Optimum Prefix Code

1. Symbols that occurs more frequently will have shorter codewords than symbols that occur less frequently
2. the two symbols that occur least frequently will have the same codeword length

Pf: p: k bits

q:

q'

∵ Prefix ∴q'≠p

∵ p & q are the two longest codewords

∴ no other codeword c

s.t. length(c) > length(p) or length(c) > length(q)

∴ we can drop those k bits of q and q' won't become the prefix of some other codeword

p m*0

q m*1

m: a string of 1s and 0s and * denote concatenation

This requirement don't violate the previous two observations

Huffman coding

Huffman Coding (1)

- Example

symbols	P(ai)	Code-1	Code-2
a_2	0.4	1	00
a_1	0.2	01	10
a_3	0.2	000	11
a_4	0.1	0010	010
a_5	0.1	0011	011

$$H = 2.122 \text{ bits/symbol}$$

$$\overline{l_1} = \overline{l_2} = 2.2 \text{ bits/symbol}$$

Sort the list of symbols in non-increasing probability order

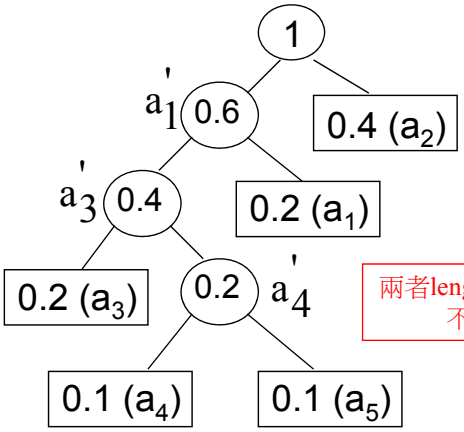
$$\sigma_1^2 = 0.4(1-2.2)^2 + 0.2(2-2.2)^2 + 0.2(3-2.2)^2 + 0.1(4-2.2)^2 + 0.1(4-2.2)^2 = 1.36$$

$$\sigma_2^2 = 0.16$$

$E(x-\bar{x})^2$

Huffman Coding (2)

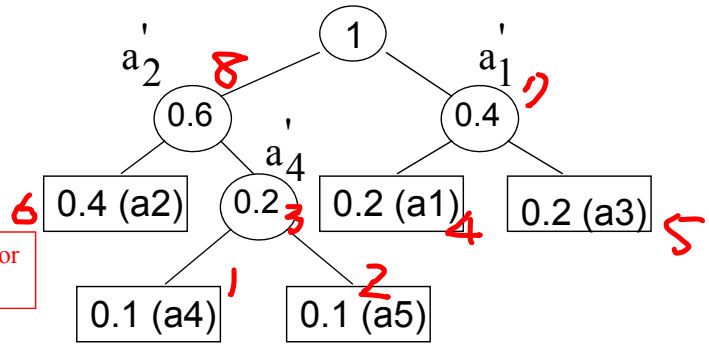
- Code-1



symbols	$P(a_i)$
a_2	0.4
a_1	0.2
a_3	0.2
a_4'	0.2

兩者length vector 不同

- Code-2: the minimum variance Huffman code



symbols	$P(a_i)$
a_2	0.4
a_4'	0.2
a_1	0.2
a_3	0.2

always put the combined symbol as high in the list as possible

Q：average codeword length = sum of weights of internal nodes

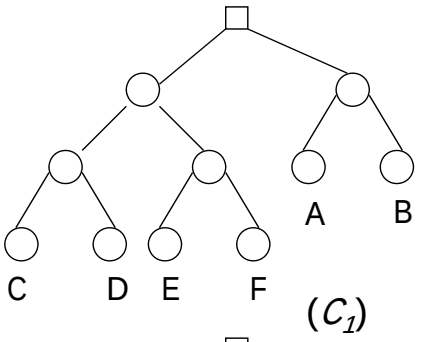
Binary Huffman Equivalent Codes

(weakly Huffman code equivalent)

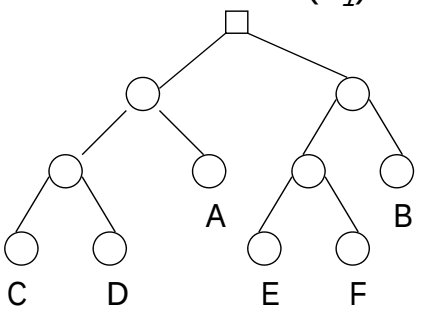
Source Letter	Prob.	C_1	C_2
A	0.3	10	01
B	0.3	11	11
C	0.1	000	000
D	0.1	001	001
E	0.1	010	100
F	0.1	011	101

