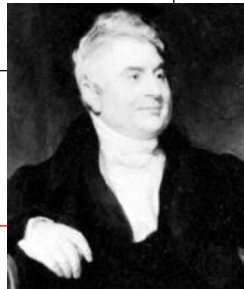
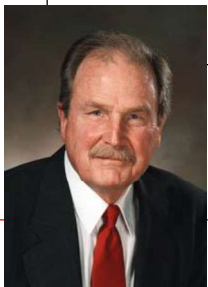


Topic 20: Huffman Coding

The author should gaze at Noah, and ... learn, as they did in the Ark, to crowd a great deal of matter into a very small compass.

Sydney Smith, Edinburgh Review



Agenda

- **Encoding**
- Compression
- Huffman Coding

2

Encoding

- UTCS
- 85 84 67 83
- 01010101 01010100 01000011 01010011
- what is a file?
- open a bitmap in a text editor

3

Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	(space)	100 0000	100	64	40	@	110 0000	140	96	60	`
010 0001	041	33	21	!	100 0001	101	65	41	A	110 0001	141	97	61	a
010 0010	042	34	22	"	100 0010	102	66	42	B	110 0010	142	98	62	b
010 0011	043	35	23	#	100 0011	103	67	43	C	110 0011	143	99	63	c
010 0100	044	36	24	\$	100 0100	104	68	44	D	110 0100	144	100	64	d
010 0101	045	37	25	%	100 0101	105	69	45	E	110 0101	145	101	65	e
010 0110	046	38	26	&	100 0110	106	70	46	F	110 0110	146	102	66	f
010 0111	047	39	27	'	100 0111	107	71	47	G	110 0111	147	103	67	g
010 1000	050	40	28	(100 1000	110	72	48	H	110 1000	150	104	68	h
010 1001	051	41	29)	100 1001	111	73	49	I	110 1001	151	105	69	i
010 1010	052	42	2A	*	100 1010	112	74	4A	J	110 1010	152	106	6A	j
010 1011	053	43	2B	+	100 1011	113	75	4B	K	110 1011	153	107	6B	k
010 1100	054	44	2C	,	100 1100	114	76	4C	L	110 1100	154	108	6C	l
010 1101	055	45	2D	-	100 1101	115	77	4D	M	110 1101	155	109	6D	m
010 1110	056	46	2E	.	100 1110	116	78	4E	N	110 1110	156	110	6E	n
010 1111	057	47	2F	/	100 1111	117	79	4F	O	110 1111	157	111	6F	o
011 0000	060	48	30	0	101 0000	120	80	50	P	111 0000	160	112	70	p
011 0001	061	49	31	1	101 0001	121	81	51	Q	111 0001	161	113	71	q
011 0010	062	50	32	2	101 0010	122	82	52	R	111 0010	162	114	72	r
011 0011	063	51	33	3	101 0011	123	83	53	S	111 0011	163	115	73	s
011 0100	064	52	34	4	101 0100	124	84	54	T	111 0100	164	116	74	t
011 0101	065	53	35	5	101 0101	125	85	55	U	111 0101	165	117	75	u
011 0110	066	54	36	6	101 0110	126	86	56	V	111 0110	166	118	76	v
011 0111	067	55	37	7	101 0111	127	87	57	W	111 0111	167	119	77	w

4

Text File

Project Gutenberg's The Adventures of Sherlock

This eBook is for the use of anyone anywhere at almost no restrictions whatsoever. You may copy it under the terms of the Project Gutenberg License or online at www.gutenberg.net

Title: The Adventures of Sherlock Holmes

Author: Arthur Conan Doyle

Posting Date: April 18, 2011 [EBook #1661]
First Posted: November 29, 2002

5

Text File???

```
50 72 6F 6A 65 63 74 20 47 75 74 65 6E 62 65 72 Project Gutenber
67 27 73 20 54 68 65 20 41 64 76 65 6E 74 75 72 g's The Adventur
65 73 20 6F 66 20 53 68 65 72 6C 6F 63 6B 20 48 es of Sherlock H
6F 6C 6D 65 73 2C 20 62 79 20 41 72 74 68 75 72 olmes, by Arthur
20 43 6F 6E 61 6E 20 44 6F 79 6C 65 0D 0A 0D 0A Conan Doyle....
54 68 69 73 20 65 42 6F 6F 6B 20 69 73 20 66 6F This eBook is fo
72 20 74 68 65 20 75 73 65 20 6F 66 20 61 6E 79 r the use of any
6F 6E 65 20 61 6E 79 77 68 65 72 65 20 61 74 20 one anywhere at
6E 6F 20 63 6F 73 74 20 61 6E 64 20 77 69 74 68 no cost and with
0D 0A 61 6C 6D 6F 73 74 20 6E 6F 20 72 65 73 74 ..almost no rest
72 69 63 74 69 6F 6E 73 20 77 68 61 74 73 6F 65 rictions whatsoe
```

6

Bitmap File



7

Bitmap File????

```
07 01 06 05 00 04 03 01 04 02 02 05 03 03 07 02 .....
05 06 02 04 05 01 00 01 00 00 02 00 00 02 00 00 .....
03 01 04 07 05 09 0C 0A 07 0A 08 02 05 03 00 04 .....
05 00 04 03 00 04 03 01 03 03 02 05 03 04 05 03 .....
05 06 02 05 06 02 08 09 07 05 06 04 01 02 00 04 .....
04 04 0E 0E 0E 0E 0E 0E 0C 0B 0D 11 10 12 06 00 .....
0D 00 15 00 8F AD 3C FF F8 A1 FB F9 BF F1 F8 B5 .....
FF FF C7 F3 FD A0 D2 EB 6B EE F9 7F FB FF 8F F7 .....
FF 94 FD FF A5 AA 9F 6B 02 03 00 00 00 16 42 81 .....
25 55 8D 3A 3D 68 25 2F 4C 1B 31 41 22 38 3F 30 .....
0F 11 0B 0D 0B 0A 09 0E 11 00 04 07 00 01 02 06 .....
0C 0B 0C 13 10 06 0D 08 00 07 00 00 04 00 00 01 .....
00 02 07 06 04 09 08 00 04 03 00 03 02 05 0A 09 .....
06 0B 0A 01 06 05 12 06 12 03 05 0D 00 0E 00 28 .....
6C 17 E5 FA 7B EE FF AA FB FB DD F5 F8 D8 FF FD .....
C5 FA F4 B3 F7 FA BC F9 FD D9 FF FF FA FC F5 F8 .....
FF FB E9 FF F8 CC C9 FC 82 F1 FA C8 FF F4 F5 FB .....
F5 E8 F9 FD DA FF FF E8 FF FF FB BC D7 CE 31 50 .....
2B 01 16 00 00 02 00 01 03 04 09 08 04 0E 0D 03 .....
03 04 00 09 08 04 0B 0C 0A 0A 0B 09 06 07 05 00 .....
01 00 00 01 00 06 07 05 0B 0C 0A 0D 0E 0C 19 11 .....
00 00 01 00 10 01 0D 00 01 00 00 00 00 00 00 00 ..... P1286
```

8

JPEG File



9

JPEG VS BITMAP

	Tower-number1-1024x768.bmp	Bitmap image	2,305 KB
	Tower-number1-1024x768.jpg	JPEG Image	283 KB

- JPEG File

F	D8	FF	E0	00	10	4A	46	49	46	00	01	01	01	00	48	00000000	JFIF	H
0	48	00	00	FF	DB	00	43	00	02	01	01	02	01	01	02	00000000	H.ÿÿ.C	
2	02	02	02	02	02	02	03	05	03	03	03	03	03	06	04	00000000		
4	03	05	07	06	07	07	07	06	07	07	08	09	0B	09	08	00000000		
8	0A	08	07	07	0A	0D	0A	0A	0B	0C	0C	0C	0C	07	09	00000000		
E	0F	0D	0C	0E	0B	0C	0C	0C	FF	DB	00	43	01	02	02	00000000	ÿÿ.C	
2	03	03	03	06	03	03	06	0C	08	07	08	0C	0C	0C	0C	00000000		
C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	00000000		
C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	00000000		
C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	FF	00000000	ÿA	
0	11	08	03	00	04	00	03	01	22	00	02	11	01	03	11	00000000	"	
1	FF	C4	00	1F	00	00	01	05	01	01	01	01	01	01	00	00000000	ÿA	
0	00	00	00	00	00	00	01	02	03	04	05	06	07	08	09	00000000		
A	0B	FF	C4	00	B5	10	00	02	01	03	03	02	04	03	05	00000000	ÿA.p	
5	04	04	00	00	01	7D	01	02	03	00	04	11	05	12	21	00000000	}.....!	
1	41	06	13	51	61	07	22	71	14	32	81	91	A1	08	23	00000000	1A Qa "q 2' i .#	
2	B1	C1	15	52	D1	F0	24	33	62	72	82	09	0A	16	17	00000000	BtA.RN\$3brI	
8	19	1A	25	26	27	28	29	2A	34	35	36	37	38	39	3A	00000000	%&'()*456789;	

10

Encoding Schemes

- "It's all 1s and 0s"
- What do the 1s and 0s mean?
- 50 121 109
- ASCII -> 2ym
- Red Green Blue-> dark teal?



