CSEP 590
Data Compression
Autumn 2007

Adaptive Huffman Coding
Adaptive Huffman Coding

- One pass
- During the pass calculate the frequencies
- Update the Huffman tree accordingly
  - Coder – new Huffman tree computed after transmitting the symbol
  - Decoder – new Huffman tree computed after receiving the symbol
- Symbol set and their initial codes must be known ahead of time.
- Need NYT (not yet transmitted symbol) to indicate a new leaf is needed in the tree.
Optimal Tree Numbering

- $a : 5$, $b : 2$, $c : 1$, $d : 3$
Weight the Nodes

• $a: 5$, $b: 2$, $c: 1$, $d: 3$
Number the Nodes

- a : 5, b: 2, c : 1, d : 3

Number the nodes as they are removed from the priority queue.
Adaptive Huffman Principle

- In an optimal tree for n symbols there is a numbering of the nodes $y_1 < y_2 < \ldots < y_{2n-1}$ such that their corresponding weights $x_1, x_2, \ldots, x_{2n-1}$ satisfy:
  - $x_1 \leq x_2 \leq \ldots \leq x_{2n-1}$
  - siblings are numbered consecutively

- And vice versa
  - That is, if there is such a numbering then the tree is optimal. We call this the node number invariant.
Initialization

• Symbols $a_1, a_2, \ldots, a_m$ have a basic prefix code, used when symbols are first encountered.
• Example: a, b, c, d, e, f, g, h, i, j
Initialization

- The tree will encode up to \( m + 1 \) symbols including NYT.
- We reserve numbers 1 to \( 2m + 1 \) for node numbering.
- The initial Huffman tree consists of a single node.
Coding Algorithm

1. If a new symbol is encountered then output the code for NYT followed by the fixed code for the symbol. Add the new symbol to the tree.
2. If an old symbol is encountered then output its code.
3. Update the tree to preserve the node number invariant.
Decoding Algorithm

1. Decode the symbol using the current tree.
2. If NYT is encountered then use the fixed code to decode the symbol. Add the new symbol to the tree.
3. Update the tree to preserve the node number invariant.
Updating the Tree

1. Let $y$ be leaf (symbol) with current weight $x$.*
2. If $y$ the root update $x$ by 1, otherwise,
3. Exchange $y$ with the largest numbered node with the same weight (unless it is the parent).**
4. Update $x$ by 1
5. Let $y$ be the parent with its weight $x$ and go to 2.

*We never update the weight of NYT
** This exchange will preserve the node number invariant
Example

• \texttt{aabcdad} in alphabet \{a,b,..., j\}

\begin{center}
\begin{tabular}{|c|}
\hline
0 & 21 \\
\hline
\end{tabular}
\end{center}

NYT

\textbf{fixed code for a}

output = 000

\textbf{fixed code}
Example

- \texttt{aabcdad}

output = \texttt{000}
Example

• $aaaabcdad$

output = 0001
Example

- \texttt{aabcdad}

output = 0001
Example

• aabcdad

NYT

fixed code for b

output = 00010001
Example

- aabcdad

output = 00010001
Example

- \texttt{aab\_c\_d\_d\_a\_d}

\begin{center}
\begin{tikzpicture}[level distance=1.5cm, sibling distance=1.5cm, every node/.style={draw,rectangle,rounded corners,minimum height=1cm}]
  \node (n0) {0} child {node (n1) {19} child {node (n5) {0} child {node (n7) {17} child {node (n13) {NYT}}}} child {node (n6) {1}} child {node (n18) {1}}};
  \node (n2) {2} child {node (n19) {21} child {node (n20) {2}}};
\end{tikzpicture}
\end{center}

output = 00010001
Example

- aabccdada

```
output = 00010001
```

CSEP 590 - Lecture 2 - Autumn 2007

19
Example

- `aabcbdad`

```
Example

- aabcbdad

output = 00010001000100

fixed code for c

NYT

output = 00010001000100
```
Example

- aabcbdad

output = 0001000100010
Example

- \texttt{aabc\textunderscore dad}

\[
\begin{array}{cc}
0 & 17 \\
\text{NYT} & 15 \\
0 & 1
\end{array}
\begin{array}{cc}
1 & 18 \\
16 & c
\end{array}
\begin{array}{c}
20 \\
a
\end{array}
\begin{array}{c}
21 \\
1
\end{array}
\begin{array}{c}
0 \\
1
\end{array}
\]

output = 0001000100010
Example

- \texttt{aab\_c\_d\_d\_ad}

\[
\begin{array}{c}
\text{output = 0001000100010}
\end{array}
\]
Example

• aabcbdad

output = 0001000100010
Example

- $\text{aab}cd\text{a}d$

Output: $0001000100010000011$

Fixed code for $d$
Example

• aabcdad

output = 0001000100010000011
Example

• aabcdad

output = 0001000100010000011
Example

- aabcdad

output = 00010001000010000011

exchange!
Example

- aabcbdad

output = 00010001000100000111
Example

• \texttt{aabcdad}

\[
\begin{array}{c}
\text{output} = 00010001000100000111
\end{array}
\]
Example

- `aabcdnad`

```
Example

• aabcdnad

output = 0001000100010000011
```
Example

- aabcdad

output = 0001000100010000011
Example

- aabcdad

```
output = 00010001000100000110
```

Note: the first a is coded as 000, the second as 1, and the third as 0.
Example

- `aabccdad`

output = 00010001000100000110
Example

- aabcdad

```
output = 000100010001000001101101
```

exchange!
Example

• aabcdad

output = 000100010001000001101101
Example

- aabcdad

output = 000100010001000001101101
Example

- aabccdada

```
output = 000100010001000001101101
```
Example

• aabcdad

\[
\text{output} = 000100010001000001101101
\]
Data Structure for Adaptive Huffman

1. Fixed code table
2. Binary tree with parent pointers
3. Table of pointers nodes into tree
4. Doubly linked list to rank the nodes
In Class Exercise

• Decode using adaptive Huffman coding assuming the following fixed code

• 00110000

• 00110000
Huffman Summary

- Statistical compression algorithm
- Prefix code
- Fixed-to-variable length code
- Optimization to create a best code
- Symbol merging
- Context
- Adaptive coding
- Decoder and encoder behave almost the same
- Need for data structures and algorithms