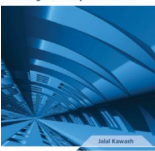


Trees & Information Coding


Peeking into Computer Science



© Jalal Kawash 2010

- Mandatory: Chapter 3 – Section 3.5

Reading Assignment



Information Coding
An application of trees

3

At the end of this section, the student will be able to:

1. Understand the need for variable-length coding (VLC)
2. Understand how compression works
3. Calculate the space savings with VLC
4. Understand decoding problems with VLC
5. Define non-prefix VLC
6. Use Huffman's algorithm and binary trees to assign optimized non-prefix VLC
7. Represent BL trees as nested lists

Objectives

Peeking into Computer Science

© Jalal Kawash 2010

- Assume we have a file that contains data composed of 6 symbols only:
- A, I, C, D, E, and S (for space)

```
ACE DICE AIDE CAID
EAD DAICED ...
```

Back to Coding (JT: Review)

Peeking into Computer Science

© Jalal Kawash 2010

5

- Assume we have a file that contains data composed of 6 symbols only:
- A, I, C, D, E, and S (for space)

```
ACESDICESAIDESCAID
EADSDAICED ...
```

Back to Coding (JT: Review)

Peeking into Computer Science

© Jalal Kawash 2010

6

- If the file has 1000 characters, how many bits (0s and 1s) are needed to code the file?

Coding (JT: Review)

Peeking into Computer Science

© Jalal Kawash 2010

7

- The first question is
- How many symbols do we need to represent each character?
- The objective is to keep the size of the file as small as possible
- We have 6 characters (messages) and two symbols (0 and 1)
- 2 is not enough, since 2^2 is 4

Coding (JT: Review)

Peeking into Computer Science

© Jalal Kawash 2010

8

- 00 for A
- 01 for S
- 10 for I
- 11 for E

- We cannot represent the rest C and D

- 3 works, since 2^3 is 8, so we can represent up to 8 characters and we only have 6



2 bits are not enough (JT: Review)

- Say
- 000 for A
- 001 for S
- 010 for I
- 011 for E
- 100 for C
- 101 for D
- 110 not used
- 111 not used



3 bits are more than enough (JT: Review)

- If the file has 1000 characters, how many bits (0s and 1s) are needed to code the file?
- Each character needs 3 bits
- Hence, we need $3 \times 1000 = 3000$ bits



Coding (JT: Review)

Peeking into Computer Science

© Jalal Kawash 2010

11

- How does file compression work?
- Huffman's codes reduce the size of a file by using variable-length codes for characters



Compression

Peeking into Computer Science

© Jalal Kawash 2010

12

- Analyzing the file we find that
 - 35% are S
 - 28% of the characters are A
 - 20% are E
 - 7% are I
 - 6% are C
 - 4% are D
- Some characters are more frequent than others



Compression

Peeking into Computer Science

© Jalal Kawash 2010

13

- If we use this coding, what is the size of the file?
- 350 S's (35% of 1000) require **700** bits (2 bits for each S)
- 200 E's require **400** bits (2 bits)
- 280 A's require **560** bits (2 bits)
- 70 I's require **210** bits (3 bits)
- 60 C's require **240** bits (4 bits)
- 40 D's require **160** bits (4 bits)
- Total is **2270** bits
- Recall that with fixed codes, the size is **3000** bits
- Compressed file size is about **76%** of original size



Compressed File Size

Peeking into Computer Science

© Jalal Kawash 2010

14

- If we use this coding, what is the size of the file?
 - 350 S's (35% of 1000) require **700** bits (2 bits for each S)
 - 200 E's require **400** bits
 - 280 A's require **560** bits
 - 70 I's require **210** bits
 - 60 C's require **240** bits
 - 40 D's require **160** bits
-
- Total is **2270** bits
 - Recall that with fixed codes, the size is **3000** bits
 - Compressed file size is about **76%** of original size



Compressed File Size

- Not any variable-length code works
 - Assume A's code is 0
 - C's code is 1
 - E's code is 01
-
- The code 0101 could correspond to ACE, EAC, ACAC, or EE



Problems with Variable-Length Codes

- Codes that work must have the property:
- No code can be the prefix of another code
- **Called Non-Prefix Codes**
- **0** is a prefix of **01**, this is why our coding failed
- Another example: **0101** is a prefix of **010111**



Prefix Codes

Peeking into Computer Science

© Jalal Kawash 2010

17

- Non-Prefix codes can be generated using a binary tree
- Start from a binary tree
- Label edges to left children with 0
- Label edges to right children with 1
- Record the labels on the path from the root to the leaves
- Each path corresponds to a non-prefix code