11

Altering files

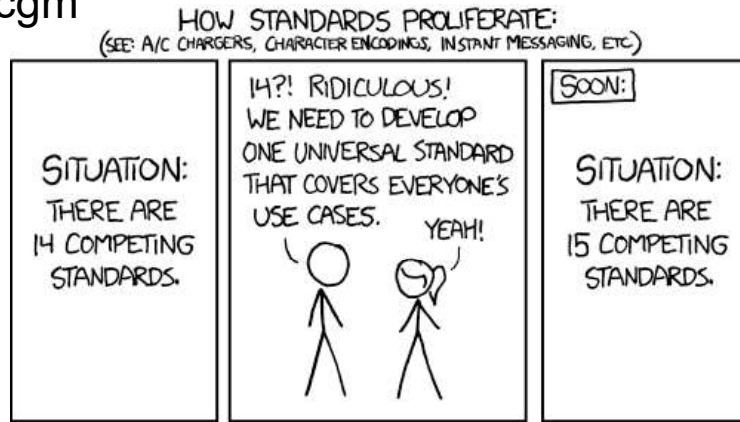
- Tower bit map (Eclipse/Huffman/Data).
Alter the first 300 characters of line
16784 to this

[illegible]

12

Why So Many Encoding Decoding Schemes?

- Image file formats: bmp, png, jpg, gif, tiff, svg, cgm



- XKCD, Standards: <https://xkcd.com/927/>

13



Agenda

- Encoding
- **Compression**
- Huffman Coding

14

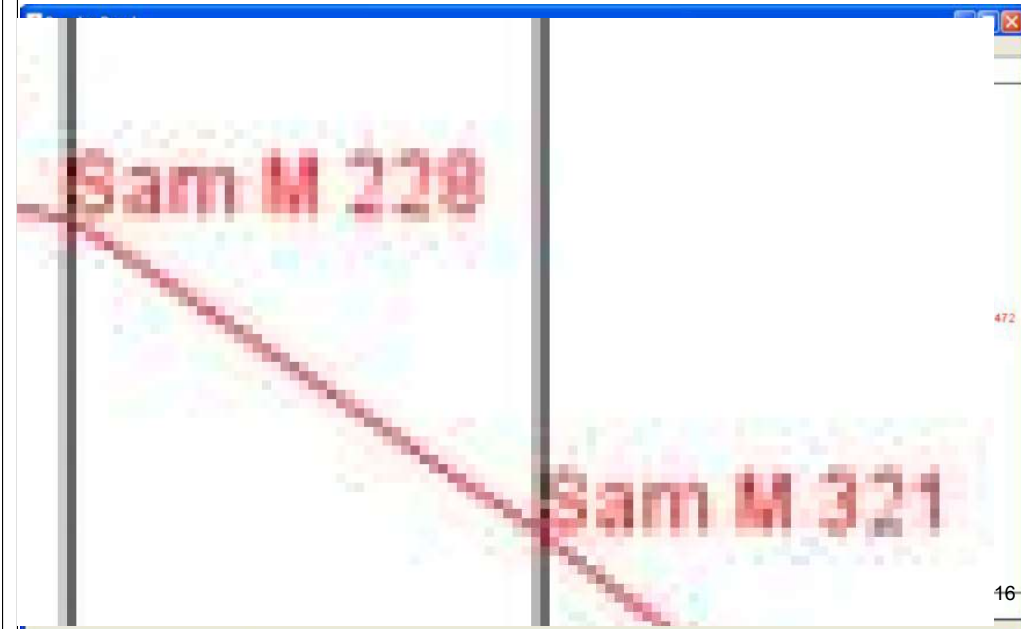
Compression

- Compression: Storing the same information but in a form that takes less memory
- lossless and lossy compression
- Recall:

	Tower-number1-1024x768.bmp	Bitmap image	2,305 KB
	Tower-number1-1024x768.jpg	JPEG Image	283 KB

15

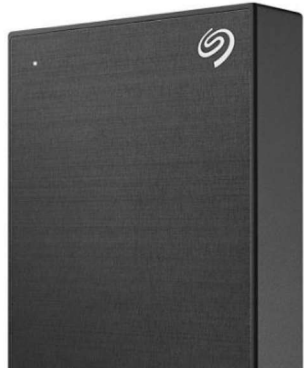
Lossy Artifacts



16

Why Bother?

- Is compression really necessary?



Seagate Backup Plus **5TB** External Hard Drive Portable HDD – Black USB 3.0 for PC Laptop and Mac, 1 year MylioCreate, 2 Months Adobe CC Photography (STHP5000400)

by Seagate
 ★★★★★ 287 ratings | 148 answered questions
 List Price: \$439.99
 Price: **\$109.99** ✓prime FREE One-Day & FREE Returns
 You Save: \$20.00 (15%)
 Get \$70 off instantly: Pay **\$39.99** ~~\$109.99~~ upon approval for the Amazon Prime Rewards Visa Card. No annual fee.
 Free Amazon tech support included
 Capacity: **5TB**
 1TB 2TB 4TB 5TB
 Color: **Black**
 Black Red Silver Blue

5 Terabytes

1250 HD, 2 hour movies or 1,250,000 songs
 Price? About \$110.00

17

Clicker 1

- With computer storage so cheap, is compression really necessary?
- A. No
 B. Yes
 C. It Depends

18

Little Pipes and Big Pumps

Home Internet Access

- 400 Mbps roughly \$85 per month
- 12 months * 3 years * \$70 = \$2,520
- 400,000,000 *bits* /second = $5 * 10^7$ bytes / sec

CPU Capability

- \$2,000 for a good laptop or desktop
- Intel® Core™ i9-7900X**
- Assume it lasts 3 years.
- Memory bandwidth 40 GB / sec = $4.0 * 10^{10}$ bytes / sec
- on the order of $6.4 * 10^{11}$ instructions / second

19

Mobile Devices?

Cellular Network

- Your mileage may vary ...
- Mega bits per second
- AT&T
17 mbps download, 7 mbps upload
- T-Mobile & Verizon
12 mbps download, 7 mbps upload
- 17,000,000 bits per second = $2.125 * 10^6$ bytes per second

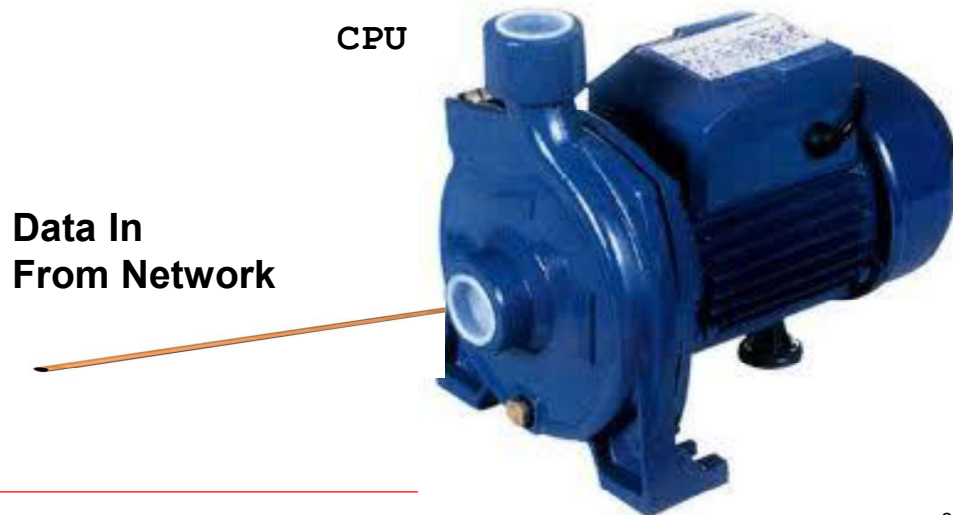
iPhone CPU

- Apple A6 System on a Chip
- Coy about IPS
- 2 cores
- Rough estimates: $1 * 10^{10}$ instructions per second

<http://tinyurl.com/q6o7wan>

20

Little Pipes and Big Pumps



21

Compression - Why Bother?

