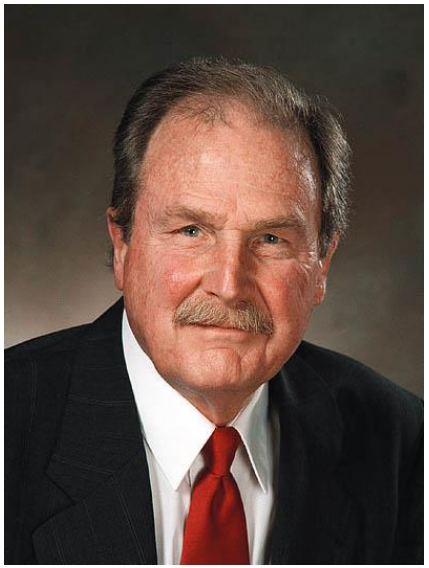
Huffman codes

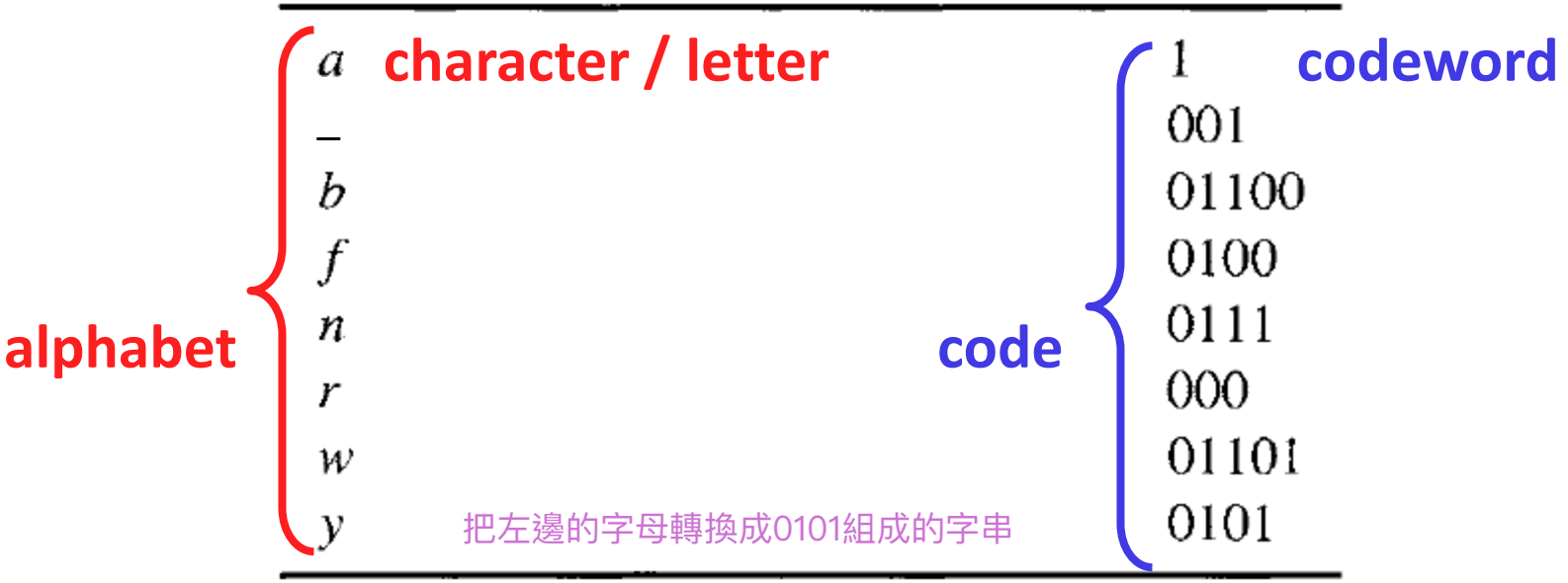


David Huffman
(1925~1999)