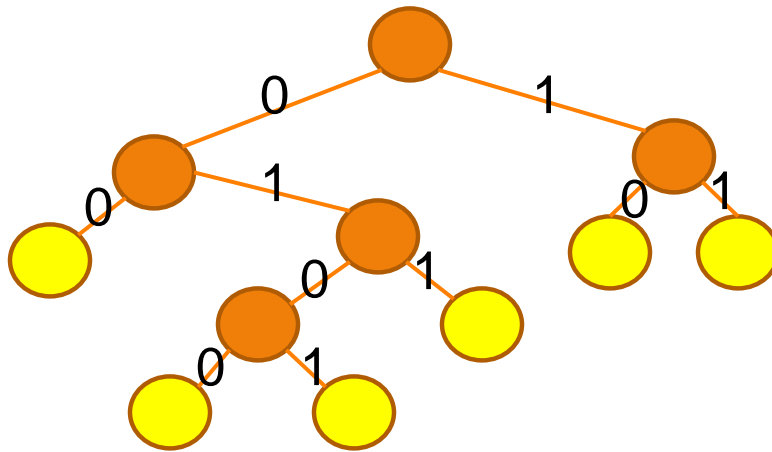


Non-Prefix Codes

Peeking into Computer Science

© Jalal Kawash 2010

18



Non-Prefix Codes from a Binary Tree

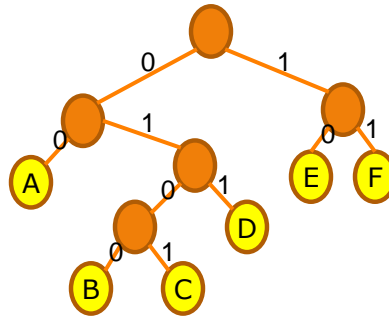


Peeking into Computer Science

© Jalal Kawash 2010

19

- A's code: 00
- B's 0100
- C's 0101
- D's 011
- E's 10
- F's 11



Non-Prefix Codes from a Binary Tree

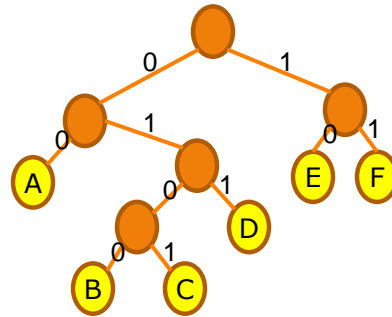


Peeking into Computer Science

© Jalal Kawash 2010

20

- 00
- 0100
- 0101
- 011
- 10
- 11



No code is a prefix of another

Peeking into Computer Science

© Jalal Kawash 2010

21

- A:00, B:0100, C:0101, D:011, E:10, F:11
- 010000011
B A D
- No other interpretation
 - Parsing left to right
 - (JT's extra: parsing refers to 'reading' or 'breaking into meaningful portions')

No Confusion

Peeking into Computer Science

© Jalal Kawash 2010

22

- A:00, B:0100, C:0101, D:011, E:10, F:11
- | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|

F E E D
- So How do we generate such codes?



No Confusion

Peeking into Computer Science

© Jalal Kawash 2010

23

- Build a binary tree
- The characters in a file are the leaves
- The most frequent characters should be closer to the root, generating shorter codes
- Assign the codes, based on this tree



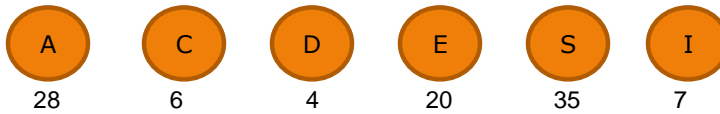
Huffman's Coding

Peeking into Computer Science

© Jalal Kawash 2010

24

1. Assign to each symbol its weight (frequency)



Each of these is a tree of size one!

Huffman's Coding – Step 1

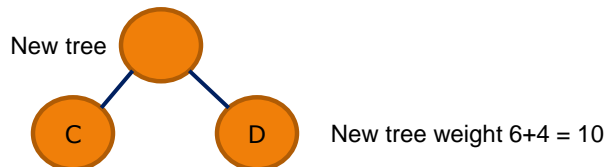
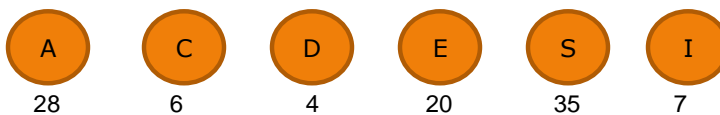
Peeking into Computer Science

© Jalal Kawash 2010

25

2. Choose two trees that have the minimum weights

- Replace these two trees with a new tree with new root
- Make the tree with the smaller weight a right child
- The weight of the new tree is the sum of old weights (JT: next slide)



