CS 218 Design and Analysis of Algorithms

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Module 1: Basics of algorithms

Huffman Coding

Problem Description

- Given a file with data coming from an alphabet (say English alphabet).
- Convert it into a binary alphabet using as few bits as possible while keeping it uniquely decodable.

Huffman Coding: Greedy strategy for prefix-free encoding

Greedy strategy 3 (Bottom-up approach)

- Start with the two least frequently occurring elements, say f, f'.
- Assign a string (suffix) 0 to one and 1 to the other.
- Combine the frequencies creating a new letter with frequency $f + f'$.
- Repeat the above steps till all the letters are assigned strings.

Greedy strategy 3 and its tree

Tree associated with the bottom-up strategy.

We can naturally associate this strategy with a tree.

The nodes of the tree are subsets of the alphabet.

The leaves are single letters.

The depth of a leaf is the length of the code word assigned to it.

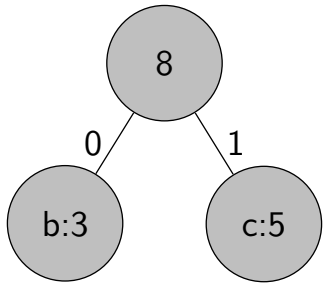
Specifying the tree completely specifies the code generated by the strategy.

Huffman Coding: Greedy strategy for prefix-free encoding

Greedy strategy 3 (Bottom-up approach)

- Start with the two least frequently occurring elements, say f, f'.
- Assign a string (suffix) 0 to one and 1 to the other.
- Combine the frequencies creating a new letter with frequency $f + f'$.
- Repeat the above steps till all the letters are assigned strings.

letters	a	b	c	d	e
frequency	100	3	5	45	20

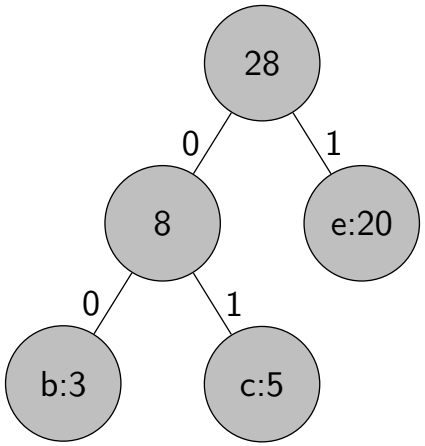


Huffman Coding: Greedy strategy for prefix-free encoding

Greedy strategy 3 (Bottom-up approach)

- Start with the two least frequently occurring elements, say f, f'.
- Assign a string (suffix) 0 to one and 1 to the other.
- Combine the frequencies creating a new letter with frequency $f + f'$.
- Repeat the above steps till all the letters are assigned strings.

letters	a	b	c	d	e
frequency	100	3	5	45	20

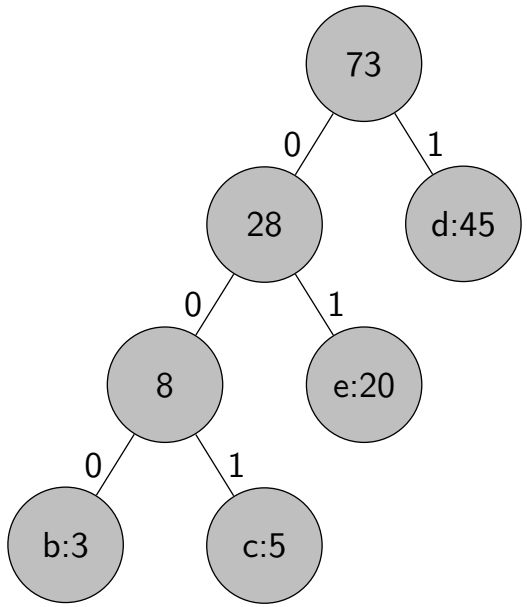


Huffman Coding: Greedy strategy for prefix-free encoding

Greedy strategy 3 (Bottom-up approach)

