5. Greedy algoritm

5.0 Basics

5.1 Minimum spanning trees

5.2 Knapsack problem

5.3 Job sequencing with deadline

5.4 Optimal merge patterns

5.5 Huffman encoding

5.5 Huffman encoding

- Example: The encoding process of MP3
  - Sample at regular rates
    - In CD, 44,100 samples per second
    - A 50 min-length symphony has 50 X 60 X 44,100 ≈ 130,000,000 samples
  - Each sample is quantized
    - Each sample value is approximated by a nearby number from a finite set T.
  - The resulting string is encoded in binary

5.5 Huffman encoding

- Size of encoding
  - Estimating the size of encoding
    - Number of samples X T.
    - If T = {A, B, C, D}, previous example has 130,000,000 X 2 bits = 37.5MByte
    - If T has M symbols, encoding for T requires ($\log_2 M$) bits.
  - How can we reduce the size of encoding?
    - Reduce sample rates
    - Reduce the length of alphabets in T

5.5 Huffman encoding

- Reduce the length of alphabets in T
  - Greedy approach

    → Variable-length encoding

Alphabet	Frequency	Conventional	Variable-length
A	70M	00	0
B	3M	01	001
C	20M	10	10
D	37M	11	11

- The resulting size of encoding becomes

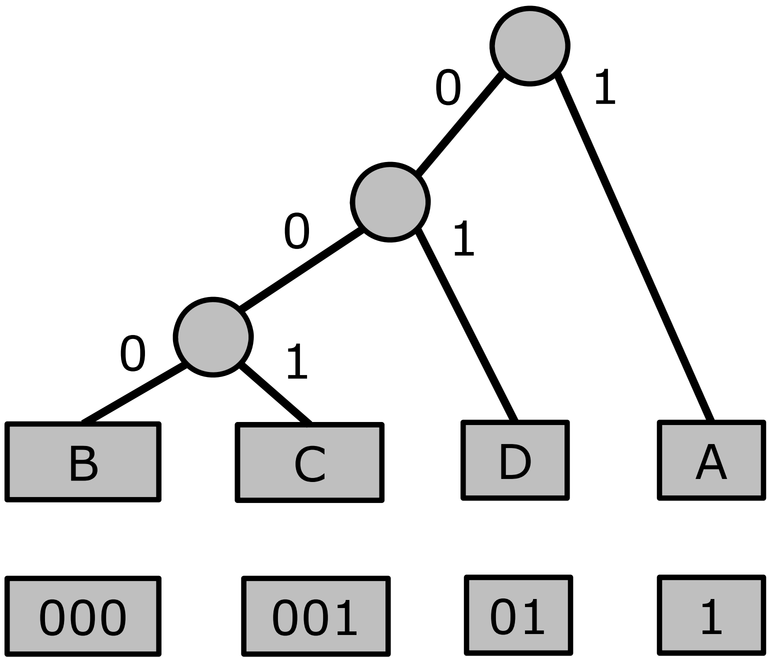
Type	Formula	Size
Conventional	130,000,000 X 2 bits	260M bits
Variable-length	70M X 1 bit + 57M X 2 bits + 3M X 3bits	193M bits

5.5 Huffman encoding

- Problem of previous encoding?
  - Ambiguity
  - What is 0010?
    - AAC or BA
  - How to avoid ambiguity?
    - Prefix-free encoding
      - No codeword can be a prefix of another codeword
      - Can be represented using full binary tree
- Huffmann encoding
  - Variable-length & prefix-free encoding
  - Similar to optimal merge pattern algorithm

5.5 Huffman encoding

- Strategy
  - Similar to optimal merge pattern
  - Sort the symbols according to the increasing order of frequency



5.5 Huffman encoding

- Huffmann encoding
  - Variable-length & prefix-free encoding

Alphabet	Frequency	Conventional	Variable-length	Huffman code
A	70M	00	0	1
B	3M	01	001	000
C	20M	10	10	001
D	37M	11	11	01

Type	Formula	Size
Conventional	130,000,000 X 2 bits	260M bits
Variable-length	70M X 1 bit + 57M X 2 bits + 3M X 3bits	193M bits
Huffman code	70M X 1 bit + 37M X 2 bits + 23M X 3bits	213M bits

All about Greedy Algorithm

	Purpose	Feasibility	Step	Optimization scheme	
Kruskal	MCSP	n-1 edges without cycle	Adding edges	Sorting edges	
Prim	MCSP	n-1 edges without cycle	Adding vertices	Sorting edges	
Knapsack		$\Sigma x_i w_i \leq M$	Selecting objects	Sorting p_i/w_i	
Job sequencing		$D(J(r)) > r$ $D(J(r_i)) \leq D(J(r_{i+1}))$	Arranging jobs	Sorting profits	yield
Optimal merge			Merging files	Sorting lengths	
Huffman			Similar to Optimal merge	Sorting frequencies	Prefix -free

Contents