Huffman Codes

Imran Rashid

University of Washington

January 30, 2008

Lecture Outline

Compression Example

- 100k file, 6 letter alphabet:
- File Size:
  - ASCII, 8 bits/char: 800kbits
  - $2^3 > 6$; 3 bits/char: 300kbits

letter	freq
a	.45
b	.13
c	.12
d	.16
e	.09
f	.05

Compression Example

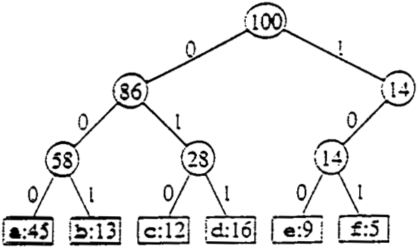
- 100k file, 6 letter alphabet:
- File Size:
    - ASCII, 8 bits/char: 800kbits
    - $2^3 > 6$; 3 bits/char: 300kbits

letter	freq	code
a	.45	00
b	.13	01
c	.12	10
d	.16	1100
e	.09	1101
f	.05	1110

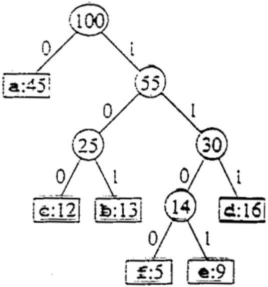
Data Compression

- 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 (unique decoding)

Prefix Codes = Trees



$$\underbrace{1\ 0\ 1}_{f}\ \underbrace{0\ 0\ 0}_{a}\ \underbrace{0\ 0\ 1}_{b}$$

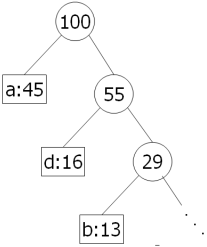


$$\underbrace{1\ 1\ 0\ 0}_{f}\ \underbrace{0}_{a}\ \underbrace{1\ 0\ 1}_{b}$$

Greedy Idea #1

- Put most frequent under root, then recurse
- Too greedy: unbalanced tree

letter	freq
a	.45
b	.13
c	.12
d	.16
e	.09
f	.05

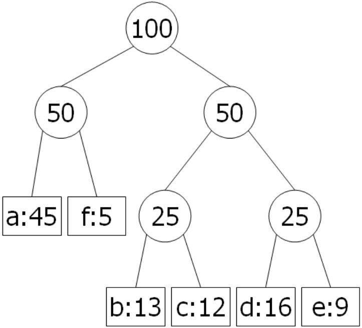


Greedy Idea #2

- Divide letters into 2 groups, with $\approx$ 50% weight in each;recurse(Shannon-Fano code)
- Again, not terrible:

$$2 * .5 + 3 * .5 = 2.5$$

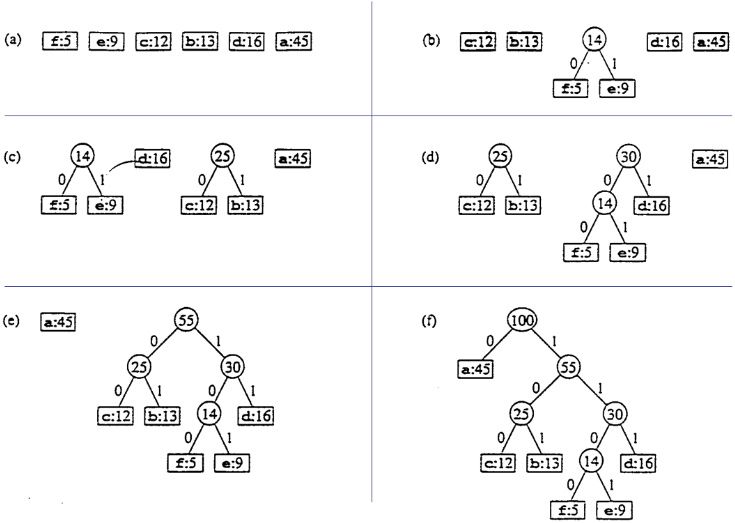
- But this tree can easily be improved! (How?)



Greedy idea #3

- Group least frequent letters near bottom

Huffman example



Huffman's Algorithm (1952)

Insert node for each letter into priority queue by freq
while queue length > 1 **do**
    Remove smallest 2 nodes, call them x, y
    Make new node z with children x, y.
    $f(z) = f(x) + f(y)$
    Insert z into queue
end while

- Analysis:
- Goal: Minimize $B(T) = \sum freq(c) * depth(c)$
- Correctness: ???

Correctness Strategy

- Optimal solution may not be unique, so cannot prove that greedy gives the only possible answer.
- Instead, show that greedy's solution is as good as any.

Inversions

- A pair of leaves x, y is in an inversion if
  $depth(x) \geq depth(y)$
  and
  $freq(x) \geq freq(y)$
- Claim: if we flip an inversion, cost never increases.

Lemma 1: "Greedy Choice Property"

- The 2 least frequent letters might as well be siblings at deepest level
  - Let a be least freq, b 2nd
  - Let u, v be siblings at max depth, $f(u) \leq f(v)$ (why must they exist?)
  - Then (a, u) and (b, v) are inversions. Swap them.

Lemma 2

- Let (C, f) be a problem instance: C an n-letter alphabet with letter frequencies $f(c)$ for $c \in C$.
- For any $x, y \in C$, let C' be the $(n-1)$ letter alphabet $C - \{x, y\} \cup \{z\}$ and for all $c \in C'$ define
- Let T' be an optimal tree for (C', f').
- Then create tree T by adding x, y as children of z in T'.
- T is optimal for (C, f) among all trees having x, y as siblings

Proof of Lemma 2

Theorem: Huffman gives optimal codes

Proof.

By Induction on $|C|$.

- Basis: $n = 1, 2$ – immediate
- Induction: $n > 2$
  - Let x, y be least frequent
  - Form $C', f', \& z$, as above
  - By induction, T' is opt for (C', f')
  - By lemma 2, T created from T' as above, is opt for (C, f) among trees with x, y as siblings
  - By lemma 1, some opt tree has x, y as siblings
  - Therefore, T is optimal.

$\square$

Data Compression

- Huffman is optimal.
- BUT still might do "better"
  - Huffman uses one encoding throughout a file. What if characteristics change?
  - What if data has structure? E.g. raster images, video,...
  - Huffman is lossless. Necessary?
- LZW, MPEG, ...