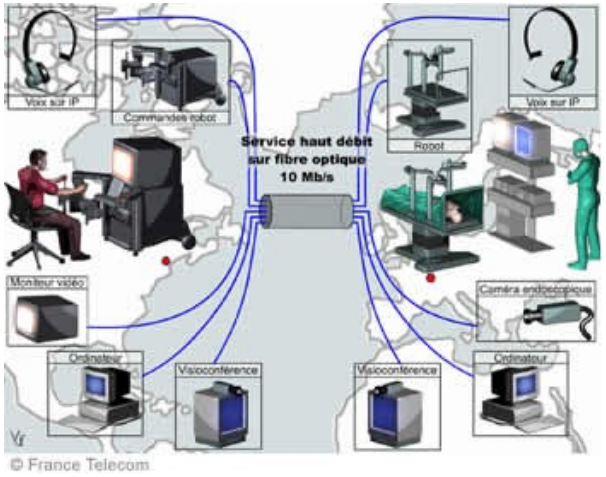
  - Kebutuhan tempat penyimpanan per halaman = 4.800 × 2 byte = 9.600 byte = 9,375 Kbyte

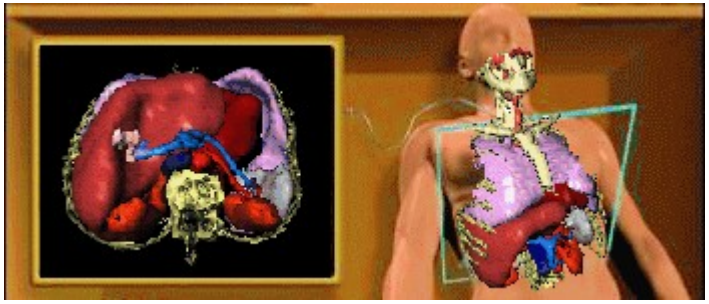
Contoh : (2)

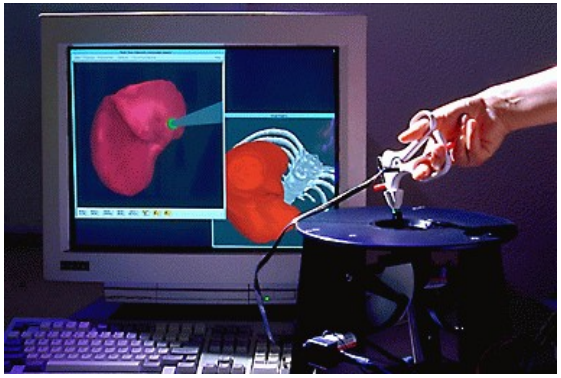
- Contoh kebutuhan data selama 1 detik pada layar resolusi 640 x 480 :
  - Color Display
    - Jenis : 256, 4.096, 16.384, 65.536, 16.777.216 warna
    - Masing-masing warna pixel memakan tempat 1 byte
    - Misal 640 x 480 x 256 warna x 1 byte = 307.200 byte = 300 KByte

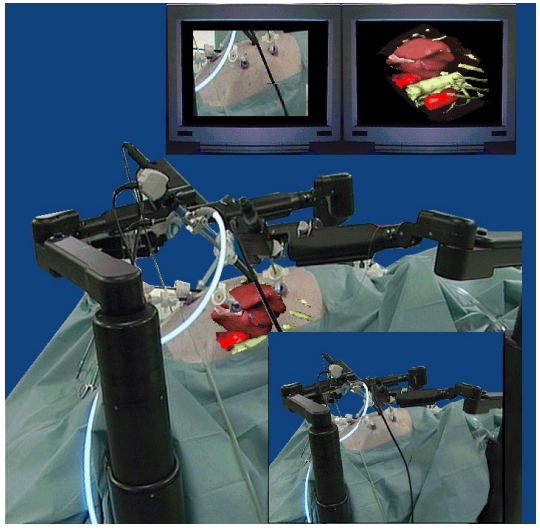

Otak : Computing, Analisis, Kontrol, Visualisasi, komunikasi. Data Multimedia (Teks, audio, citra, video)
Peraba (sensasi sentuhan)
Vision (citra & video)
Penciuman (bau)
pendengaran (audio & musique)
Perasa (rasa)

Aplikasi kedokteran :

