IR UNIT – 3 (Index Compression and Dynamic Inverted Indices) – END-SEM PYQ Answers

➢ MAY / JUN 2022

Q1) a) Explain Data Compression with Huffman Coding. [9]

Data Compression is a process of reducing the size of data so that it occupies fewer bits. The goal is to remove redundancy and represent information more efficiently. Compression improves storage efficiency and transmission speed.

Huffman Coding is a lossless compression technique based on variable-length encoding. It assigns shorter codes to more frequent characters and longer codes to less frequent characters, achieving optimal prefix-free encoding.

Steps in Huffman Coding

1. Calculate Frequency:
   Count how many times each symbol appears in the data.
2. Build Min-Heap (Priority Queue):
   Insert all symbols as nodes with their frequencies.
3. Construct Huffman Tree:
   - Remove two nodes with the smallest frequency.
   - Create a new internal node whose frequency is the sum of the two.
   - Insert it back into the queue.
   - Repeat until only one node (root) remains.
4. Generate Huffman Codes:
   - Traverse the tree:
     - Assign 0 for left branch
     - Assign 1 for right branch
   - The code for each symbol is collected along the path from root to leaf.

Properties of Huffman Codes

- Prefix-free: No code is a prefix of another, reducing decoding ambiguity.
- Optimality: Produces minimal average code length compared to other prefix codes.
- Lossless: Original data can be reconstructed exactly.

Example:

Assume characters and frequencies:
A:5, B:9, C:12, D:13, E:16, F:45
Huffman tree assigns codes like:
F:0, C:100, D:101, A:1100, B:1101, E:111
(Frequent symbols get shorter codes)

Advantages:

- Very effective for text compression.
- Simple and fast encoding/decoding.
- Widely used in ZIP files, JPEG, MP3, etc.

Conclusion: Huffman Coding is a widely used lossless data compression algorithm that reduces data size by assigning variable-length, prefix-free codes based on symbol frequency, offering an optimal encoding method.

Q1) b) What are the Context-Aware Compression Methods? Explain it in detail.[9]

Context-Aware Compression uses information about the surrounding data (context) to predict upcoming symbols and encode them more efficiently. Unlike simple frequency-based methods, these techniques analyze patterns, correlations, and dependencies within the data.

These methods are highly effective for text, images, audio, and executable files where context plays a role.

1. Prediction by Partial Matching (PPM): PPM is a statistical compression model that predicts the next symbol based on previously seen symbols (context).

- Uses variable-length context models (e.g., last 1, 2, 3 characters).
- For each context, it stores frequencies of possible next symbols.
- Uses escape symbols to handle unknown symbols.
  Advantages: High compression ratio for text.
  Disadvantages: Large memory usage.

2. Dynamic Markov Compression (DMC): DMC uses a finite-state machine that grows dynamically as data is processed.

- Each state represents a context and stores probabilities for '0' or '1'.
- When a symbol is read, states split and update their prediction model.
  Advantages: Good adaptation to changing data patterns.
  Used in: Some file compressors and predictive coders.

3. Burrows–Wheeler Transform (BWT) + Move-to-Front + Huffman: BWT is technically not compression by itself but a reordering transform that makes the data more context-friendly.

- Rearranges characters so that similar symbols cluster together.
- After BWT, Move-to-Front (MTF) encoding reduces repeated patterns.
- Finally, Huffman Coding is applied.
  Used in: bzip2 compression tool.

4. Lempel-Ziv (LZ) Family (LZ77 / LZ78 / LZW): LZ methods use dictionary-based context:

- The algorithm searches for longest matching substrings in previously seen data.
- Replaces repeated sequences with references (offset, length).
  Advantages: Fast, widely used.
  Used in: ZIP, GIF, PNG, PDF.

5. Context Mixing / CM Models: These use multiple context models and mix their predictions using weighted averages.

- Produces extremely accurate probability estimates.
- Leads to superior compression.
  Used in: PAQ series compressors (one of the highest compression ratios).

6. Arithmetic Coding with Context Models: Here, arithmetic coding encodes entire messages as a single number based on symbol probabilities. When combined with context models, probabilities become more accurate and compression improves.