Huffman's Coding – Step 2

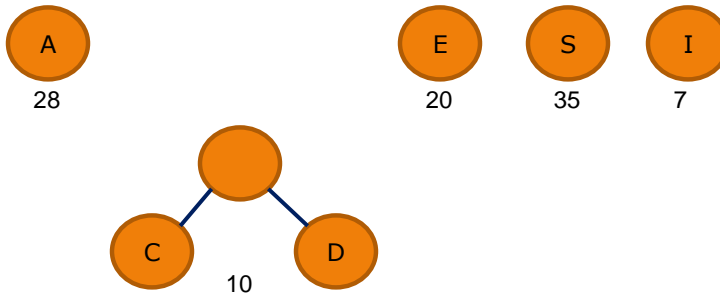
Peeking into Computer Science

© Jalal Kawash 2010

26

2. Choose two trees that have the minimum weights

- Replace these two trees with a new tree with new root
- Make the tree with the smaller weight a right child
- The weight of the new tree is the sum of old weights



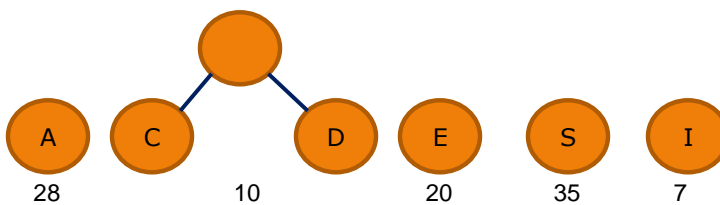
Huffman's Coding – Step 2

Peeking into Computer Science

© Jalal Kawash 2010

27

- Repeat Step 2 until we have a single tree



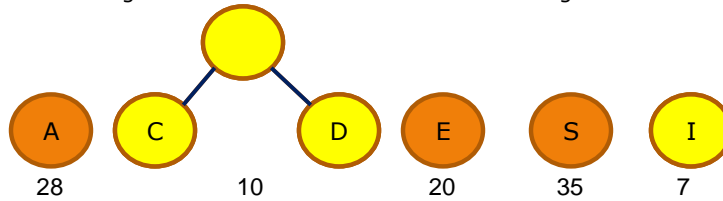
Huffman's Coding – Repeat

Peeking into Computer Science

© Jalal Kawash 2010

28

2. Choose two trees that have the minimum weights
 - Replace these two trees with a new tree with new root
 - Make the tree with the smaller weight a right child
 - The weight of the new tree is the sum of old weights



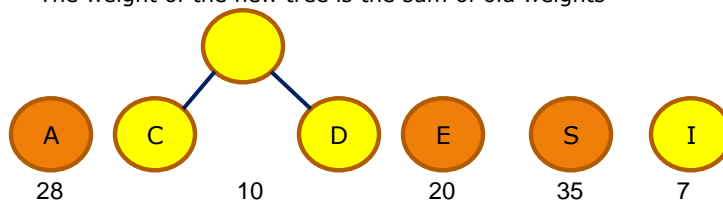
Step 2 Again

Peeking into Computer Science

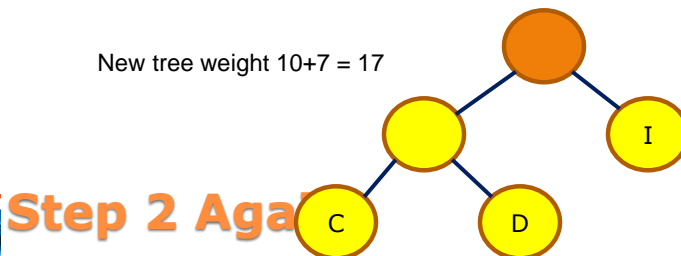
© Jalal Kawash 2010

29

2. Choose two trees that have the minimum weights
 - Replace these two trees with a new tree with new root
 - Make the tree with the smaller weight a right child
 - The weight of the new tree is the sum of old weights



