Information Storage and Retrieval

Unit 05 – Web Searching

Nov – Dec 2022

Q5) a) Write a short note on Searching the Web.

Ans. **Searching the web** refers to the process of using an online search engine (like Google, Bing, DuckDuckGo, etc.) to locate information stored on the internet. Here's a clear explanation of how it works and what it involves:

How Web Searching Works

1. **You enter a query**
    a. A query is the word(s) or question you type, such as *"best restaurants near me"* or *"how do solar panels work?"*
2. **The search engine processes it**
    a. Search engines use software called **web crawlers** (or spiders) that scan the internet and store pages in a large **index** (like a giant library catalogue).
3. **The search engine matches results**
    a. It looks through its index and tries to find the pages most relevant to your query.
4. **Results are ranked**
    a. A search algorithm decides the order of results based on factors like:
        i. Relevance to the query
        ii. Quality of content
        iii. Website reliability
        iv. Popularity and links from other sites
5. **Results are displayed**
    a. You receive a list of links called a **Search Engine Results Page (SERP)**, which may include:
        i. Web pages
        ii. Images
        iii. Videos

iv. Maps
v. News
vi. Ads
vii. Featured snippets or instant answers

Why Web Searching is Useful

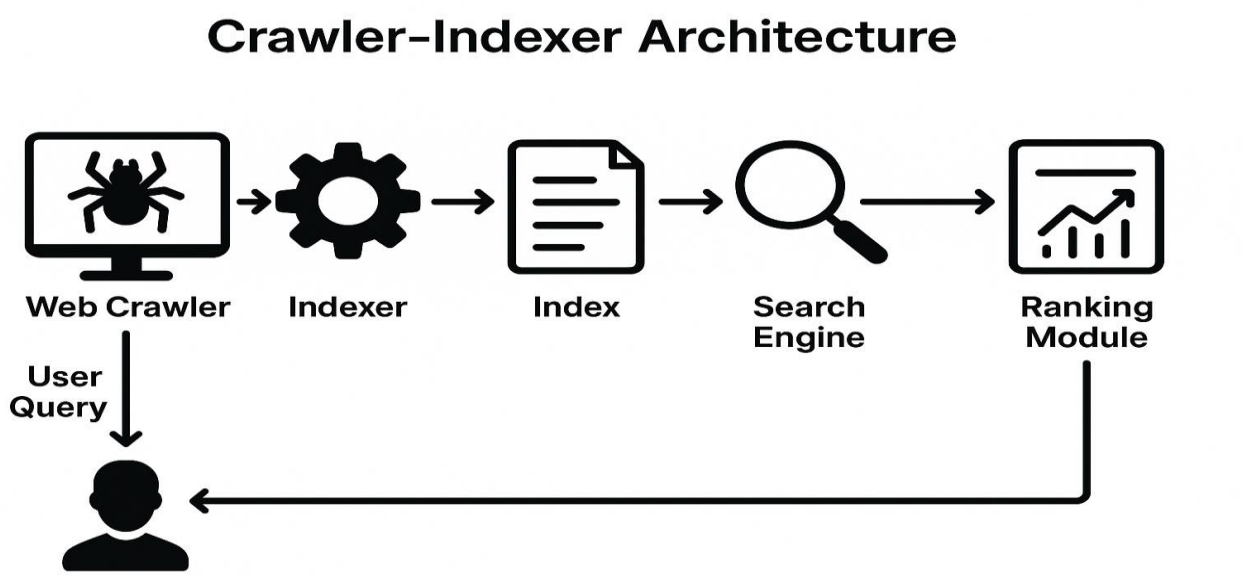
- Quickly find information on almost any topic
- Compare products or services
- Discover answers, guides, or tutorials
- Stay current with news and events
- Research for school, work, or personal projects

Tips for Better Searching

Tip	Example
Use specific keywords	*symptoms of vitamin d deficiency*
Use quotes for exact phrases	*"climate change effects"*
Use minus sign to exclude words	*jaguar -car*
Ask questions	*How to change a tire?*
Use filters	date, location, image type, etc.

Q5) b) Explain Crawler-Indexer Architecture with neat diagram.

Ans. The **Crawler–Indexer Architecture** is a common structure used by search engines (like Google or Bing) to collect, organize, and make web content searchable. It describes how search engines gather information and prepare it so users can quickly find relevant pages when they search.



Main Components

1. Web Crawler (Spider or Bot)

A **web crawler** is an automated program that travels across the internet by following links from page to page.

Key responsibilities:

- Visit webpages and download their content
- Discover new pages by following hyperlinks
- Detect updated or deleted pages
- Respect rules set by websites (e.g., in `robots.txt` files)
- Decide which pages to crawl first based on priority and frequency

Think of it like a robot librarian traveling to every website and copying the pages.

2. Indexer

After a crawler collects webpage data, it sends it to the **indexer**, which processes and organizes the information into a structured database called an **index**.

Key responsibilities:

- Parse (read) text, metadata, images, links, and structure of webpages
- Remove duplicates or spam
- Analyze keywords and their context
- Store important information in the index so it can be quickly retrieved
- Map words to locations in documents for fast searching

This is like a librarian organizing books in a catalog so people can find them quickly.

How the Two Parts Work Together

Workflow

1. **Crawler** visits/webpage and downloads content
2. Sends raw content to **Indexer**
3. **Indexer** analyzes and stores the content in an **index**
4. When a user enters a query in a search engine, the **search engine queries the index** (not the live web!)
5. Rankings and algorithms determine the best results to display

```
User Query → Search Engine → Index → Search Results
        ↑                      |
     Ranking                Indexer
        ↑                      |
      Results ← Search Engine ← Crawler
```

Why This Architecture is Necessary

Reason	Benefit
Web is huge and constantly changing	Continuous crawling keeps search results fresh
Searching the live web would be too slow	Index allows lightning-fast answers
Need relevance and ranking	Indexer extracts meaningful signals

Real World Example

If you searched for **“electric car benefits”**, the search engine wouldn’t scan the entire internet in real time. Instead, it investigates its **pre-built index**, finds relevant documents, ranks them, and shows results in milliseconds.