Advantages of Context-Aware Methods

- High compression ratio
- Adapts to different types of data
- Effective for complex patterns

Applications

- Text files
- Multimedia compression
- Modern file compressors (bzip2, 7zip, rar)

Q2) a) Write a short note on: [9]

i) Decoding Performance ii) Document Reordering iii) Arithmetic Coding

i) Decoding Performance

Decoding performance refers to the speed and efficiency with which compressed data is reconstructed back into its original form. It depends on:

- Algorithm complexity: Simple algorithms like Huffman decode faster than arithmetic coding.
- Memory usage: Efficient memory access speeds up decoding.
- Data structure design: Prefix-free codes allow direct and fast decoding.
- Hardware capability: Faster processors reduce decoding time.

Good decoding performance is essential in real-time applications such as video streaming, communication systems, and search engines. It ensures minimal delay, low computational overhead, and smooth user experience.

ii) Document Reordering

Document Reordering refers to rearranging documents in a compressed index or storage to improve compression and retrieval efficiency.

- Often used in search engine indexes where documents are renumbered based on similarity or connectivity.
- Documents with similar content or link patterns are placed closer so that posting lists compress better due to smaller gaps (d-gaps).
- Reordering also improves cache locality, leading to faster query processing.

Common techniques:

- URL-based sorting
- Graph-based clustering (e.g., BFS ordering in a web graph)
- Content-based reordering

Its main goal is to reduce index size and speed up search operations.

iii) Arithmetic Coding

Arithmetic Coding is an entropy-based lossless compression technique that represents an entire message as a single fractional number between 0 and 1.

- Instead of giving each symbol a codeword, it repeatedly narrows down an interval based on symbol probabilities.
- More frequent symbols shrink the interval less, achieving compression close to the theoretical limit (Shannon entropy).
- Steps:
    1. Assign probability ranges to symbols.
    2. Process each symbol to reduce the interval.
    3. Final interval uniquely represents the message.
- Achieves better compression than Huffman coding when probabilities are skewed.

Used in video codecs (H.264), image compression, and text compression.

b) Explain in detail Invalidation List, Garbage Collection, Document Modifications in document deletion.

When documents are deleted or modified in large information retrieval systems (such as search engines), the index must be updated efficiently. Directly rebuilding the index every time is costly, so special mechanisms are used.

1. Invalidation List

An invalidation list keeps track of documents that are deleted, outdated, or modified.

- The main index remains unchanged initially.
- Deleted document IDs are added into the invalidation list.
- During query processing, the search engine filters out results that appear in this list.

Advantages:

- Quick update (no need to rebuild index immediately)
- Efficient for large-scale systems

Usage:

- Web search engines
- Dynamic text collections

2. Garbage Collection

Garbage collection refers to the process of removing invalid or deleted entries from an index to free space and maintain efficiency.

- Performed periodically, not after every deletion.
- The system scans the index and eliminates entries whose document IDs appear in the invalidation list.
- Compacting the index improves:
    - Storage utilization
    - Query speed
    - Compression ratio

Types:

- Online GC: Runs continuously in the background
- Offline GC: Runs at specific intervals

3. Document Modifications in Document Deletion

Document modifications involve:

- Updating document content
- Removing outdated fields
- Replacing old data with new data

In indexing systems, modification is treated as:

1. Delete old version (add its ID to invalidation list)
2. Insert new version as a fresh document with a new document ID

This approach avoids expensive in-place updates because:

- Posting lists are compressed and ordered
- Direct modification breaks compression and index order

Process:

- Mark old document as invalid
- Add new document data to the index
- During garbage collection, old postings are removed permanently

Advantages:

- Maintains index consistency
- Reduces update cost
- Works well for dynamic document collections

➢ **MAY/JUNE 2023**

Q1) a) Enlist the General-Purpose Data Compression techniques and explain with suitable examples. [9]

General-purpose compression techniques are algorithms designed to compress any type of data (text, images, binaries, executables, logs, web data, etc.) without being specific to one domain. These algorithms focus on reducing redundancy and improving storage/transmission efficiency.

1. Dictionary-Based Compression (LZ Family): These methods replace repeated strings with references from a dictionary built during compression.

Examples:

- LZ77: Uses sliding window, replaces repeated patterns with (offset, length).
- LZ78 / LZW: Maintains a dictionary of strings; replaces strings with dictionary indexes.

Used in: ZIP, PNG, GIF.

2. Statistical Compression: These methods assign shorter codes to frequent symbols and longer codes to rare symbols.

Examples:

- Huffman Coding: Creates prefix-free variable codes based on symbol frequency.
- Arithmetic Coding: Encodes whole messages into a fractional interval for higher compression.