New tree weight $10+7 = 17$

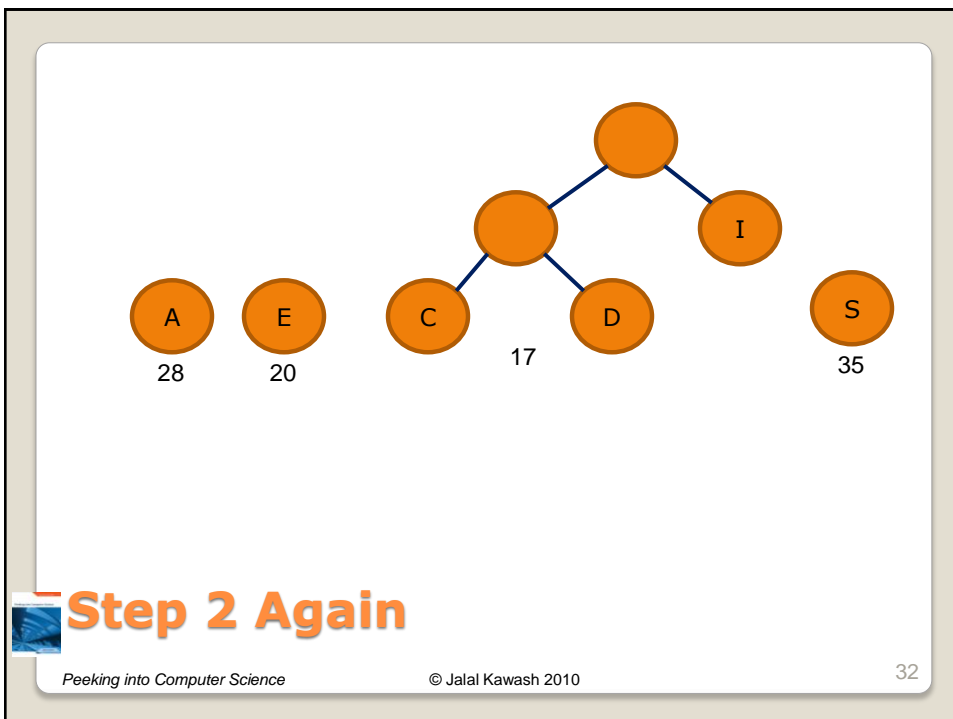
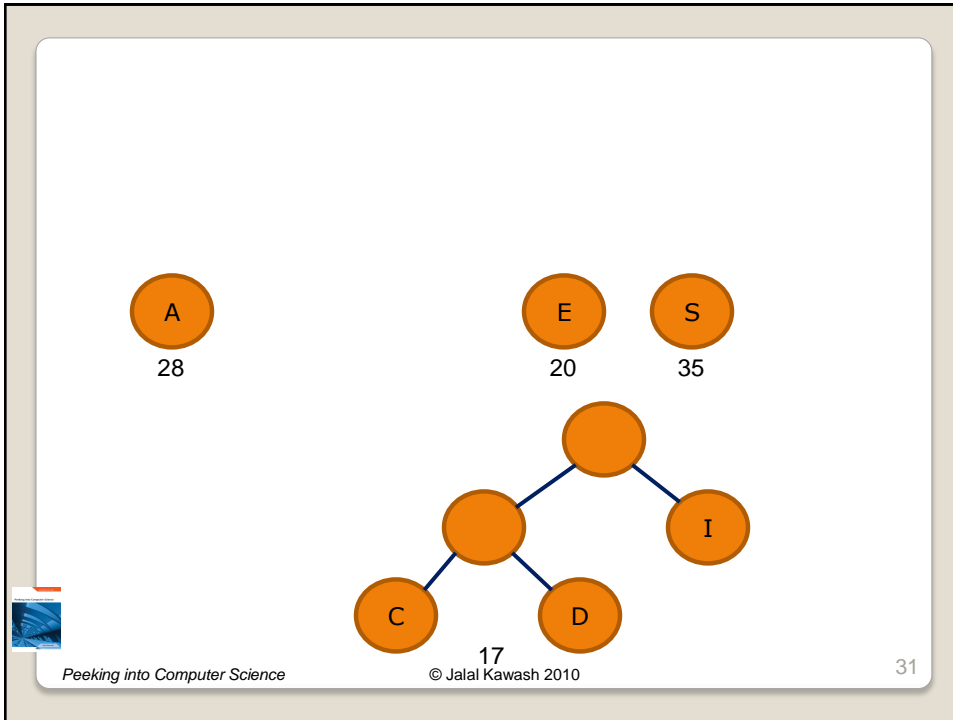


