Programming and Data Structures - II (CS3201)

Lecture 3

Kripabandhu Ghosh

CDS, IISER Kolkata

HUFFMAN CODING

Storage of a character file

	a	b	c	d	e	f
Frequency in ‘000s	45	13	12	16	9	5

Storage of a character file: coding schemes

	a	b	c	d	e	f
Frequency in '000s	45	13	12	16	9	5
Fixed length	000	001	010	011	100	101
Variable length	0	101	100	111	1101	1100

- **Fixed length code**: For 100,000 characters, 3 x 100,000 = 300,000 bits are needed
- **Variable length code**: (45x1 + 13x3 + 12x3 + 16x3 + 9x4 + 5x4)x1000 = 224,000 bits are needed ($\approx$ 25% savings!!)

Encoding

Definition

In a character-coding scheme, given a code (with an alphabet Γ) and a *codeword* and a message, produce the *encoded* message

Alphabet Γ

$\Gamma = \{a, b, c, d\}$

Code

Character	a	b	c	d
Codeword C_1	00	01	10	11
Codeword C_2	1	110	10	111

Encoding

- Input message: bad
- Encoded message using C_1: 010011
- Encoded message using C_2: 1101111

Decoding

Definition

In a character-coding scheme, given a code (with an alphabet Γ) and a *codeword* and an encoded message, produce the *original* message

Code

Character	a	b	c	d
Codeword C_1	00	01	10	11
Codeword C_2	1	110	10	111

Decoding using C_1

- Encoded message: 010011 $\Rightarrow$ 010011
- Decoded message : bad

Decoding (contd.)

Code

Character	a	b	c	d
Codeword C_1	00	01	10	11
Codeword C_2	1	110	10	111

Decoding using C_2

- Encoded message: 1101111
- 1. Interpretation 1: 1101111, decoded message: bad
  2. Interpretation 2: 1101111, decoded message: acda
  3. Interpretation 3: 1101111, decoded message: acad

Decoding (contd.)

Code

Character	a	b	c	d
Codeword C_1	00	01	10	11
Codeword C_2	1	110	10	111

Decoding using C_2

- Encoded message: 1101111
- 1. Interpretation 1: 1101111, decoded message: bad
  2. Interpretation 2: 1101111, decoded message: acda
  3. Interpretation 3: 1101111, decoded message: acad

C_2 is NOT uniquely decodable !!!

Decoding (contd.)

Code

Character	a	b	c	d
Codeword C_1	00	01	10	11
Codeword C_2	1	110	10	111

Decoding using C_2

- Encoded message: 1101111
- 1. Interpretation 1: 1101111, decoded message: bad
  2. Interpretation 2: 1101111, decoded message: acda
  3. Interpretation 3: 1101111, decoded message: acad

C_2 is NOT uniquely decodable !!!

C_1 is uniquely decodable

Prefix codes

Definition

A code where no codeword is a prefix of another

Prefix codes

Definition

A code where no codeword is a prefix of another

Misnomer!!

Prefix-free code (instead of *prefix code*) would have been a better nomenclature, but the latter is standard in literature

Prefix codes

Definition

A code where no codeword is a prefix of another

Misnomer!!

Prefix-free code (instead of *prefix code*) would have been a better nomenclature, but the latter is standard in literature

Example

$\{a = 0, b = 110, c = 10, d = 111\}$ is a prefix code

Advantages

- Uniquely decodable/decipherable
- Achieves optimal data compression

Cost of a coding

- **C**: set of characters in the input message
- **c**: c $\in$ C is a character
- **c.freq**: frequency of character c in the input message
- **T**: Tree corresponding to the optimal prefix code (full binary tree: every non-leaf node has two children)
- $d_T(c)$: The depth of c's leaf in T

Number of bits required to encode a message with characters in C

B(T) $= \sum_{c \in C} c.freq \; d_T(c)$

Optimal coding problem

Find a (binary) prefix tree T (equivalently an optimal prefix code) such that B(T) is minimized