Average codeword length = 2.4

Length vector = (0, 2, 4)



(C_1)



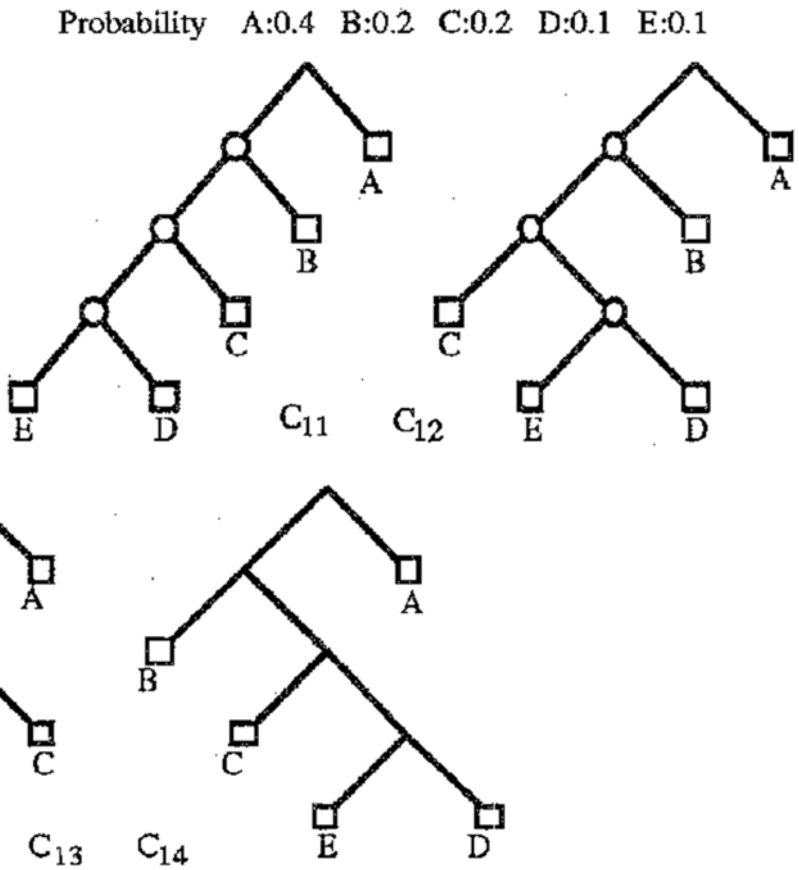
(C_2)

Can't be obtained via. the Huffman algorithm

Binary Huffman Equivalent Codes

(strongly Huffman code equivalent)

All C_{11}~C_{14} can be obtained via. the Huffman algorithm



Optimality of Huffman Codes (1)

- The necessary conditions for an optimal binary prefix code
  - Given any two letters a_j and a_k, if $P[a_j] \geq P[a_k]$, then $l_j \leq l_k$, where l_j is the number of bits in the codeword for a_j.
  - The two least probable letters have codewords with the same maximum length l_m.
  - In the tree corresponding to the optimum code, there must be two branches stemming from each intermediate node.
  - Suppose we change an intermediate node into a leaf node by combining all the leaves descending from it into a composite letter of a reduced alphabet. Then, if the original tree was optimal for the original alphabet, the reduced tree is optimal for the reduced alphabet. (⟶)

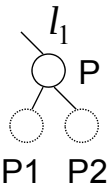
•Prove Huffman code is optimal ⟵

The original tree was optimal for the original alphabet
if and only if
the reduced tree is optimal for the reduced alphabet.