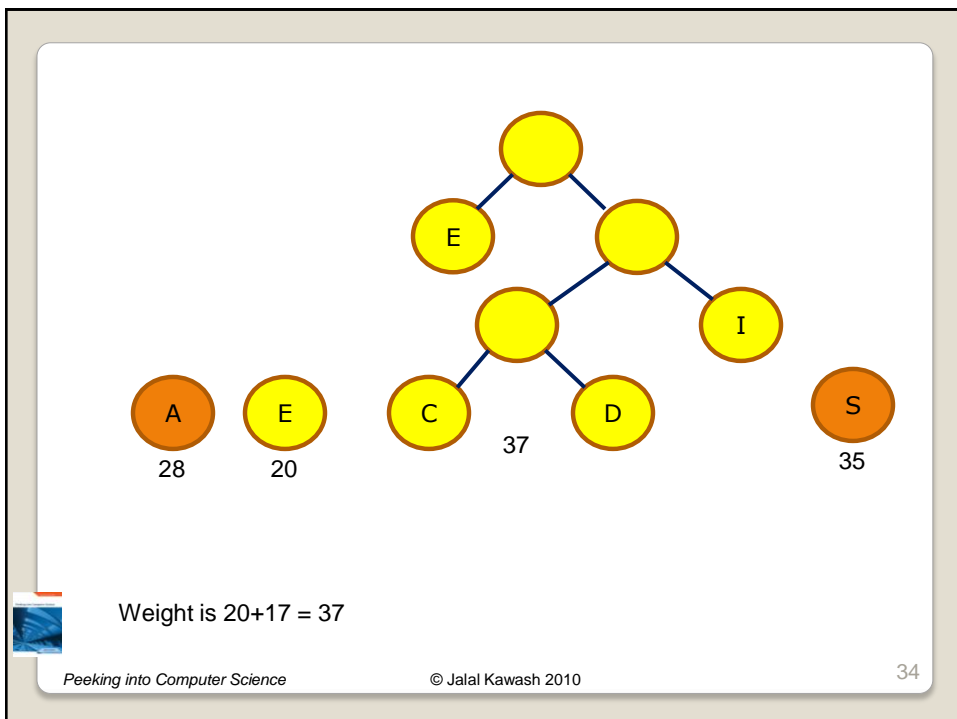
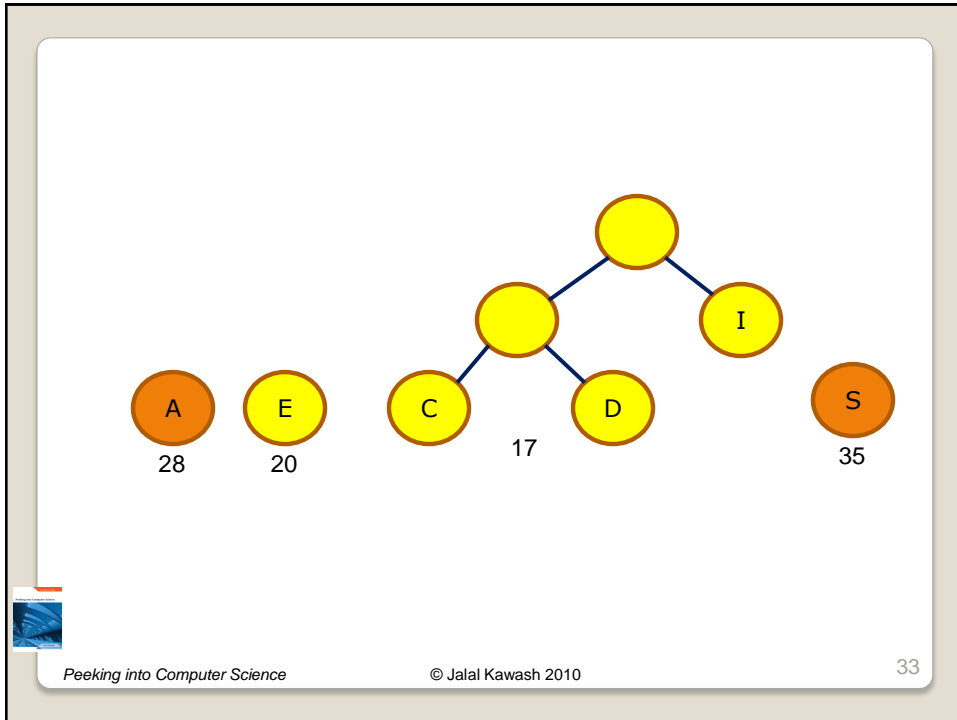
Step 2 Again

