Trees and Codes

Make Money Fast!

Stock Fraud
Ponzi Scheme
Bank Robbery
Binary Trees and Prefix Codes

◆ Imagine that you have an alphabet of symbols
  ■ e.g. A, B, ..., Z, a, b, ..., z, ‘’, ‘’
◆ We wish to represent a string of these symbols as a string of bits
  ■ e.g. “This is a string of characters” becomes “01100111000101010111001”
Method 1: Use a fixed number for each symbol

- Map A -> 1, B -> 2, ...
- “This is a string of characters” becomes a string of numbers
  - “20,34,35,...”
- I required commas to separate the characters!
  - Use a fixed width, padded with leading 0’s instead
  - “020034035...”
Method 1: Fixed width

- How wide to my characters need to be
  - Using a string of bits
  - How many bits are needed to represent the largest character?
- $\text{ceil}(\log_2(n))$ bits
- Use that many bits for each character
- This is the system used within the computer with 8 bits for each ASCII code
Method II: Prefix codes

- For each symbol, we’ll use a code with a special property
  - No code is the prefix of any other code
- How does this work?
- Decoding:
  - We read in the codes one bit at a time
  - When we have a code we recognise, it must be the end of a symbol
    - It cannot be part of a longer symbol because no code is the prefix of another code
Method II: Prefix codes

Example

- A:00, B:010, C:011, D:10, E:11

- ADBECABADE

- 00100101101100010001011

- A D B E C A B A D E

- The string decodes
Making a Prefix Code

- We want the code to be efficient
  - No strings longer than necessary
  - No wasted strings

- A code is a set of strings of binary digits, such that no string corresponding to one symbol is the prefix of a string corresponding to another symbol

- In a tree, leaf nodes have no children
  - No path from the root to a leaf is the prefix of a path from the root to another node
Binary Trees and Prefix Codes

- Binary trees are in one to one correspondence with Prefix Codes
  - A:00, B:010, C:011, D:10, E:11
Prefix Trees

Binary Trees
- The left child corresponds to 0, the right to 1
- Each leaf contains a symbol
- The code for a symbol corresponds to the path from the root to the leaf containing that symbol
Encoding and Decoding

- Imagine the encoder and decoder running in parallel

**Encoding**
- Start from the root
- While you are not at the symbol’s leaf
  - If the symbol you wish to send is a left descendant, send 0 and move to your left child, else send 1 and move to your right child

**Decoding**
- Start from the root
- While you are not at a leaf
  - Read a bit. If it is 0 then move to your left child, else move to your right child
Encoding and Decoding: ACD

Encoding: ACD

Decoding:
Encoding and Decoding: ACD

Encoding: ACD

Decoding:

Trees
Encoding and Decoding: ACD

Encoding: ACD

Decoding:
Encoding and Decoding: ACD

Encoding: ACD

Decoding:

Trees
Encoding and Decoding: ACD

- **Encoding:** CD
- **Decoding:** A
Encoding and Decoding: ACD

- Encoding: CD
- Decoding: A
Encoding and Decoding: ACD

- Encoding: CD
- Decoding: A
Encoding and Decoding: ACD

- Encoding: CD
- Decoding: A
Encoding and Decoding: ACD

Encoding: CD

Decoding: A
Encoding and Decoding: ACD

- Encoding: CD
- Decoding: A
Encoding and Decoding: ACD

Encoding: CD

Decoding: A
Encoding and Decoding: ACD

Encoding: D

Decoding: AC

Trees
Encoding and Decoding: ACD

- **Encoding:** D
- **Decoding:** AC

Trees
Encoding and Decoding: ACD

- Encoding: D
- Decoding: AC

Trees
Encoding and Decoding: ACD

Encoding: D
Decoding: AC

Trees
Encoding and Decoding: ACD

Diamond Encoding: D

Diamond Decoding: AC

Trees
Encoding and Decoding: ACD

Encoding:

Decoding: ACD

Trees
Back to Method I: Balanced Tree

- Method I was to use fixed length code words.
- Each path from the root to a leaf is the same length: a balanced tree.
- Balanced trees are good for worst case path length. Are they good for coding?
  - Yes, if you assume the worst case.
  - But we can normally do better...
Statically optimal codes

- Want common symbols to have short codes
- This will make uncommon symbols have longer codes
  - In a tree with a fixed number of leaves/symbols, moving one leaf/symbol closer to the root will move others further away
Huffman codes

- From Shannon’s information theory, the optimal static code assigns $-\log_2(p)$ bits to a symbol that occurs with probability $p$.
- It is possible to make a Huffman code tree with this property.
  - Will look at this later in the course.
Adaptive Codes

As long as the same change is made in both sending and receiving trees/codes, there is no reason why the tree/code must remain static

- Send a character using the initial tree
- Update the tree using that character
  - Can also be updated in the receiver as it already has the character
- Send the next character
Encoding and Decoding: ACD

Encoding: ACD

Decoding:

Trees
Encoding and Decoding: ACD

Encoding: CD

Decoding: A

Trees
Encoding and Decoding: ACD

Encoding: CD

Decoding: A

Make same change in both trees:
Rotate A’s parent
Encoding and Decoding: ACD

Encoding: CD  
Decoding: A

Trees
Encoding and Decoding: ACD

Encoding: D

Decoding: CA

Trees
Encoding and Decoding: ACD

Encoding: D

Decoding: CA

Trees
Encoding and Decoding: ACD

_encoding:D

Decoding:CA
Encoding and Decoding: ACD

Encoding:

Decoding: CAD

Trees