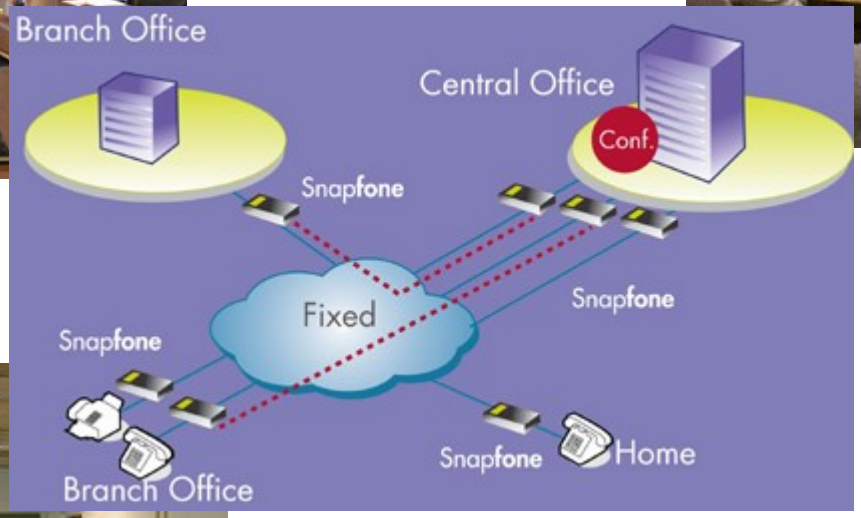








Aplikasi Video conference



- Bandwidth jaringan : terbatas dan mahal
- Delay waktu transmisi : sangat besar
- Volume data multimedia : sangat besar

Kompresi Data

- Suatu teknik pemampatan data sehingga diperoleh file dengan ukuran yang lebih kecil daripada ukuran aslinya.

- Proses pengubahan sekumpulan data menjadi bentuk kode dengan tujuan untuk menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data (memperkecil kebutuhan bandwidth).

Kompresi Data

- Contoh kompresi sederhana yang biasa kita lakukan misalnya adalah menyingkat kata-kata yang sering digunakan tapi sudah memiliki konvensi umum. Misalnya: kata “*yang*” dikompres menjadi kata “*yg*”.

- Pengiriman data hasil kompresi dapat dilakukan jika pihak pengirim/yang melakukan kompresi dan pihak penerima memiliki aturan yang sama dalam hal kompresi data

Kompresi Data

- Pengkodean (coding) data atau informasi yang memiliki redundancy (kerangkapan) kedalam jumlah bit yang lebih kecil.

- Beberap contoh coding : Huffman, arithmetic, statistik, RLE (*run-length encoding*), Lempel-Ziv, Lempel-Ziv-Welch

Lossless compression :

Huffman Coding (David Albert Huffman 1952)

- Berbasis pada perhitungan statistik
- Mengunakan bantuan pohon biner
- Data yang frekuensi munculnya paling banyak di kode dengan jumlah bit terkecil
- Data yang frekuensi munculnya paling sedikit dikode dengan jumlah bit terbesar

Lossless compression :