Used in: JPEG, text compression, multimedia codecs.

3. Transform-Based Compression: Transforms data into a form where repeated patterns are grouped.

Examples:

- Burrows–Wheeler Transform (BWT): Rearranges data so similar characters cluster.
- Usually combined with Move-to-Front (MTF) + Huffman Coding.

Used in: bzip2.

4. Run-Length Encoding (RLE): Replaces consecutive repeating characters with (value, count).

Example:

String: AAAAABBBB → A5B4
Effective when data has long runs.

Used in: TIFF images, fax documents.

5. Delta / Differential Compression: Stores changes between consecutive data values.

Example:

100, 101, 103, 106 → store differences: 1, 2, 3
Reduces redundancy in signals or numeric data.

6. Entropy Coding: Represents data using probability-based encoding.

Examples:

- Arithmetic coding
- Range coding

Used as a final stage for many compressors.

Conclusion: General-purpose compression techniques include dictionary-based, statistical, transform-based, run-length, and entropy coding methods, each effective for different types of data. Modern compressors often combine multiple techniques to achieve high compression ratios.

b) Write a short note on. [9]
i) Nonparametric Gap Compression **ii) Parametric Gap Compression**
iii) Context-Aware Compression Methods

i) Nonparametric Gap Compression: Nonparametric gap compression encodes gaps (d-gaps) between sorted document IDs without assuming any fixed distribution.
Used in inverted indexes in search engines.
Key Methods:

- Unary Coding: Represents number n as n ones followed by 0.
- Gamma Coding: Uses log-based prefix + binary suffix.
- Delta Coding: More compact than gamma; encodes length of binary first.

Features:

- No assumptions about how gaps are distributed.
- Good when gaps vary widely.
- Simple and effective.

Example:
Gap = 10
Gamma code = prefix (1110) + suffix (010)
Used in web indexing and IR systems.

ii) Parametric Gap Compression: Parametric methods assume a probability distribution for the gaps. This allows encoding them more compactly when the assumption fits real data.

Common Distributions:

- Geometric distribution (for small frequent gaps)
- Zipfian distribution (common in web data)

Techniques:

- Use Golomb Coding: optimal for geometric distributions.
- Use Rice Coding: simplified Golomb when divisor is power of 2.

Features:

- Highly efficient when gaps are small and follow expected distribution.
- Better compression than nonparametric if model is accurate.

Example: If average gap ≈ 5, Golomb coding compresses very effectively.

Used in search engines like Google for inverted lists.

iii) Context-Aware Compression Methods: Context-aware compression uses previous symbols (context) to estimate the probability of next symbols, improving compression quality.

Main Techniques:

1. **PPM (Prediction by Partial Matching)**
   - Builds variable-length contexts from previous characters.
   - Predicts next symbol using frequency tables.
   - Very high compression for text.
2. **DMC (Dynamic Markov Compression)**
   - Uses Markov states that adapt dynamically as data is processed.
   - Each state represents a context and predicts next symbol.
3. **CM (Context Mixing)**
   - Combines multiple context models, weighted by their accuracy.
   - Achieves best compression ratios (used in PAQ compressors).
4. **BWT-based Context Transform**
   - Burrows–Wheeler Transform rearranges data so similar characters group.
   - Followed by MTF + Huffman coding.

Features:

- Uses context patterns to improve probability estimation.
- Works for text, images, and binary data.
- Produces high compression ratios.

Q2) a) Write a short note on. [9]

i) Modeling and Coding ii) Huffman Coding iii) Arithmetic Coding

i) Modeling and Coding: Modeling and Coding are the two major components of any data compression system.

Modeling:

- Modeling identifies the statistical structure or patterns in the data.
- It predicts the probability of occurrence of each symbol.
- A good model reduces redundancy and improves compression.

Types of Models:

- Fixed models: Probabilities do not change.
- Adaptive models: Probabilities update as data is read.
- Context-based models: Probability depends on previous symbols (PPM, Markov models).

Role of Modeling:

- Provides a probability distribution for symbols
- Drives the coding mechanism
- Higher accuracy → better compression ratio

ii) Huffman Coding: Huffman Coding is a lossless, variable-length, prefix-free compression technique. It assigns shorter codes to frequent symbols and longer codes to rare symbols.

