CMPT307: Greedy Algorithms

Week 8-2

Xian Qiu

Simon Fraser University

xianq@sfu.ca

Encoding Characters

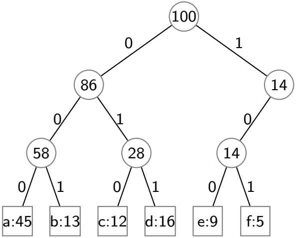
to encode 100 characters: a-f only

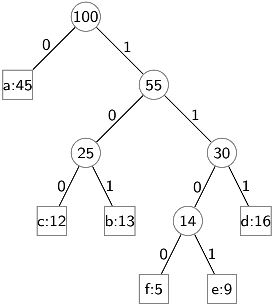
using binary

	a	b	c	d	e	f
frequency	45	13	12	16	9	5
fixed length codeword	000	001	010	011	100	101
variable length codeword	0	101	100	111	1101	1100

- fixed length: 300 bits
- variable length: 224 bits
- consider prefix code: no codeword is a prefix of another

Decoding





characters are leaves

$110001001101 = ?$

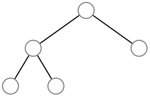
$f_c =$ frequency, $d_T(c) =$ depth, $\forall c \in C =$ alphabet

total bits: $B(T) = \sum_{c \in C} f_c \cdot d_T(c)$

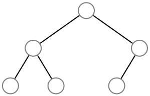
- input: alphabet C, with frequency $f_c > 0$ for all $c \in C$
- output: prefix codes, minimizing the total bits
  equivalently, represent by a binary tree T minimizing $B(T)$

Definition

A binary tree is full if each internal node has exactly two children.



full



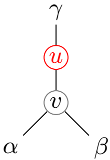
complete

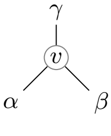
- denote by T^* the optimal binary tree

Lemma 1. T^* is full.

Proof

assume T^* is not full





T' is a feasible encoding and has less bits than T^*

Properties of OPT

Lemma 2

T^* has $|C|$ leaves and $|C|-1$ internal nodes.

- removing all leaves of T^* yields a new tree T_1^* with $\frac{|C|}{2}$ leaves
- removing all leaves of T_1^* yields T_2^* with $\frac{|C|}{2^2}$ leaves
- continue this way until T_h^* with $\frac{|C|}{2^h} = 1$ leave (singleton)
- total number of internal nodes is

$$\frac{|C|}{2} + \frac{|C|}{2^2} + \cdots + \frac{|C|}{2^h} = |C| \frac{\frac{1}{2}\left(1 - \frac{1}{2^h}\right)}{1 - \frac{1}{2}}$$
$$= |C| \left(1 - \frac{1}{|C|}\right) = |C| - 1$$

Problem Reformulation

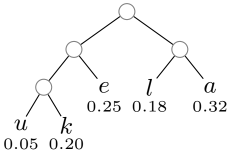
- given n nodes $c_1, \dots, c_n$ and their frequencies $f_{c_1}, \dots, f_{c_n}$
- construct a full binary tree T^* of $2n-1$ nodes s.t.
  1. $c_1, \dots, c_n$ are all leaves of T^*
  2. $B(T^*) = \sum_{i=1}^{n} f_{c_i} \cdot d_{T^*}(c_i)$ is minimized

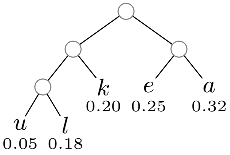
greedy strategies

- since T^* is full, consider two nodes at a time
- top-down: letters with high frequency should be near the top
- bottom-up: lowest frequency letters should be at the lowest level

algorithm: divide C into sets C_1 and C_2 with (almost) equal frequencies, and recursively build tree for C_1 and C_2

Example. $f_a = 0.32, f_e = 0.25, f_k = 0.20, f_l = 0.18, f_u = 0.05.$





the algorithm does not yield optimal solution!

observations

- for $n > 1$, the lowest level always contains at least two leaves
- the order in which items appear in a level does not matter

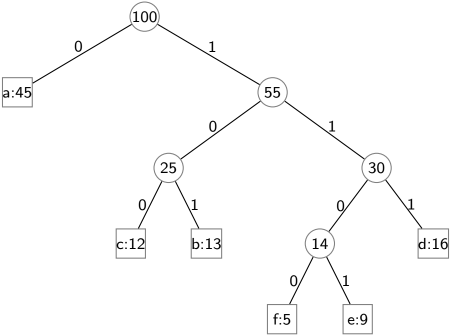
Lemma

Assume $x, y \in C$ have lowest frequencies, then x and y are siblings of T^*.