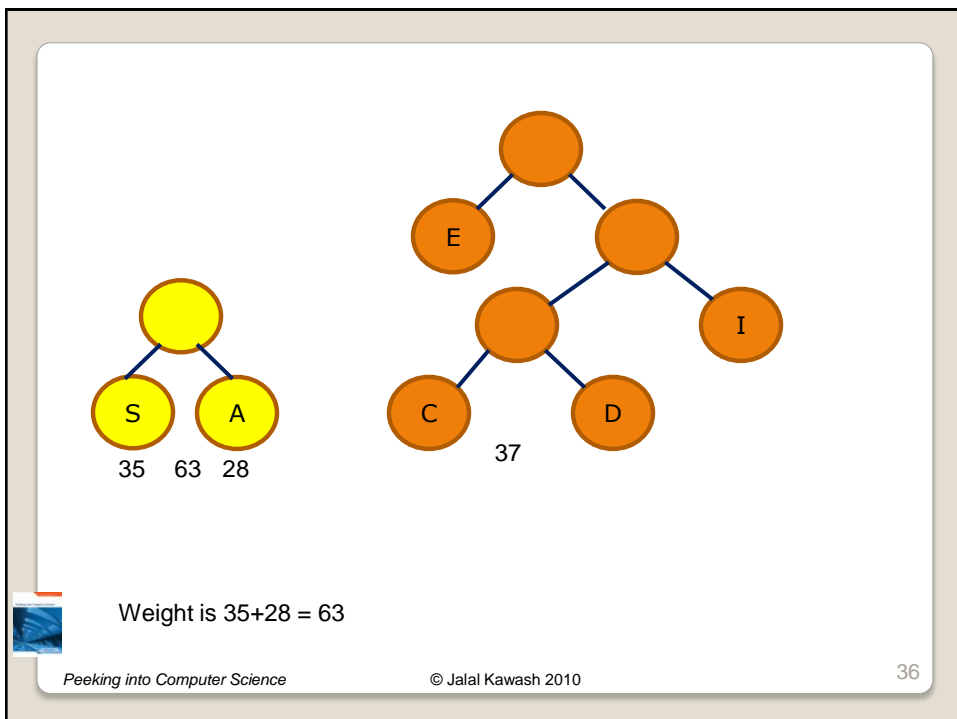
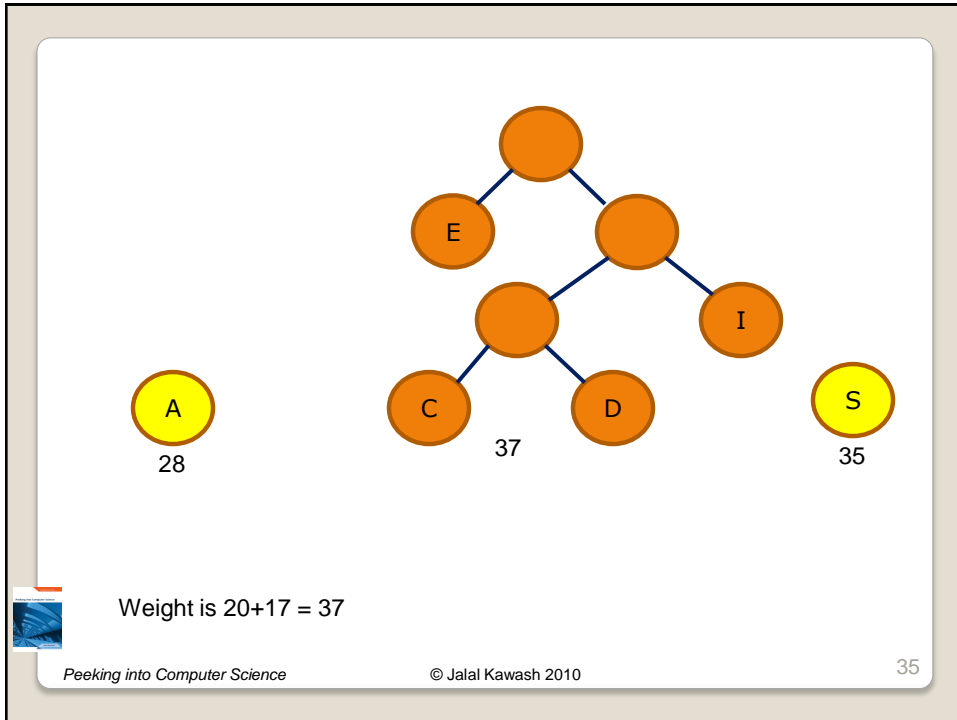
Peeking into Computer Science