Optimality of Huffman Codes (2)

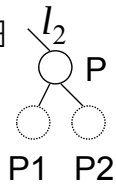
Proof:

(大→小)



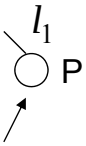
$\bar{l}_1 = \ldots + Pl_1$ (for reduced tree) $\Rightarrow$ $\bar{l} = \bar{l}_1 - Pl_1 + P1(l_1+1) + P2(l_1+1) = \bar{l}_1 + P$ (for original tree)

If $\exists$



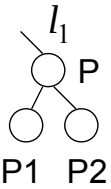
with $\bar{l}_2 = \ldots + Pl_2 (\bar{l}_2 < \bar{l}_1)$ then $\bar{l}' = \bar{l}_2 + P < \bar{l}$ $(\rightarrow\leftarrow)$

(大←小)



the reduced tree is optimal for the reduced alphabet with $\bar{l}_1$

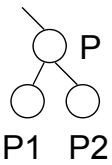
If



is not optimal for the original alphabet

where $\bar{l} = \bar{l}_1 - Pl_1 + P1(l_1+1) + P2(l_1+1) = \bar{l}_1 + P$

$\therefore \exists$



with $\bar{l}' < \bar{l}$

then $\bar{l}' - P < \bar{l}_1$ $(\rightarrow\leftarrow)$

where

$\bar{l}' - P$ is the average codeword length of the reduced alphabet

Length of Huffman Codes (1)

- For a source S with alphabet $A = \{a_1, a_2, \ldots, a_k\}$, and probability model $\{P(a_1), \{P(a_2), \ldots, \{P(a_k)\}$

$\Rightarrow \quad H(S) \le \bar{l} < H(S) + 1$

$f(x)$ is a concave function

(*) Jensen's inequality

$E[f(X)] \le f(E[X])$

(**) Kraft inequality

$\sum 2^{-l_i} \le 1$

Proof (a) :

$$\bar{l} = \sum_{i=1}^{k} P(a_i) l_i$$

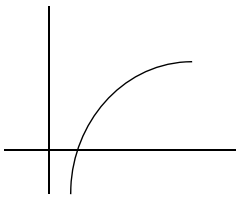
$$H(s) - \bar{l} = -\sum_{i=1}^{k} P(a_i) \log_2 P(a_i) - \sum_{i=1}^{k} P(a_i) l_i$$

$$= \sum_{i=1}^{k} P(a_i) \log_2 \left[\frac{2^{-l_i}}{P(a_i)} \right] \overset{by\,(*)}{\le} \log_2 \left[\sum_{i=1}^{k} 2^{-l_i} \right] \overset{by\,(**)}{\le} 0 \Rightarrow H(s) \le \bar{l}$$

Let $f(x) = \log_2 x$, $x_i = \dfrac{2^{-l_i}}{P(a_i)}$, and $P(X = x_i) = P(a_i)$.

$$E[X] = \sum P(X = x_i) x_i = \sum P(a_i) \frac{2^{-l_i}}{P(a_i)} = \sum 2^{-l_i}$$

$$E[f(X)] = \sum P(a_i) \log_2 \frac{2^{-l_i}}{P(a_i)}$$



Length of Huffman Codes (2)

Proof (b) :

$$Let \quad l_i = \left\lceil \log_2 \frac{1}{P(a_i)} \right\rceil \quad \because \lceil x \rceil = x + \varepsilon \quad 0 \le \varepsilon < 1$$

$$\therefore \log_2 \frac{1}{P(a_i)} \le l_i < \log_2 \frac{1}{P(a_i)} + 1$$

$$\Downarrow$$

$$2^{-l_i} \le P(a_i)$$

$$\therefore \sum_{i=1}^{k} 2^{-l_i} \le 1$$

$$\bar{l} = \sum_{i=1}^{k} P(a_i) l_i < \underbrace{\sum_{i=1}^{k} \left[\log_2 \frac{1}{P(a_i)} + 1 \right] \times P(a_i)}_{\Updownarrow \; H(s)+1}$$

$$Q: If \quad \begin{array}{ll} P_{\max} \ge 0.5 & \bar{l} < H(s) + P_{\max} \\ P_{\max} < 0.5 & \bar{l} < H(s) + P_{\max} + 0.086 \end{array}$$

Extended Huffman Codes (1)

Let $A = \{a_1, a_2, \ldots, a_m\}$

$A^{(n)} = \{\underbrace{a_1 a_{1\ldots} a_1}_{\text{n times}}, a_1 a_{1\ldots} a_2, \ldots, a_1 a_{1\ldots} a_m, \ldots, a_m a_{m\ldots} a_m\}$

of the blocks in $A^{(n)} = m^n$

For the source $S^{(n)}$ with extended alphabet $A^{(n)}$, we have

$$H(S^{(n)}) \le R^{(n)} < H(S^{(n)}) + 1,$$

where $R^{(n)}$ is the number of bits required to code one block (n letters) and

$$H(S^{(n)}) = -\sum_{i_1=1}^{m} \sum_{i_2=1}^{m} \ldots \sum_{i_n=1}^{m} P(a_{i_1}, a_{i_2}, \ldots, a_{i_n}) \log[P(a_{i_1}, a_{i_2}, \ldots, a_{i_n})]$$

