HW3 Huffman Encoding Tree

編碼(encode)

- 固定長度編碼
- ASCII：將1,2,…,9,a,b,…,A,B,Z 用1~128之間的數字表示
- 不管甚麼字，編碼長度都一樣

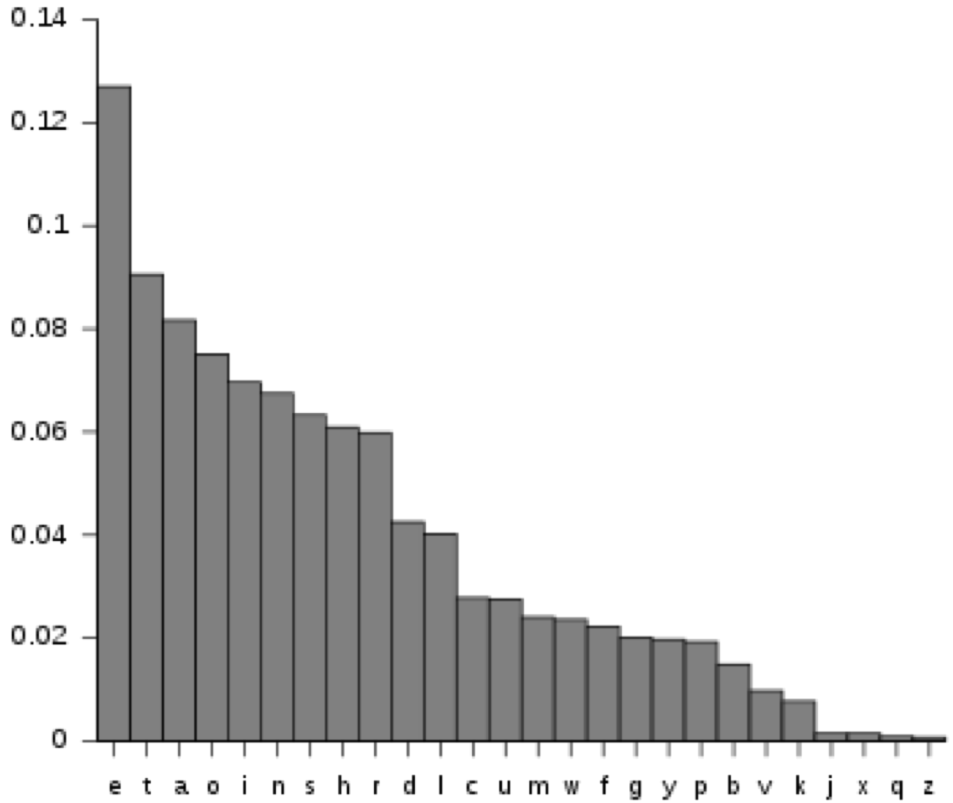
-> 資料量相同

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

www.alpharithms.com

編碼(encode)

- 變動長度編碼
- 實際上，每個英文字母的出現頻率並不相同
  - e出現的頻率遠大於xqz
- 如果我們能對不同的字母採用不同的編碼長度，可以大幅縮減資料量



編碼(encode)

- Ex: 考慮一個只有四個字母的語言abcd
- 固定長度編碼：a=00, b=01,c=10,d=11
- 動態長度編碼：a=0, b=111,c=110,d=10
- 考慮句子：aaaaaaaacaaaabddaacd
  - 固定長度：00000000000000010000000001111100001011 共40碼
  - 變動長度：00000000110000011110100011010 共29碼
  - 資料量少了1/4

- 是否存在一個動態編碼方式可以保證資料長度為最短？

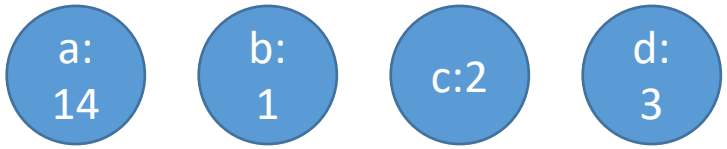
Huffman Encoding Tree

- 利用二元樹的特性，保證
  1. 前綴碼不會有共用部分字串的編碼；意即，解碼的時候選項是單一的
  2. 保證編碼後的壓縮比為最佳

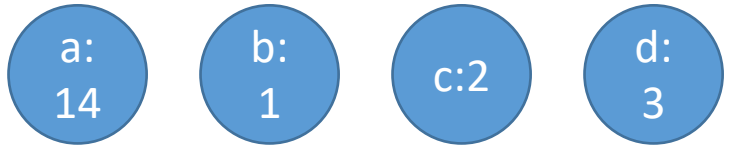
- 為了達到以上目的需要先統計好所有字母的出現次數
- aaaaaaaacaaaabddaacd
  - a: 14
  - b: 1
  - c: 2
  - d: 3