Huffman Coding

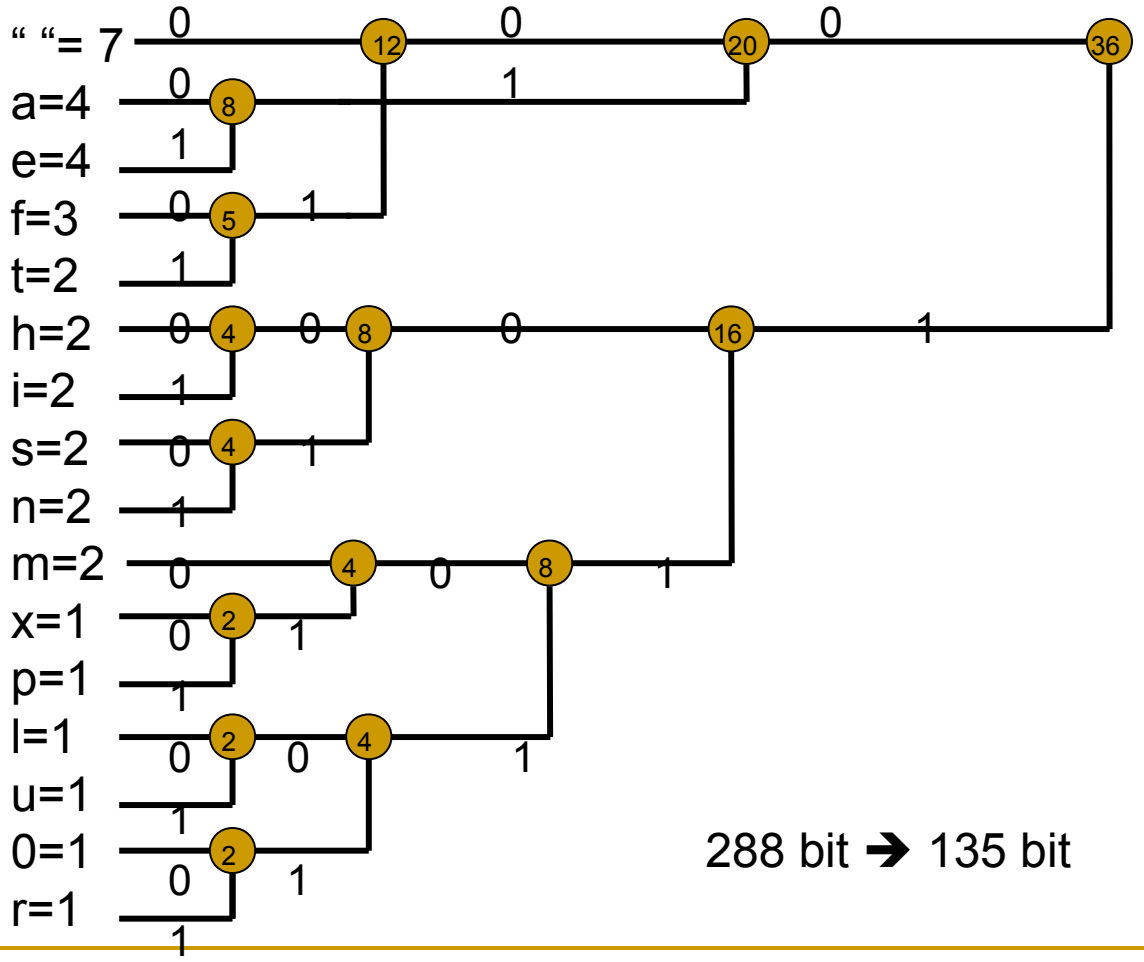
Contoh : "this is an example of a huffman tree"

- statistik munculnya karakter : “ “= 7, a=4, e=4, f=3, t=2, h=2, i=2, s=2, n=2, m=2, x=1, p=1, l=1, u=1, 0=1, r=1.

- Probabilitas munculnya karakter : “ “= 0.1944…, a=e=0.1111…, f=0.0833…, t=h=i=s=n=m=0.0556, x=p=l=u=o=r=0.0278.

Lossless compression :

- Huffman Coding
- pohon biner :



288 bit ➔ 135 bit

" "= 000
a = 010
e = 011
f = 0010
t = 0011
h = 1000
i = 1001
s = 1010
n = 1011
m = 1100
x = 11010
p = 11011
l = 11100
u = 11101
o = 11110
r = 11111

- this is an example of a huffman tree

Dikodekan :

T	: 0011	« _ »	: 000		
H	: 1000	O	: 11110		
I	: 1001	F	: 0010		
S	: 1010	« _ »	: 000		
« _ »	: 000	A	: 010		
I	: 1001	« _ »	: 000		
S	: 1010	H	: 1000		
« _ »	: 000	U	: 11101		
A	: 010	F	: 0010		
N	: 1011	F	: 0010		
« _ »	: 000	M	: 1100		
E	: 011	A	: 010		
X	: 11010	N	: 1011		
A	: 010	« _ »	: 000		
M	: 1100	T	: 0011		
P	: 11011	R	: 11111		
L	: 11101	E	: 011		
E	: 011	E	: 011	Total bit = **135** bit	

Tabel Kode
" "= 000
a = 010
e = 011
f = 0010
t = 0011
h = 1000
i = 1001
s = 1010
n = 1011
m = 1100
x = 11010
p = 11011
l = 11100
u = 11101
o = 11110
r = 11111

- **0011 1000 1001 1010 000 1001 1010 000 010 1011 000 011 11010 010 1100 11011 11101 011 000 11110 0010 000 010 000 1000 11101 0010 0010 1100 010 1011 000 0011 11111 011 011**

Latihan

- “kukikiskikiskukukakikakeku”

- Lakukan Kompresi dengan Algoritma Huffman Coding !