Process:

1. Count frequency of each symbol.
2. Build a min-heap and construct a binary Huffman Tree.
3. Left edge = 0, Right edge = 1.
4. Generate codes for each symbol based on tree traversal.

Example:

If A is frequent → Code = 0
If Z is rare → Code = 11011

Features:

- Simple and efficient
- Optimal among prefix-free codes
- Used in ZIP files, JPEG, MP3

iii) Arithmetic Coding: Arithmetic Coding is an entropy-based compression method that encodes an entire message into a single fractional interval (0,1).

Concept:

- Each symbol is assigned a probability range.
- The interval is repeatedly divided according to symbol probabilities.
- Final interval uniquely identifies the entire message.

Features:

- Closer to Shannon's optimal compression
- More efficient than Huffman when symbol probabilities are skewed
- Used in H.264, H.265, JPEG2000

Advantages:

- Handles fractional bit lengths
- Very high compression ratio

b) Describe Dynamic Inverted Indices like Incremental Index Updates, Contiguous Inverted Lists and Noncontiguous Inverted. [9]

Dynamic inverted indices deal with continuous updates, such as document additions, deletions, and modifications in a search engine.

1. Incremental Index Updates: Incremental updates allow adding or deleting documents without rebuilding the entire index.

Process:

- New documents are stored in a separate index partition (auxiliary index).
- Periodically, these partitions are merged with the main index.

Advantages:

- Fast updates
- No interruption to search queries
- Efficient for frequently changing document sets

Used in: Web search engines, dynamic databases.

2. Contiguous Inverted Lists: In a contiguous layout, the postings list of a term is stored in one continuous block of memory.

Characteristics:

- Excellent cache performance
- Fast query processing
- Easy compression (gap encoding works very well)

Drawback:

- Not suitable for frequent updates.
  If new postings are added, the entire block may need to be moved or resized.

Best used when: Indexes are mostly static.

3. Noncontiguous Inverted Lists: Here, postings for a term are stored in separate blocks or segments, linked via pointers.

Characteristics:

- Supports fast dynamic updates
- New entries can be appended to new blocks
- Avoids relocation of large memory chunks

Drawbacks:

- Slightly slower retrieval because of pointer following
- Lower compression efficiency compared to contiguous lists

Used in: Large-scale search engines that require frequent updates.

Summary

Method	**Strength**	**Weakness**	**Best Use**
Incremental Updates	Fast addition/deletion	Requires merge step later	Dynamic collections
Contiguous Lists	Fast query, high compression	Poor update support	Static indexes
Noncontiguous Lists	Flexible updates	Slightly slower query	Web-scale IR systems

➢ **NOV/DEC 2023**

Q1) a) What is Data Compression? Explain Modeling and Coding. [9]

Data Compression is the process of reducing the size of data by eliminating redundancy.
Its main goals are to:

- Save storage space
- Reduce transmission time
- Improve efficiency in data retrieval and processing

Compression can be lossless (no data lost) or lossy (some information is discarded). To perform compression effectively, two major components are used: Modeling and Coding.

1. Modeling

Modeling identifies the statistical structure, patterns, and redundancy present in the data.
It predicts the probability of each symbol, which helps the coding step compress more efficiently.

Types of Models:

- Fixed Model: Probabilities remain constant.
- Adaptive Model: Updates probabilities as data is processed.
- Context-Based Model: Uses previous symbols (context) to predict the next one (e.g., PPM, Markov models).

Role of Modeling:

- Creates an accurate probability distribution
- Improves compression ratio
- Reduces uncertainty in data

A better model → better prediction → better compression.

2. Coding

Coding converts the probability information from the model into compressed binary representations.

Popular Coding Techniques:

- Huffman Coding: Assigns shorter codes to frequent symbols and longer to rare ones.
- Arithmetic Coding: Represents the entire message as a single number in the interval (0,1) using symbol probabilities.
- Run-Length Encoding (RLE): Encodes consecutive repeating symbols.