- Apostolos "Toli" Leros
- Facebook Engineer
- Heads image storage group
- jpeg images already compressed
- look for ways to compress even more
- 1% less space = millions of dollars in savings



Agenda

- Encoding
- Compression
- **Huffman Coding**

23

Huffman Coding

- Proposed by Dr. David A. Huffman
 - Graduate class in 1951 at MIT with Robert Fano
 - term paper or final
 - term paper: prove min bits needed for binary coding of data
 - *A Method for the Construction of Minimum Redundancy Codes*
- Applicable to many forms of data transmission
 - Our example: text files
 - still used in fax machines, mp3 encoding, others

24

The Basic Algorithm

- Huffman coding is a form of statistical coding
 - Not all characters occur with the same frequency, in typical text files. (can be true when reading raw bytes as well)
 - Yet in ASCII all characters are allocated the same amount of space
 - 1 char = 1 byte, be it **e** or **X**
- fixed width encoding

25

The Basic Algorithm

- Any savings in tailoring codes to frequency of character?
- Code word lengths are no longer fixed like ASCII or Unicode
- Code word lengths vary and will be shorter for the more frequently used characters
- Examples use characters for clarity, but in reality just read raw bytes from file.

26

The Basic Algorithm

1. Scan file to be compressed and determine frequency of all values.
2. Sort or prioritize values based on frequency in file.
3. Build Huffman code tree based on prioritized values.
4. Perform a traversal of tree to determine new codes for values.
5. Scan file again to create new file using the new Huffman codes

27

Building a Tree

Scan the original text

- Consider the following short text

Eerie eyes seen near lake.

- Determine frequency of all numbers (values or in this case characters) in the text

28

Building a Tree

Scan the original text

Eerie eyes seen near lake.

- What characters are present?

E e r i space
y s n a r l k .

29

Building a Tree

Scan the original text

Eerie eyes seen near lake.

- What is the frequency of each character in the text?

Char	Freq.	Char	Freq.	Char	Freq.
E	1	y	1	k	1
e	8	s	2	.	1
r	2	n	2		
i	1	a	2		
space	4	l	1		

30

Building a Tree

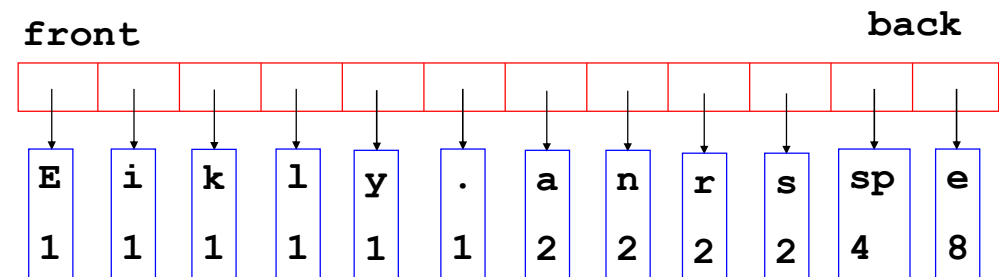
Prioritize characters

- Create binary tree nodes with a value and the frequency for each value
- Place nodes in a priority queue
 - The **lower** the frequency, the **higher** the priority in the queue

31

Building a Tree

- The queue after enqueueing all nodes



- Null Pointers are not shown
- sp = space

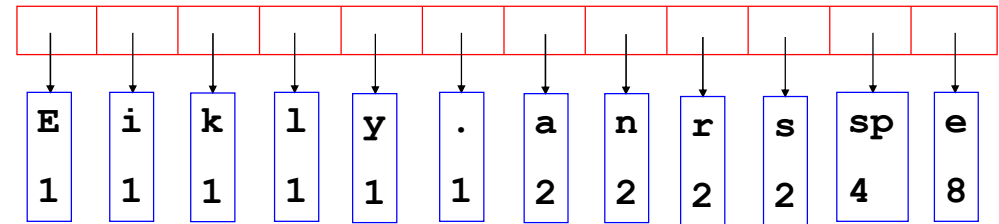
32

Building a Tree

- While priority queue contains two or more nodes
 - Create new node
 - Dequeue node and make it left child
 - Dequeue next node and make it right child
 - Frequency of new node equals sum of frequency of left and right children
 - New node does not contain value
 - Enqueue new node back into the priority queue

