Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Nama :…………………………
NIM :…………………………
T.tangan:…………………………

Solusi Kuis ke-4 IF2091 Struktur Diskrit (3 SKS) – Pohon, Kompleksitas Algoritma
Dosen: Rinaldi Munir, Harlili
Rabu, 5 Desember 2012
Waktu: 60 menit

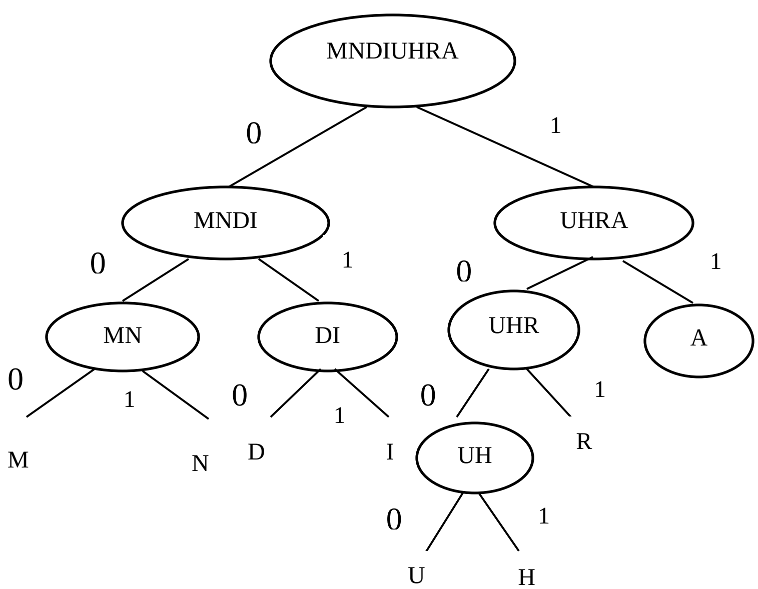
1. Proses pembuatan pohon Huffman dalam proses kompresi kalimat “Rahma dan Rara ada di rumah ini” mengikuti aturan: simbol dengan peluang lebih kecil sebagai anak kiri (berlabel 0) dan simbol dengan peluang lebih besar menjadi anak kanan berlabel 1. Tuliskan kode Huffman untuk setiap huruf yang merepresentasikan kalimat diatas. (spasi diabaikan). (20)

Jawaban:

Tabel frekuensi kemunculan tiap huruf tertera sebagai berikut:

Huruf	Frekuensi
R	4/25
A	8/25
H	2/25
M	2/25
D	3/25
N	2/25
I	3/25
U	1/25

Pohon Huffman yang terbentuk diperlihatkan sebagai berikut:



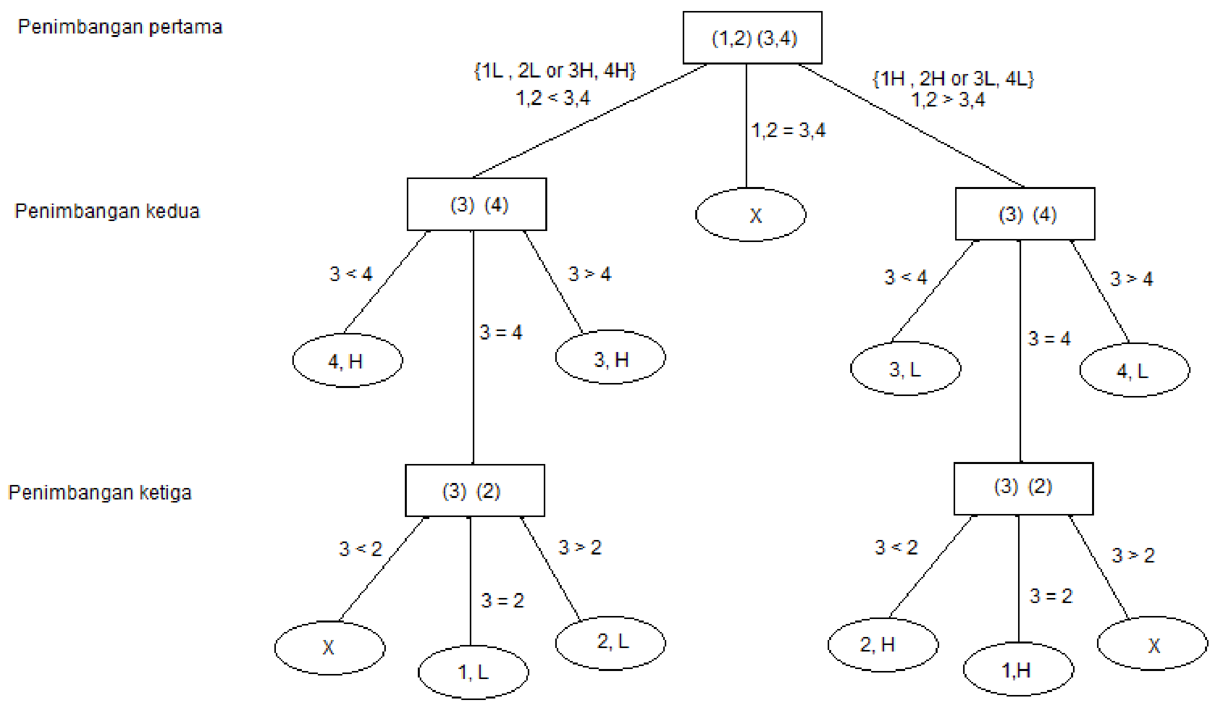
Dari konstruksi pohon diatas didapat hasil pengkodean tiap huruf sebagai berikut:

Huruf	Frekuensi
R	101
A	11
H	1001
M	000
D	010
N	001
I	011
U	1000

2. Diberikan 4 buah koin yang identik antara satu dengan yang lainnya, namun ternyata satu di antaranya adalah koin yang palsu. Koin yang palsu memiliki berat yang berbeda dengan koin yang asli, namun tidak diketahui apakah koin palsu tersebut lebih berat / lebih ringan daripada yang asli. Untuk menentukan mana yang palsu, diberikan sebuah timbangan, namun hanya dapat digunakan sebanyak 3 kali penimbangan. Dengan menggunakan decision tree, tentukan semua kemungkinan koin yang palsu berdasarkan penimbangan, dan apakah koin palsu tersebut lebih berat / lebih ringan dari yang asli. (20)

Jawaban:

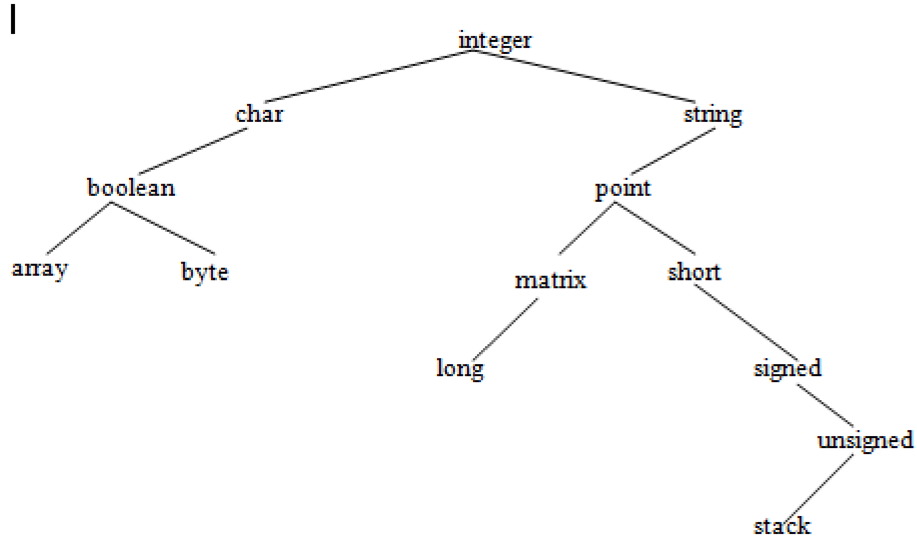
Semua kemungkinan yang ada, dapat direpresentasikan menggunakan pohon keputusan di bawah ini :



Dengan L artinya lighter (koin palsunya lebih ringan), dan H adalah heavier (koin palsunya lebih berat). X menyatakan kondisi yang tidak mungkin terjadi.