Huffman coding

Developed by David A. Huffman

Intuition

- Tree T is build (Min-Heap) from C in a bottom-up manner
- Min-priority-queue is used to identify and merge the two least frequency nodes
- The merged node has frequency as the sum of the two nodes merged
- The final tree T is is the optimal prefix code
- The codeword is the sequence of edge labels (0 for the left, 1 for the right subtree) on a simple path from the root to the node

Characteristics

Greedy method $\Rightarrow$ finds locally optimal solutions (which looks best at the moment) in the hope it will achieve globally optimal solution

Huffman coding

Algorithm

1. $n = |C|$
2. $Q = C$
3. for i = 1 to n - 1
4. allocate a new node z
5. $z.left = x =$ EXTRACT-MIN(Q)
6. $z.right = y =$ EXTRACT-MIN(Q)
7. $z.freq = x.freq + y.freq$
8. INSERT(Q, z)

Time Complexity

- Step 2 invokes BUILD-MIN-HEAP which takes $\mathcal{O}(nlogn)$
- In the for loop EXTRACT-MIN ($\mathcal{O}(logn)$) is called n-1 times $\Rightarrow \mathcal{O}(nlogn)$
- INSERT takes $\mathcal{O}(logn)$

Huffman coding

Algorithm

1. $n = |C|$
2. $Q = C$
3. for $i = 1$ to n - 1
4.   allocate a new node z
5. $z.left = x =$ EXTRACT-MIN(Q)
6. $z.right = y =$ EXTRACT-MIN(Q)
7. $z.freq = x.freq + y.freq$
8.   INSERT(Q, z)

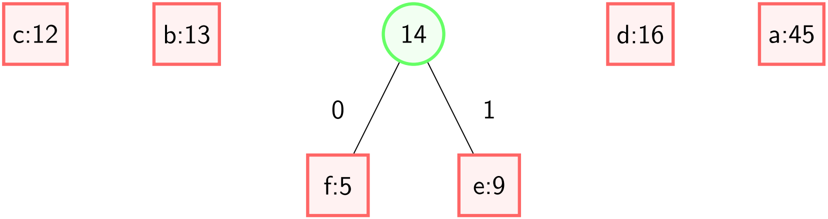
Time Complexity

- Step 2 invokes BUILD-MIN-HEAP which takes $\mathcal{O}(nlogn)$
- In the for loop EXTRACT-MIN ($\mathcal{O}(logn)$) is called n-1 times $\Rightarrow \mathcal{O}(nlogn)$
- INSERT takes $\mathcal{O}(logn)$
- **Huffman coding on a character C set of n elements has a running time of $\mathcal{O}(nlogn)$**

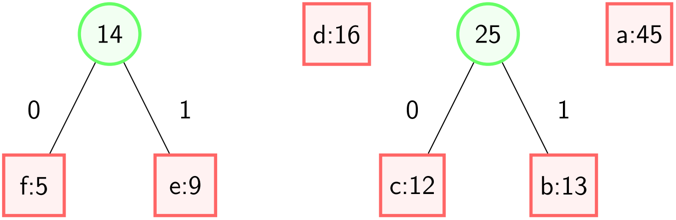
Huffman coding: example

f:5 e:9 c:12 b:13 d:16 a:45

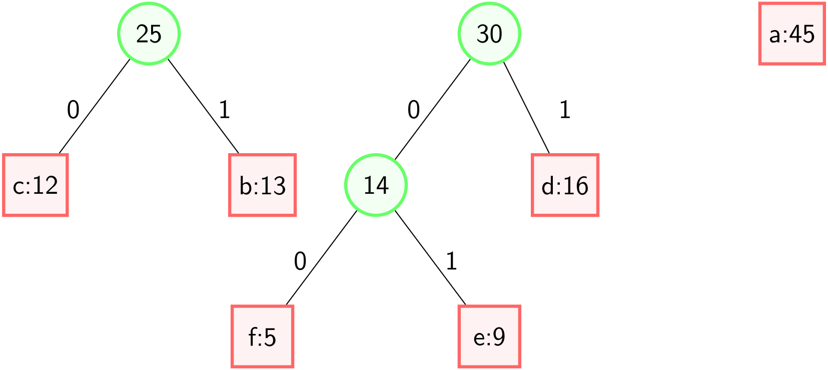
Huffman coding: example (contd.)