Role of Coding:

- Converts data into compact bitstreams

- Ensures prefix-free or uniquely decodable codes
- Implements the model's probability information efficiently

Conclusion: Data compression relies on the combination of Modeling (to understand data behavior) and Coding (to convert that behavior into compact form). Together, they minimize redundancy and achieve optimal compression.

b) Write a short note on: [9]

i) Invalidation List ii) Garbage Collection iii) Document Modifications

i) Invalidation List: The Invalidation List is a structure used in dynamic search systems to keep track of documents that are deleted or outdated.
Instead of immediately removing the document from the index (which is expensive), its ID is added to an invalidation list.
Features:

- Faster updates since the index is not rebuilt immediately
- During querying, search engines filter out invalid or deleted document IDs
- Useful when frequent changes occur

Benefits:

- Low update overhead
- Maintains real-time responsiveness

Used widely in large search engines before full index rebuild.

ii) Garbage Collection: Garbage Collection refers to the process of periodically cleaning the index by removing entries that belong to deleted or invalid documents.

Process:

- Scan the inverted index
- Remove postings whose document IDs appear in the invalidation list
- Compact the index by merging remaining postings
- Free up storage space

Benefits:

- Improves compression efficiency
- Speeds up query processing
- Keeps index clean and consistent

Garbage collection is usually performed offline or in the background.

iii) Document Modifications: Document Modification refers to updating the content of an existing document in the index.
Since inverted indexes store compressed, ordered posting lists, in-place updates are expensive.

Approach Used:

Document modification is treated as:

1. Delete the old version
    - Add its ID to the invalidation list
2. Insert the updated document
    - Add it as a new document with a new document ID

Why this approach?

- Maintains sorted posting lists
- Avoids breaking compression structure
- Makes updates efficient and consistent

Advantages:

- Easier handling of updates
- Ensures index stays fresh
- Works well for dynamic web documents

Q2) a) Write a short note on: [9]
i) Decoding Performance ii) Document Reordering iii) Arithmetic Coding
i) Decoding Performance: Decoding performance refers to the speed and efficiency with which compressed data is reconstructed into its original form.
Good decoding performance depends on:

- Algorithm Complexity: Simple codes (Huffman) decode faster than arithmetic coding.
- Memory Access Patterns: Efficient data structures reduce overhead.
- Prefix-Free Codes: Allow direct table-based decoding.
- Hardware Efficiency: Faster CPU and cache improve decoding.

High decoding performance is essential in real-time applications like video playback, search engines, and communication systems. Good performance ensures low latency, quick access, and smooth system operation.

ii) Document Reordering: Document Reordering is the process of rearranging document IDs so that documents with similar properties appear closer in the index.
The main purpose is to improve compression and retrieval efficiency.

Why it is used?

- Posting lists store d-gaps (differences between sorted document IDs).
- If similar documents are placed together, gaps become smaller.
- Smaller gaps → better compression using gap-encoding methods.

Methods:

- URL-based reordering
- Graph-based clustering (e.g., BFS ordering in web graphs)
- Content-based grouping

Benefits:

- Smaller index size
- Better cache locality
- Faster query processing

iii) Arithmetic Coding: Arithmetic Coding is a lossless entropy coding technique that represents an entire message as a single fractional number between 0 and 1.

Working Principle:

1. Assign probability intervals to each symbol (e.g., A:0–0.5, B:0.5–0.8, C:0.8–1).
2. For each symbol in the message, the interval is subdivided according to symbol probabilities.
3. Final narrow interval uniquely represents the entire message.

Features:

- Achieves compression close to Shannon's limit
- More efficient than Huffman when symbol probabilities are skewed
- Allows fractional bit output

Applications: Used in image/video compression (JPEG2000, H.264) and high-performance file compressors.

b) Differentiate between Contiguous Inverted Lists & Noncontiguous Inverted. [5]

Q2 (b) Differentiate between Contiguous Inverted Lists & Noncontiguous Inverted Lists. [5]

Contiguous Inverted Lists	Noncontiguous Inverted Lists
Posting list of a term is stored in **one continuous block** of memory.	Posting list is stored in **separate blocks or segments**, linked via pointers.
Provides **fast query processing** due to good locality and fewer cache misses.	Slightly **slower retrieval** because multiple blocks must be accessed.
Highly **compressible** since gaps follow smooth patterns.	Compression efficiency is lower because lists are broken into chunks.
Poor support for **dynamic updates** (insertions require shifting or rebuilding).	Very good for **frequent updates**, as new blocks can be appended easily.
Best suited for **static or rarely updated indexes**.	Best suited for **dynamic, large-scale systems** like search engines.

c) Differentiate between Non parametric Gap Compression & Parametric Gap Compression. [4]

Q2 (c) Differentiate between Nonparametric Gap Compression & Parametric Gap Compression. [4]

Nonparametric Gap Compression	Parametric Gap Compression
Does **not assume any probability distribution** of gaps.	Assumes a **specific probability distribution** (e.g., geometric).
Encoding is based purely on gap size.	Encoding is optimized based on expected distribution.
Examples: **Unary, Gamma, Delta coding.**	Examples: **Golomb coding, Rice coding.**
Works well when gaps vary widely or unpredictably.	Gives better compression when gaps follow the assumed model.