In Summary

Crawler	Indexer
Collects and downloads content	Processes, analyzes, and stores content
Follows links to discover web pages	Builds searchable index
Visits sites repeatedly	Removes duplicates and spam

Q5) c) What is role of crawler in web searching? Explain the strategies used by the web crawler.

Ans. **Role of Crawler in Web Searching**

A **web crawler** (also called a spider, bot, or robot) is a software program used by search engines to automatically explore and collect information from webpages across the internet. It plays a **foundational role** in web searching by discovering new and updated content so that it can be indexed and made searchable for users.

Main Roles of a Web Crawler

1. **Discover Web Pages**
    a. Crawlers visit web pages and follow hyperlinks to find additional pages.
2. **Download and Collect Data**
    a. They fetch the content (HTML, images, metadata, etc.) from the web pages.
3. **Update Index**
    a. Crawlers send collected data to the **indexer**, which analyzes and stores it in the search engine index.
4. **Monitor Website Changes**

    a. Crawlers frequently revisit pages to detect updates, deletions, or newly added links.
5. **Respect Site Rules and Policies**
    a. Crawlers follow website guidelines defined in `robots.txt` files and avoid overloading servers.

Strategies Used by Web Crawlers

To efficiently crawl billions of web pages, web crawlers use intelligent strategies. The main ones include:

1. Breadth-First Search (BFS)

- Crawls pages level by level, starting from seed URLs.
- Good for collecting high-level structure of the web early.

2. Depth-First Search (DFS)

- Follows a link path deeply before backtracking.
- Useful for focused site crawling but may go too deep into a single website.

3. Priority-Based Crawling / Best-First Crawling

- Pages are assigned priority scores based on factors such as:
    - Page rank
    - Number of backlinks
    - Update frequency
    - Content importance
- Crawler visits higher priority pages first.

4. Focused Crawling

- Crawls only pages relevant to a specific topic or domain.
- Used for thematic search engines (e.g., medical or academic portals).

5. Incremental Crawling

- Periodically revisits pages to update only changed content.
- Reduces bandwidth and improves efficiency.

6. Distributed Crawling

- Multiple crawler processes run on different machines.
- Speeds up crawling and avoids duplication through coordination.

7. Politeness Policy / Rate Control

- Controls request frequency to avoid overwhelming a server.
- Respects crawl-delay directives in `robots.txt`.

8. URL Scheduling

- Maintains a **URL Frontier** (queue of URLs to crawl).
- Decides optimal order based on freshness, priority, or domain rules.

In Summary

Role of Crawler	Strategies Used
Discover and collect web pages	BFS, DFS, Best-First

Feed data to indexer	Priority / Focused Crawling
Update index with fresh content	Incremental Crawling
Avoid overloading servers	Politeness & scheduling
Explore web through links	Distributed crawling

Conclusion

The web crawler is essential in the architecture of search engines. Without crawlers, search engines would not have up-to-date or comprehensive content, making it impossible to deliver relevant results to user queries.

Q6) a) What is hyperlink? Explain structure of hyperlink and also explain searching using hyperlinks.

Ans. **What is a Hyperlink?**

A **hyperlink** (or simply *link*) is a reference in a webpage that users can click to move from one document or resource to another. Hyperlinks connect web pages and allow navigation across the World Wide Web. They can link to:

- Another webpage
- A different website
- A specific section within the same page
- Files (PDF, image, video, etc.)
- Email addresses

Hyperlinks are fundamental to the web because they create the network structure that allows users and crawlers to browse information easily.

Structure of a Hyperlink

A hyperlink is created using the `<a>` (anchor) tag in HTML.

General Syntax

```
<a href="URL">Link Text</a>
```

Components

Component	Meaning
`<a>`	Anchor tag, used to define a hyperlink
`href` attribute	Specifies the destination URL (where link points)
`URL`	Address of the target webpage or resource
`Link Text`	Visible text that users click

Example

```
<a href="https://www.wikipedia.org">Visit Wikipedia</a>
```

Internal Link Example

```
<a href="#section2">Go to Section 2</a>
```

Email Link Example

```
<a href="mailto:info@example.com">Send Email</a>
```

Searching Using Hyperlinks

Hyperlinks play an important role in how search engines **crawl and search** the web. When a search engine crawler visits a webpage, it:

1. **Reads the page content**
2. **Extracts hyperlinks present on the page**
3. **Adds the discovered URLs to the crawling queue (URL Frontier)**
4. **Follows hyperlinks to find new pages and resources**

5. **Builds a web of connected pages**

This process allows search engines to discover billions of pages without manually listing them.

Importance of Hyperlinks in Searching

Function	Explanation
Page Discovery	Crawlers follow links to find new or updated content
Ranking (Popularity)	Pages with more inbound links are considered more important (basis of Google's PageRank)
Topic Relevance	Link text (anchor text) is used to understand what the target page is about
Navigation	Links help users search and browse related content

Example: Searching Using Hyperlinks Workflow

```
User query → Search engine consults index → Retrieves relevant pages
                                    ↓
Crawler explores hyperlinks → Discovers related pages
                                    ↓
Ranking based on link popularity and content relevance
```