Huffman Encoding Tree

- Key Idea: 越少出現的字母，編碼時應該要放在樹的越下面(level越高)
- aaaaaaaacaaaabddaacd

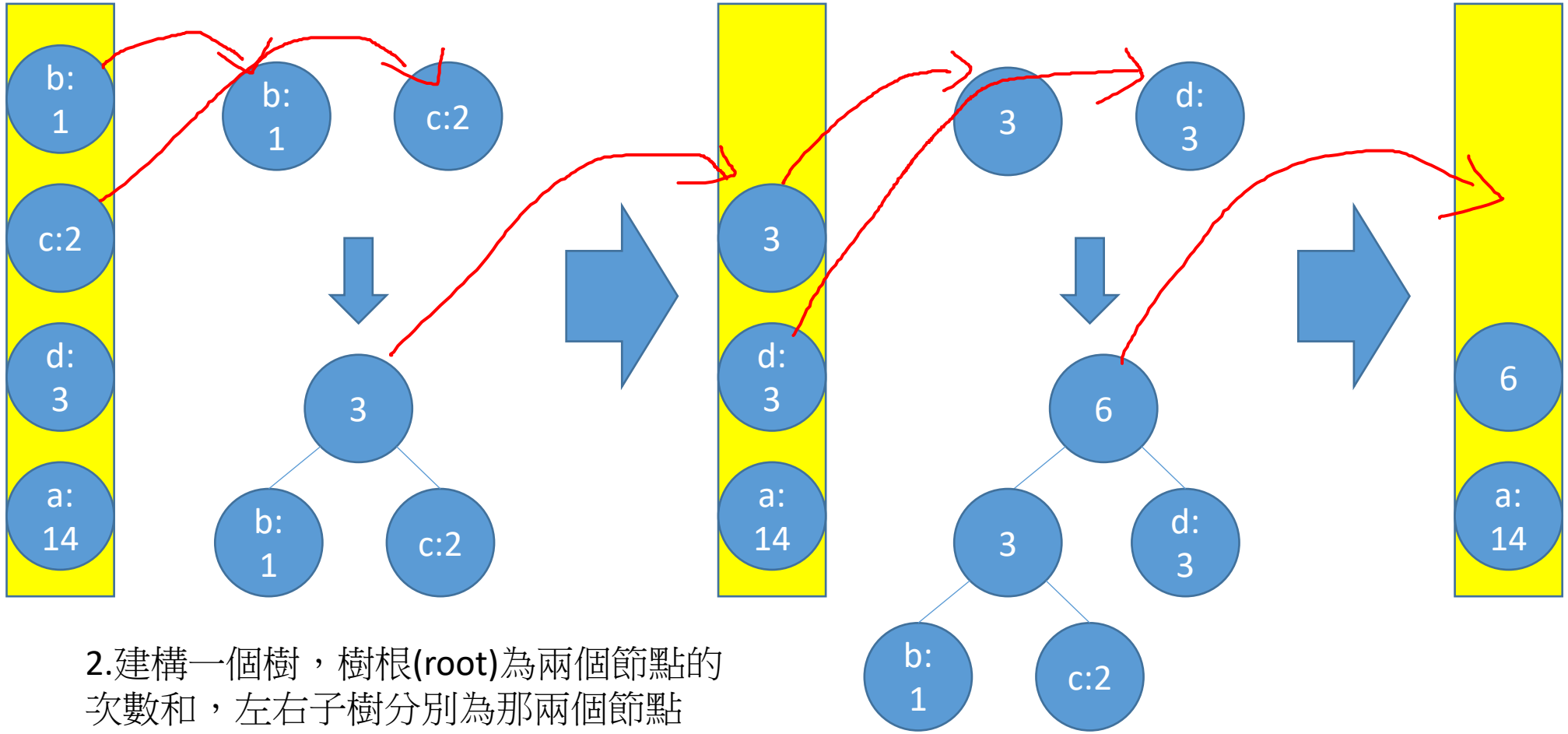


- Iteratively進行直到沒有可以合併的對象：
  - 將所有節點丟入Priority Queue, 以頻率aka出現次數排序，小的在前面
  - 每次挑選最小的兩個節點：
    - 建構一個樹，樹根(root)為兩個節點的次數和，左右子樹分別為那兩個節點
    - 將新建好的樹的樹根節點放回Priority Queue



