Huffman Code

- Data Compression
  - Lossy and Lossless Compression
  - Adaptive and non-Adaptive Compression
  - Fixed and Variable length Codes
- Prefix Free Codes
  - Binary Tree Representation
  - Goodness Measure
- Generic Greedy Algorithm
- Huffman Code
- Optimality and Implementation

IMDAD ULLAH KHAN

Data Compression

Fixed Length Binary Code

- Fixed num of bits for each symbol ▷ e.g. ASCII and Unicode

Variable Length Binary Code

- Variable num of bits for each symbol ▷ uses fewer bits for frequent symbols

Prefix free code

- no code is a prefix of another

If a code is prefix free, then it is uniquely decodable

Prefix free code as Binary Tree

- Prefix free code can be represented by a rooted binary tree
- Leaves are labeled with characters and edges with bits
- The bits along the path from root a leaf is code of the symbol

Compression from Codes (Binary Trees)

- Alphabet $\Sigma = \{a_1, \ldots, a_n\}$
- A document $D \in \mathcal{D}$
- f: frequency distribution, $f(a_i)$: freq. of a_i in D
- C: a compression scheme with code given as a tree
- $B_C(D) = B(D)$: number of bits to encode D with C
- $C(a_i)$: the code for a_i and $len(C(a_i))$: is its length
- $L(a_i) = len(C(a_i))$= length of root to leaf (a_i) path, $depth(a_i)$

Total number of bits needed to encode the document D is

$$B(D) \;=\; \sum_{a_i \in \Sigma} f(a_i)L(a_i) \;=\; \sum_{i=1}^{n} f(a_i) \cdot \text{ [depth of } a_i \text{ in } T]$$

Problem Formulation

Input: Given an alphabet Σ and a frequency distribution $f : \Sigma \to \mathbb{Z}$

Output: A prefix free code C with minimum $\sum_{i=1}^{n} f(a_i) \cdot$ [depth of a_i in T], where T is the tree representation of C

Equivalently

Input: A document D

Output: A prefix free code C with minimum $B(D)$

Equivalence follows from the fact that Σ and f can be computed with a single scan of D

Greedy Algorithm

Algorithm Generic Algorithm ($\mathcal{D}$)

```
Make every symbol a_i a tree T_{a_i}
for i = 1 to n − 1 do
    Select two tree T_x and T_y
    MERGE(T_x, T_y)                ▷ Make them left/right child of a new node
return the only remaining tree T
```

Clearly constructs a prefix free code

▷ Symbols always and only remain at leaves

Which two subtrees to merge?

Huffman Coding

- Have to take into account the frequency distribution
- Merging two trees increases code lengths of leaves therein by one
- Code length of a symbol is the number of merges its tree undergoes
- **Would like frequent symbols go through few merges**

1. Huffman Coding (greedily) chooses two symbols **x** and **y** with lowest frequencies (min and second min)
2. Inserts a new meta-symbol **z** for the merged tree
3. Delete x and y and their frequencies
4. $f(z) \leftarrow f(x) + f(y)$
5. Repeat on the reduced set of symbols

Huffman Coding

$\mathcal{S}$ is input symbols with associated frequencies

▷ $\mathcal{S}$ can be readily populated from input $\mathcal{D}$

Algorithm Huffman-Tree ($\mathcal{S}$)

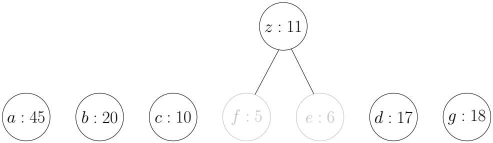
```
for x ∈ S do                                  ▷ x is both symbol and pointer
    MAKE-NODE(x)
for i = 1 to n − 1 do
    x ← FINDMIN(S)                            ▷ find the symbol with minimum freq.
    S ← S \ {x}
    y ← FINDMIN(S)
    S ← S \ {y}
    MAKENODE(z)
    z · freq ← x · freq + y · freq
    S ← S ∪ {z}
return the only node in S
```

Huffman Coding

for $x \in \mathcal{S}$ **do**
  MakeNode(x) ▷ x is both symbol and pointer
for $i = 1$ to $n - 1$ **do**
  $x \leftarrow$ FindMin($\mathcal{S}$)
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x\}$
  $y \leftarrow$ FindMin($\mathcal{S}$)
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$
  MakeNode(z)
  $z \cdot freq \leftarrow x \cdot freq + y \cdot freq$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
return the only node in $\mathcal{S}$

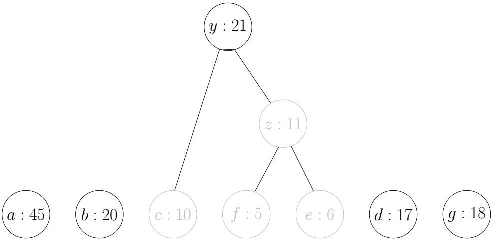
Huffman Coding

for $x \in \mathcal{S}$ **do**
  $\textsc{MakeNode}(x)$ ▷ x is both symbol and pointer
for $i = 1$ to $n-1$ **do**
  $x \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x\}$
  $y \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$
  $\textsc{MakeNode}(z)$
  $z \cdot freq \leftarrow x \cdot freq + y \cdot freq$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
return the only node in $\mathcal{S}$