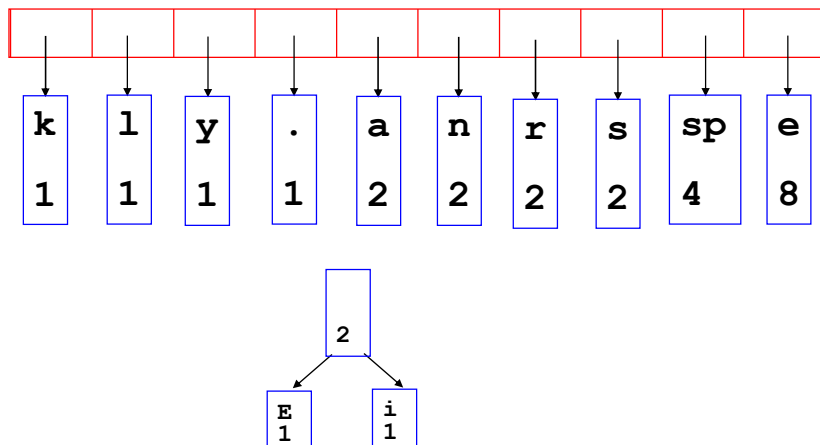
33

Building a Tree



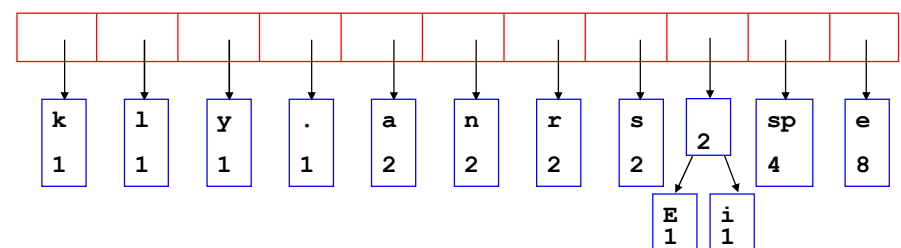
34

Building a Tree



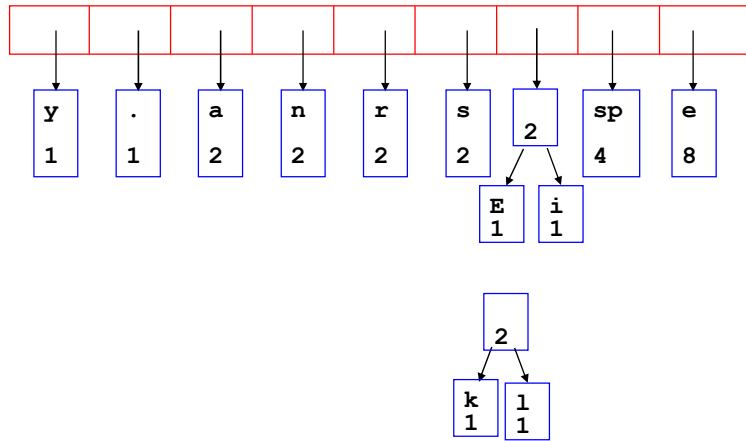
35

Building a Tree



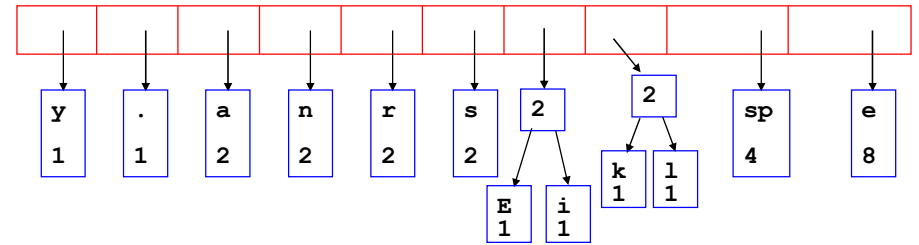
36

Building a Tree



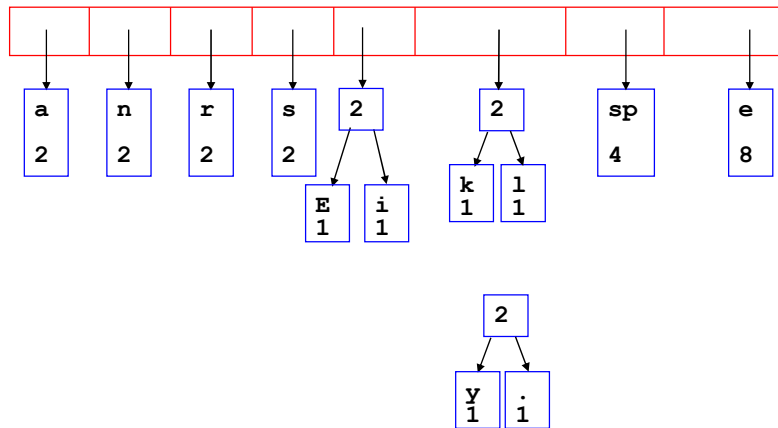
37

Building a Tree



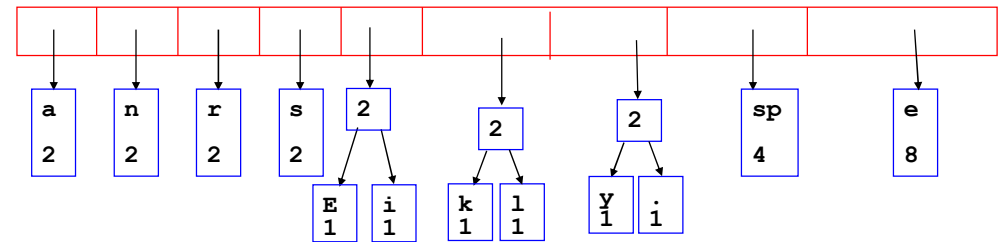
38

Building a Tree



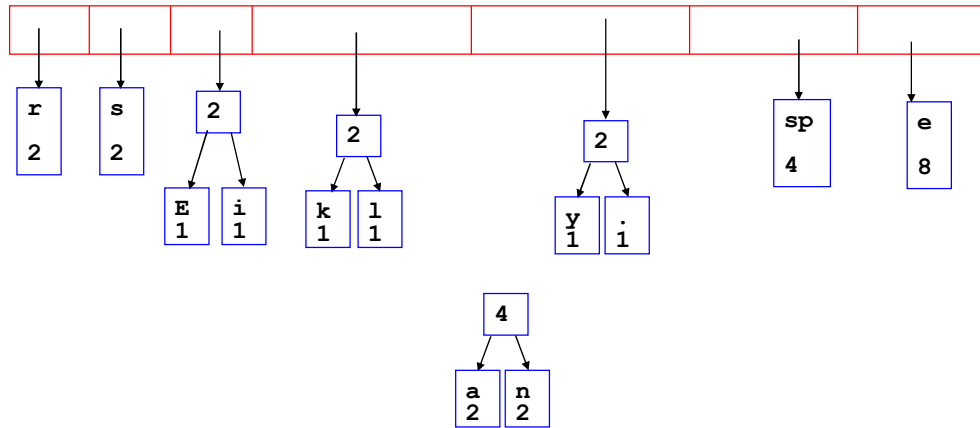
39

Building a Tree



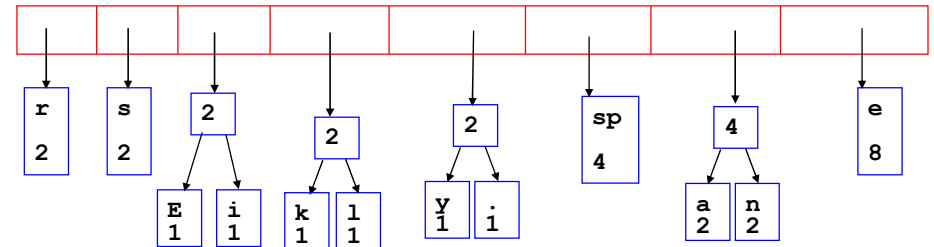
40

Building a Tree



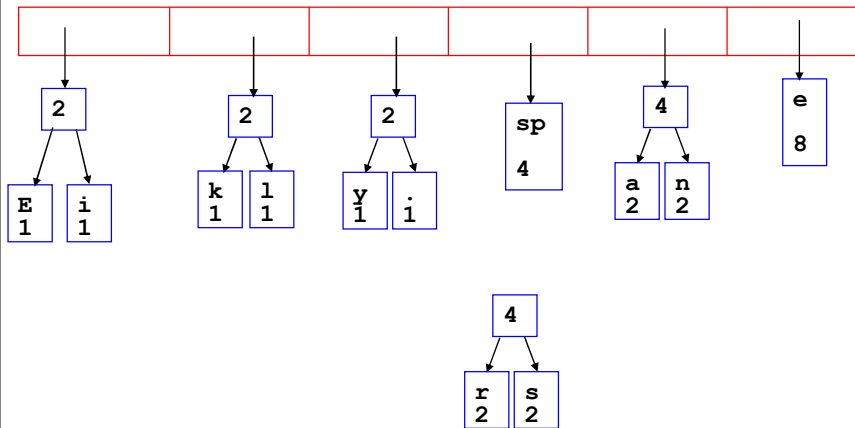
41

Building a Tree



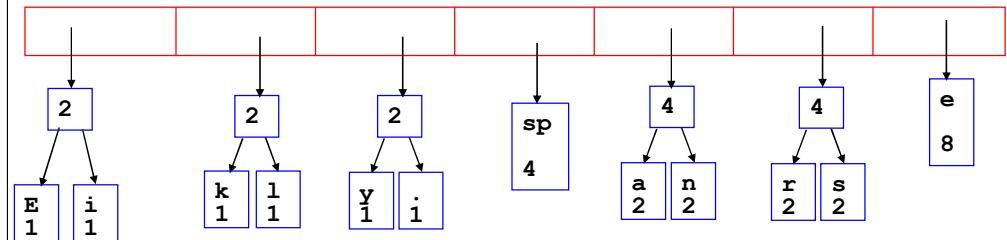
42

Building a Tree



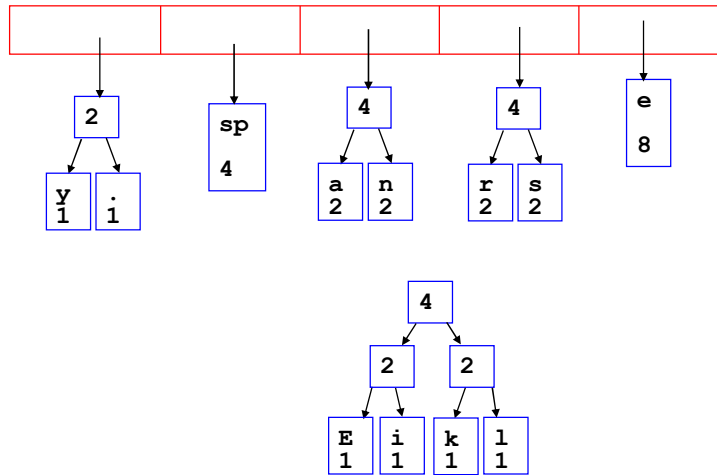
43