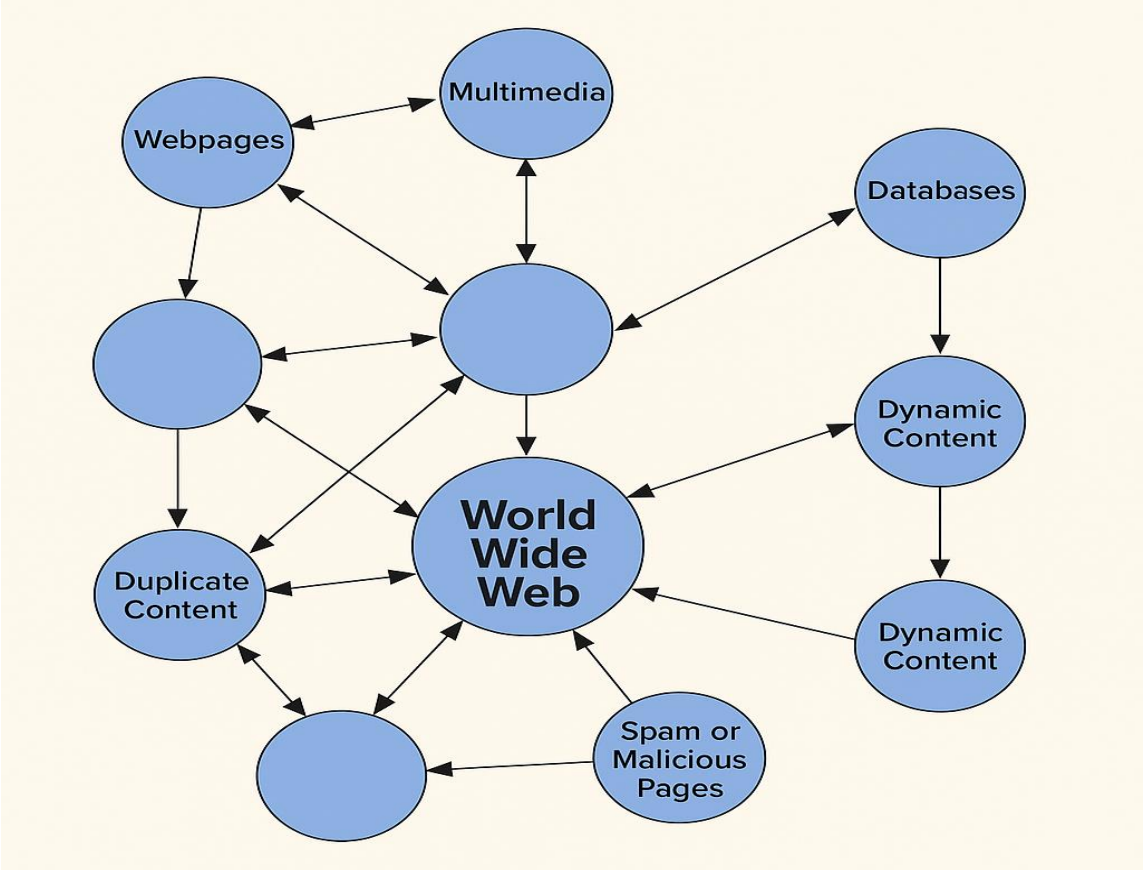
Summary

Hyperlink	Structure	Role in Searching
A clickable reference on the web	`<a href="URL">Text</a>`	Helps search engines discover and rank pages
Connects web pages	Uses URL & anchor text	Crawlers follow links to build web index

Q6) b) Write a note on characterizing the web.

Ans. **Characterizing the Web**

The **World Wide Web** is a massive, dynamic, and continuously expanding collection of interconnected documents and resources linked through hyperlinks. Characterizing the web means understanding its structure, properties, and behavior to support tasks like searching, crawling, indexing, and ranking.



Key Characteristics of the Web

1. Huge Size and Rapid Growth

- The web contains **billions of webpages**, and new pages are added every second.
- Information is created in diverse formats — text, images, audio, video, databases, etc.
- Because of this enormous size, it is **impossible to crawl and index the entire web completely**.

2. Dynamic and Continuously Changing

- Web content changes frequently (pages updated, deleted, or moved).
- Dynamic web technologies (like CMS, APIs, social media) generate content in real time.
- Search engines must **continuously re-crawl** pages to keep indexes fresh.

3. Distributed and Heterogeneous

- The web is spread across servers all over the world.
- No central authority controls the entire web.
- Content uses different technologies and formats (HTML, XML, JSON, multimedia, scripts).
- Quality varies widely — from official research papers to blogs and spam sites.

4. Highly Interlinked Structure

- Webpages connect through **hyperlinks**, forming a huge directed graph.
- These connections help users and search engines navigate.
- Link structure is used for **ranking algorithms** like PageRank to measure importance.

5. Contains Semi-Structured Data

- Web documents usually have HTML tags, headings, hyperlinks, metadata, etc.
- This makes information extraction possible but not easy because formatting is inconsistent.

6. Duplicate and Redundant Content

- Many sites replicate content (mirrors, copies, shared news articles).

- Crawlers must detect **duplication** to reduce storage and increase efficiency.

7. Accessibility and Availability Variability

- Some content is freely accessible, while others are behind paywalls or login pages.
- Part of the web exists as the **Deep Web**, which search engines cannot fully access.

8. Contains Spam and Malicious Pages

- The web includes harmful content like phishing, malware, and SEO spam.
- Search engines use filters to identify and remove low-quality or dangerous pages.

Why Characterizing the Web is Important

Purpose	Benefit
Efficient crawling and indexing	Saves time and resources
Better ranking and search results	Improves quality for users
Detect changes and new pages	Keeps index up to date
Handle duplicates and spam	Improves reliability
Understand link structure	Enhances relevance and connectivity