Suppose iid assumption

$$= -\sum_{i_1=1}^{m} P(a_{i_1}) \log P(a_{i_1}) [\sum_{i_2} P(a_{i_2}) \sum_{i_3} P(a_{i_3}) \ldots \sum_{i_n} P(a_{i_n})]$$

$$- \sum_{i_2=1}^{m} P(a_{i_2}) \log P(a_{i_2}) [\sum_{i_1} P(a_{i_1}) \sum_{i_3} P(a_{i_3}) \ldots \sum_{i_n} P(a_{i_n})] - \ldots$$

$$- \sum_{i_n=1}^{m} P(a_{i_n}) \log P(a_{i_n}) [\sum_{i_1} P(a_{i_1}) \sum_{i_2} P(a_{i_2}) \ldots \sum_{i_{n-1}} P(a_{i_{n-1}})]$$

$$= nH(S) \qquad \because R = \frac{1}{n} R^{(n)} \quad \Rightarrow \quad H(S) \le R < H(S) + 1/n$$

Extended Huffman Codes (2)

- Example : Consider a source that put out iid letters from the alphabet A={ a_1, a_2, a_3} with probability mode P(a_1)=0.8, P(a_2)=0.02, P(a_3)=0.18.

H(S)=0.816 bits/symbol

Symbol	P(a_i)	codeword
a_1	0.80	0
a_2	0.02	11
a_3	0.18	10

$$\bar{l} = 1.2 \text{ bits/letter}$$

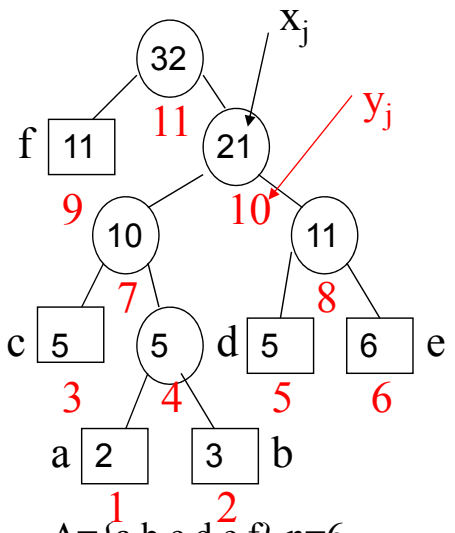
Symbol	P($a_i a_j$)	codeword
a_1a_1	0.6400	0
a_1a_2	0.0160	10101
a_1a_3	0.1440	11
a_2a_1	0.0160	101000
a_2a_2	0.0004	10100101
a_2a_3	0.0036	1010011
a_3a_1	0.1440	100
a_3a_2	0.0036	10100100
a_3a_3	0.0324	1011

$\bar{l}$ = 1.7228 bits/symbol

(0.8614 bits/letter)

Adaptive Huffman Coding(1)

- To develop the Huffman code based on the statistics of the symbols already encountered (just one-pass coding)



A={a,b,c,d,e,f} n=6

weight : number of occurrence

node number : 1 ~ 2n-1

* the weight of each internal node = the sum of the weight of it's children

* the nodes can be numbered in non - decreasing order by weight as

node number : $y_1, y_2, \ldots\ldots\ldots\ldots y_{2n-1}$, s.t.

1. if x_j is the weight of node y_j, we have

$x_1 \le x_2 \le \ldots\ldots\ldots\ldots \le x_{2n-1}$

Furthermore,

2. nodes y_{2j-1} & y_{2j} are siblings $(1 \le j \le n-1)$, and

3. their common parent node is higher in the numbering

Q: A prefix code is a Huffman code ~~if and only~~ if it satisfies the sibling property (1 & 2). (←, ~~→~~) (see pp. 3 code-2) 2&3

Adaptive Huffman Coding(2)