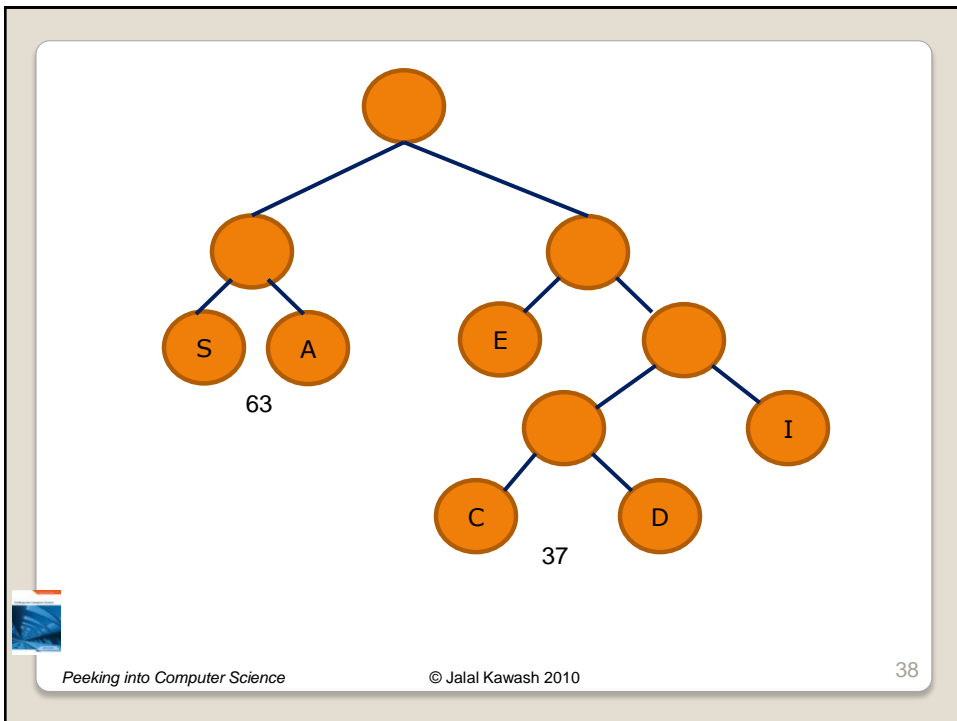
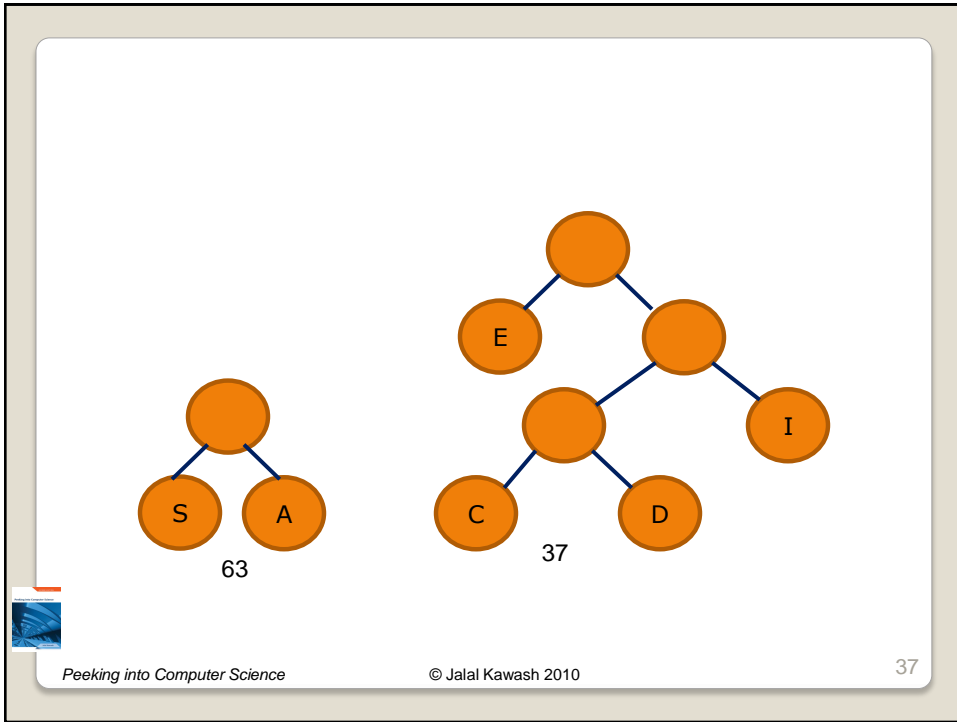
© Jalal Kawash 2010