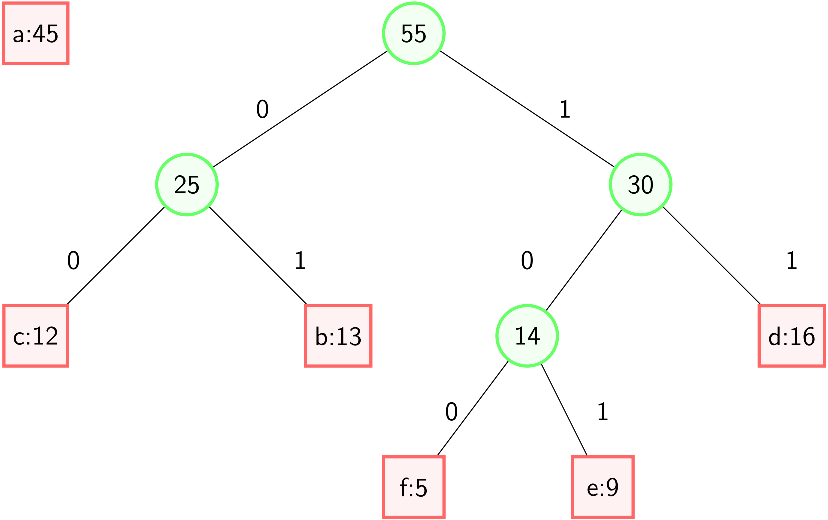
Huffman coding: example (contd.)



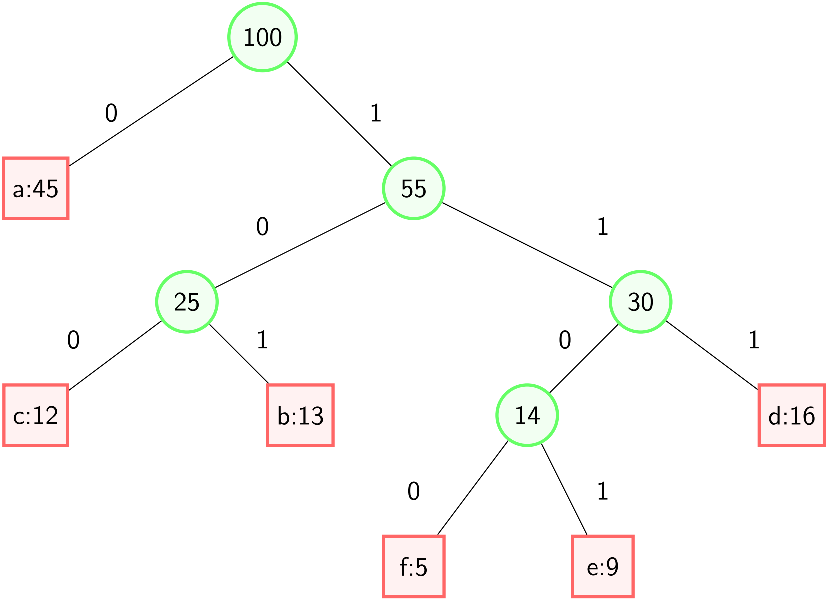
Huffman coding: example (contd.)



Huffman coding: example (contd.)