Building a Tree



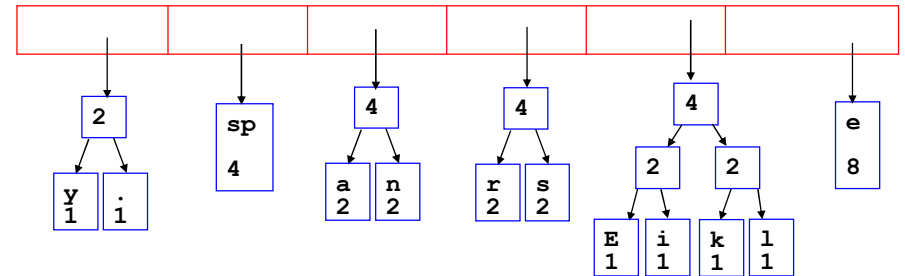
44

Building a Tree



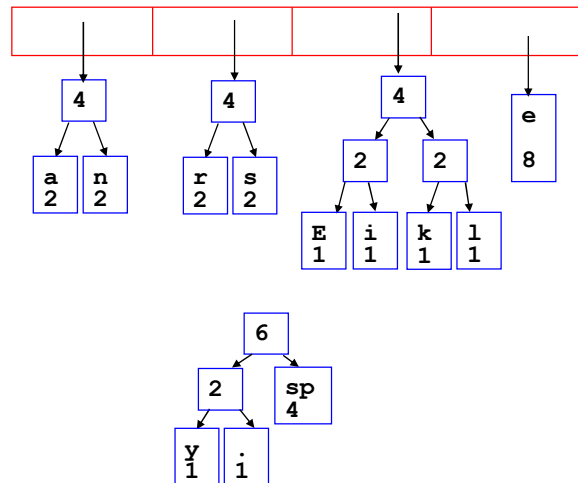
45

Building a Tree



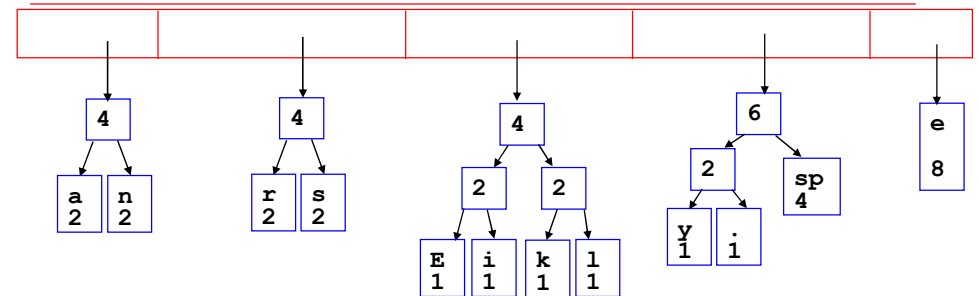
46

Building a Tree



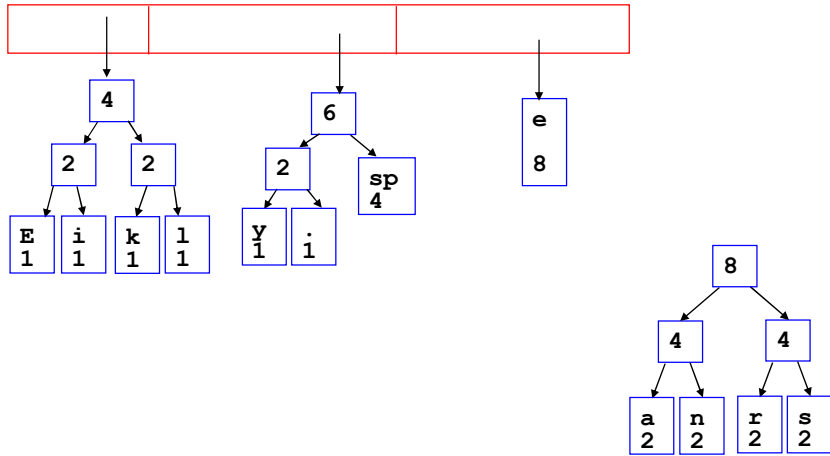
47

Building a Tree



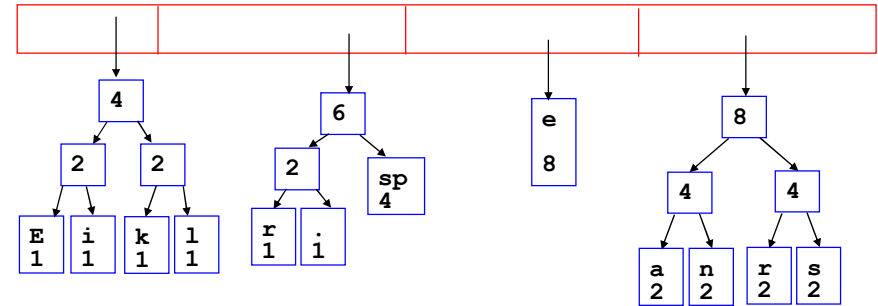
48

Building a Tree



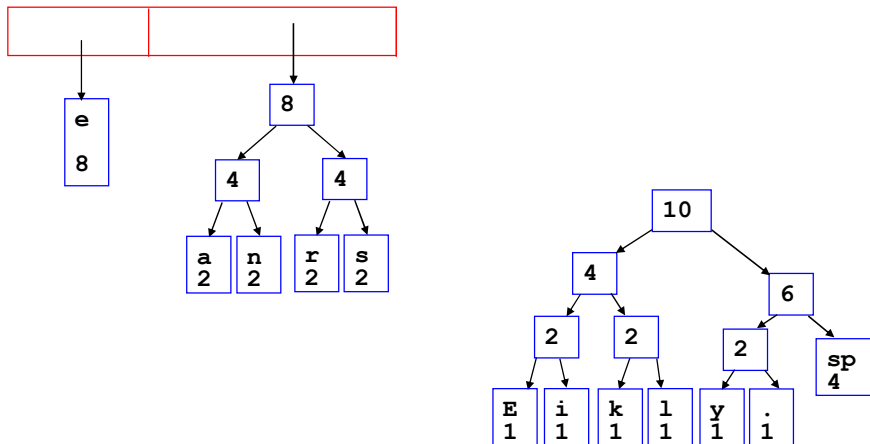
49

Building a Tree



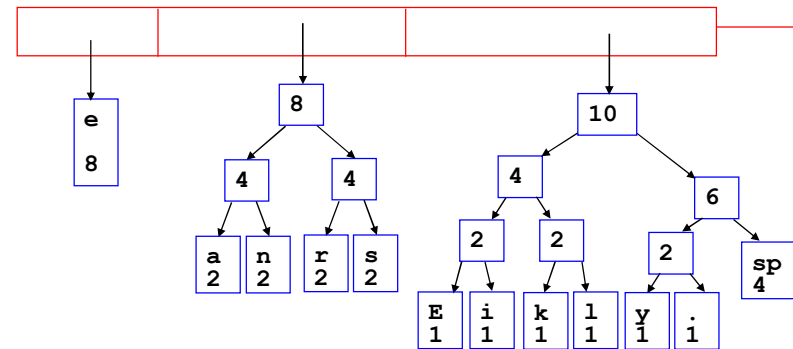
50

Building a Tree



51

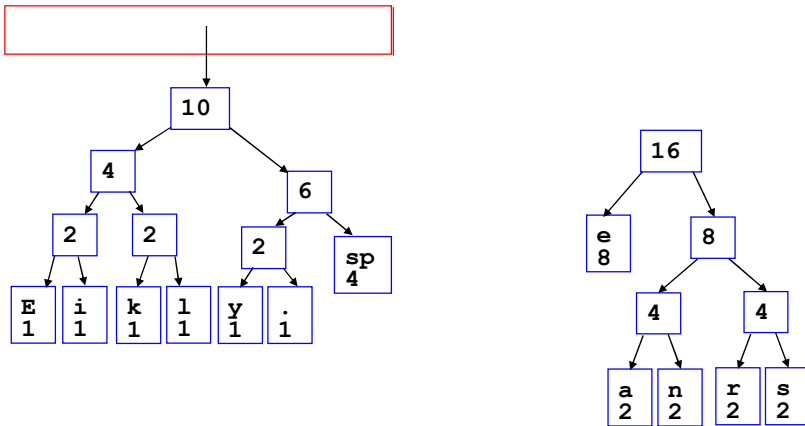
Building a Tree



Clicker 2 - What is happening to the values with a low frequency compared to values with a high freq.?
 A. Small Depth B. Large Depth
 C. Something else