現在使用的jpg檔案是失真的，有壓縮，壓縮的過程有使用這個huffman code

Coding

- The assignment of ***binary sequences*** to ***characters*** of an alphabet



4/20 (四) START

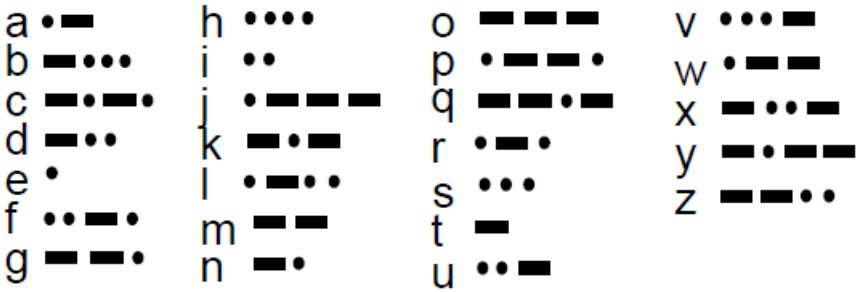
Coding

- Fixed-length code
  - Example: ASCII Code, Big5, UTF

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

- Variable-length code
  - Example: Morse Code (1838)

早期是航海用的，用閃光或鳴笛來送出訊息。長度是2~4不等



Coding

站在儲存或傳送資料的角度，會覺得檔案越小越好

- Which one is more effective in terms of ***compression***?

	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed length	000	001	010	011	100	101
Variable length	0	101	100	111	1101	1100

3：每個長度是3碼

Fixed-length code: 3 * (45 + 13 + 12 + 16 + 9 + 5) = 300 k bits

用fixed length來存會需要300k bits

Variable-length code: 1 * 45 + 3 * (13 + 12 + 16) + 4 * (9 + 5) = 224 k bits

*Assign **short** codeword to **frequent** characters!*

所以站在壓縮的角度variable length是比較好的

Coding

Characters	Code 1	Code 2	Code 3
A	0	0	0
C	0	1	10
T	1	00	110
G	10	11	111

Ambiguous

Not uniquely decodable

Prefix code

在壓縮上是很有效的編碼法

100 → C T

→ C A A

code 2的編碼法會解出兩種可能，所以不行

Prefix code

- No codeword is a prefix of some other codeword.
- Example:

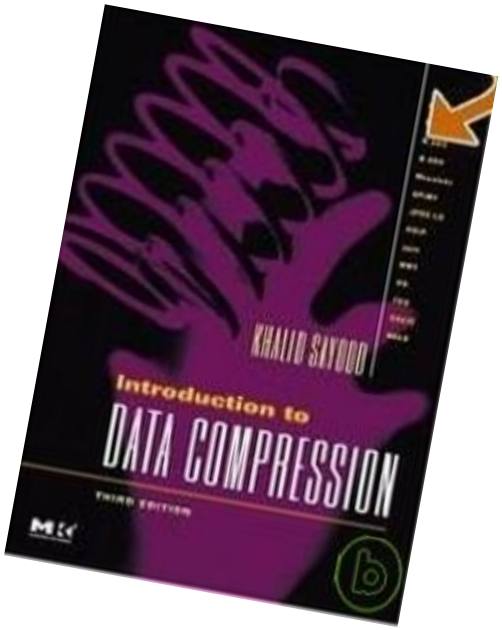
prefix code：找不到任何code word是別的core word的prefix的編碼法

  - *Which one is prefix code?* **Code3 !**

Letters	Code 2	Code 3	Code 4
a_1	0	0	0
a_2	1	10	01
a_3	00	110	011
a_4	11	111	0111

Prefix code

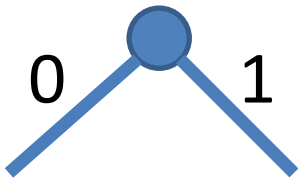
- Prefix code can always achieve the ***optimal*** data compress among any code!