- Updating the tree
  - both encoder & decoder start with the same tree
  - process next symbol
  - both encoder & decoder modifying the tree using sibling property
- NYT(not yet transmitted) code
  - weight = 0
  - each uncompressed symbol is preceded by a variable-size NYT code
  - the smallest code number is assigned to NYT
- Uncompressed Code
  - for a source alphabet ($a_1, a_2, \ldots, a_m$) of size m

$$m = 2^e + r \quad 0 \le r < 2^e$$

$$a_k \text{ is encoded as} \begin{cases} k-1 & in \quad (e+1) \quad bits \quad 1 \le k \le 2r \\ k-r-1 & in \quad e \quad bits \quad 2r < k \le m \end{cases}$$

Adaptive Huffman Coding(3)

Example : $m = 26 = 2^4 + 10 \quad (e = 4, r = 10)$

Q:

20

k	symbols	codewords	
1	a	00000	$(=0)$
2	b	00001	$(=1)$
3	c	00010	$(=2)$
4	d	00011	$(=3)$
.	.	.	. $=k-1.$
19	s	10010	$(=18)$
20	t	10011	$(=19)$

= y

XX…X0

or XX…X1

2^e-y+2y=m

$\Rightarrow$ y= m- 2^e

$\Rightarrow$i.e. take y=r

6

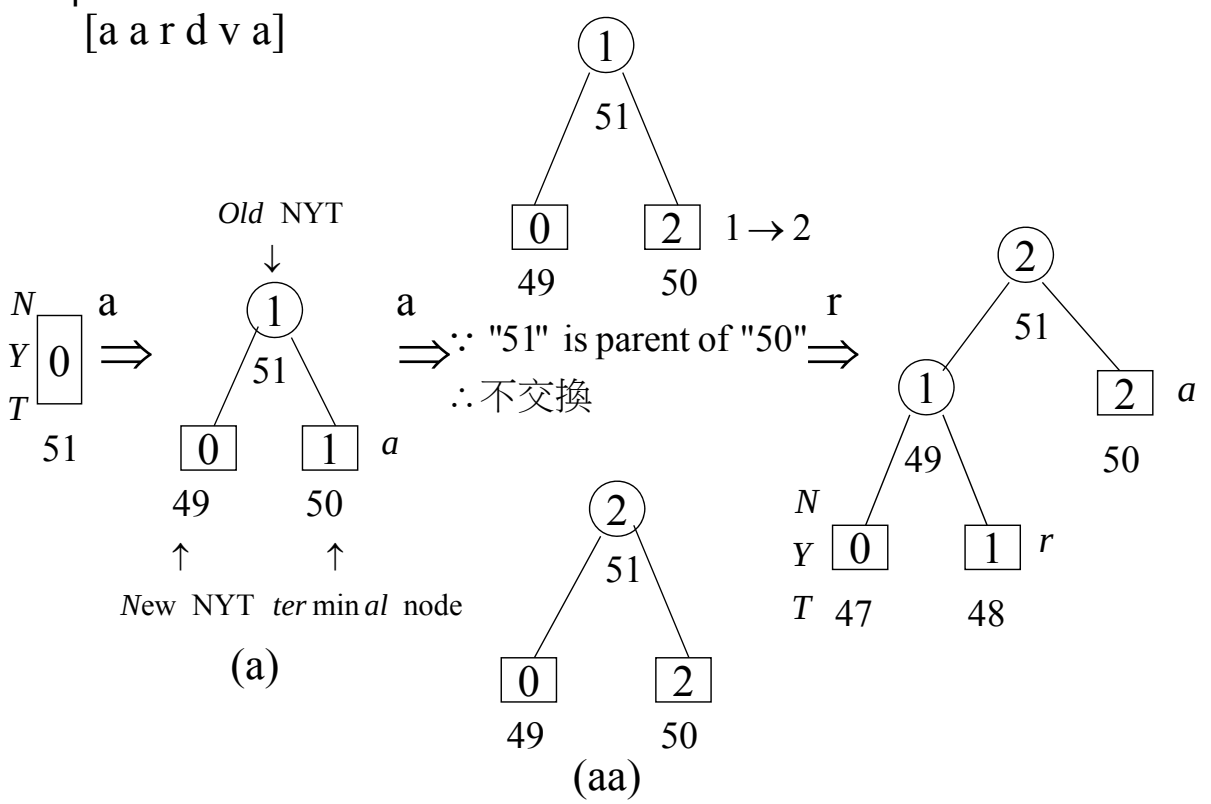
21	u	1010	$(=10)$
22	v	1011	$(=11)$
.	.	.	. $=k-r-1$
25	y	1110	$(=14)$
26	z	1111	$(=15)$