3. Buat pohon pencarian biner (BST) sesuai urutan kamus untuk string berikut !
“integer”, “char”, “string”, “boolean”, “point”, “matrix”, “array”, “short”, “long”, “byte”, “signed”, “unsigned”, “stack” (15)

Jawaban:



4. Apakah $T(n) = 2n.\log(n^2+1)+n^2 \log n$ merupakan $O(n^2 \log n)$? Buktikan jawaban Anda! (10)

Jawaban:

Misalkan, f(n) = 2n, g(n) = $\log(n^2+1)$, h(n) = n^2, i(n)=log n

f(n) = n + 5, maka f(n) = O(n)

Untuk g(n) = $\log(n^2+1)$,

$$\log(n^2+1) <= \log(2n^2) = \log 2 + \log n^2 = \log 2 + 2 \log n <= \log n$$

maka g(n) = O(log n)

Untuk h(n) = n^2 ,h(n) = $O(n^2)$

Untuk i(n) = log n,h(n) = O(log n)

Notasi Big-O untuk T(n) = f(n)g(n) + h(n)i(n) = max(nlog n, n^2log n) = $O(n^2 \log n)$

5. Tentukan notasi O, Ω, dan θ untuk :

a. $T(n) = 7n^2 + 7n \log n$

b. $T(n) = 2T(n-1) + C$, dengan *C* adalah konstanta dan *T*(*n*) bernilai 1 untuk *n* = 0. (15)

Jawaban:

a. Karena 7n log n < $7n^2$, maka $7n^2 + 7n \log n < 14 n^2$, dengan mengambil C = 14, didapat :

$7n^2 + 7n \log n = O(n^2)$

Karena $7n^2 + 7n \log n > 7n^2$, untuk C = 7, didapat :

$7n^2 + 7n \log n = \Omega (n^2)$

Karena $7n^2 + 7n \log n = O(n^2)$ dan $7n^2 + 7n \log n = \Omega(n^2)$, maka $7n^2 + 7n \log n = \Theta(n^2)$

b. T(n) = 2T(n-1) + C dengan C konstanta.

$= 2*2T(n-2) + 2C=4*2T(n-3)+3C = 2^n*T(0) + (n-1)C = O(2^n) + O(n) = O(2^n)$

6. Berapa kali operasi perbandingan dan berapa kali operasi pertukaran dilakukan untuk algoritma BubbleSort di bawah ini untuk kasus terburuk? Selanjutnya, nyatakan hasilnya dalam notasi Big-Oh: (20)

```
procedure bubbleSort( A : list of sortable items )
    n = length(A)
    repeat
       newn = 0
       for i = 1 to n-1 inclusive do
          if A[i-1] > A[i] then
             swap(A[i-1], A[i])
             newn = i
          end if
       end for
       n = newn
    until n = 0
end procedure
```

Jawaban: Pada kasus terburuk, semua elemen (kecuali elemen pertama) akan ”terlempar ke atas”. Jumlah perbandingan elemen (`A[i-1] > A[i]`) sama seperti pada algoritma Selection Sort, yaitu:

$T(n) = 1 + 2 + (n-2) + (n-1) = n(n-1)/2 = O(n^2)$

Pada kasus terburuk, semua elemen mengalami pertukaran (`swap(A[i-1], A[i])`), yaitu pada kondisi `if A[i-1] > A[i]` selalu true). Jumlah pertukaran itu sama dengan jumlah operasi perbandingan yang terjadi:

$T(n) = 1 + 2 + (n-2) + (n-1) = n(n-1)/2 = O(n^2)$