Prefix (Optimal) code

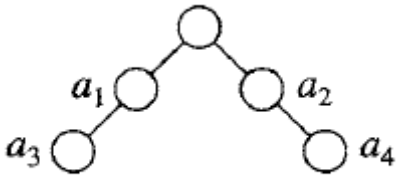
Q. 如何判斷一組code是不是prefix code呢？

- Can always be represented by a ***binary tree***
- ***Characters can only appear at leaf nodes***

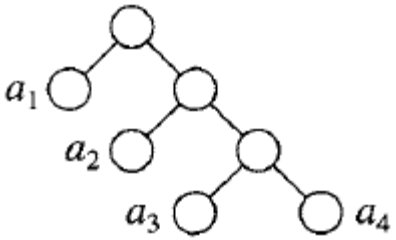


Letters	Code 2	Code 3	Code 4
a_1	0	0	0
a_2	1	10	01
a_3	00	110	011
a_4	11	111	0111

A. 讓這一組code長出一顆樹，最後再去檢查是不是每個letters都在樹葉端上

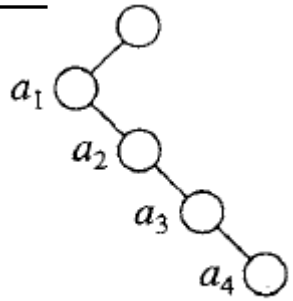


Code 2



Code 3

coed 3 所有的letters都只會出現在樹葉端



Code 4

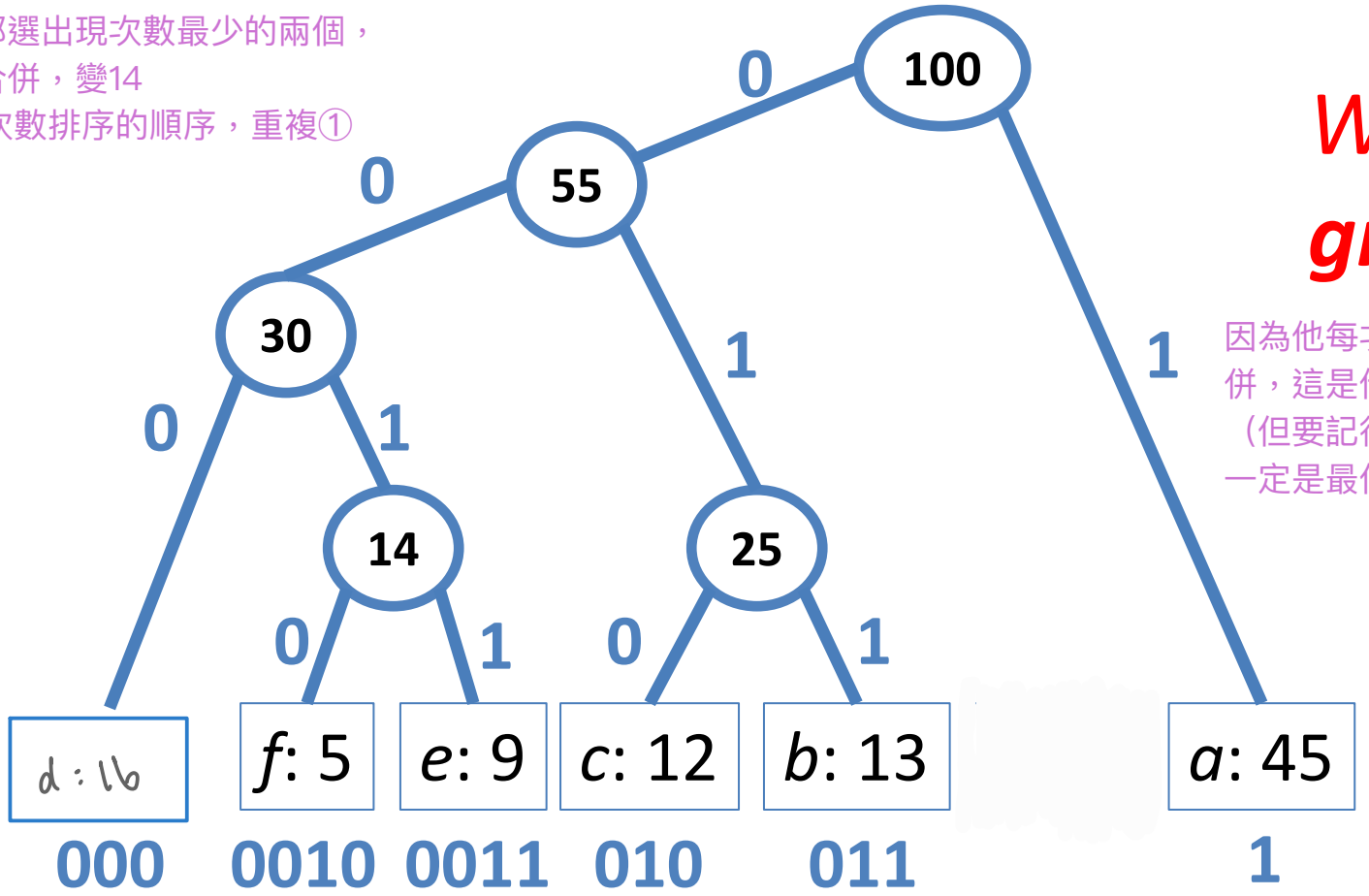
code 4都是別人的prefix

Huffman code

每個字母出現的次數（千次）

- $A = \{f: 5, e:9, c:12, b:13, d: 16, a:45\}$

1.每次都選出現次數最少的兩個，
然後合併，變14
2.按照次數排序的順序，重複①



*Why is it **greedy**?*

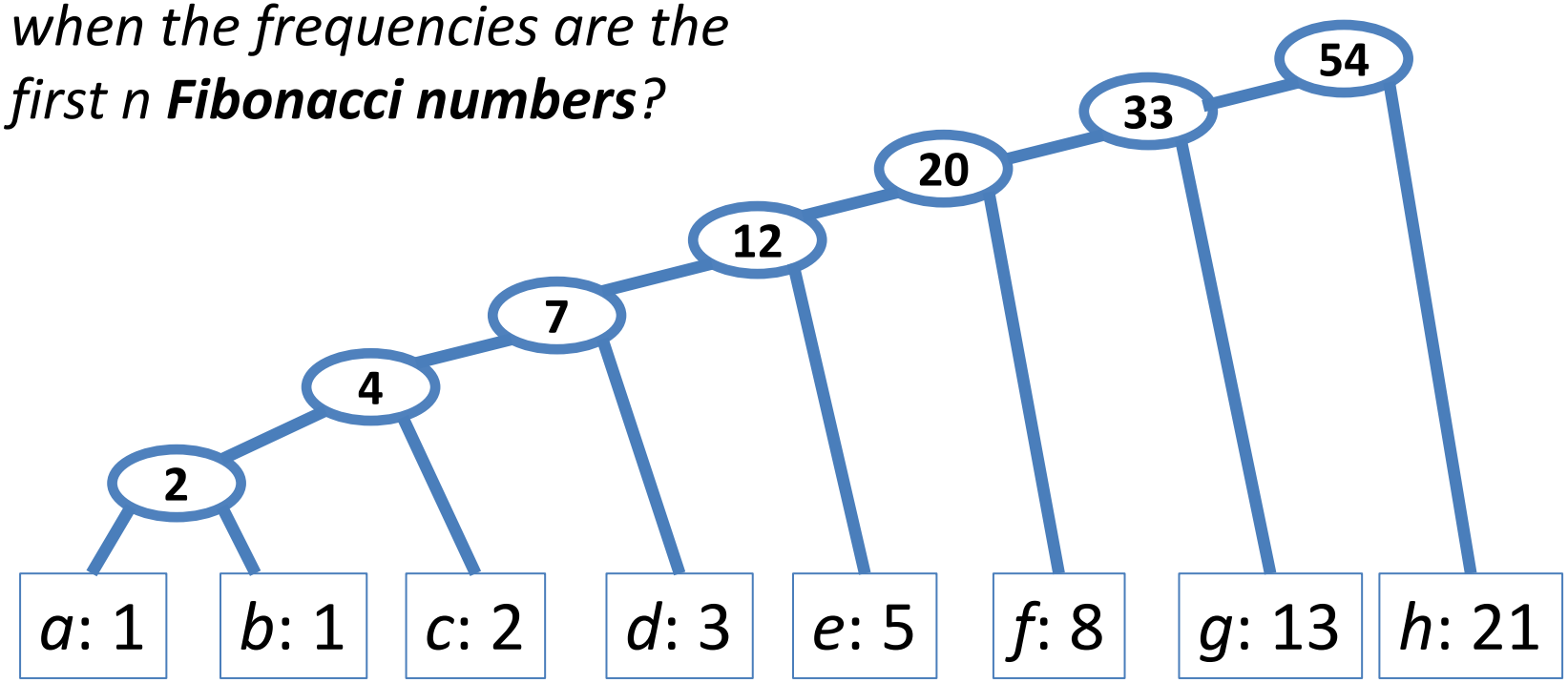
因為他每次都把兩個最小的合併，這是他的greedy choice（但要記得貪婪法並沒有保證一定是最佳解）