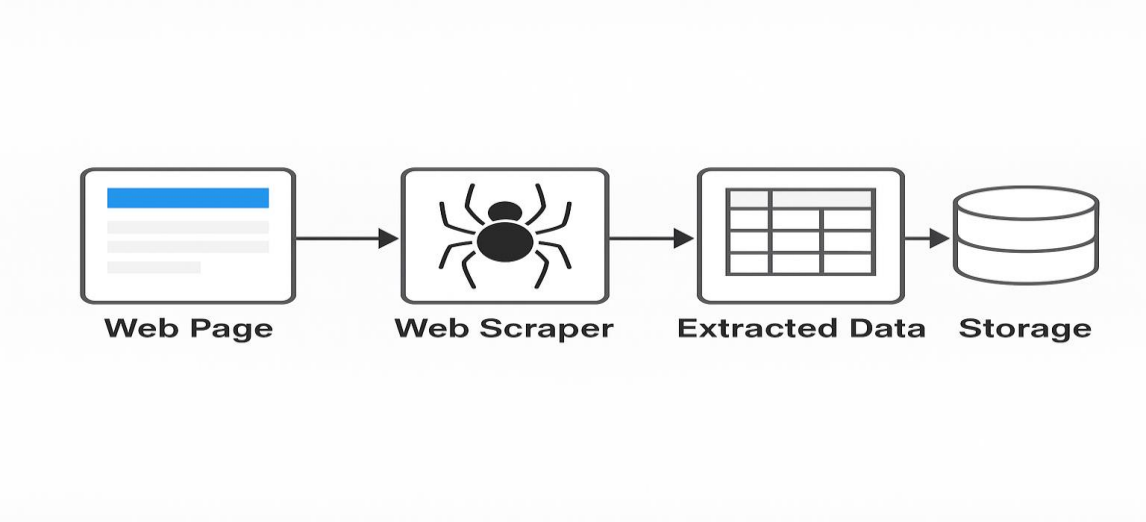
Conclusion

Characterizing the web helps search engines understand how to **organize, navigate, and search** the vast and ever-changing information space. Because the web is large, dynamic, decentralized, and complex, search engines rely on intelligent crawling, indexing, and ranking techniques to provide meaningful and accurate results.

Q6) c) Explain Web Scrapping with suitable example.

Ans. **Web Scraping**

Web scraping is the process of automatically extracting useful information from websites. Instead of manually copying data from web pages, automated tools or scripts collect structured information (such as product prices, news headlines, reviews, or statistics) and store it in a database, spreadsheet, or file.



Web scraping is widely used for:

- Price comparison (e-commerce)
- Market and business analytics
- Academic research or data mining
- Monitoring news or trends
- Collecting contact or product information

How Web Scraping Works

Typical steps in a scraping process:

1. **Send a request** to a webpage URL
2. **Download the HTML content**
3. **Parse the HTML** to locate and extract needed data
4. **Store/Save the extracted data** to Excel, CSV, or database

Example of Web Scraping Using Python

The following example extracts the **titles of news headlines** from a website.

Python Example Using `requests` and `BeautifulSoup`

```
import requests
from bs4 import BeautifulSoup

# Step 1: Send request to webpage
url = "https://example.com/news"
response = requests.get(url)

# Step 2: Parse HTML content
soup = BeautifulSoup(response.text, "html.parser")

# Step 3: Extract headline text
headlines = soup.find_all("h2", class_="headline")

# Step 4: Print results
for h in headlines:
    print(h.text)
```

Explanation

- `requests.get()` retrieves the HTML content of the webpage
- `BeautifulSoup` parses the HTML structure
- `find_all()` searches for tags (here `<h2 class="headline">`)
- The extracted text is displayed or stored

Real-Life Example

Scraping Product Prices

A script could collect prices from multiple shopping websites like:

```
Product Name | Price | Website
Phone X      | $699  | Amazon
Phone X      | $720  | Flipkart
Phone X      | $680  | Walmart
```

This can be used to find the cheapest product.

Ethical & Legal Considerations

- Always check a site's **robots.txt** file to see allowed scraping rules
- Do not overload servers with too many requests
- Respect copyright and privacy policies
- Only scrape publicly available data

Summary

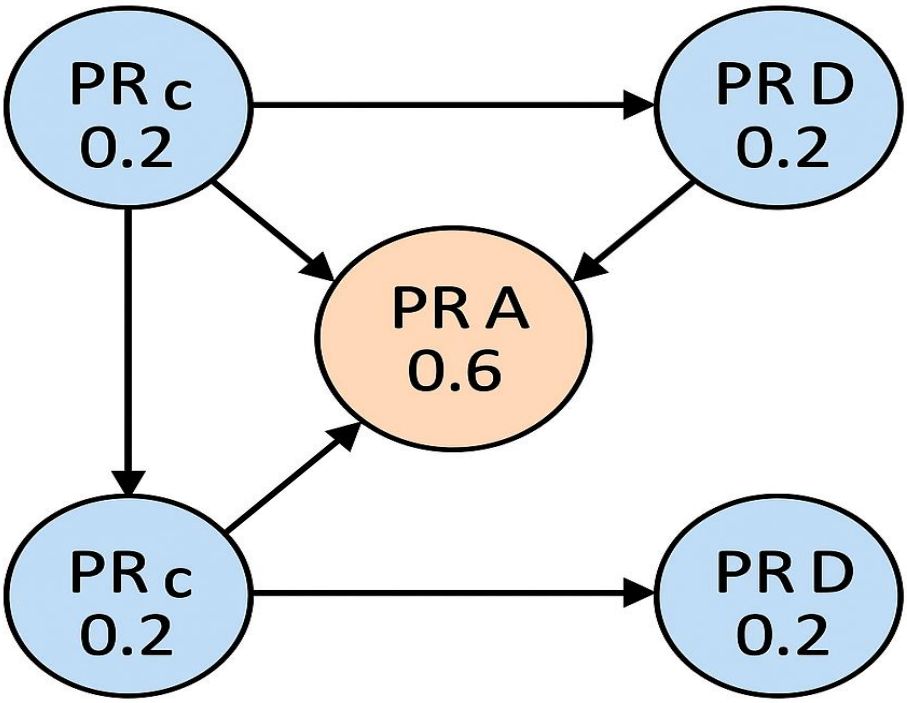
Feature	Explanation
Definition	Automated extraction of data from websites
Why used	Price comparison, research, marketing, analytics
Example	Scraping news titles, product prices
Tools	Python, BeautifulSoup, Scrapy, Selenium

May – Jun 2023

Q5) a) What is page ranking? Explain role of ranking in web searching with algorithm.

Ans. **What is Page Ranking?**

Page Ranking



Page ranking is the process of determining the **importance, quality, and relevance** of web pages so that search engines can display the **most useful results** at the top of the Search Engine Results Page (SERP). Since millions of pages may contain similar information, page ranking is essential to decide which pages appear first when a user enters a query.