- **MAY/JUNE 2024**

Q1) a) Explain Data Compression with Arithmetic Coding. [9]

Data Compression is the technique of reducing the size of data by eliminating statistical redundancy.
Goals of data compression:

- To reduce storage space
- To improve transmission speed
- To efficiently encode data for storage/retrieval

Compression can be lossless (reversible) or lossy (irreversible).
Arithmetic Coding is one of the most powerful lossless statistical compression techniques.

Arithmetic Coding: Arithmetic Coding encodes an entire message into a single fractional number between 0 and 1.
Instead of assigning a fixed code to each symbol (as in Huffman coding), it uses the probability of symbols to continuously narrow down an interval.

Working Principle

1. Assign probability ranges to each symbol.
   Example:
   A (0.0–0.5), B (0.5–0.8), C (0.8–1.0)
2. Start with interval [0,1).
3. For each symbol in the input string:
   - Subdivide the current interval into sub-intervals proportional to symbol probabilities.
   - Choose the sub-interval corresponding to the current symbol.
4. Final interval uniquely represents the entire message.
   Any number within this final interval acts as the encoded output.

Example

Message = “AB”

Probabilities: A = 0.5, B = 0.3, C = 0.2
Initial interval = [0,1)

Step 1 (A): take 0–0.5
Step 2 (B): subdivide 0–0.5 and take its 0.5–0.8 range

Final interval might be: [0.25, 0.35)
Any number here represents the message “AB”.

Advantages

- Achieves compression closer to the theoretical limit (Shannon Entropy).
- More efficient than Huffman coding for skewed or complex probability distributions.
- Supports fractional bit output.

Applications

- JPEG 2000
- H.264 / H.265 video compression
- Advanced file compressors

Conclusion: Arithmetic Coding is a highly efficient entropy coding technique that compresses data by representing entire messages as fractions, using symbol probabilities for optimal compression.

b) What is Parametric and Non-Parametric Gap Compression? Explain it in detail. [9]

In search engines and Information Retrieval systems, inverted lists store document IDs in sorted order.
Gaps between consecutive document IDs (d-gaps) are encoded to reduce index size.

There are two major types of gap compression techniques:

- Non-Parametric Gap Compression
- Parametric Gap Compression

1. Non-Parametric Gap Compression

Definition: Non-parametric methods do not assume any probability distribution of gaps.
They encode the gaps directly based on their size.

Key Techniques

1. **Unary Coding**
   Represents integer n as n ones followed by a zero.
   Example: Gap = 4 → 11110
2. **Gamma Coding**
   Represents numbers using a logarithmic prefix and binary suffix.
   Good for small to medium gaps.
3. **Delta Coding**
   More compact than Gamma coding; uses encoding of the length of the binary representation.

Characteristics

- Simple to implement
- Distribution-free → works for any type of data
- Efficient for small and moderately varying gaps
- Common in static or semi-static inverted lists

Application: Used in many IR systems where gap patterns vary unpredictably.

2. Parametric Gap Compression

Definition: Parametric methods assume that gap values follow a specific probability distribution.
The encoding is optimized using this mathematical model.

Common Assumed Distributions

- Geometric distribution (very common for document gaps)
- Zipfian distribution (frequent in web data)

Key Techniques

1. **Golomb Coding**
   Optimal when gaps follow geometric distribution.
   Uses quotient and remainder for encoding.
2. **Rice Coding**
   Special case of Golomb coding where the divisor is a power of 2 → simplifies computation.

Characteristics

- More compact than non-parametric methods when the model fits well
- Excellent for large posting lists with many small gaps
- Slightly more computation compared to gamma/delta codes

Application: Used in modern search engines (Google-style indexing) due to predictable gap behavior in web data.