有先排序過

Practice

- $A = \{a: 1, b: 1, c: 2, d: 3, e: 5, f: 8, g: 13, h: 21\}$

Can you find the optimal code when the frequencies are the first n ***Fibonacci numbers****?*



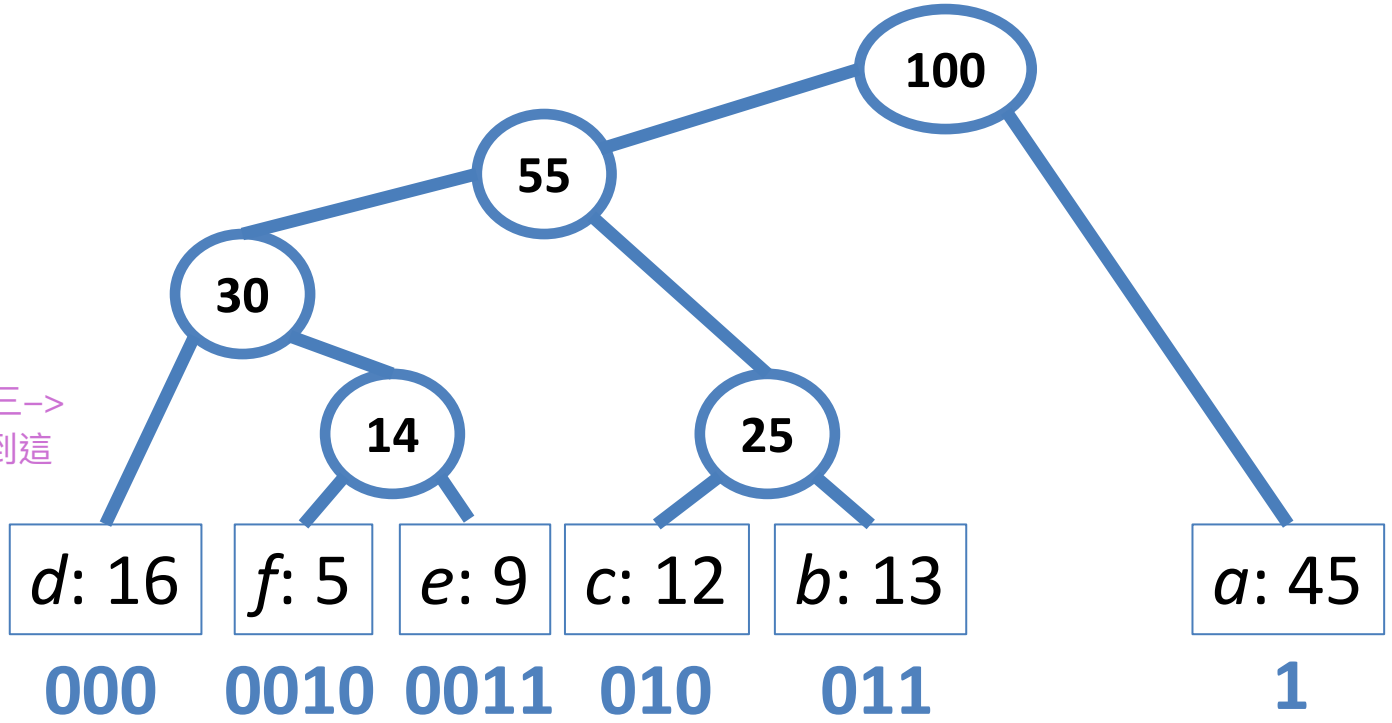
3 * 0.16 + 4 * 0.05 + 4 * 0.09 + 3 * 0.12 + 3 * 0.13 + 1 * 0.45 = 2.24

