Discussion 7

Ning Liu
The idea of Counter

• You may find it useful to put save some counters
  – Suppose you have 100Bytes + 3 bits in your compressed file, how do you know which bit is the end of the codes?
  – Stop decode when you reach the total count
How to implement HCTree::decode()

/**
  * Return symbol coded in the next sequence of bits from the stream.
  * PRECONDITION: build() has been called, to create the coding
  * tree, and initialize root pointer and leaves vector.
  */

int decode(BitInputStream& in) const;
int decode(BitInputStream& in) const{
    HCNode* curr= root; // start from the root of the tree
    int bit;
    while(curr is not the leaf) {
        bit = in.readBit(); // read one bit from the compressed file
        // if bit is 0, curr goes to its left child
        // if bit is 1, curr goes to its right child
    }
    return curr->symbol;
}
How to implement readBit()

• We talked about the implementation of writeBit()

• The two functions are very similar
Make your own test cases

• You need to make your own test cases
• First do compress, and then do uncompressed
• Compare with the original file using diff command
How many ways you can store your header?

• 1. Most naïve way:
Store all the 256 frequencies.
Easy to implement and construct the tree
But takes too much space.

Consider this situation: the file only contains lots of A
• 2. Store the frequencies of the chars that only appear in the file

• Butter than the first one, the header will become smaller

• You need to know the size of your header. Why?
• Assume the file is
  AAAAAAAAAABBBBBBCCCCCC

• What would your header look like?

• Is this the best one?
• 3. Can you think about one way to store the ‘structure’ of the tree?