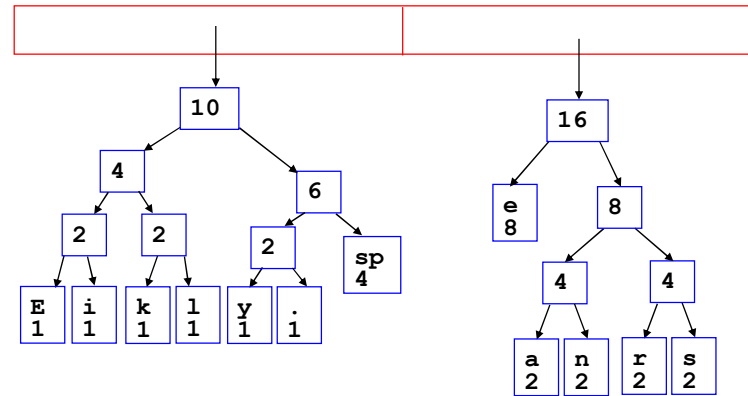
52

Building a Tree



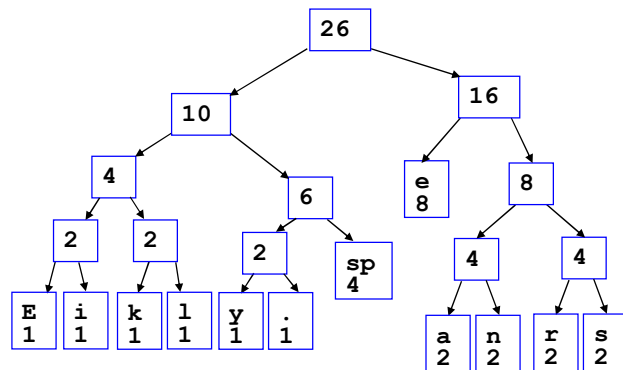
53

Building a Tree



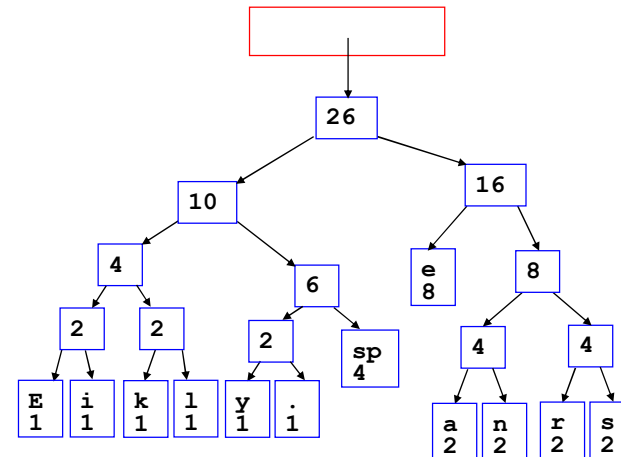
54

Building a Tree



55

Building a Tree



•After enqueueing this node there is only one node left in priority queue.

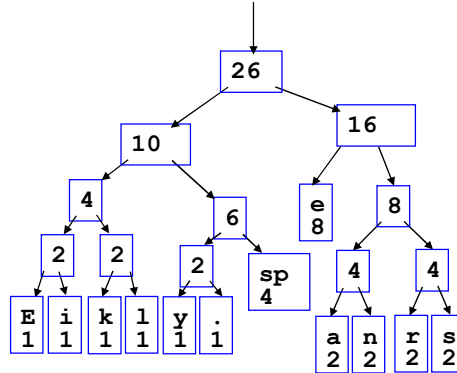
56

Building a Tree

Dequeue the single node left in the queue.

This tree contains the new code words for each character.

Frequency of root node should equal number of characters in text.

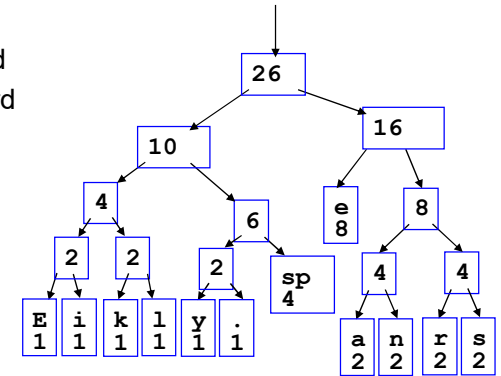


Eerie eyes seen near lake. 4 spaces,
26 characters total

57

Encoding the File Traverse Tree for Codes

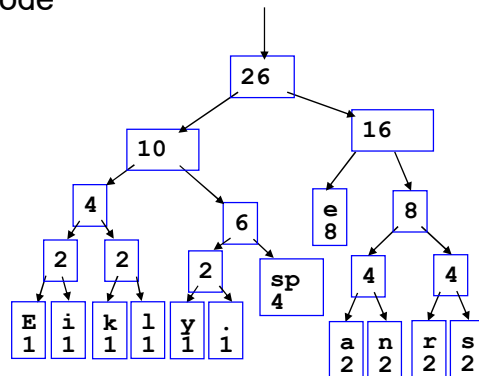
- Perform a traversal of the tree to obtain new code words (sequence of 0's and 1's)
- left, append a 0 to code word
- right append a 1 to code word
- code word is only complete when a leaf node is reached



58

Encoding the File Traverse Tree for Codes

Original Value	New Code
E	0000
i	0001
k	0010
l	0011
y	0100
.	0101
space	011
e	10
a	1100
n	1101
r	1110
s	1111



59

Prefix free codes. The code for a value is never the prefix of another code.

Encoding the File

- Rescan text and encode file using new code words

Eerie eyes seen near lake.

```
000010111000011001110
01001011110111111010
110101111011011001110
011001111000010100101
```

Char	New Code
E	0000
i	0001
k	0010
l	0011
y	0100
.	0101
space	011
e	10
a	1100
n	1101
r	1110
s	1111

60

Encoding the File Results

- Have we made things any better?
- 82 bits to encode the text
- ASCII would take $8 * 26 = 208$ bits

```
000010111000011001110
01001011110111111010
110101111011011001110
011001111000010100101
```

- If modified code used 4 bits per character are needed. Total bits $4 * 26 = 104$. Savings not as great.

61

Decoding the File

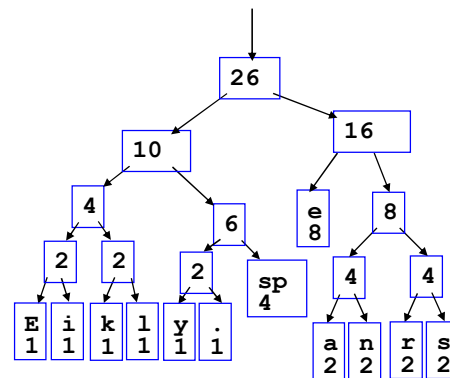
- How does receiver know what the codes are?
- Tree constructed for each text file.
 - Considers frequency for each file
 - Big hit on compression, especially for smaller files
- Tree predetermined
 - based on statistical analysis of text files or file types

62

Clicker 3 - Decoding the File

- Once receiver has tree it scans incoming bit stream
- 0 \Rightarrow go left
- 1 \Rightarrow go right

```
101000100111100011111
110111100001010
```



- elk nay sir
- eek a snake
- eek kin sly
- eek snarl nil
- eel a snarl

63

Assignment Hints

- reading chunks not chars
- header format
- the pseudo eof value
- the GUI

64

Assignment Example

- "Eerie eyes seen near lake." will result in different codes than those shown in slides due to:
 - adding elements in order to PriorityQueue
 - required pseudo eof character (PEOF)

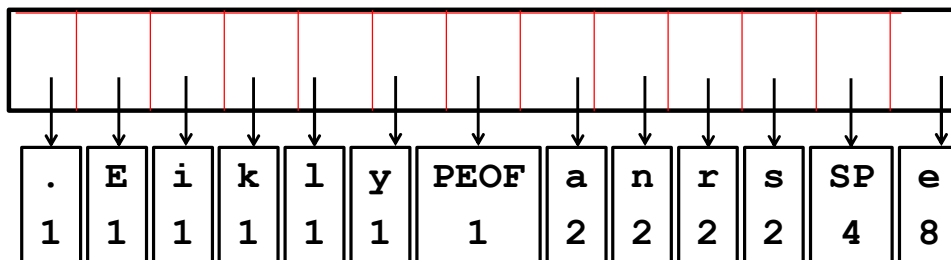
65

Assignment Example

Char	Freq.	Char	Freq.	Char	Freq.
E	1	y	1	k	1
e	8	s	2	.	1
r	2	n	2	PEOF	1
i	1	a	2		
space	4	l	1		