Latihan

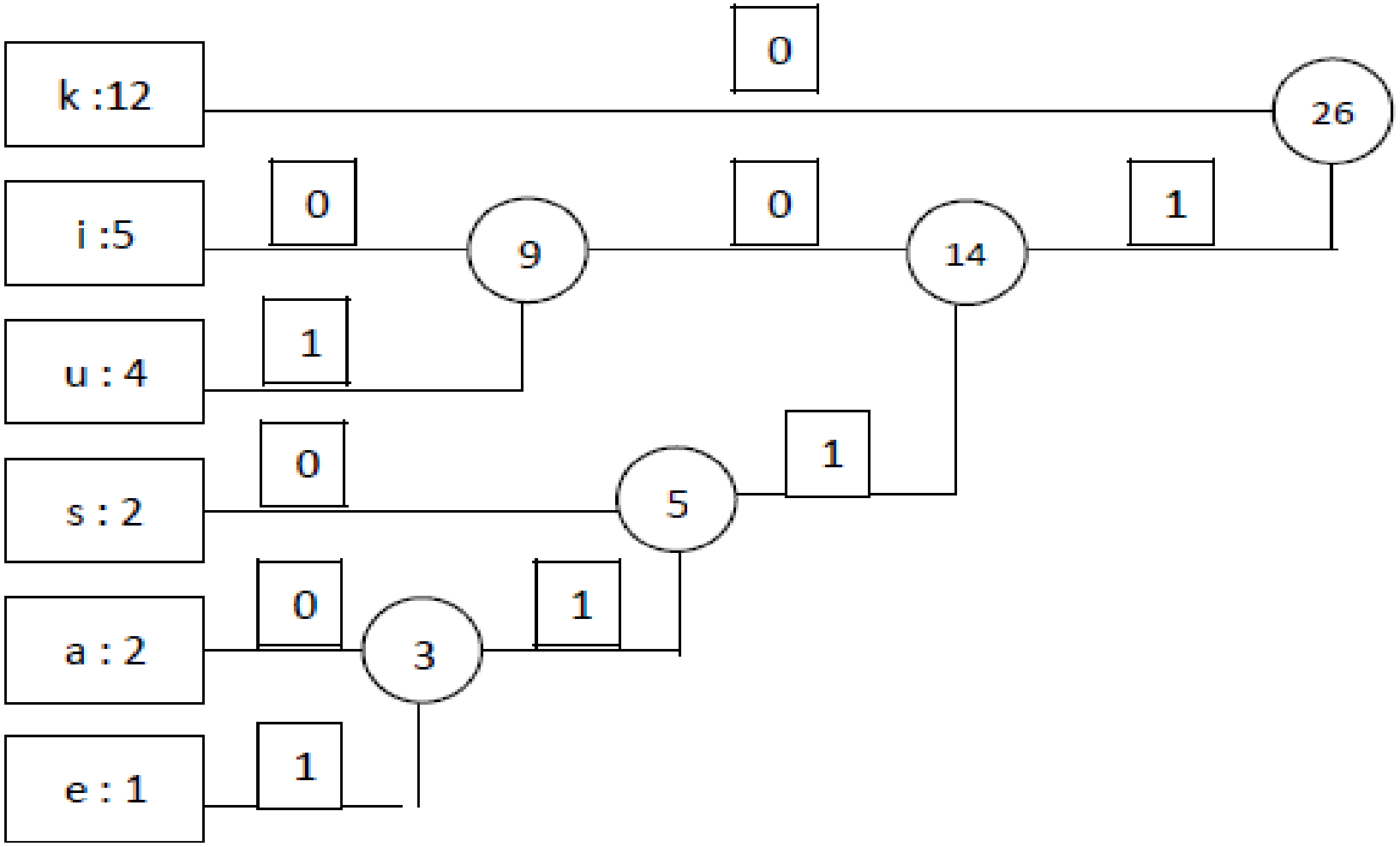
- statistik munculnya karakter :

  k = 12, i = 5, u = 4, s = 2, a = 2, e = 1

  Total = 26
- Probabilitas munculnya karakter :

  k = 12/26 = 0.461, **i** = 5/26 = 0.192, **u** = 4/26 = 0.154, **s** dan **a** = 2/26 = 0.077, **e** = 1/26 = 0.038

- Pohon Biner :



k = 0, i = 100, u = 101, s = 110, a = 1110, e = 1111

01010100010011001000100110010101010111001000111001111010 1

Loseless Compression :

Huffman Coding

- **statik** : code setiap karakter ditentukan langsung oleh algoritma (contoh : teks berbahasa Prancis, dimana frekuensi kemunculan huruf e sangat banyak sehingga code bitnya kecil.