Q2) a) Write a short note on: [9]

i) Decoding Performance ii) Document Deletion iii) Symbol wise Coding

i) Decoding Performance: Decoding performance refers to the speed, efficiency, and resource usage involved in reconstructing compressed data back to its original form.
It determines how quickly a system can read and interpret compressed files.
Factors Affecting Decoding Performance

- Algorithm Complexity: Simpler algorithms like Huffman decode faster; arithmetic coding is slower.
- Data Structure Efficiency: Prefix-free trees and lookup tables speed up decoding.
- Memory Access Patterns: Contiguous memory improves cache locality.
- Processor Speed & Hardware Acceleration: Modern CPUs, SIMD instructions enhance decoding throughput.

Importance

- Essential for real-time applications (video streaming, IR systems).
- Reduces latency in search engines.
- Improves user-perceived performance.

Good decoding performance ensures fast, low-latency, and efficient retrieval.

ii) Document Deletion: Document deletion refers to removing a document from a search engine's inverted index. Since inverted lists are sorted and compressed, in-place deletion is expensive. Therefore, deletion is handled using efficient indirect methods.

Approach Used

1. Invalidation List: Instead of modifying the index immediately, deleted document IDs are added to an invalidation list.
2. Filtering During Query: When returning search results, the system filters out invalid or deleted IDs.
3. Garbage Collection: Periodically, the system cleans the index by removing postings related to deleted documents and compacts the lists.

Advantages

- Fast and cheap deletion
- Maintains index consistency
- Avoids expensive real-time rebuilding

Document deletion ensures dynamic IR systems can handle continuous document updates.

iii) Symbol-wise Coding: Symbol-wise coding is a type of entropy coding where each symbol is encoded individually based on its probability. Only one symbol is processed at a time, and a separate codeword is assigned per symbol.

Features

- Used in Huffman coding, Shannon-Fano, and simple prefix codes.
- Each symbol gets a variable-length code:
    - Frequent symbol → short code
    - Rare symbol → long code

Advantages

- Easy to implement
- Fast encoding and decoding
- Suitable for real-time systems

Limitation

- Less efficient than block-based methods like arithmetic coding because it cannot encode fractional bits per symbol.

Symbol-wise coding is widely used in text compression and many standard file compression tools.

b) Explain in detail Dynamic Inverted Indices [9]

Dynamic inverted indices are indexing structures that support continuous updates such as document insertions, deletions, and modifications.
Unlike static indexes, they must handle real-time changes without full rebuilding.

Dynamic IR systems (e.g., search engines) use three main mechanisms:

1. Incremental Index Updates

Instead of updating the main index immediately:

- New documents are added to a separate auxiliary index.
- Queries search both the main index and the incremental index.
- Periodically, both indexes are merged to maintain efficiency.

Advantages

- Fast insertion
- Main index remains stable
- Efficient for large-scale, frequently updated systems

2. Contiguous Inverted Lists: A contiguous inverted list stores all postings for a term in one continuous memory block.

Characteristics

- Excellent query speed
- Good compression (smooth d-gaps)
- Great cache performance

Drawback: Poor update support (requires shifting or rebuilding when new postings arrive)

Used mainly when the index is mostly static.

3. Noncontiguous Inverted Lists: In this structure, the postings list is divided into multiple segments linked together.

Characteristics

- Very good for dynamic updates
- New segments can be appended without restructuring
- Supports fast insertions and deletions

Drawbacks

- More pointer chasing → slower queries
- Compression slightly weaker than contiguous lists

Used in large-scale web search engines.

Conclusion: Dynamic inverted indices enable modern IR systems to efficiently index constantly changing document collections, ensuring fast search performance while supporting continuous updates.

➢ **NOV/DEC 2024**

Q1) a) List out the General-Purpose Data Compression techniques and explain with suitable example. [6]

General-purpose compression techniques work for all types of data (text, images, binaries).
Common techniques include:

1. Dictionary-Based Compression (LZ Family): Builds a dictionary of repeated strings and replaces them with references.
Example:

- LZ77: Uses sliding window → (offset, length).
- Application: ZIP, PNG, GIF files.

2. Statistical Compression: Assigns short codes to frequent symbols.
Example:

- Huffman Coding: Prefix-free variable-length codes.
- Application: JPEG, MP3.

3. Arithmetic Coding: Encodes entire message as one fractional number.
Application: JPEG2000, H.264.

4. Run-Length Encoding (RLE): Stores repeated characters as (value, count).
Example:
AAAAA → A5
Application: TIFF, fax images.