The ranking procedure uses algorithms that evaluate factors such as:

- Relevance to the search query
- Number and quality of incoming links
- Content quality and freshness
- User engagement signals (click-through rate, bounce rate)
- Page structure and keywords

Role of Ranking in Web Searching

Ranking plays a central role in search engines because:

Role	Explanation
Improves search accuracy	Users quickly find relevant information
Organizes large web data	Prioritizes useful pages over irrelevant ones
Measures page importance	Uses link structure and popularity
Enhances user experience	Top-ranked pages increase search satisfaction
Prevents spam and low-quality results	Filters out manipulated or irrelevant pages

Without ranking, search results would appear in **random** order, making searching slow and inefficient.

Page Ranking Algorithm

One of the most popular ranking algorithms is **Google's PageRank Algorithm** (developed by Larry Page and Sergey Brin).

Concept

PageRank measures the **importance of a web page based on the number and quality of links** pointing to it. A page linked by many important pages is considered more important.

Basic PageRank Formula

$$PR(A) = (1 - d) + d \times (PR(T_1)/C(T_1) + PR(T_2)/C(T_2) + \ldots + PR(T_n)/C(T_n))$$

Where:

Symbol	Meaning
PR(A)	PageRank of page A
PR(T_1)	PageRank of a page pointing to A

$C(T_1)$	Number of outbound links from page T_1
d	Damping factor (usually 0.85)

Steps in PageRank Algorithm

1. **Assign initial rank to all pages** (usually 1)
2. **Calculate rank based on links** pointing to each page
3. **Distribute rank equally among outgoing links**
4. **Apply damping factor** to simulate random surfing
5. **Iterate until convergence** (stable rank values)

Example

Assume 3 pages **A, B, C**:

- B → A
- C → A, B
- A → C

Each page starts at PR = 1. Using iteration and formula, the PageRank eventually stabilizes, giving **higher rank to pages with more inbound links**.

Why PageRank Matters

- Helps search engines decide **which pages appear first**
- Evaluates page **value based on human-like recommendations**
- Maintains **fair and logical ranking** across billions of pages

Conclusion

Page ranking is a critical component of web searching that ensures users receive **accurate, useful, and relevant results quickly**. Algorithms like **PageRank** analyze link structure and page importance to sort results effectively.

Q5) b) Write a note on i) Request module and beautiful soup library. ii) Web scraping

Ans. Below is a clear explanation of **Requests module** and **BeautifulSoup library**, commonly used in web scraping in Python.

1. Requests Module

The **Requests module** is a popular Python library used to **send HTTP requests** to web servers and retrieve web content such as HTML pages, JSON data, files, or API responses. It is simple, user-friendly, and handles connections efficiently.

Purpose of the Requests Module

- To send requests to a webpage (GET, POST, PUT, DELETE, etc.)
- To download the content of webpages
- To access APIs securely using headers, cookies, and authentication
- To handle response codes and errors

Features

Feature	Description
HTTP requests	GET, POST, PUT, DELETE etc.
Retrieve HTML	`response.text` or `response.content`
Check server status	`response.status_code`
Send headers & parameters	Helps avoid blocking
Handle sessions & cookies	Useful for login-based scraping

Example Using Requests

```
import requests

url = "https://example.com"
response = requests.get(url)

print(response.status_code)    # prints HTTP status code
print(response.text)           # prints HTML content of the page
```

2. Beautiful Soup Library

BeautifulSoup is a Python library used to **parse HTML and XML documents**. Once a webpage is downloaded using Requests, BeautifulSoup helps navigate through the HTML structure and **extract specific data** like headings, links, tables, images, etc.

Purpose of BeautifulSoup

- Parse HTML or XML pages
- Locate and extract information from tags
- Clean and format extracted data
- Search by tags, classes, ids, attributes, and text

Key Features

Feature	Description
Parse HTML/XML	Creates a structured parse tree
Find tags	`find()` or `find_all()`
Extract text or attributes	`.text`, `.get('href')`
Supports CSS selectors	`select("div.classname")`

Example Using BeautifulSoup

```
from bs4 import BeautifulSoup
import requests
```

```
url = "https://example.com"
response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

# Extract all headings
headings = soup.find_all("h2")

for h in headings:
    print(h.text)
```

How Requests and BeautifulSoup Work Together

Step	Action
1	Use **Requests** to download the webpage
2	Use **BeautifulSoup** to parse and extract data

Simple Workflow Diagram

```
Web Page → Requests (downloads HTML)
                ↓
          BeautifulSoup (parses & extracts data)
                ↓
             Structured Data (CSV / Database)
```

Conclusion

- **Requests** handles communication with the website and retrieves raw HTML.
- **BeautifulSoup** parses that HTML and extracts meaningful structured content.

Together, they form a powerful combination for **web scraping, automation, and data mining**.

Q6) a) Explain difference between centralized and distributed architecture of search engine.

Ans.

Comparison Table: Centralized vs Distributed Architecture of Search Engines

Parameter	Centralized Architecture	Distributed Architecture
Definition	All search engine components such as crawler, indexer, and ranking run on a single main server	Components are spread across multiple servers working together in parallel
Scalability	Low – cannot handle very large amount of data	Very high – easily scalable across many systems
Performance	Slower when traffic increases	High performance due to parallel processing
Data Storage	Stored in one central repository	Distributed across multiple storage systems
Fault Tolerance	Low – failure of central system stops service	High – failure of one server does not stop the system
Crawling Process	Performed by a single centralized crawler	Performed by multiple distributed crawlers
Indexing	Single index stored in one place	Index partitioned across many servers
Query Processing	Performed by one processor/server	Queries processed in parallel across multiple servers
Load Handling	Poor under high load	Excellent due to load balancing
Response Time	Slower as data grows	Faster and more efficient
Synchronization	Simple, little coordination required	Complex; requires coordination and consistency
Cost	Low hardware and maintenance cost	High cost due to multiple servers and infrastructure
Maintenance	Easy to monitor and manage	More difficult to maintain
Use Cases	Small and medium-scale search systems	Large commercial search engines

Examples	Organizational website search, early search engines	Google, Bing, Yahoo

Q6) b) What is web searching? Define and explain the following terms with respect to web searching i) Crawling ii) Web directories

Ans.