- **semi-adaptatif** : teks harus dibaca terlebih dulu untuk menghitung frekuensi munculnya setiap karakter, kemudian membentuk pohon binernya.

Loseless Compression :

Huffman Coding

- **adaptatif** : Metode ini memberikan rasio kompresi yang tinggi karena pohon biner dibentuk secara dinamik mengikuti tahapan compresi. Namun dari sisi kecepatan eksekusi membutuhkan waktu yang lebih lama karena setiap saat pohon binernya akan berubah mengikuti perubahan frekuensi munculnya setiap karakter.

Lossless compression :

- Kelemahan Huffman Coding

- Bila frekuensi munculnya setiap karakter dalam suatu dokumen adalah sama semua.
- File kompresinya bisa sama atau lebih besar dari file aslinya
- Solusi yang mungkin adalah kompresi per blok karakter dari dokumen tersebut

Lossless compression :

- Run-length encoding

- RLE coding telah diaplikasikan khususnya pada scanner hitam putih (biner)
- Prinsip dasarnya adalah menghitung jumlah/panjang data yang sama dalam serangkain data yang akan dikompres
- Contoh pada dokumen hitam H (tulisan) dan putih P (latar belakang dokumen), berikut misalnya data pada satu baris dokumen yang direpresntasikan dalam pixel :

PPPPPPPPPPPPHPPPPPPPPPPPPPPHHHPPPPPPPPPPPPPPPPPPPPPHPPPPPPPPPP

- Bentuk kompresinya adalah : 12P1H14P3H23P1H11P

Lossless compression :

- **Lempel-Ziv-Welch coding**

- Asumsi setiap karakter dikode dengan 8 bit (nilai code 256)
- Membentuk table gabungan karakter (kata dalam kamus)
- Tabel ini menyimpan kode kata dengan jumlah bit tetap (umumnya maksimum 12 bit)
- Contoh : TOBEORNOTTOBEORTOBEORNOT

Algoritma kompresi LZW :

c	w	wc	output	Kamus
T	<NIL>	T		
O	T	TO	T	TO = <256>
B	O	OB	O	OB = <257>
E	B	BE	B	BE = <258>
O	E	EO	E	EO = <259>
R	O	OR	O	OR = <260>
N	R	RN	R	RN = <261>
O	N	NO	N	NO = <262>
T	O	OT	O	OT = <263>
T	T	TT	T	TT = <264>
O	T	TO		
B	TO	TOB	<256>	TOB = <265>
E	B	BE		

c	w	wc	output	Kamus
O	BE	BEO	<258>	BEO = <266>
R	O	OR		
T	OR	ORT	<260>	ORT = <267>
O	T	TO		
B	TO	TOB		
E	TOB	TOBE	<265>	TOBE = <268>
O	E	EO		
R	EO	EOR	<259>	EOR = <269>
N	R	RN		
O	RN	RNO	<261>	RNO = <270>
T	O	OT		
	OT		<263>	

Lossless compression :

- Lempel-Ziv-Welch coding

- Contoh : TOBEORNOTTOBEORTOBEORNOT

Hasil pengkodean :

TOBEORNOT<256><258><260><265><259><261><263>

Jumlah bit 16 * 9 = 144 bits.

Algoritma Rekonstruksi LZW :

TOBEORNOTTOBEORTOBEORNOT

k	w	input	w+input	output	Kamus
T		T		T	
O	T	O	TO	O	TO = <256>
B	O	B	OB	B	OB = <257>
E	B	E	BE	E	BE = <258>
O	E	O	EO	O	EO = <259>
R	O	R	OR	R	OR = <260>
N	R	N	RN	N	RN = <261>
O	N	O	NO	O	NO = <262>
T	O	T	OT	T	OT = <263>
<256>	T	TO	TT	TO	TT = <264>
<258>	TO	BE	TOB	BE	TOB = <265>
<260>	BE	OR	BEO	OR	BEO = <266>
<265>	OR	TOB	ORT	TOB	ORT = <267>
<259>	TOB	EO	TOBE	EO	TOBE = <268>
<261>	EO	RN	EOR	RN	EOR = <269>
<263>	RN	OT	RNO	OT	RNO = <270>

Tugas

- “kukikiskikiskukukakikakeku”

- Lakukan Kompresi dengan algoritma LZW
- Kirimkan ke email : narendro@staff.gunadarma.ac.id
- Sebelum tanggal : 6 Maret 2010

The End