- Start with the two least frequently occurring elements, say f, f'.
- Assign a string (suffix) 0 to one and 1 to the other.
- Combine the frequencies creating a new letter with frequency $f + f'$.
- Repeat the above steps till all the letters are assigned strings.

letters	a	b	c	d	e
frequency	100	3	5	45	20

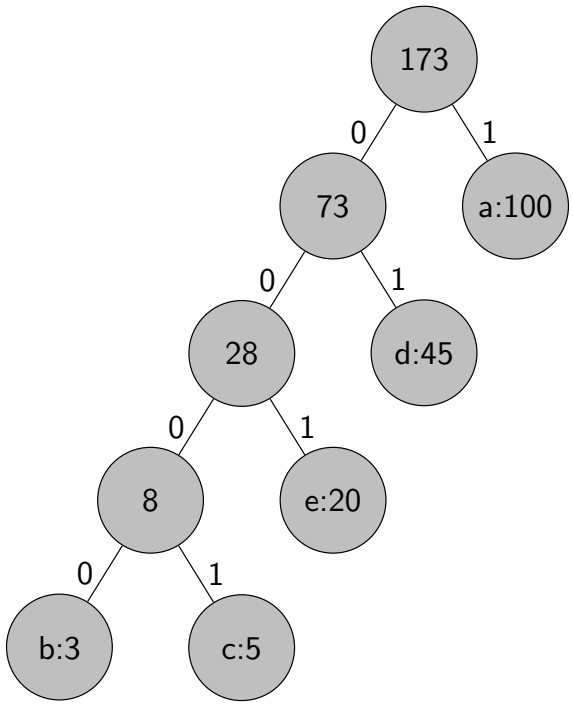


Huffman Coding: Greedy strategy for prefix-free encoding

Greedy strategy 3 (Bottom-up approach)

- Start with the two least frequently occurring elements, say f, f'.
- Assign a string (suffix) 0 to one and 1 to the other.
- Combine the frequencies creating a new letter with frequency $f + f'$.
- Repeat the above steps till all the letters are assigned strings.

letters	a	b	c	d	e
frequency	100	3	5	45	20



a: 1, b: 0000, c: 0001, d:01, e:001.

Huffman Coding: Greedy strategy for prefix-free encoding

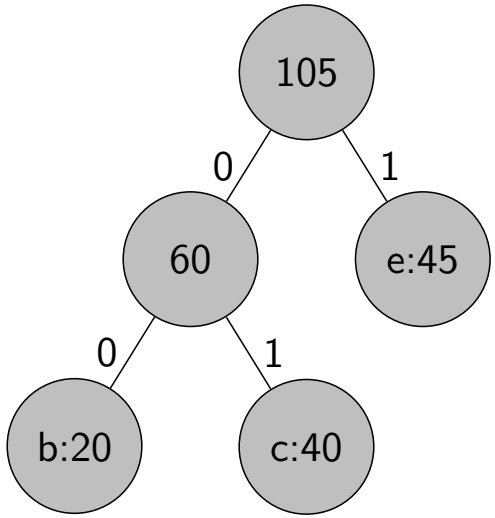
Greedy strategy 3 (Bottom-up approach)

letters	a	b	c	d	e
frequency	100	20	40	80	45

Huffman Coding: Greedy strategy for prefix-free encoding

Greedy strategy 3 (Bottom-up approach)