Web searching is the process of using a search engine (such as Google, Bing, or Yahoo) to locate information on the World Wide Web. Users enter keywords or queries, and the search engine returns a list of relevant web pages from its index. Web searching helps users quickly find useful information from billions of available pages.

Terms with Respect to Web Searching

i) Crawling

Crawling is the process performed by automated programs called **web crawlers** or **spiders**, which browse the internet systematically to discover and collect web pages.

How Crawling Works

- The crawler starts with a list of known URLs called **seed URLs**
- It downloads web pages and extracts hyperlinks from them
- Newly discovered links are added to a **URL frontier (queue)**
- The pages collected are sent to the **indexer** for processing and storage

Purpose of Crawling

- To discover new web pages
- To detect updated or deleted content
- To keep the search engine index fresh and accurate

Example

Googlebot is Google's web crawler that continuously scans and updates the web index.

ii) Web Directories

Web directories are structured collections of websites organized into categories and subcategories by human editors rather than by automated search engines.

Characteristics

- Websites are grouped by topics such as Education, Health, Technology, Sports, etc.
- Users can browse through categories to find information rather than typing search queries
- Listings are reviewed and approved manually

Purpose of Web Directories

- Provide organized, topic-wise access to websites
- Useful for research and browsing specific information domains
- Helps find high-quality, curated resources

Examples

Web Directory	Description
Open Directory Project (DMOZ)	One of the largest human-edited web directories
Yahoo Directory (historical)	Early popular directory before modern search engines

Summary Table

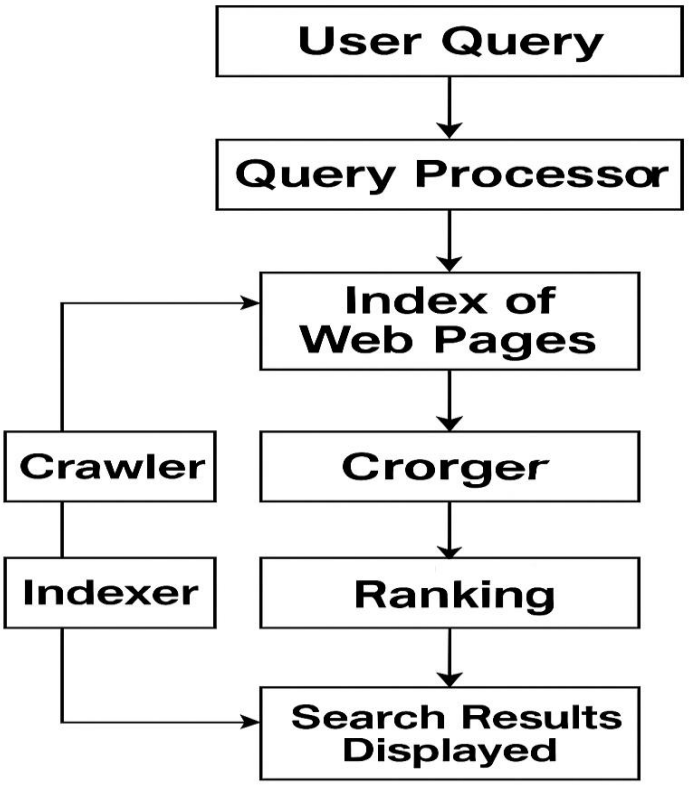
Term	Definition	Purpose
Crawling	Automatic process of discovering webpages using a crawler bot	Build and update search engine index

Web Directories	Manually organized lists of websites grouped by topics	Provide structured browsing of categorized sites

May - Jun 2024

Q5) a) Explanation of search engine mechanism- [4 M] Diagram- [2 M].

Ans. **Explanation of Search Engine Mechanism**



Search Engine Mechanism

A **search engine mechanism** refers to the internal process through which a search engine discovers, analyses, organizes, and retrieves information from the web to provide relevant results to users. When a user types a query into a search engine, several automated components work together to deliver fast and accurate search results.

Main Components of Search Engine Mechanism

1. Crawling

- Search engines use automated programs called **crawlers** or **spiders** to browse the web.
- Crawlers visit webpages, follow hyperlinks, and collect page data.
- They continuously search for new, updated, or deleted pages to keep the index current.

2. Indexing

- The data collected by the crawler is processed by an **indexer**.
- The indexer analyses content (text, keywords, tags, images, links) and stores it in a structured database called an **index**.
- The index is like an organized library catalog that allows fast searching.

Indexing tasks include:

- Tokenizing and parsing text
- Removing stop words and duplicates
- Extracting metadata and ranking signals
- Storing content in an **inverted index**

3. Query Processing

- When a user enters a search query, the search engine interprets it.
- It may apply spell correction, autosuggestions, and query expansion (related terms or synonyms).
- Converts the query into internal format for processing.

4. Ranking

- The search engine compares the query with documents in the index.
- It evaluates relevance and importance of pages using **ranking algorithms** such as PageRank, TF-IDF, HITS, etc.
- Factors considered include keyword matching, backlink structure, content quality, page freshness, user engagement, etc.