Cost (average bit rate)

平均來講每個字會需要2.24的bit，可以把文件傳出去

Q. 為什麼用最小的兩個字合併會是最佳解？

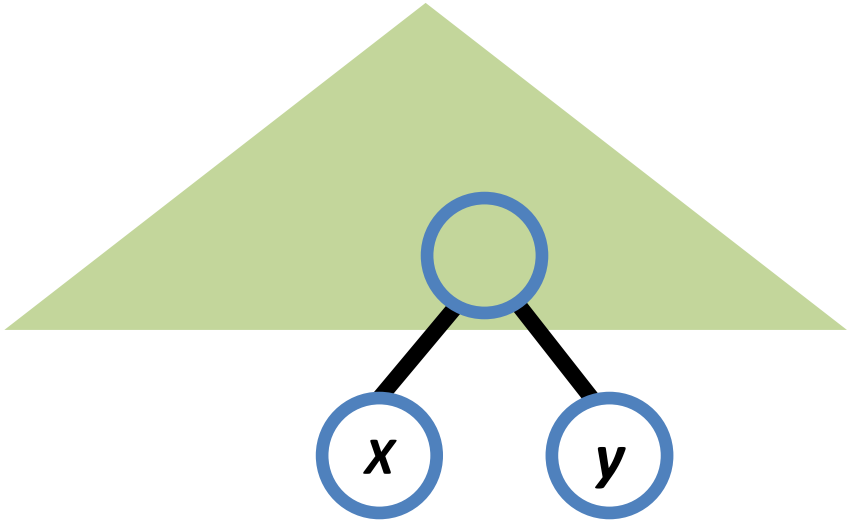
- The cost of a tree *T*: the sum of weighted depth of leaves
- An optimal tree has the *minimal* cost.