= 2^e-y

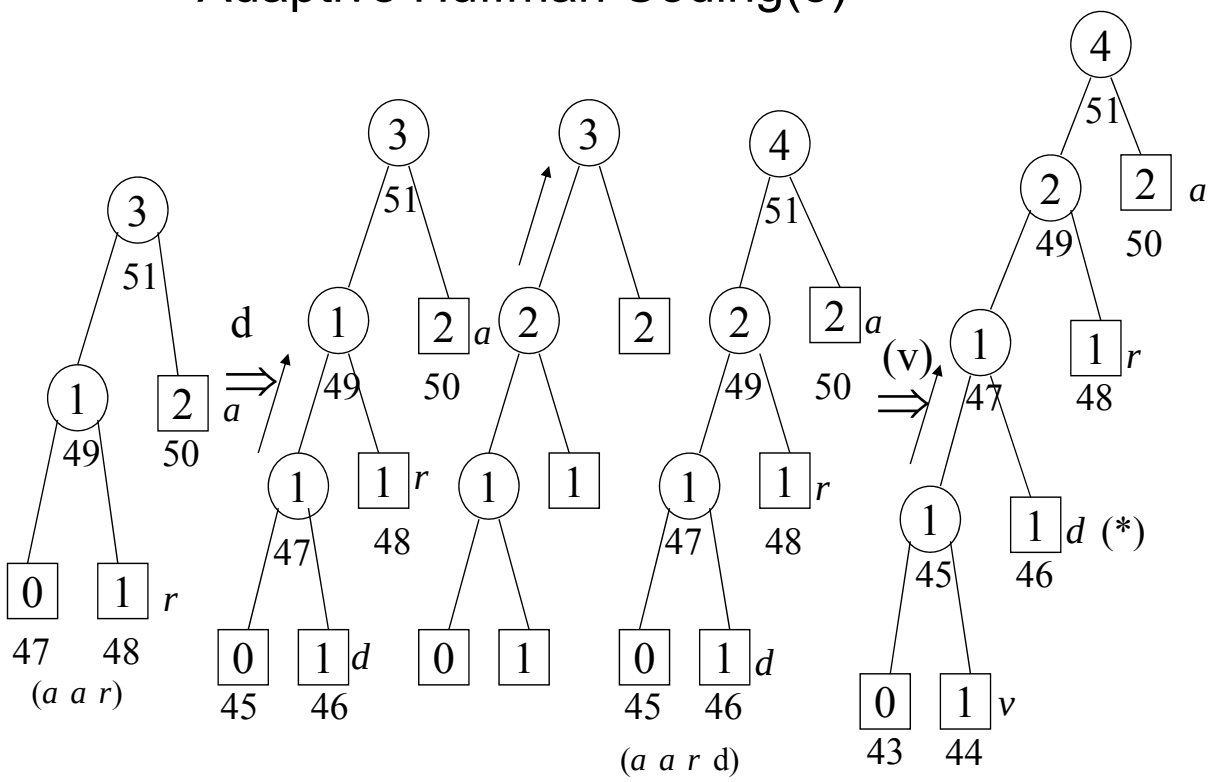
Adaptive Huffman Coding(4)