Huffman coding: example (end)



Huffman coding: codeword generation

	a	b	c	d	e	f
Frequency in ‘000s	45	13	12	16	9	5
Huffman coding	0	101	100	111	1101	1100

Huffman coding: proof of correctness

Lemma 1

- Let C be an alphabet and x and y be the two characters in C with the *lowest* frequencies
- **To prove**: *There exists an optimal prefix code for C in which the codewords for x and y have the same length that differ only in the last bit*

Proof idea

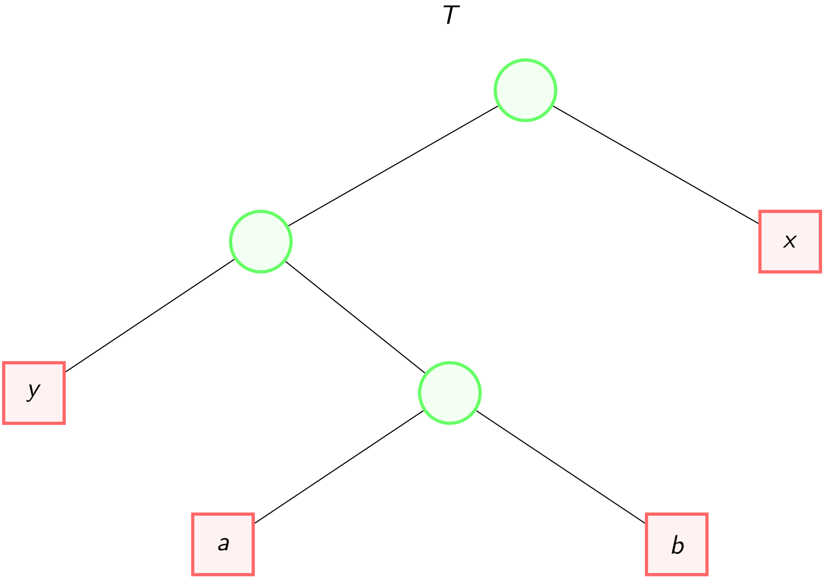
- Take a tree T representing an arbitrary optimal prefix code
- Modify T into another tree representing another optimal prefix code such that x and y appear as sibling leaves of maximum depth in the new tree

Huffman coding: proof of correctness (Lemma 1)

Proof

- Let a and b be two characters that are sibling leaves of the maximum depth in T
- We assume $a.freq \leq b.freq$ and $x.freq \leq y.freq$
- Since $x.freq$ and $y.freq$ are the lowest, we have $x.freq \leq a.freq$ and $y.freq \leq b.freq$

Huffman coding: proof of correctness (Lemma 1): tree T

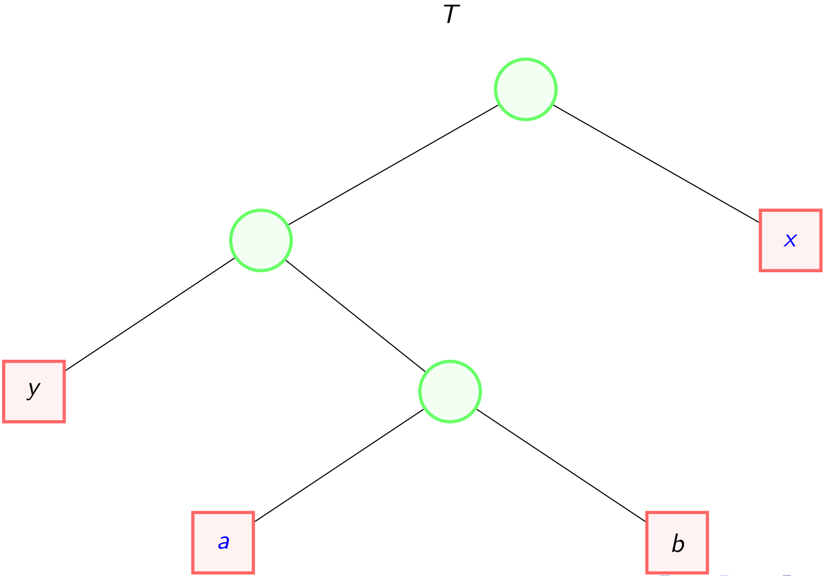


Huffman coding: proof of correctness (Lemma 1)