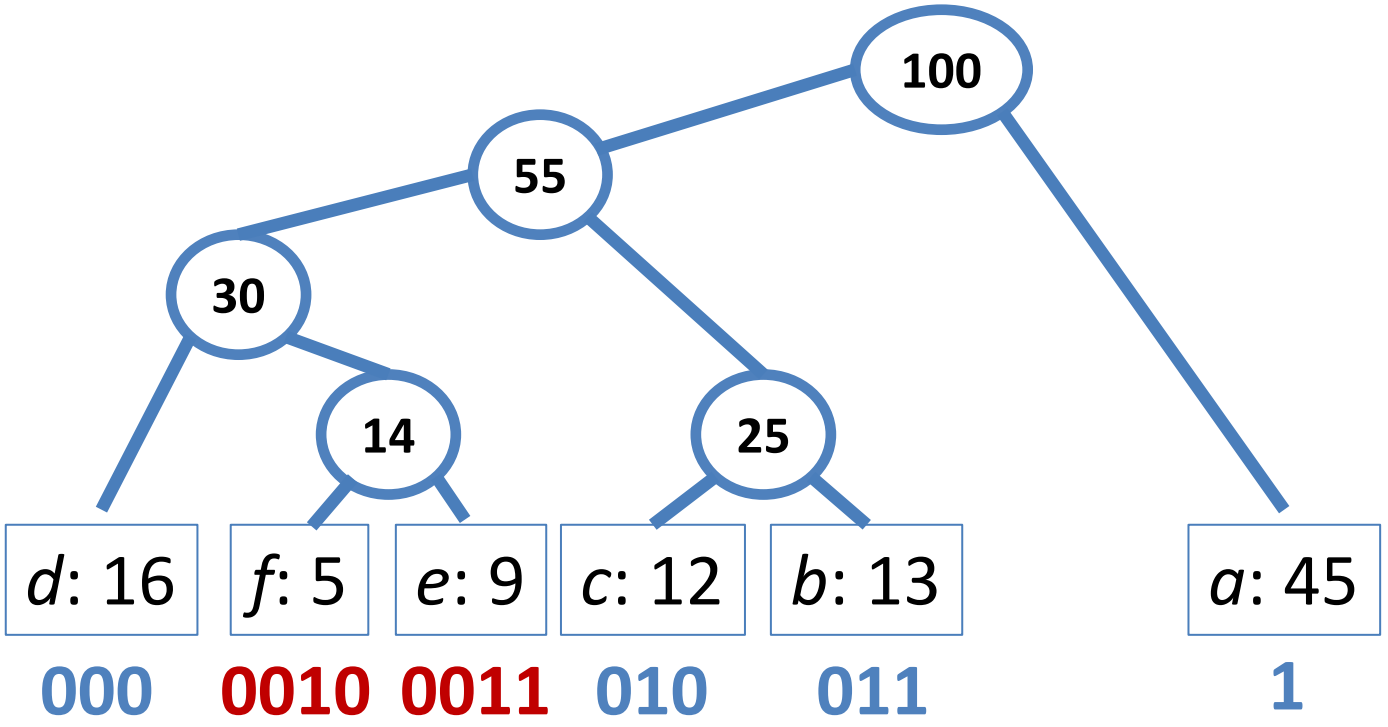
000 長度為三–>從根節點走到這裡是第三層

Correctness

- Claim: Let *x* and *y* be the lowest frequencies. There exists an optimal prefix code in which the code-words for *x* and *y* have the same length and differ only in the last bit.



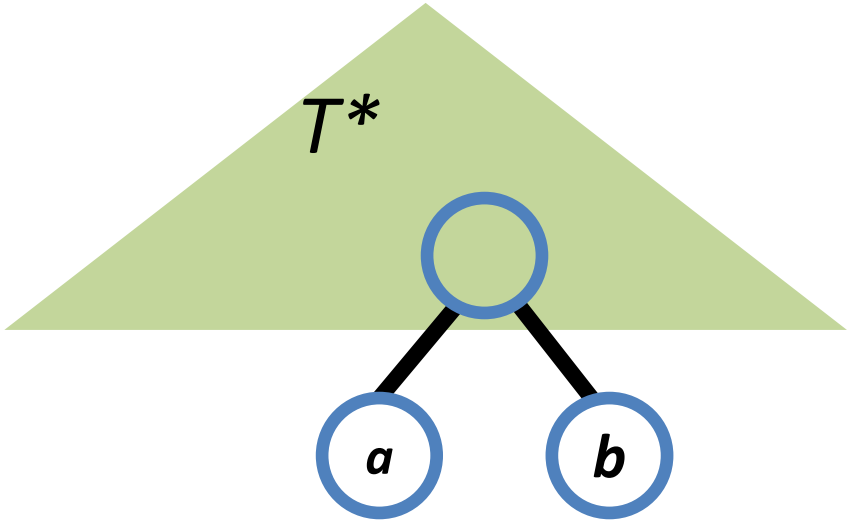
- Claim: Let *x* and *y* be the lowest frequencies. There exists an optimal prefix code in which the code-words for *x* and *y* have the same length and differ only in the last bit.



Correctness

- Proof: Let T^* be an optimal tree. Let a and b be two sibling leaves that are deepest in T^*.

