More more greed!

10/24/24

Recall: Greedy algorithms

- Use a simple rule to pick part of the solution, generally in a locally-best way
- Then, prune choices this makes impossible and repeat
- Greedy algorithms don't always work, but they do for some problems

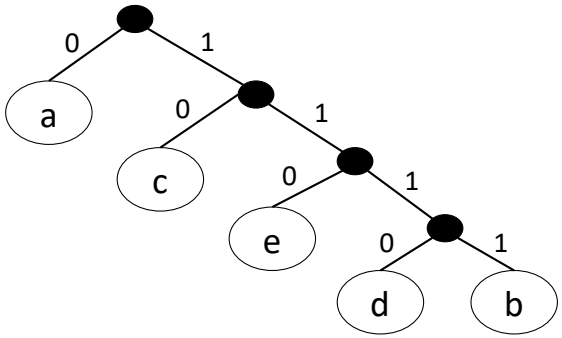
- Proving they work: Suppose the greedy rule doesn't allow an optimal solution. Take a solution that is optimal and change it to include the greedy choice. Show that this creates an optimal solution that (now) includes the greedy choice, contradicting the original assumption.

Recall: Huffman encoding

char	a	b	c	d	e
#times	47	11	20	5	17
encoding	0	1111	10	1110	110

- Length: 47 + 2*20 + 3*17 + 4*(11+5) = 202

Each term: ~~length of encoding~~ * #times
depth in tree

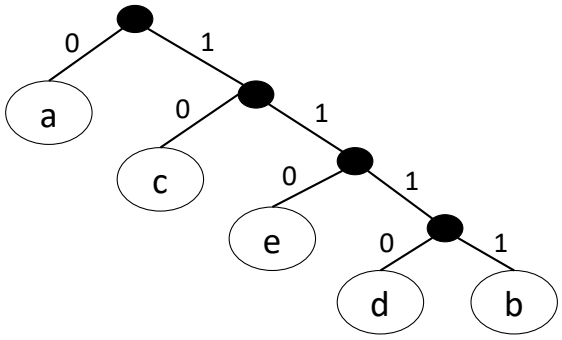


Huffman encoding

char	a	b	c	d	e
#times	47	11	20	5	17
encoding	0	1111	10	1110	110

- Length: 47 + 2*20 + 3*17 + 4*(11+5) = 202

Each term: ~~length of encoding~~ * #times
depth in tree



Start with a node for each char
While there is more than one node,
Combine 2 nodes w/ lowest frequencies
//Combined node has sum of frequencies

Create a Huffman encoding for characters a-e with the frequency distribution below

letter	a	b	c	e	d	f
frequency	18	6	31	12	10	23

Picking letter depths

Give an input for the Huffman coding algorithm that will cause it to have one letter at depth 1, three at depth 3, and 2 at depth 4. (Recall that the depth of a node is the number of edges that must be traversed from the tree's root to reach it.) Include the instance (the letters and their frequencies) and describe how the algorithm constructs the Huffman encoding tree for this input.

Are you here?

A. Yes

B. Affirmative

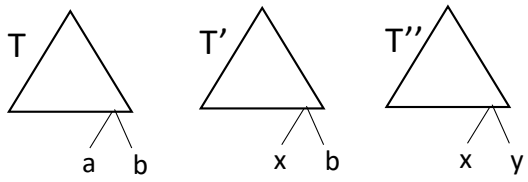
C. Absolutely

D. Definitely

E. More than one of the above

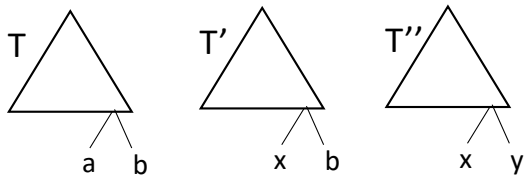
Key part of the argument

Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit



Key part of the argument

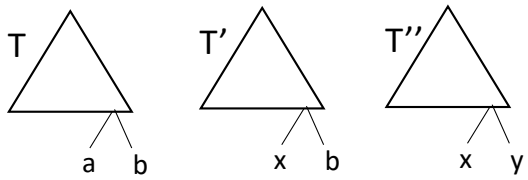
Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit



Proof: WLOG, assume x.freq $\leqslant$ y.freq
Let T be the tree of an optimal prefix code.
Let a and b be sibling nodes at the lowest level with a.freq $\leqslant$ b.freq.

Key part of the argument

Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit

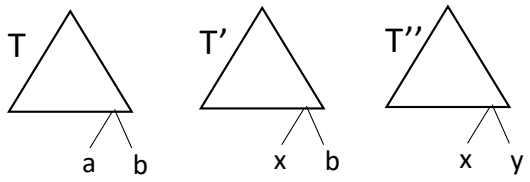


Proof: WLOG, assume x.freq $\leqslant$ y.freq
Let T be the tree of an optimal prefix code.
Let a and b be sibling nodes at the lowest level with a.freq $\leqslant$ b.freq.
Case 1: x.freq = b.freq

Then a.freq = b.freq = x.freq = y.freq and can freely substitute.