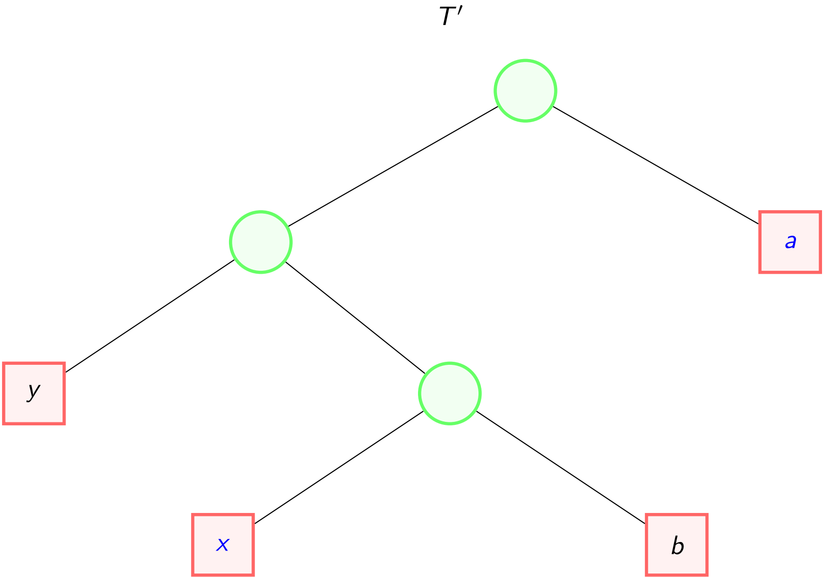
Proof

- Let a and b be two characters that are sibling leaves of the maximum depth in T
- We assume $a.freq \leq b.freq$ and $x.freq \leq y.freq$
- Since $x.freq$ and $y.freq$ are the lowest, we have $x.freq \leq a.freq$ and $y.freq \leq b.freq$
- We exchange the positions of a and x in T to produce a tree T'

Huffman coding: proof of correctness (Lemma 1): tree T (before exchange)



Huffman coding: proof of correctness (Lemma 1): tree T′



Huffman coding: proof of correctness (Lemma 1)

$B(T)$ - $B(T') = \sum_{c \in C} c.freq\ d_T(c)$ - $\sum_{c \in C} c.freq\ d_{T'}(c)$

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

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

$= (a.freq$ - $x.freq)\ (d_T(a)$ - $d_T(x)) \geq 0$

$\Rightarrow B(T) \geq B(T')$

(since $x.freq \leq a.freq$ and a is a leaf of the maximum depth in T)

Since T is optimal we have $B(T) \leq B(T')$

Huffman coding: proof of correctness (Lemma 1)

$B(T)$ - $B(T') = \sum_{c \in C} c.freq\ d_T(c)$ - $\sum_{c \in C} c.freq\ d_{T'}(c)$

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

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

$= (a.freq$ - $x.freq)\ (d_T(a)$ - $d_T(x)) \geq 0$

$\Rightarrow B(T) \geq B(T')$

(since $x.freq \leq a.freq$ and a is a leaf of the maximum depth in T)

Since T is optimal we have $B(T) \leq B(T')$

Therefore, $B(T) = B(T')$ (that is, T' is also optimal)

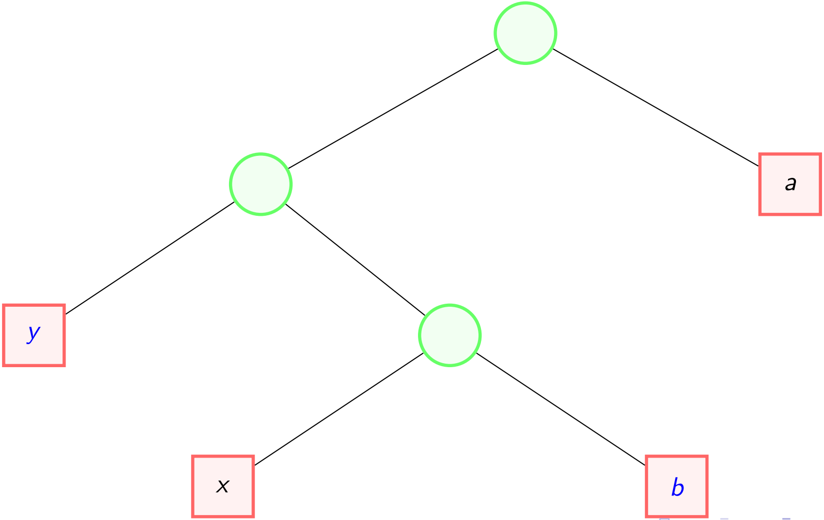
Huffman coding: proof of correctness (Lemma 1)