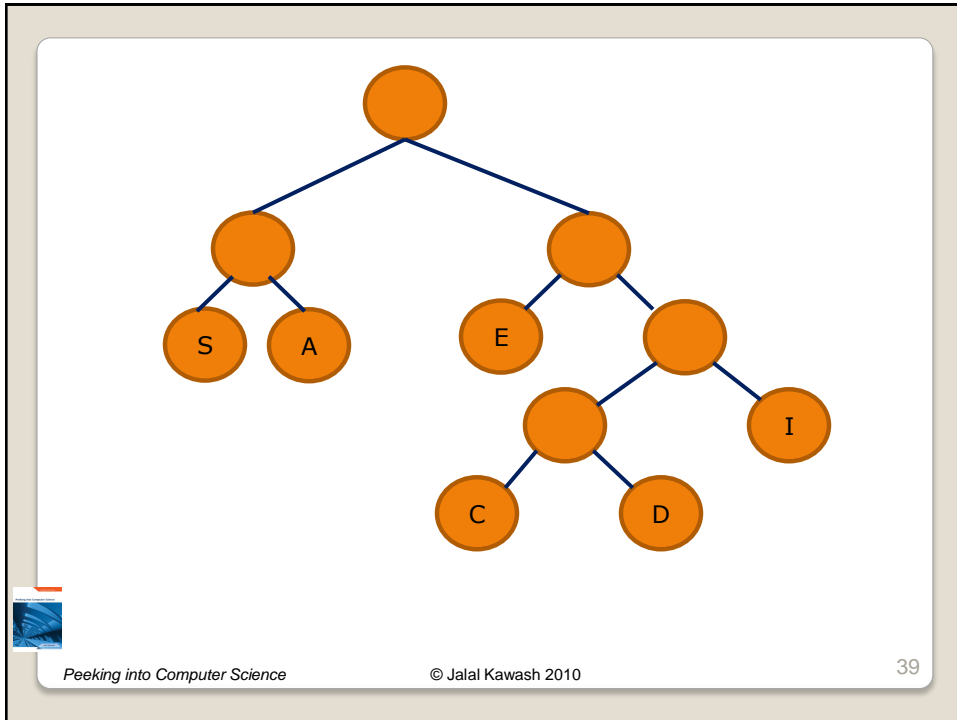
30



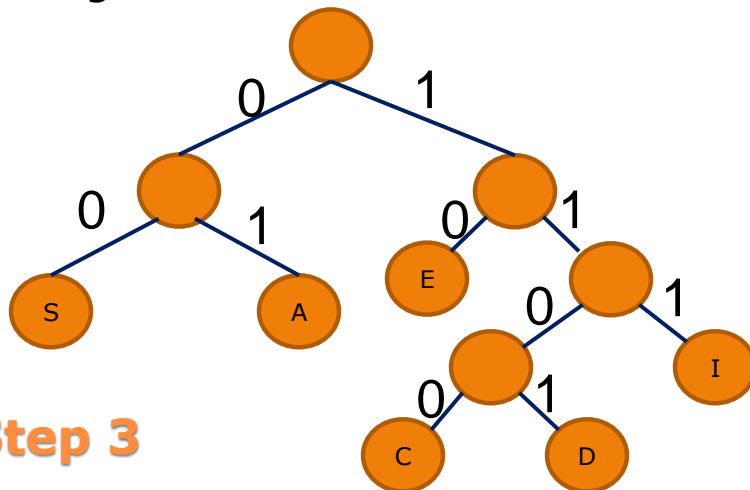








3. Label left edges with 0 and right edges with 1

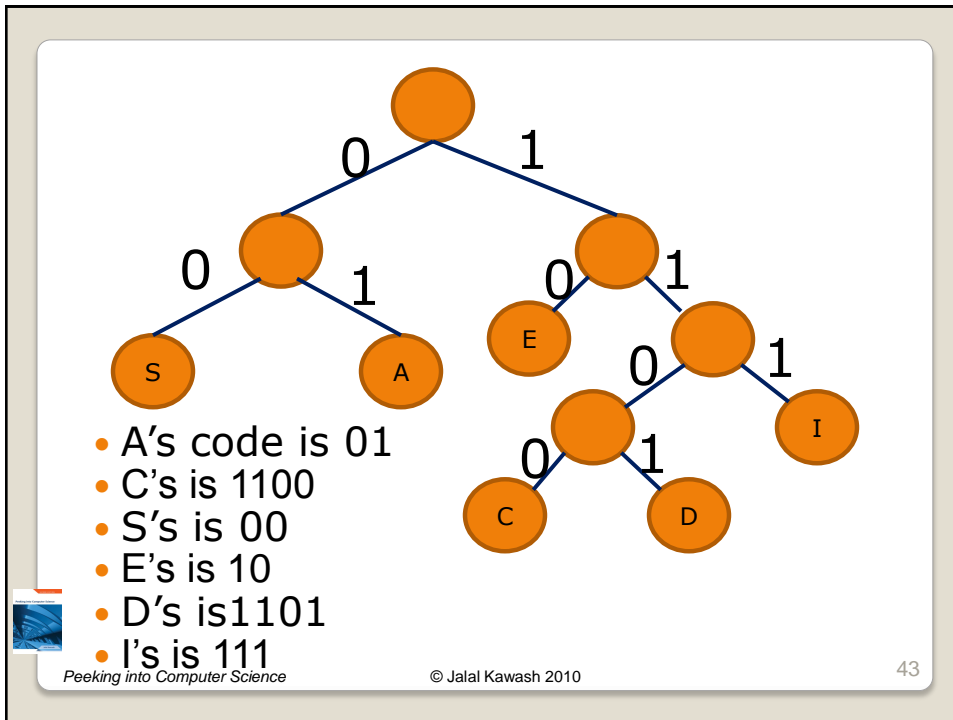


- Read the labels on the path from the root to each leaf, this is its code

Peeking into Computer Science © Jalal Kawash 2010 41

- A's code is 01

Peeking into Computer Science © Jalal Kawash 2010 42



- 35% are S
- 28% are A
- 20% are E
- 7% are I
- 6% are C
- 4% are D

More Frequent Characters have shorter codes

Peeking into Computer Science

© Jalal Kawash 2010

44

- If we use this coding, what is the size of the file?
 - 350 S's (35% of 1000) require **700** bits (2 bits for each S)
 - 280 A's require **560** bits
 - 200 E's require **400** bits
 - 70 I's require **210** bits
 - 60 C's require **240** bits
 - 40 D's require **160** bits
- Total is **2270** bits
 - Recall that with fixed codes, the size is **3000** bits
 - Compressed file size is about **76%** of original size



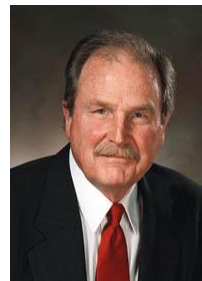
Recall This Analysis?

Peeking into Computer Science

© Jalal Kawash 2010

45

- 1925-1999
- US Electrical Engineer
- Contributions in coding theory, signal design for radar and communications, and logic circuits
- He wrote his coding algorithm as a graduate student at MIT



David Huffman

Peeking into Computer Science

© Jalal Kawash 2010

46

- Even though coding gave C and D codes of length 4 (compared to 3 in fixed coding), it was beneficial
- No code generated by Huffman's method can be a prefix of another code
- Many compression tools use a combination of different coding methods, Huffman's is among them



Concluding Notes