Key part of the argument

Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit



Proof: WLOG, assume x.freq $\leqslant$ y.freq
Let T be the tree of an optimal prefix code.
Let a and b be sibling nodes at the lowest level with a.freq $\leqslant$ b.freq.
Case 2: x.freq != b.freq (and hence x != b)

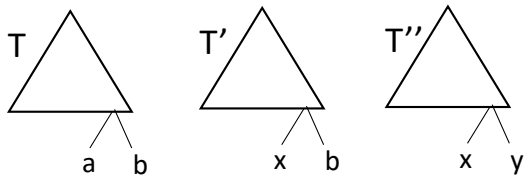
Let T' be T, but swapping a and x. The difference in the encoded length is

$$x.freq * d_T(x) + a.freq * d_T(a) - x.freq * d_{T'}(x) - a.freq * d_{T'}(a)$$

Depth of a in T'

Key part of the argument

Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit



Proof: WLOG, assume x.freq $\leqslant$ y.freq

Let T be the tree of an optimal prefix code.

Let a and b be sibling nodes at the lowest level with a.freq $\leqslant$ b.freq.

Case 2: x.freq != b.freq (and hence x != b)

Let T' be T, but swapping a and x. The difference in the encoded length is

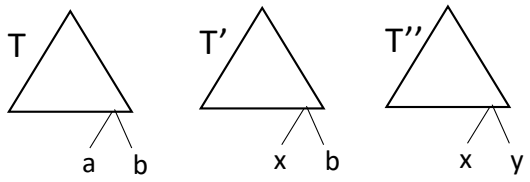
$$x.freq * d_T(x) + a.freq * d_T(a) - x.freq * d_{T'}(x) - a.freq * d_{T'}(a)$$

$$= x.freq * d_T(x) + a.freq * d_T(a) - x.freq * d_T(a) - a.freq * d_T(x)$$

Depth of a in T'

Key part of the argument

Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit



Proof: WLOG, assume x.freq $\leqslant$ y.freq

Let T be the tree of an optimal prefix code.

Let a and b be sibling nodes at the lowest level with a.freq $\leqslant$ b.freq.

Case 2: x.freq != b.freq (and hence x != b)

Let T' be T, but swapping a and x. The difference in the encoded length is

Depth of a in T'

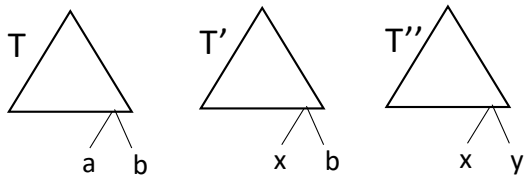
$$x.freq * d_T(x) + a.freq * d_T(a) - x.freq * d_{T'}(x) - a.freq * d_{T'}(a)$$

$$= x.freq * d_T(x) + a.freq * d_T(a) - x.freq * d_T(a) - a.freq * d_T(x)$$

$$= (a.freq - x.freq)(d_T(a) - d_T(x))$$

Key part of the argument

Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit



Proof: WLOG, assume x.freq $\leqslant$ y.freq

Let T be the tree of an optimal prefix code.

Let a and b be sibling nodes at the lowest level with a.freq $\leqslant$ b.freq.

Case 2: x.freq != b.freq (and hence x != b)

Let T' be T, but swapping a and x. The difference in the encoded length is

Depth of a in T'

$$x.freq * d_T(x) + a.freq * d_T(a) - x.freq * d_{T'}(x) - a.freq * d_{T'}(a)$$

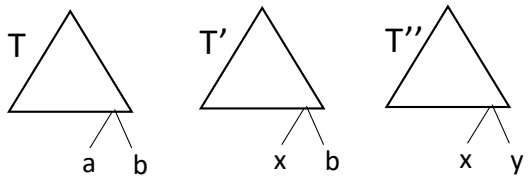
$$= x.freq * d_T(x) + a.freq * d_T(a) - x.freq * d_T(a) - a.freq * d_T(x)$$

$$= (a.freq - x.freq)(d_T(a) - d_T(x))$$

$$\geq 0$$

Key part of the argument

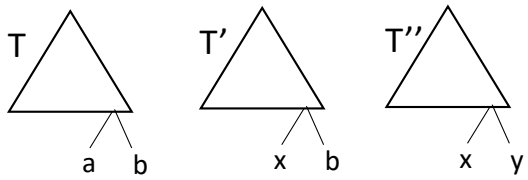
Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit



Case 2 continued:
Let T'' be T', but swapping b and y. The difference in the encoded length is

Key part of the argument

Lemma: Let x and y be two characters having the lowest frequencies. Then there exists an optimal prefix code in which the codewords for x and y have the same length and differ only in the last bit



Case 2 continued:

Let T'' be T', but swapping b and y. The difference in the encoded length is

$$\begin{aligned} & y.freq * d_{T'}(y) + b.freq * d_{T'}(b) - y.freq * d_{T''}(y) - b.freq * d_{T''}(b) \\ = \; & y.freq * d_{T'}(y) + b.freq * d_{T'}(b) - y.freq * d_{T'}(b) - b.freq * d_{T'}(y) \\ = \; & (b.freq - y.freq)(d_{T'}(b) - d_{T'}(y)) \\ \geq \; & 0 \end{aligned}$$

More scheduling

In another version of scheduling with deadlines, job Ji has a release time r_i before which it cannot be run, a duration p_i which is how long it must run, and a deadline d_i before which it must finish. The system is allowed to switch between jobs in the middle (called preemption); at each time step, it must select one unfinished job to work on. Job J_i is finished after it has been selected p_i times.

In this setting, the Earliest Deadline First (EDF) algorithm works on the job with the earliest deadline among the jobs that have been released and not yet finished. Prove that this version of EDF will complete every job by its deadline if this is possible. (You want to assume that there is a schedule that completes every job by its deadline and follows the EDF schedule as long as possible. Then look at the first time this schedule chooses a different job than EDF and construct a schedule that follows the EDF algorithm for another time step and still completes all jobs by their deadline.)