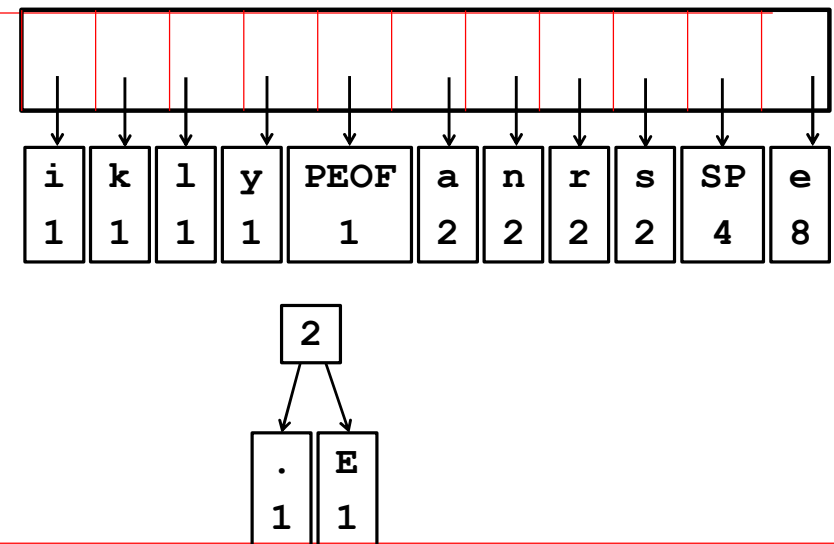
66

Assignment Example



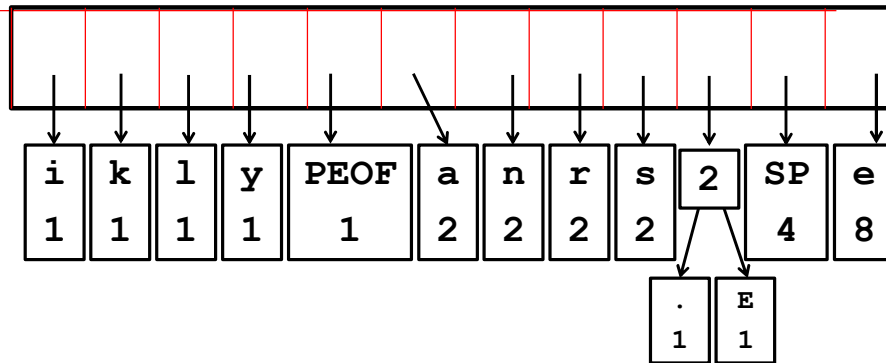
67

Assignment Example



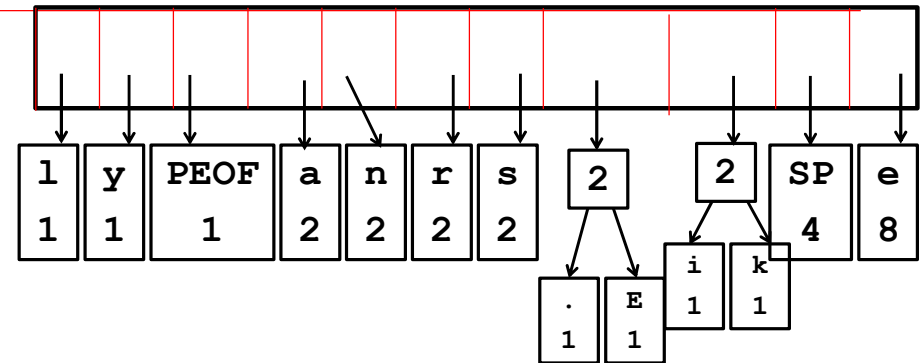
68

Assignment Example



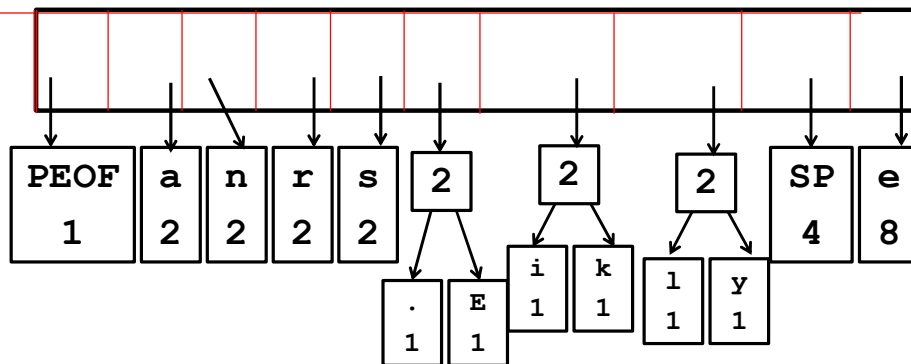
69

Assignment Example



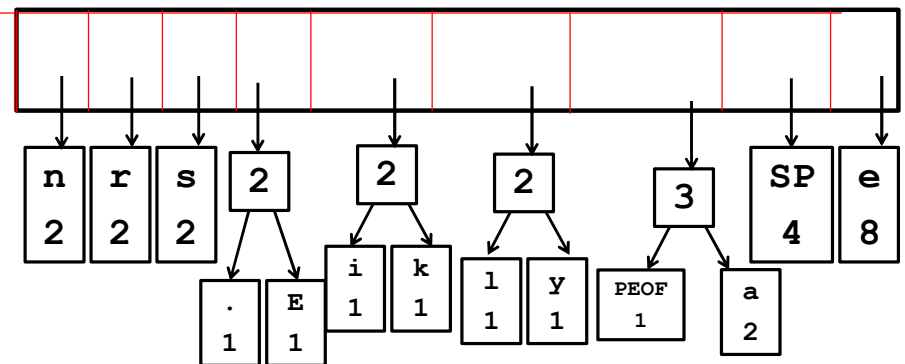
70

Assignment Example



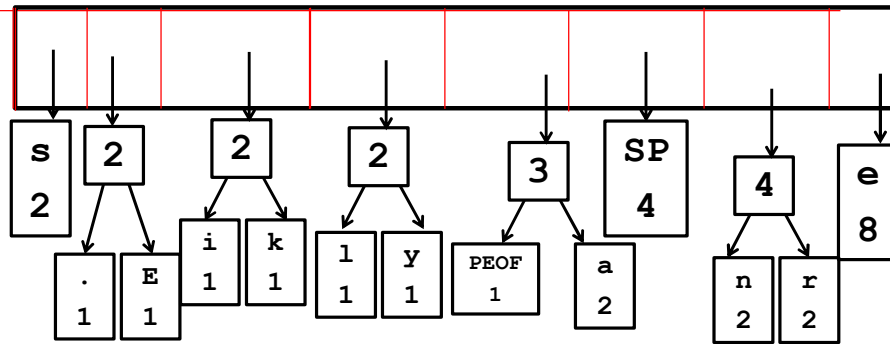
71

Assignment Example



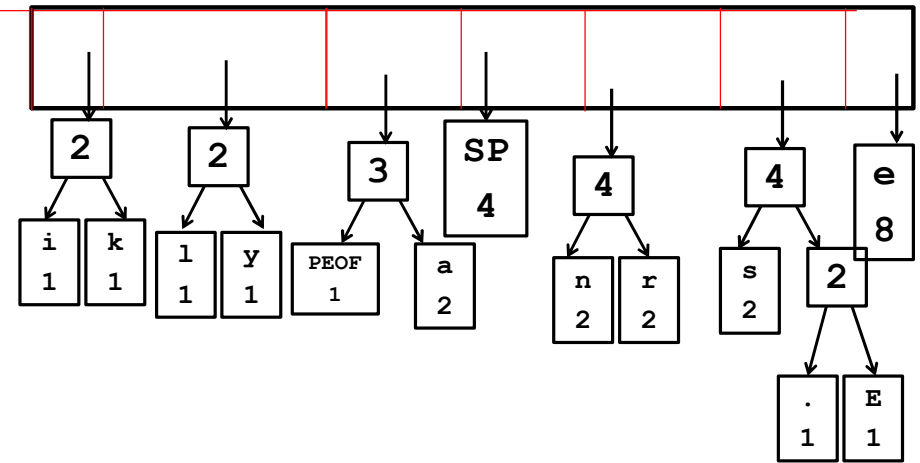
72

Assignment Example

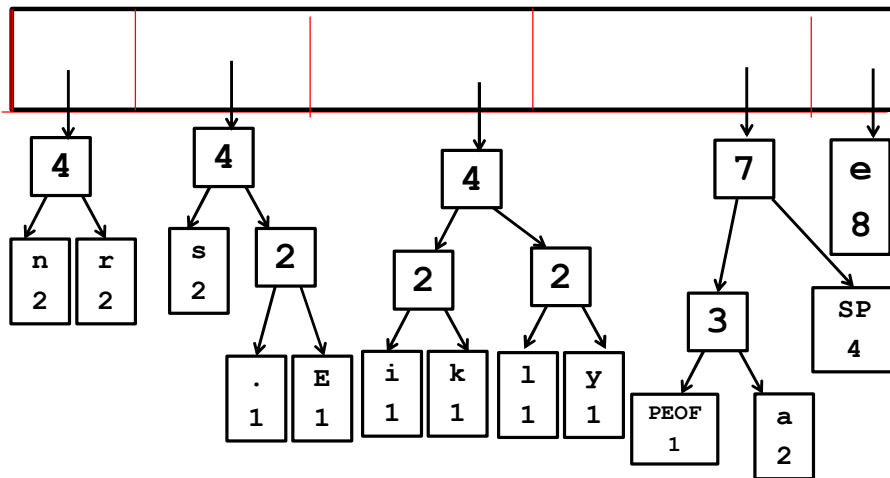


73

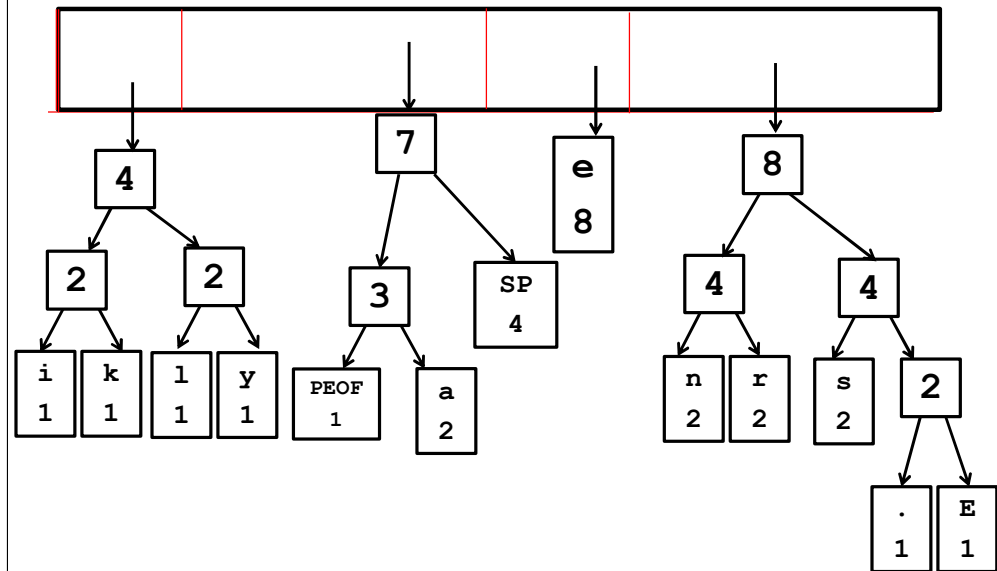
Assignment Example



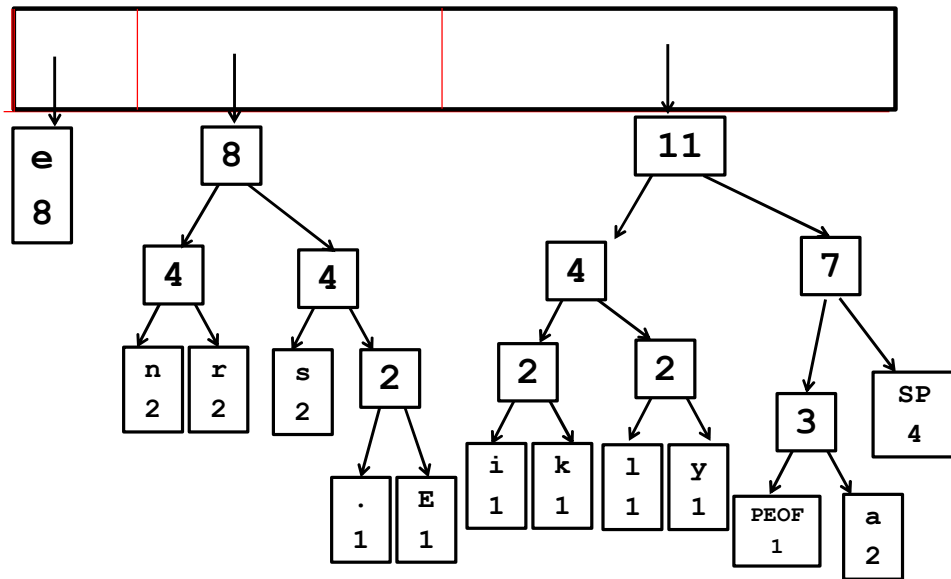
74



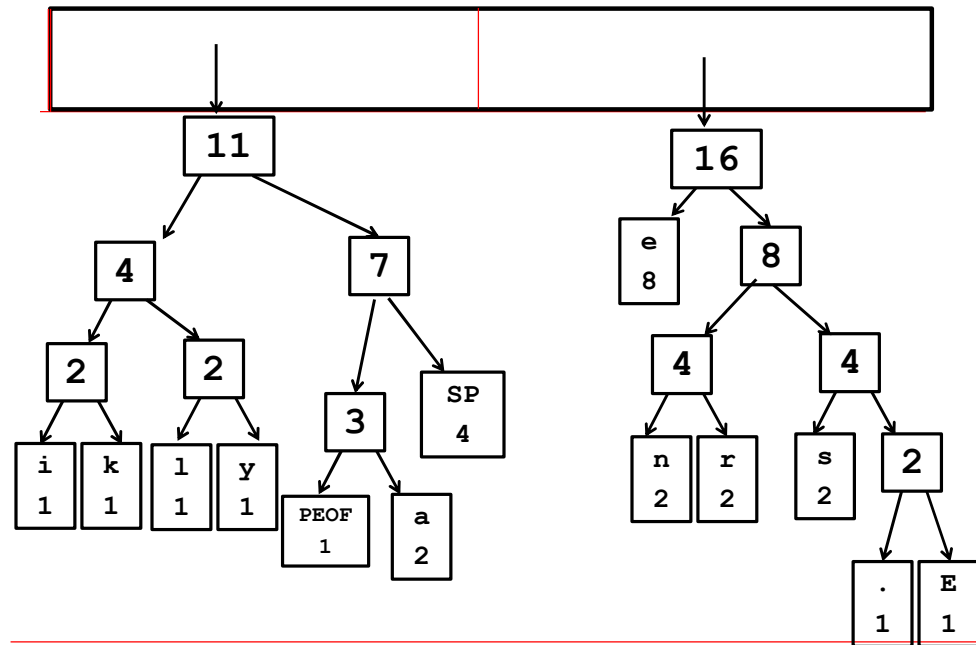
75



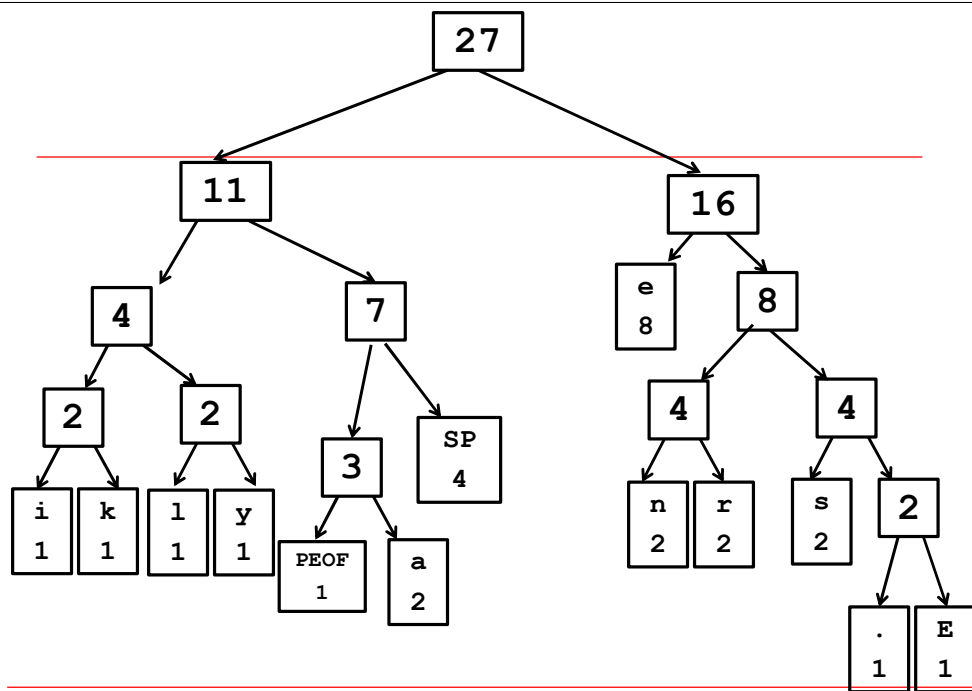
76



77



78



79

Codes

value: 32, equivalent char: , frequency: 4, new code 011
 value: 46, equivalent char: ., frequency: 1, new code 11110
 value: 69, equivalent char: E, frequency: 1, new code 11111
 value: 97, equivalent char: a, frequency: 2, new code 0101
 value: 101, equivalent char: e, frequency: 8, new code 10
 value: 105, equivalent char: i, frequency: 1, new code 0000
 value: 107, equivalent char: k, frequency: 1, new code 0001
 value: 108, equivalent char: l, frequency: 1, new code 0010
 value: 110, equivalent char: n, frequency: 2, new code 1100
 value: 114, equivalent char: r, frequency: 2, new code 1101
 value: 115, equivalent char: s, frequency: 2, new code 1110
 value: 121, equivalent char: y, frequency: 1, new code 0011
 value: 256, equivalent char: ?, frequency: 1, new code 0100

80