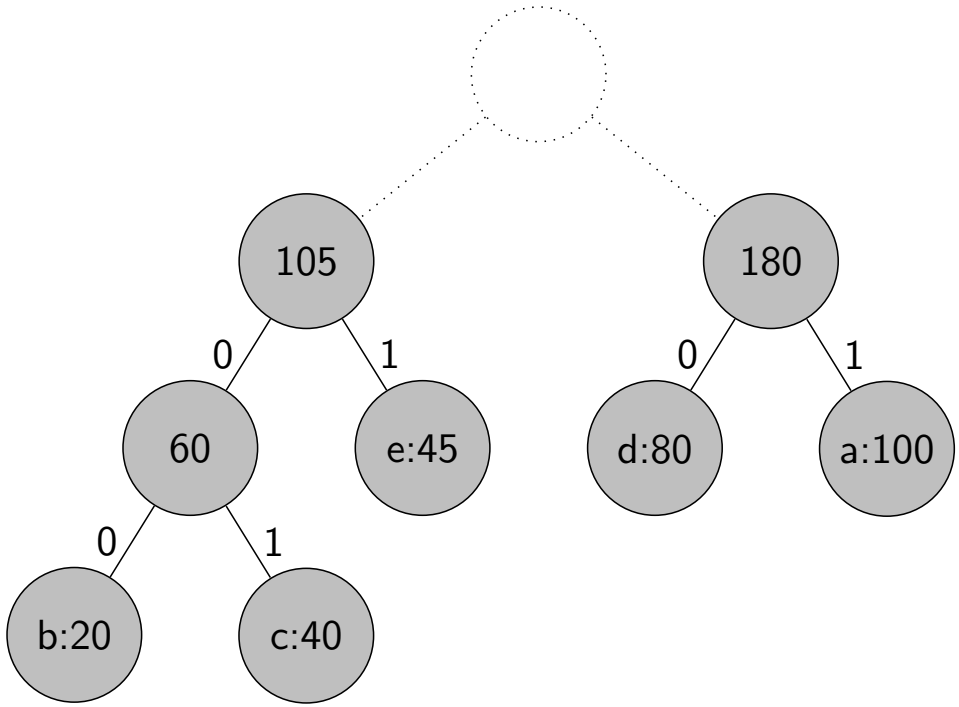
letters	a	b	c	d	e
frequency	100	20	40	80	45



Huffman Coding: Greedy strategy for prefix-free encoding

Greedy strategy 3 (Bottom-up approach)

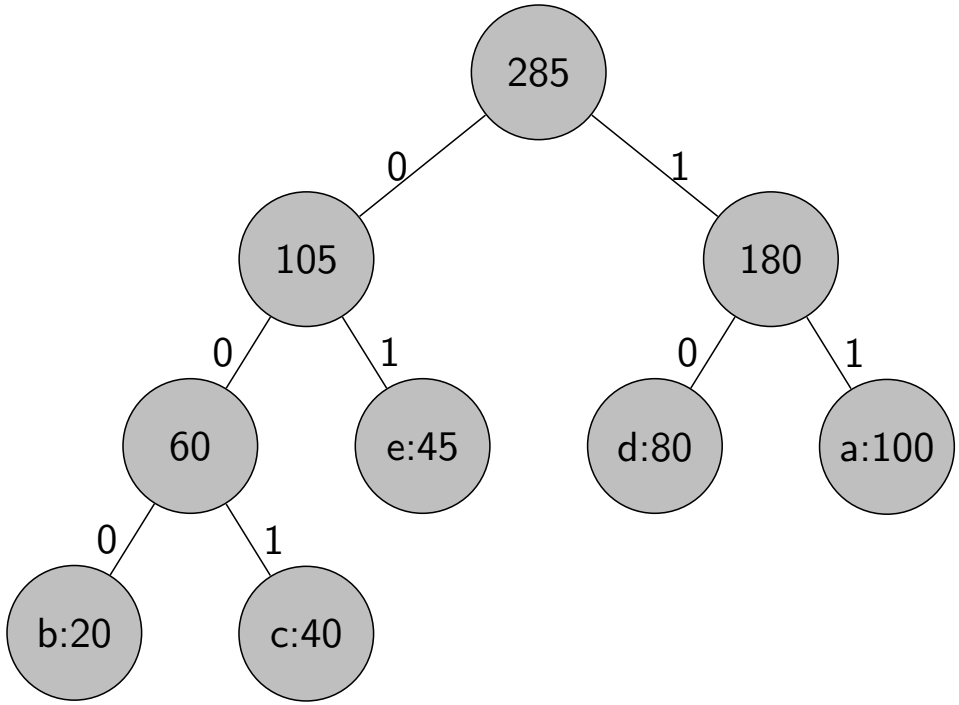
letters	a	b	c	d	e
frequency	100	20	40	80	45



