

CS126 Precept 6

Meeting 15: Data Compression Assignment

Randy Wang

Fixed Length Encoding Scheme

- ASCII encoding
 - 8 bits for representing each character
 - Can represent 256 characters

a 01100001	d 01100100
b 01100010	...
c 01100011	z 01111010

- What if one character shows up a lot more often?

Variable Length Encoding Scheme

character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100

- Use variable number of bits to represent each character
- More frequently used characters are represented by fewer characters

Decoding

character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100

Code book

0111110010110101001111100100

Message

- How to decode given an encoded message and a code book?

Decoding

character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100

Code book

0111110010110101001111100100

Message

|0|111|110|0|1011|0|1010|0|111|110|0|100
a b r a c a d a b r a !

Decoding

- Look for the prefix at the head of the encoded message that has a corresponding entry in the code table

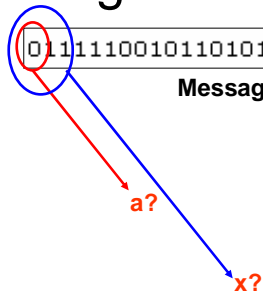
Preventing ambiguity during decoding

character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100
x	011

Code book

0111110010110101001111100100

Message

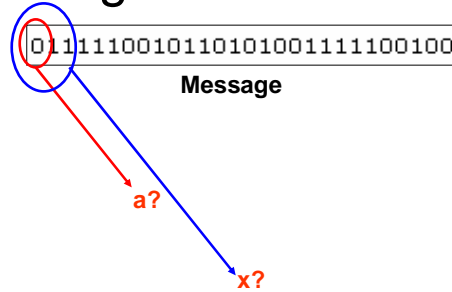


- Multiple possible ways of dealing with such ambiguity

Preventing ambiguity during decoding

character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100
x	011

Code book

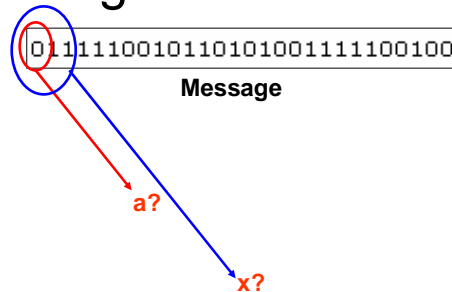


- Multiple possible ways of dealing with such ambiguity
- One way is to outlaw such ambiguities in the code book to begin with

Preventing ambiguity during decoding

character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100
x	011

Code book



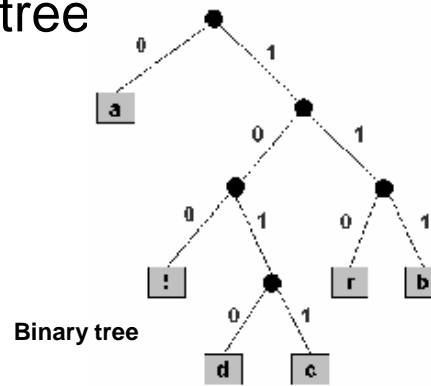
Forbid any entry to be a prefix of any other

- Multiple possible ways of dealing with such ambiguity
- One way is to outlaw such ambiguities in the code book to begin with

Representing the code book with a binary tree

character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100

Code book



- Your tasks:
 - Building a tree
 - Print the code table from the tree
 - Decode a message using the tree

Study ParseTree.java

- Show and trace ParseTree.java

Step 1: Write Constructor

- Very similar to `ParseTree` constructor
- Use `CharStdIn.java`

- Show and trace `charexample.java`

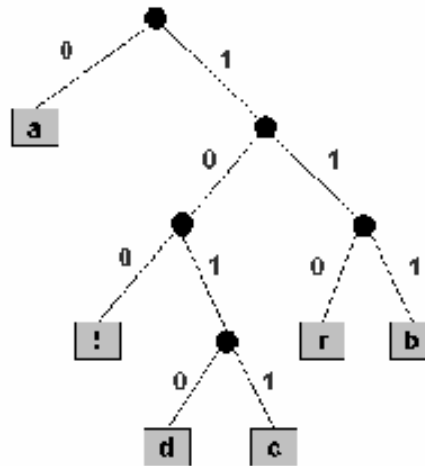
Step 2: write preorder traversal

- Very similar to `ParseTree.toString()`
- Test constructor and preorder traversal with simple `main()` program

- Run `PrefixTree1.java`

Step 3: modify preorder () to print code table

- Modify preorder () to add a string argument: preorder (String prefix)
- The prefix string argument accumulates partial answers seen at internal nodes---it grows longer with successive recursive calls



• Run PrefixTree2.java

Step 4: write uncompress ()

- No recursion necessary
- A single loop suffices: as long as there are more input 0 or 1 bits to read, read a bit and use it to traverse down the tree...

• uncompress.java