5. Transform-Based Compression: Transforms data to cluster similar symbols.
Example:

- Burrows–Wheeler Transform (BWT) + Move-To-Front + Huffman.
- Application: bzip2.

6. Delta Coding: Stores differences between numbers instead of raw values.
Example: 100, 105 → store +5.

b) Describe in detail Invalidation List, Garbage Collection, Document Modifications in document deletion. [6]

1. Invalidation List

- Used in search engines to track deleted or outdated documents.
- When a document is deleted, its ID is added to the invalidation list instead of immediately removing it from the index.
- During querying, such IDs are filtered out.
- Efficient, avoids costly index updates.

2. Garbage Collection

- A periodic clean-up process to remove invalid document entries.
- Scans inverted lists, removes postings whose document IDs are in the invalidation list.
- Compacts the index, improving storage, compression, and query speed.
- Runs offline or in background.

3. Document Modifications

- Direct modification is expensive because inverted lists are sorted and compressed.
- So modification is handled as:
    1. Delete old version → add its ID to invalidation list
    2. Insert new version as a fresh document
- Ensures index consistency and efficient updates.

c) Write a short note on. [6]

i) Modeling and coding **ii) Arithmetic Coding**

i) Modeling and Coding

Compression consists of two components:

Modeling

- Discovers patterns and redundancy in data.
- Assigns probability distribution to symbols.
- Types: Fixed, Adaptive, Context-based.

Coding

- Converts model probabilities into compressed bit sequences.
- Uses prefix-free or entropy coding.
- Examples: Huffman coding, arithmetic coding.

Modeling predicts → Coding encodes.

ii) Arithmetic Coding

- A statistical compression technique that represents an entire message as a fractional interval (0,1).
- Each symbol narrows the interval based on its probability.
- Final interval uniquely encodes the data.

Advantages

- Achieves compression close to Shannon entropy.
- Better than Huffman for skewed probability distributions.

Applications: JPEG2000, H.264, text compressors.

Q2) a) Explain software Architecture of the IR system in detail. [6]

A typical Information Retrieval (IR) system architecture consists of:

1. Document Processing Component

- Crawling, tokenization, stemming, stop-word removal.
- Converts documents into indexable terms.

2. Indexer

- Builds inverted index (term → list of document IDs).
- Applies compression to postings.

3. Query Processor

- Parses user query, expands terms, removes stopwords.
- Computes term matching and ranking scores.

4. Ranking Component

- Uses scoring models such as TF-IDF, BM25, and cosine similarity.
- Orders documents based on relevance.

5. Retrieval Engine

- Fetches top-ranked documents from index.
- Handles Boolean, phrase, and proximity queries.

6. User Interface

- Displays search results, snippets, filters.

b) Explain in details: [6]

i) Decoding performance ii) Document Reordering

i) Decoding Performance: Decoding performance measures how efficiently compressed data can be decompressed.

Depends on:

- Algorithm complexity (Huffman faster than arithmetic).
- Memory access and cache behavior.
- Size of lookup tables and tree structures.

- Hardware acceleration.

Good decoding performance → low latency → fast IR results.

ii) Document Reordering: Document reordering rearranges document IDs to improve compression and retrieval.

Why?

- In inverted lists, document IDs are stored as d-gaps (differences).
- Smaller gaps → better compression.

Techniques

- URL-based reordering
- Graph clustering (BFS on web graph)
- Content-based clustering

Benefits

- Reduced index size
- Faster query execution
- Better cache locality

c) Describe data compression with Huffman coding. [6]

Huffman Coding is a lossless compression method based on assigning shorter codes to frequent symbols and longer codes to infrequent ones.

Steps

1. Count frequency of each symbol.
2. Build a min-heap of nodes with these frequencies.
3. Construct Huffman Tree by repeatedly merging two lowest-frequency nodes.
4. Assign 0 to left branch and 1 to right branch.
5. Read tree paths to generate variable-length prefix-free codes.

Example: For characters {A:5, B:9, C:12}:
A might get 110, B → 10, C → 0 (frequent symbol → shorter code).

Advantages

- Optimal among prefix-free codes.
- Simple and fast decoding.

Used in ZIP, JPEG, MP3.

NOTE: Please verify all answers before referring.