Huffman Coding: Greedy strategy for prefix-free encoding

Greedy strategy 3 (Bottom-up approach)

letters	a	b	c	d	e
frequency	100	20	40	80	45



Turns out that this is the optimal strategy! (Huffman's coding).

Pseudocode for Huffman coding

Input: an alphabet $\Sigma = \{a_1, \ldots, a_n\}$ and for each $i \in [n]$ frequency f_i for the letter a_i .

Output: Binary encoding of the letters for best compression.

1: **for** $i \in [n]$ **do**
2: Create a leaf node for a_i.
3: Insert it into the Min-heap H with f_i as the key.
4: **end for**
5: **while** H has > 1 elements **do**
6: Extract the two nodes with the lowest key values from H, say u, v.
7: Let f_u and f_v be their keys.
8: Create a new internal node w. Let $f_w \leftarrow f_u + f_v$.
9: Let u be w's left child and v its right child.
10: Insert w into H with key f_w.
11: **end while**
12: return$H$

Correctness of Huffman coding

Cost of a tree T.

- Let T be any tree corresponding to the prefix-free encoding of Σ.
- Let

$$\text{cost}(T) = \sum_{a_i \in \Sigma} f_i \times d_T(a_i),$$

where $d_T(a_i)$ is the depth of the node corresponding to a_i in T.

Simple Observation: Any optimal tree is a full binary tree.

Suppose there is a node u which is the only child of its parent v.

Delete v and make u the child of v's parent.

This only decreases the total cost of the tree.

Correctness of Huffman coding

First step of the greedy.

The first choice of the greedy algorithm is optimal.

Lemma

Consider the two letters x and y with the smallest frequencies. There is an optimal code tree T^* in which these two letters are sibling leaves in the tree in the lowest level.

Optimal substructure property.

Assuming the first step is optimal, the next step is also optimal.

Lemma

Let T be a tree corresponding to the optimal encoding of Σ. Let x, y be sibling leaves of T and z their parent in T. Let $f_z = f_x + f_y$, $T' = T \setminus \{x, y\}$ and $\Sigma' = \Sigma \setminus \{x, y\} \cup \{z\}$. Then T' corresponds to the optimal encoding of Σ'.

Correctness of Huffman coding

Lemma

Consider the two letters x and y with the smallest frequencies. There is an optimal code tree T^* in which these two letters are sibling leaves in the tree in the lowest level.

Let a, b be the two letters at the lowest level of an optimal tree T.
They are siblings of each other. Let their frequencies be f_a and f_b.

Assume $f_x \leq f_y$ and $f_a \leq f_b$. We know that $f_x \leq f_a$ and $f_y \leq f_b$.
Let T' be a tree formed from T by switching a with x.

Proof

We also know that $d_T(a) \geq d_T(x)$ and $d_T(b) \geq d_T(y)$.

$$\begin{aligned}
\text{cost}(T) &\leq \text{cost}(T') \\
&= \text{cost}(T) - [d_T(x)f_x + d_T(a)f_a - d_{T'}(x)f_x - d_{T'}(a)f_a] \\
&= \text{cost}(T) - [d_T(x)f_x + d_T(a)f_a - d_T(a)f_x - d_T(x)f_a] \\
&= \text{cost}(T) - [d_T(x)(f_x - f_a) + d_T(a)(f_a - f_x)] \\
&\leq \text{cost}(T)
\end{aligned}$$