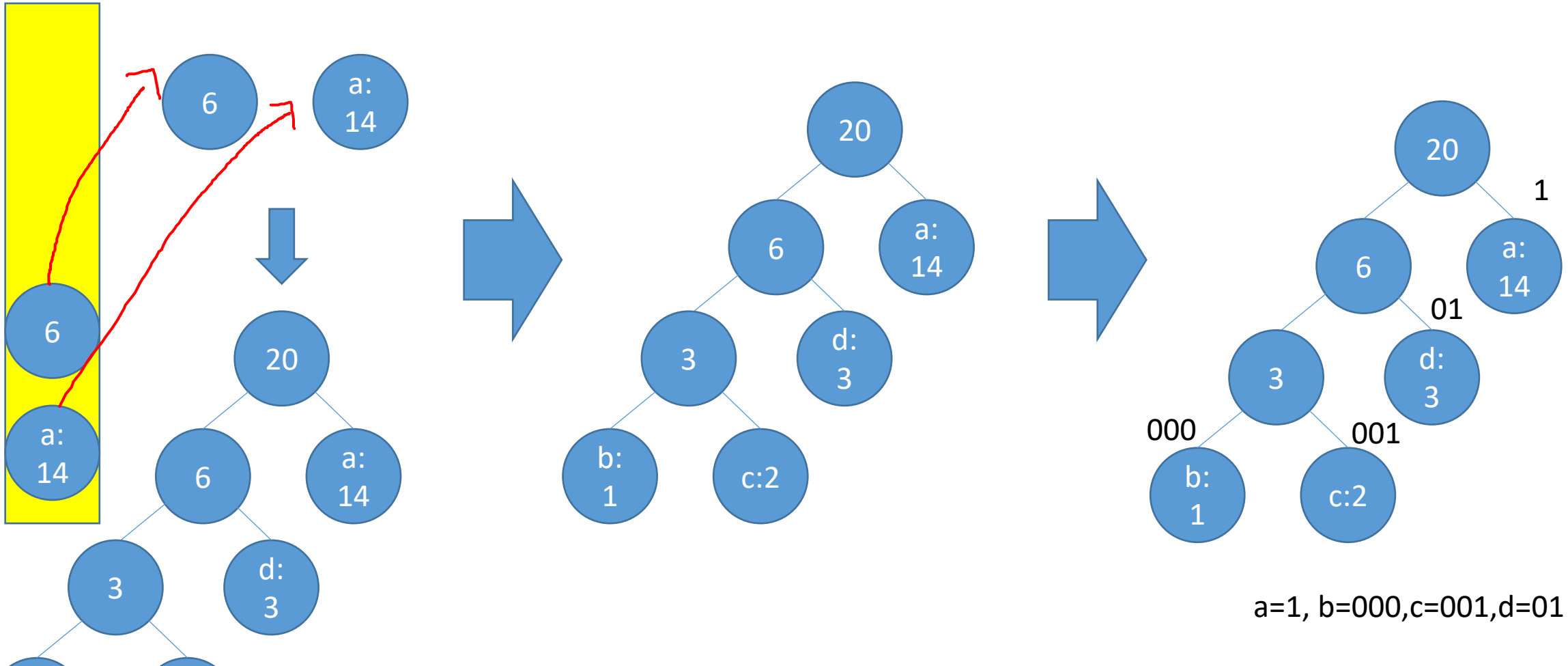
1.每次挑選最小的兩個節點

3.新建好的樹的樹根節點放回Priority Queue



2.建構一個樹，樹根(root)為兩個節點的次數和，左右子樹分別為那兩個節點

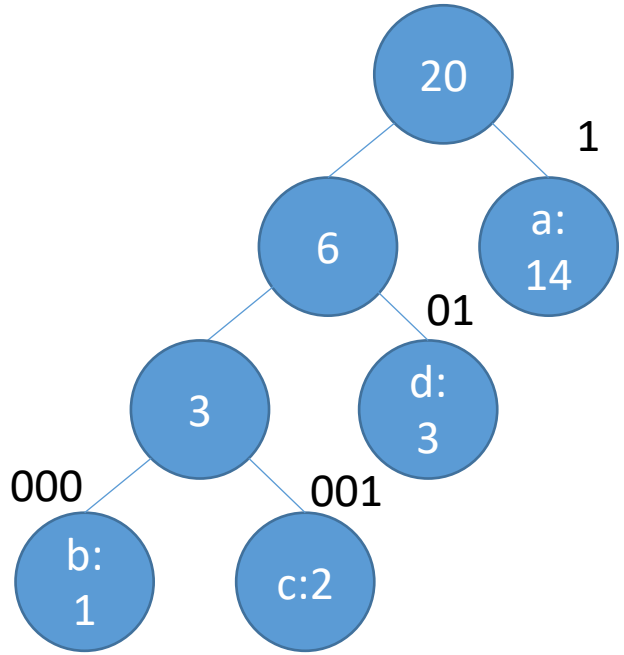
最後給予每一個leaf node
編碼：left=0,right=1



a=1, b=000,c=001,d=01

觀察最後的結果

- 資料都儲存在葉節點上
- 每一個節點degree等於2(Internal)或0(External)
- 越底層(level越高)的葉，其出現機率越低
- 左右子樹交換不影響編碼長度，也就是encode並非唯一解