5. Retrieval and Display of Results

- Based on ranking scores, the search engine retrieves the most relevant pages.
- The results are shown on the **Search Engine Results Page (SERP)**.
- SERP may include:
    - Organic results
    - Advertisements
    - Images, videos, news
    - Featured snippets or knowledge panels

Search Engine Process Flow (Simplified)

```
User Query
     ↓
Query Processor
     ↓
Index of Web Pages  ←  Indexer  ←  Crawler
     ↓
Ranking Algorithm
     ↓
Search Results Displayed (SERP)
```

Key Objectives of Search Engine Mechanism

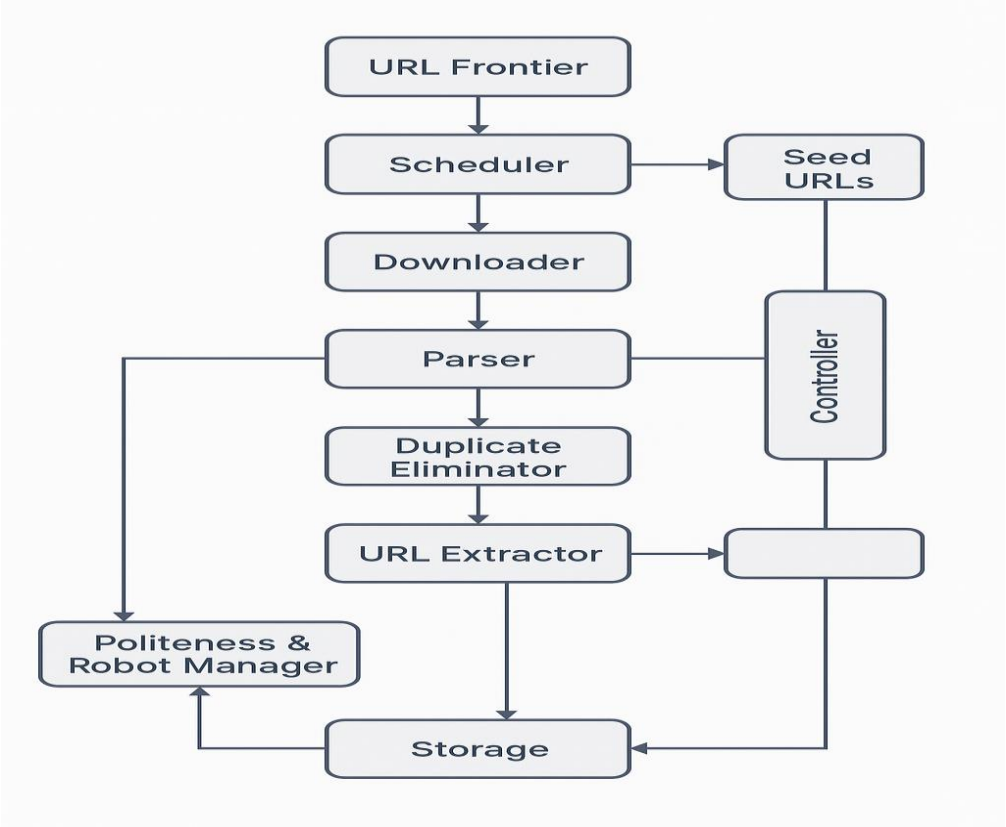
Objective	Description
Speed	Provide results in milliseconds
Relevance	Show most useful and accurate content
Coverage	Index as much of the web as possible
Freshness	Keep results up to date
Quality	Block spam and low-value pages

Conclusion

The search engine mechanism is a complex multi-stage system involving **crawling**, **indexing**, **query processing**, **ranking**, and **result presentation**. All components work together to provide fast and meaningful search results from billions of web pages.

Q6) c) What are the different components of Web crawler.

Ans.



Different Components of a Web Crawler

A **Web Crawler** (also known as a spider or bot) is a software system that automatically browses the internet to discover and download web pages for indexing by search engines. To perform this task efficiently, a crawler is made up of several key components:

1. URL Frontier (Queue)

- Stores the list of URLs to be crawled.
- Manages the scheduling and prioritization of pages.
- Uses strategies like BFS, DFS, and priority-based crawling.

2. Seed URLs

- Initial set of starting URLs from where the crawling begins.
- Chosen based on domain importance or category.

3. Downloader / Fetcher

- Sends **HTTP requests** to retrieve web pages from servers.
- Handles response codes, timeouts, cookies, headers and retries.
- Uses politeness policies to avoid overloading websites.

4. Parser

- Analyzes downloaded pages and extracts useful data.
- Identifies content sections, metadata, and hyperlinks.
- Removes HTML tags, scripts, and unnecessary objects.

5. URL Extractor (Link Extractor)

- Extracts hyperlinks from the downloaded pages.
- Adds newly discovered URLs to the frontier queue if not already visited.

6. Duplicate Eliminator / URL Filter

- Avoids crawling the same page multiple times.
- Uses hash tables or Bloom filters to check previously visited URLs.

7. Storage System

- Stores fetched pages, parsed content, and metadata.
- Used later by the **indexer** for search engine indexing.

8. Scheduler

- Decides the order and timing of crawling tasks.
- Considers crawling frequency, priority, domain rules, and freshness.

9. Politeness & Robot Manager

- Ensures crawler respects `robots.txt` restrictions and crawl-delay rules.
- Prevents excessive requests to the same server.

10. Controller / Management Module

- Coordinates all components.
- Maintains logs and monitors crawling progress and errors.

Summary Table

Component	Description
URL Frontier	Stores and manages URLs to be crawled
Seed URLs	Starting pages for crawling
Downloader	Fetches pages using HTTP requests
Parser	Extracts content and structure from HTML
URL Extractor	Collects hyperlinks from webpages
Duplicate Eliminator	Avoids re-crawling already visited pages
Storage	Saves downloaded data for indexing
Scheduler	Controls crawl priority and timing
Politeness Manager	Respects robots.txt and avoids overload
Controller	Oversees entire crawling system

May – Jun 2025

Q6) c) Discuss the challenges faced in wed searching.

Ans.

Challenges Faced in Web Searching

Web searching is a complex task due to the massive and dynamic nature of the World Wide Web. Search engines face several challenges while trying to provide fast, relevant, and accurate results to users.

1. Huge Size and Growth of the Web

- The web contains **billions of pages** and grows rapidly every second.
- It is impossible to crawl and index the entire web fully.
- Managing storage and fast retrieval becomes difficult.

2. Dynamically Changing Content

- Web pages are frequently updated, moved, or deleted.
- Search engines must continuously re-crawl to keep indexes fresh.
- Maintaining the latest version of data is challenging.

3. Heterogeneity of Data

- Web content comes in various formats: text, images, videos, scripts, databases, PDFs, etc.
- Data structure is inconsistent, and different pages use different layouts.
- Extracting meaningful information becomes difficult.