- Example

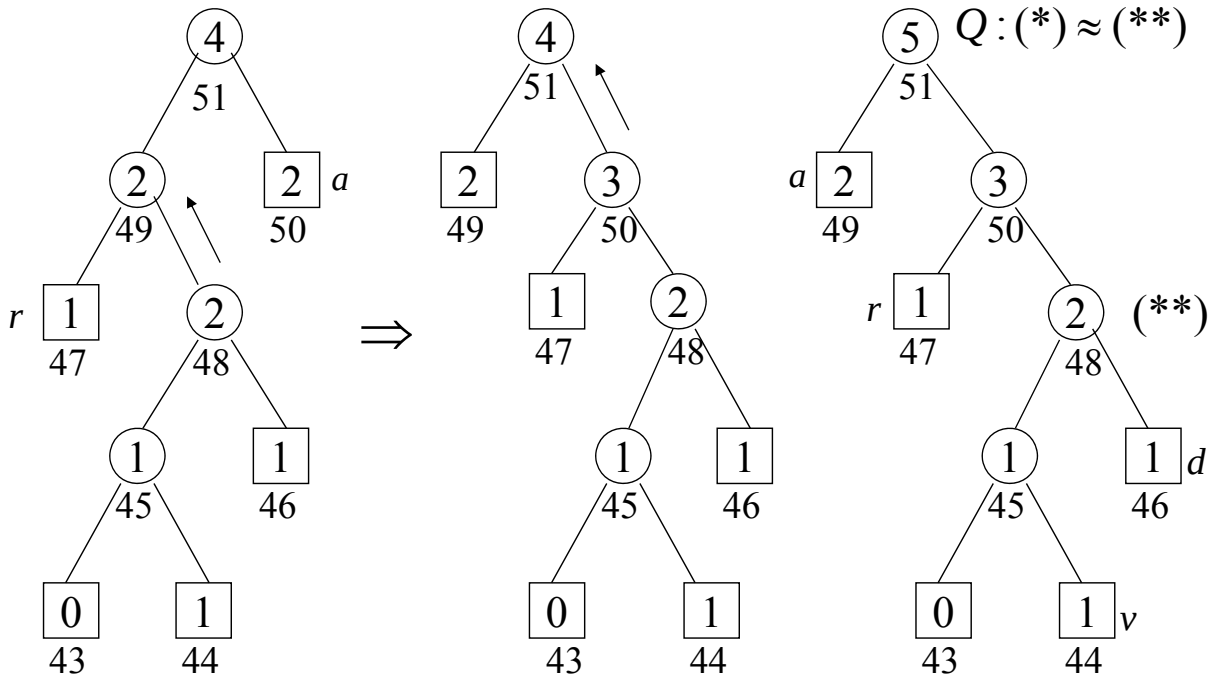
[a a r d v a]



Adaptive Huffman Coding(5)



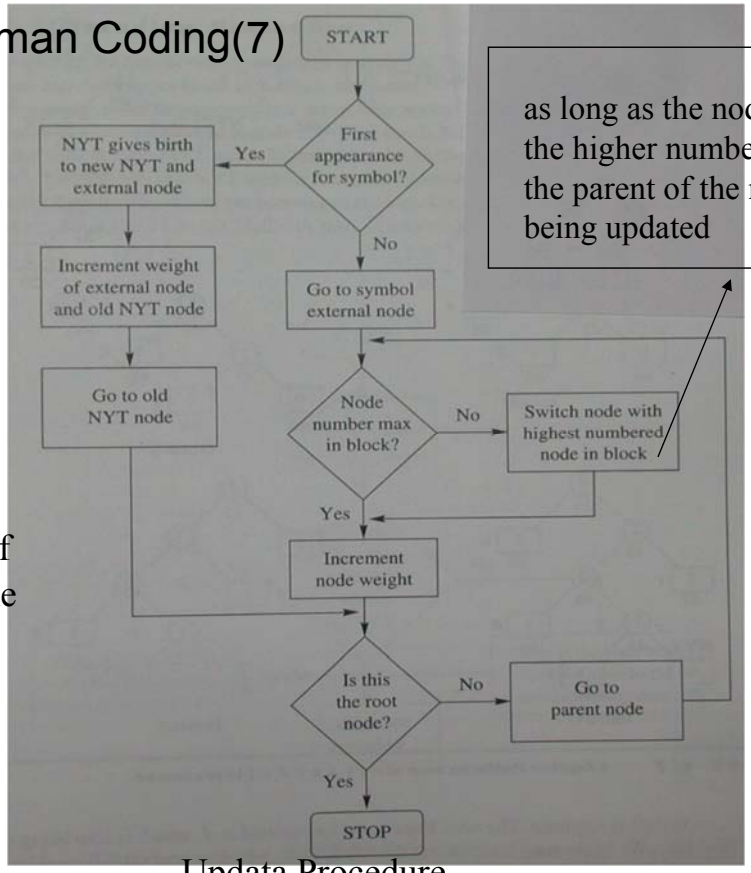
Adaptive Huffman Coding(6)



00000 1 010001 0000011 0001011 0 (a a r d v)

a a r d v a

Adaptive Huffman Coding(7)



as long as the node with the higher number is not the parent of the node being updated

a block: the set of nodes with the same weight

Updata Procedure