Proof

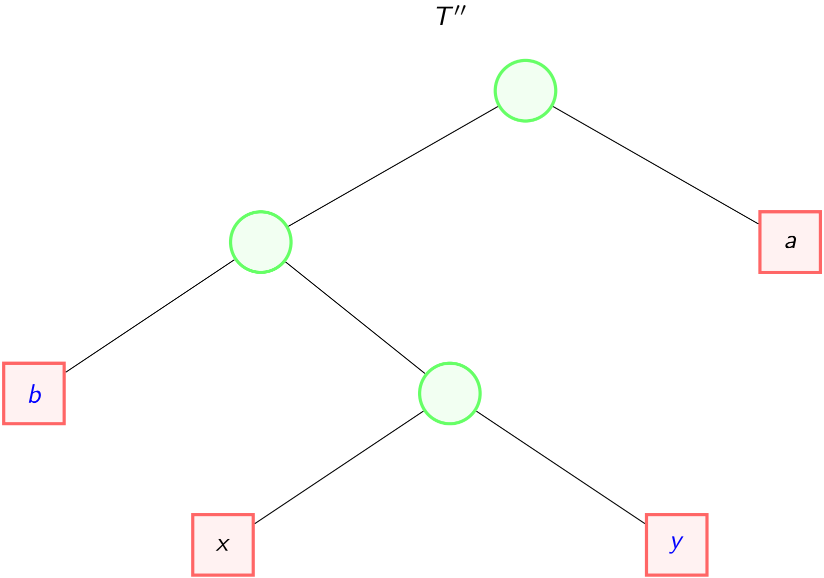
- Let a and b be two characters that are sibling leaves of the maximum depth in T
- We assume $a.freq \leq b.freq$ and $x.freq \leq y.freq$
- Since $x.freq$ and $y.freq$ are the lowest, we have $x.freq \leq a.freq$ and $y.freq \leq b.freq$
- We exchange the positions of a and x in T to produce a tree T'
- We exchange the positions of b and y in T' to produce a tree T''

Huffman coding: proof of correctness (Lemma 1): tree T′ (before exchange)

T'



Huffman coding: proof of correctness (Lemma 1): tree T″



Huffman coding: proof of correctness (Lemma 1) (proof complete)

$B(T') - B(T'') = \sum_{c \in C} c.freq\ d_{T'}(c) - \sum_{c \in C} c.freq\ d_{T''}(c)$

$= b.freq\ d_{T'}(b) + y.freq\ d_{T'}(y) - b.freq\ d_{T''}(b) - y.freq\ d_{T''}(y)$

$= b.freq\ d_T(b) + y.freq\ d_T(y) - b.freq\ d_T(y) - y.freq\ d_T(b)$

$= (b.freq - y.freq)\ (d_T(b) - d_T(y)) \geq 0$

$\Rightarrow B(T') \geq B(T'')$

(since $y.freq \leq b.freq$ and b is leaf of the maximum depth in T)

Since T' is optimal we have $B(T') \leq B(T'')$

Huffman coding: proof of correctness (Lemma 1) (proof complete)

$B(T')$ - $B(T'') = \sum_{c \in C}$ $c.freq$ $d_{T'}(c)$ - $\sum_{c \in C}$ $c.freq$ $d_{T''}(c)$