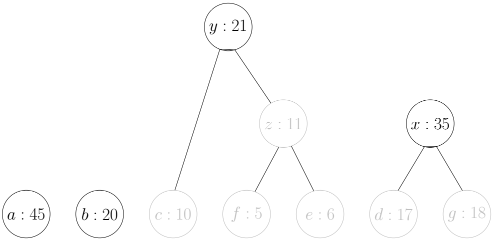
Huffman Coding

for $x \in \mathcal{S}$ **do**
  $\textsc{MakeNode}(x)$ ▷ x is both symbol and pointer
for $i = 1$ to $n-1$ **do**
  $x \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x\}$
  $y \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$
  $\textsc{MakeNode}(z)$
  $z \cdot freq \leftarrow x \cdot freq + y \cdot freq$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
return the only node in $\mathcal{S}$



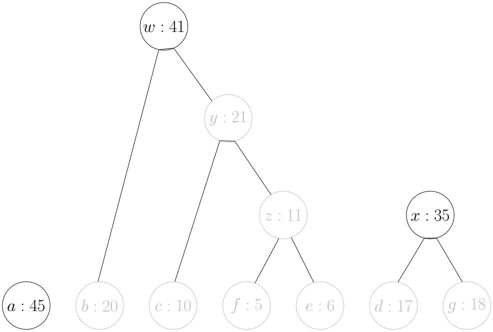
Huffman Coding

for $x \in \mathcal{S}$ **do**
  $\textsc{MakeNode}(x)$ ▷ x is both symbol and pointer
for $i = 1$ to $n - 1$ **do**
  $x \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x\}$
  $y \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$
  $\textsc{MakeNode}(z)$
  $z \cdot freq \leftarrow x \cdot freq + y \cdot freq$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
return the only node in $\mathcal{S}$



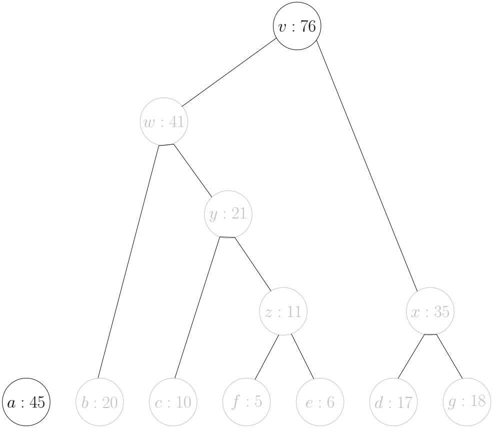
Huffman Coding

for $x \in \mathcal{S}$ **do**
  MakeNode(x) ▷ x is both symbol and pointer
for $i = 1$ to $n - 1$ **do**
  $x \leftarrow$ FindMin($\mathcal{S}$)
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x\}$
  $y \leftarrow$ FindMin($\mathcal{S}$)
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$
  MakeNode(z)
  $z \cdot freq \leftarrow x \cdot freq + y \cdot freq$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
return the only node in $\mathcal{S}$



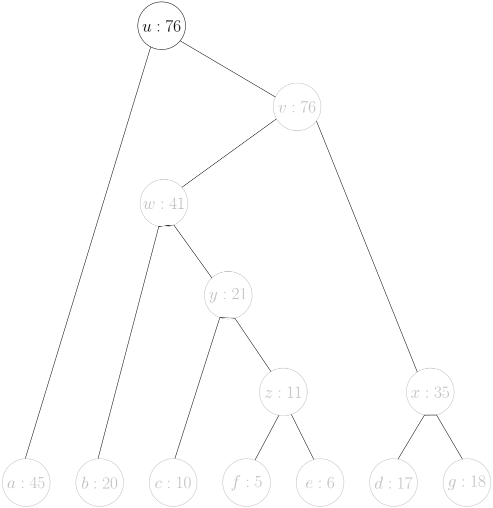
Huffman Coding

for $x \in \mathcal{S}$ **do**
  $\textsc{MakeNode}(x)$ ▷ x is both symbol and pointer
for $i = 1$ to $n-1$ **do**
  $x \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x\}$
  $y \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$
  $\textsc{MakeNode}(z)$
  $z \cdot freq \leftarrow x \cdot freq + y \cdot freq$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
return the only node in $\mathcal{S}$



Huffman Coding

for $x \in \mathcal{S}$ **do**
  $\textsc{MakeNode}(x)$ $\triangleright$ x is both symbol and pointer
for $i = 1$ to $n-1$ **do**
  $x \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x\}$
  $y \leftarrow \textsc{FindMin}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$
  $\textsc{MakeNode}(z)$
  $z \cdot freq \leftarrow x \cdot freq + y \cdot freq$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
return the only node in $\mathcal{S}$



Huffman Coding

for $x \in \mathcal{S}$ **do**
  MakeNode(x) ▷ x is both symbol and pointer
for $i = 1$ to $n - 1$ **do**
  $x \leftarrow$ FindMin($\mathcal{S}$)
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x\}$
  $y \leftarrow$ FindMin($\mathcal{S}$)
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$
  MakeNode(z)
  $z \cdot freq \leftarrow x \cdot freq + y \cdot freq$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{z\}$
return the only node in $\mathcal{S}$