4. Duplicate and Redundant Content

- Many sites host the same or similar information.
- Duplicate content wastes storage and slows indexing.
- Detecting and removing redundancy is necessary.

5. Spam and Malicious Pages

- The web contains harmful content created to manipulate rankings (SEO spam, phishing, malware).

- Search engines must filter spam without removing legitimate pages.

6. Ranking and Relevance

- Determining the most relevant pages for a query is difficult.
- Requires complex ranking algorithms and machine-learning models.
- Understanding user intent and context is challenging.

7. Lack of Structure on Web Pages

- Most webpages are **semi-structured**, making information extraction complex.
- Hard to accurately identify important content vs. advertisements or noise.

8. Language and Multilingual Issues

- Web content exists in many languages and formats.
- Requires language detection, translation, and linguistic analysis.

9. Scalability and Performance

- Must process millions of queries per second.
- Needs distributed systems and large-scale parallel computing.

10. Access Restrictions

- Many pages are inside **Deep Web** (behind login, paywalls, or dynamic content).
- Crawlers cannot access content requiring interaction or authentication.

11. Privacy and Legal Constraints

- Search engines must obey laws and policies like GDPR.
- Must respect robots.txt and copyright issues.

Summary Table

Challenge	Description
Web size	Billions of pages, continuous growth
Dynamic content	Frequent updates require constant crawling
Heterogeneous data	Different formats and structures
Duplication	Repeated information wastes resources
Spam pages	Need detection and filtering
Ranking	Hard to match relevance & user intent
Multilingual content	Requires translation & NLP
Deep web	Limited access to restricted pages
Scalability	Must handle huge workload
Legal issues	Privacy and permissions constraints

Conclusion

Web searching involves many challenges related to scale, content variety, relevance, and security. Search engines use advanced algorithms, distributed architectures, and continuous updating strategies to overcome these difficulties and deliver fast, accurate, and meaningful results.

1. Web Characteristics

The **World Wide Web (WWW)** is a massive, dynamic, and decentralized information system. Its main characteristics are:

Characteristic	Explanation
Huge size	Billions of documents distributed globally
Heterogeneous	Content in multiple formats: text, images, video, audio, PDF, XML, JSON
Dynamic	Pages frequently updated, added, deleted
Semi-structured	Contains HTML tags but lacks uniform schema
Hyperlinked network	Pages connected via hyperlinks forming a graph
Redundancy & duplication	Same information exists in many locations
Contains spam & unreliable data	Not all sources are trustworthy
Scalability & high growth rate	Expands rapidly and continuously

2. User Interfaces

User interfaces allow users to interact with search engines and access information.

Features of Search Engine User Interface

- Search box for typing queries
- Autosuggestions and auto-correction
- Filters and advanced search options
- Tabs for Images, Videos, News, Maps, Books, etc.
- Voice search & multilingual support

Goal

To help users retrieve relevant information **quickly**, **accurately**, and **easily**.

3. Indices

An **index** is a structured data repository built by search engines to store processed information from crawled web pages.

Functions of Index

- Organize web content for fast search
- Store keywords, URLs, metadata, ranking values
- Enable efficient matching between a query and documents

Inverted Index

Most search engines use **inverted index**, which maps words → list of pages containing those words.

4. Browsing

Browsing refers to **navigating through web pages** by following links rather than searching with keywords.

Types of Browsing

- **Link-based browsing** (via hyperlinks)
- **Category-based browsing** (through directories)
- **Navigation-based browsing** (menus and site structure)

Example

Wikipedia browsing through linked topics.

5. Meta-searchers

A **Meta-search engine** does not maintain its own index. Instead, it forwards user queries to multiple search engines, collects results, merges them, and removes duplicates.

Features

- Combines output from several engines
- Provides broader coverage and comparison
- No web crawling or indexing required

Examples

Meta-search Engine	Description
Dogpile	Fetches from Google, Yahoo, Yandex
Metacrawler	Early meta search tool
Ixquick / Startpage	Privacy-focused

6. Trends and Research Issues in Web Searching

Modern web search systems face new challenges and opportunities:

Emerging Trends

- AI-based search & semantic search
- Personalized search results
- Voice & conversational search (Chatbots)
- Visual search (images instead of text)
- Mobile-first search indexing
- Real-time indexing & news search

Research Issues

Issue	Explanation
Understanding user intent	Difficult due to ambiguous queries
Handling dynamic data	Frequent updates require new techniques

Spam detection	Distinguishing quality vs fake pages
Multilingual search	Automatic translation & global access
Ranking fairness	Ethical and bias-free ranking
Privacy and regulation	GDPR, data protection, consent

7. Python for Web Scraping

Python is widely used for web scraping because of:

- Simple syntax
- Strong community support
- Libraries for HTTP requests, parsing, automation

Common Python Libraries for Scraping

Library	Purpose
`requests`	Download web pages
`BeautifulSoup`	Parse HTML
`Scrapy`	Large-scale crawling & scraping framework
`Selenium`	Automating dynamic web pages
`pandas`	Storing and processing data

8. Requests Module

The **Requests** library allows Python to send HTTP requests to servers.

Example

```
import requests
response = requests.get("https://example.com")
print(response.text)
```

Functions

- Send GET/POST requests

- Read status codes
- Manage cookies, headers, and sessions

9. HTML Parsing

HTML parsing extracts meaningful content from HTML documents.

BeautifulSoup Library Example

```
from bs4 import BeautifulSoup
import requests

html = requests.get("https://example.com").text
soup = BeautifulSoup(html, "html.parser")

headings = soup.find_all("h2")
for h in headings:
    print(h.text)
```

Purpose of Parsing

- Extract headings, links, tables, prices, news, etc.
- Convert unstructured HTML → structured data

Conclusion

Web searching involves complex processes such as crawling, indexing, ranking, and retrieving information. Search engines continuously evolve using AI, machine learning, and advanced algorithms to provide accurate and fast results. Python enables powerful web scraping capabilities using tools like Requests and BeautifulSoup to extract and analyze online data.