- note that x and y are leaves

algorithm: make two leaves for x and y, recursively build tree for the rest using a meta-letter "xy"

Example



how to find the min-frequency node?

Pseudocode

c.f $=$ frequency of $c \in C$

Q is min-priority queue keyed by c.f for all $c \in C$

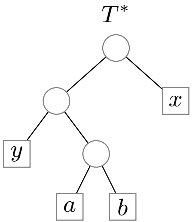
$\textsc{Huffman}(C)$

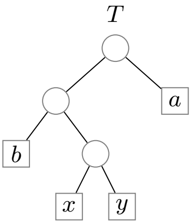
$n = |C|$;
$Q = C$;
for $i = 1$ *to* $n - 1$ **do**
  allocate a new node z;
  z.left $= x = \textsc{Extract-Min}(Q)$;
  z.right $= y = \textsc{Extract-Min}(Q)$;
  z.freq $= x$.f $+ y$.f;
  $\textsc{Insert}(Q, z)$;
return $\textsc{Extract-Min}(Q)$; `// return the root of the tree`

Correctness

Lemma

Assume $x, y \in C$ have lowest frequencies, then x and y are siblings of T^*.





- a, b are siblings of maximum depth
- assume $a \neq x$ and $y \neq b$ (what if not?)
- to show $B(T) \leq B(T^*)$

$$
\begin{aligned}
B(T) - B(T^*) &= \sum_{c \in C} c.\mathsf{f} \cdot d_T(c) - \sum_{c \in C} c.\mathsf{f} \cdot d_{T^*}(c) \\
&= x.\mathsf{f} \cdot d_T(x) + a.\mathsf{f} \cdot d_T(a) - x.\mathsf{f} \cdot d_{T^*}(x) - a.\mathsf{f} \cdot d_{T^*}(a) \\
&\qquad + y.\mathsf{f} \cdot d_T(y) + b.\mathsf{f} \cdot d_T(b) - y.\mathsf{f} \cdot d_{T^*}(y) - b.\mathsf{f} \cdot d_{T^*}(b) \\
&= x.\mathsf{f} \cdot d_{T^*}(a) + a.\mathsf{f} \cdot d_{T^*}(x) - x.\mathsf{f} \cdot d_{T^*}(x) - a.\mathsf{f} \cdot d_{T^*}(a) \\
&\qquad + y.\mathsf{f} \cdot d_{T^*}(b) + b.\mathsf{f} \cdot d_{T^*}(y) - y.\mathsf{f} \cdot d_{T^*}(y) - b.\mathsf{f} \cdot d_{T^*}(b) \\
&= (x.\mathsf{f} - a.\mathsf{f})(d_{T^*}(a) - d_{T^*}(x)) + (y.\mathsf{f} - b.\mathsf{f})(d_{T^*}(b) - d_{T^*}(y)) \\
&\leq 0
\end{aligned}
$$

next we show $\textsc{Hoffman}(C)$ is correct

- merge x and y into a new character z
- let $C_1 = C - x - y + z$ and T_1 be optimal for C_1
- to show: T obtained from T_1 by splitting z is optimal for C
- by definition, $B(T) = B(T_1) + x.\mathsf{f} + y.\mathsf{f}$

assume $B(T') < B(T)$ $\qquad$ $B(T') = B(T'') + x.\mathsf{f} + y.\mathsf{f}$

- $B(T'') < B(T) - x.\mathsf{f} - y.\mathsf{f} = B(T_1)$, a contradiction