Huffman Codes

An Optimal Data Compression Method

Data Compression

• Binary character code (“code”)
  – each k-bit source string maps to unique code word (e.g. k=8)
  – “compression” alg: concatenate code words for successive k-bit “characters” of source

• Fixed/variable length codes
  – all code words equal length?

• Prefix codes
  – no code word is prefix of another (simplifies decoding)
2

Strategy

1. Opt solution maybe not unique
   (so cannot prove greedy
gives only possible answer.)

2. Show greedy's solution is as good as any.

Huffman Code Objective

\[
\text{compressed length} = \sum c \cdot \text{length}(c)
\]
\[
\text{code} = \sum c \cdot \text{depth}(c) = \mathcal{B}(T)
\]
Defn: a pair of leaves \( x, y \) is an inversion if
- \( \text{depth}(x) = \text{depth}(y) \)
- \( \text{freq}(x) > \text{freq}(y) \)

Claim: if we flip on inversion, cost never increases.

Why?:
- All other things being equal, better to give more frequent letter shorter code.

Before:
- \( d(x)f(x) + d(y)f(y) \)

After:
- \( d(x)f(x) + d(y)f(y) - (d(x)f(y) + d(y)f(x)) \)
- \( (d(x) - d(y))(f(x) - f(y)) \) \( \geq 0 \)

i.e. cost doesn't go down.

Lemma 17.2
2. Least frequent chain might as well be siblings, at deepest level.

Proof:
- Let \( a \) be least freq. letter, \( b \) 2nd.
- Let \( U \) be least freq leaf at max depth, \( v \) its sibling.
- Then \( (a, u) \) & \( (b, v) \) are inversions. Sweep them.
Modified Lemma 19.3

Let \( C \) be an \( n \)-letter alphabet with frequencies \( f(c) \).

For \( c, y \in C \), let \( C' \) be \((n-1)\)-letter alphabet \( C' = \{x, y, z \} \) with

frag \( f'(c) = \{ f(c), y \} \) if \( c \neq z \\
= \{ f(z), x \} \) if \( c = z \)

Let \( T' \) be optimal tree for \( C' \). Then \( (T - z) + z \) is opt for \( C \) among trees having \( x, y \) as siblings.

Proof:

\[
B(T) = \sum_{c \in C} d(c) \cdot f(c)
\]

\[
B(T) - B(T') = d_T(x) f(x) + f(y)
- d_T(z) f(z)
\]

\[
= (d_T(x)+1) f(z)
- d_T(z) f(z)
\]

\[
= f(z) \text{ (having } x, y \text{ as siblings)}
\]

Suppose \( T' \) better than \( T \): \( B(T') < B(T) \)

Collapse \( x, y \) to \( z \) forming \( \bar{T}' \).

\[
B(\bar{T}') - B(\bar{T}) = f(z)
\]

Then

\[
B(\bar{T}) = B(\bar{T}') + f(z) < B(T) - f(z) = B(T)
\]

Contradiction.
Theorem 17.4
Huffman alg. gives optimal code.

Proof: by induction on # letters

basis: n = 1 trivial

ind: n ≥ 2

• let x, y be least frequent

• Form z, C’ as above

• By induction T’ is opt for C’

• By 17.3, T’ = T is opt for C among trees with x, y sibs

• By 17.2 some opt tree does

• T is optimal.

Data Compression

Huffman is optimal.

But still might do better!

1. Huffman encodes
   fixed length blocks.

2. Huffman uses one
   encoding throughout
   a file. What if characteristics change?

3. What if data has structure?
   Eg. raster images
   video

Lossless
   2iv-lempel, MPEG, ...