a=1, b=000,c=001,d=01

- 驗證：
- encoded: 1110100011011001
- decoded: aaadbaadaac

特色

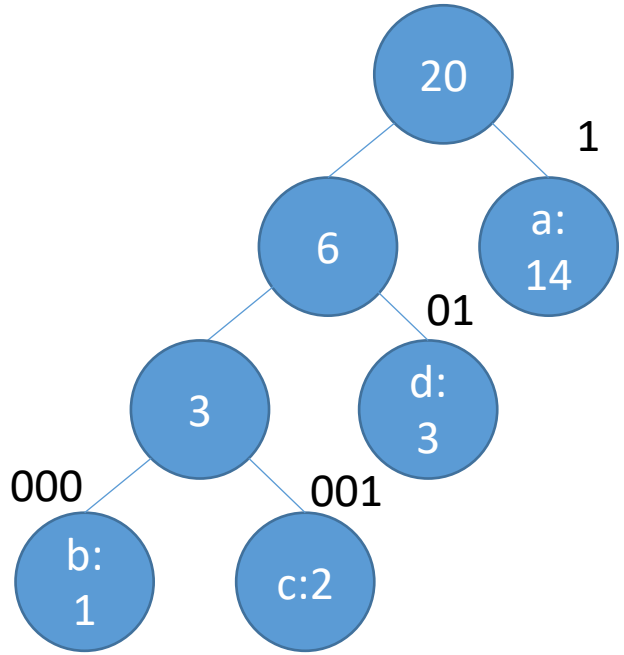
- 每個leaf的編碼長度，等於他的樹高度-1
- 每個internal的值，等於他會被經過的次數
  - aaaaaaaacaaaabddaacd
  - 6: 表示b,c,d一共有6個
  - 20: 表示a,b,c,d一共有20個
- 總長度= a: 14*1

  +d: 3*2

  +b: 1*3

  +c: 2*3 = 29

  = 20+6+3 也就是internal node的值的總和



作業要求

- 實作Huffman Encoding Tree，包含壓縮和解壓縮

- Input:
  - 一個純文字檔 input.txt，內容由ASCII code的文字組成(不會有中文)
- Output:
  - 一個純文字檔 code.txt，紀錄每一個ASCII code的字和其編碼方式
  - 一個二進位檔 compressed，為壓縮後的檔案 (注意二進位)
  - 一個純文字檔 output.txt，為從compressed解壓縮後的檔案

函數實作

- void compress()
  - 將輸入的純文字檔壓縮，輸出code.txt和compress
- void decompress()
  - 讀入code.txt和compress，根據code.txt的內容將compress解碼並存成output.txt
- code.txt的格式：
  - 有n行，表示文件中有n種符號(a,b,c,A,...@,#...等)
  - 每一行是兩個數字，分別為該符號的ASCII code(數字)和 編碼(0跟1組成)
  - Ex: 如果c的編碼是001，則輸出99 001

內容包含：
統計每個字元(char)出現的次數
根據次數建立Huffman Tree
根據Tree將每個符號編碼
根據編碼將輸入檔案壓縮

內容包含：
讀取code.txt
一次讀一個bit，根據code.txt
將compress內容解壓縮並輸出

code.txt
儲存 a=1, b=000,c=001,d=01

```
97 1
98 000
99 001
100 01
```

注意

- compress必須是一個二進位檔才有壓縮的意義
  - 不然原本aabc 只有四個字元大小，壓縮完變成11000001八個字元大小不是來鬧的嗎…
- 所以寫入和讀取都要特別注意
  - fp = fopen(“compress", "wb");
  - 小心將編碼後結果轉成字元(bytes)後寫入
- Huffman Tree不是唯一解，但編碼後的總長度不變

a=1, b=000,c=001,d=01

97 1
98 000
99 001
100 01

繳交方式&評分標準

- 請將source code(一定要含註解否則不計分)/執行結果/心得寫成報告上傳至iLearn2
- Deadline：12/25(日) 23:59:59
- Demo：12/26~12/28的晚上