$= b.freq\ d_{T'}(b) + y.freq\ d_{T'}(y)$ - $b.freq\ d_{T''}(b)$ - $y.freq\ d_{T''}(y)$

$= b.freq\ d_T(b) + y.freq\ d_T(y)$ - $b.freq\ d_T(y)$ - $y.freq\ d_T(b)$

$= (b.freq$ - $y.freq)$ $(d_T(b)$ - $d_T(y)) \geq 0$

$\Rightarrow B(T') \geq B(T'')$

(since $y.freq \leq b.freq$ and b is leaf of the maximum depth in T)

Since T' is optimal we have $B(T') \leq B(T'')$

Therefore, $B(T') = B(T'') = B(T)$ (that is, T'' is also optimal)

Huffman coding: proof of correctness (Lemma 2)

Lemma 2

- Let C be an alphabet and x and y be the two characters in C with the *lowest* frequencies
- Let C' be the alphabet C with the characters x and y removed and a new character z added, so that $C' = C$ - $\{x, y\} \cup \{z\}$
- Define
$$z.freq = x.freq + y.freq \tag{1}$$
- Let T' be a tree representing the optimal prefix code for C'
- **To prove**: *The tree T, obtained from T' by replacing the leaf node for z with an internal node having x and y as children, represents an optimal prefix code for C*

Huffman coding: proof of correctness (Lemma 2) (proof)

Proof

- For each character $c \in C$ - $\{x, y\}$, we have $d_T(c) = d_{T'}(c)$
  Therefore, $c.freq\ d_T(c) = c.freq\ d_{T'}(c)$
- Also,
$$d_T(x) = d_T(y) = d_{T'}(z) + 1 \tag{2}$$
- Therefore, $x.freq\ d_T(x) + y.freq\ d_T(y)$
  $= (x.freq + y.freq)\ (d_{T'}(z) + 1)$ (Using equation (2))
  $= (x.freq + y.freq)\ d_{T'}(z) + (x.freq + y.freq)$
  $= z.freq\ d_{T'}(z) + (x.freq + y.freq)$ (Using equation (1))
- We conclude $B(T) = B(T') + x.freq + y.freq$, equivalently
$$B(T') = B(T) - x.freq - y.freq \tag{3}$$

Huffman coding: proof of correctness (Lemma 2) (proof complete)

Proof (contd.)

- We prove by contradiction.
- **Suppose T does not represent an optimal prefix code for C**
- Then there exists an optimal tree T'' such that

$$B(T'') < B(T) \tag{4}$$

- Without any loss of generality, by lemma 1, T'' has x and y as siblings
- Let T''' be the tree T'' with the common parent of x and y replaced by a leaf z with frequency $z.freq = x.freq + y.freq$
- Then $B(T''') = B(T'')$ - $x.freq$ - $y.freq$ (Using equation (3))
  $< B(T)$ - $x.freq$ - $y.freq$ (Using equation (4))
  $= B(T')$ (Using equation (3))
  contradiction, since T' is assumed to represent an optimal prefix code for C'

T must represent an optimal prefix code for C

Huffman coding: proof of correctness (proof complete)

Lemma 1

- Let C be an alphabet and x and y be the two characters in C with the *lowest* frequencies
- **To prove**: *There exists an optimal prefix code for C in which the codewords for x and y have the same length that differ only in the last bit*

Lemma 2

- Let C be an alphabet and x and y be the two characters in C with the *lowest* frequencies
- Let C' be the alphabet C with the characters x and y removed and a new character z added, so that $C' = C$ - $\{x, y\} \cup \{z\}$
- Define

$$z.freq = x.freq + y.freq \tag{5}$$

- Let T' be a tree representing the optimal prefix code for C'
- **To prove**: *The tree T, obtained from T' by replacing the leaf node for z with an internal node having x and y as children, represents an optimal prefix code for C*

The correctness of Huffman coding follows from Lemmas 1 and 2

THANK YOU !!!