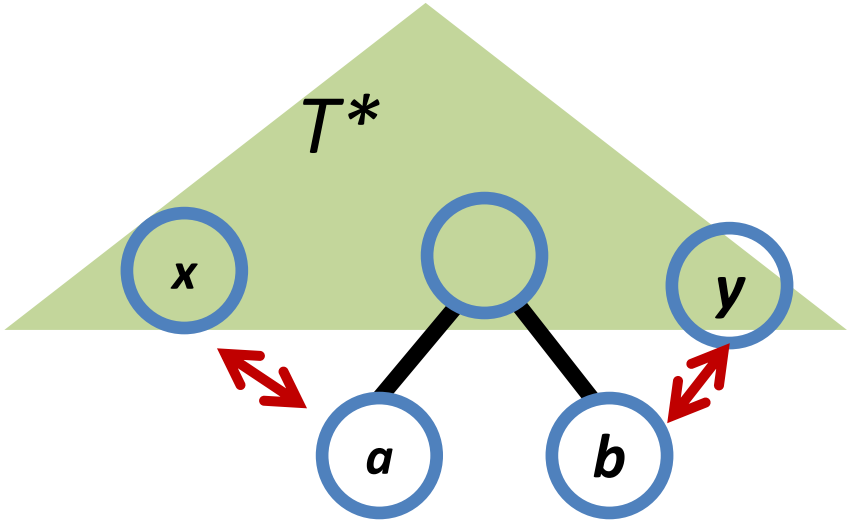
假設T*是optimal tree，a, b是最深的兩個節點



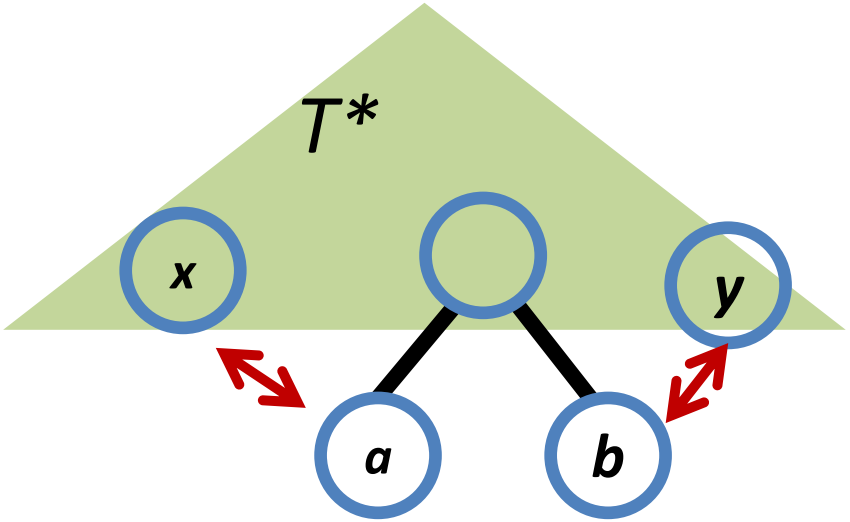
x, *y*: the lowest frequencies
a, *b*: the deepest nodes in *T**

Correctness

- Proof (continued):
  - Create *T''* by swapping *a* and *x*, *b* and *y*
  - cost (*T**) ≥ cost (*T''*) *why?* ab跟xy換位子後cost會往下掉 Q. why?
  - Since *T** is optimal, which implies cost (*T**) = cost (*T''*)



x, *y*: the lowest frequencies
a, *b*: the deepest nodes in *T**



cost (*T**) – cost(*T''*)

$$= \sum_{c \in C} c.freq \cdot d_{T^*}(c) - \sum_{c \in C} c.freq \cdot d_{T''}(c)$$

每一個樹的節點的頻率*深度

$$= (a.freq - x.freq)(d_{T^*}(a) - d_{T^*}(x)) + (b.freq - y.freq)(d_{T^*}(b) - d_{T^*}(y))$$

p. 434

這四個()都會大於等於0。
因為我們已經是知道xy是最低頻率了，也知道ab的深度是最深的。（左上角有寫）

$$\geq 0$$

Conclusion

- An optimal solution for the coding problem can be found by a greedy algorithm.