# Graph Clustering

Defination: Cutting a graph into pieces where each piece is a cluster.

Objective: Vertices in the same cluster are well connected and vertices in different clusters are not necessarily well connected.

Applications :→

1.) Social Network Analysis → Identify communities in a social network
2.) Search Engines → Find hubs and authoritative web pages on the web, which is treated as a graph.

## Cuts and Clusters

→ graph $G = (V, E)$ — vertices, Edges

→ cut $C = (S, T)$ is a partitioning of $V$

i.e $V = S \cup T$ and $S \cap T = \phi$

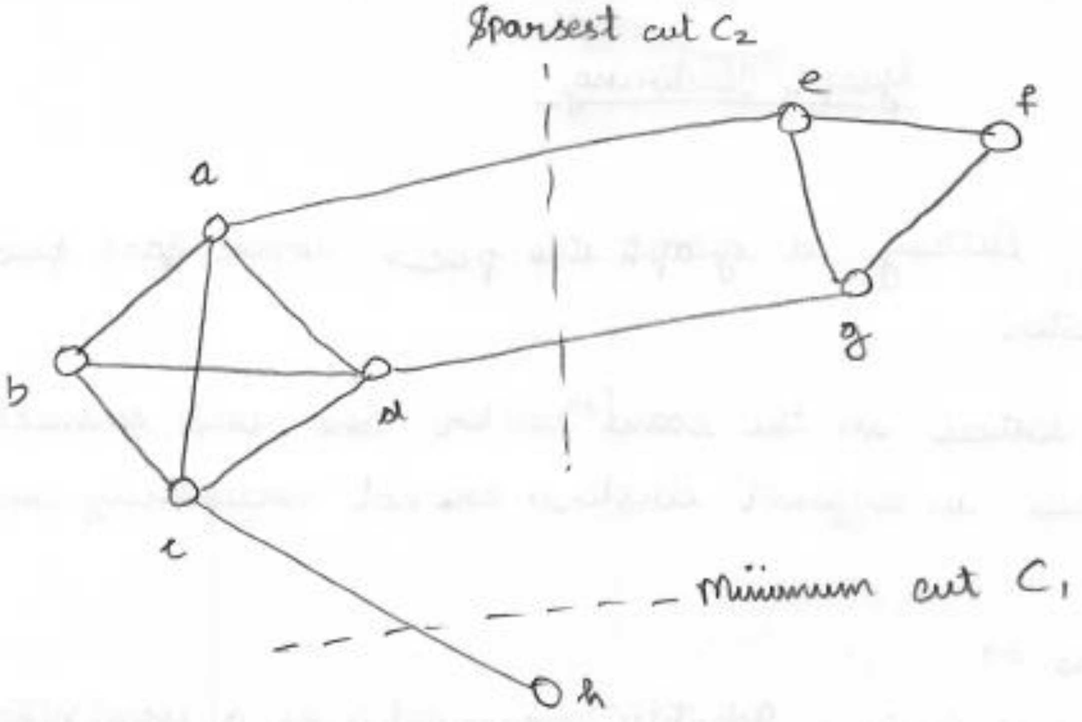
cut set is $\{(u,v) \in E \mid u \in S, v \in T\}$

→ The number of edges in the cut set is called the size of the cut (cut size).

→ Minimum cut: a cut with the smallest cut size

$$\text{Sparsity} = \frac{\text{cut size}}{\min\{|S|, |T|\}}$$

Sparsest cut: cut with smallest sparsity.




$$\text{sparsity of } C_1 = \frac{1}{1} = 1$$

$$\text{sparsity of } C_2 = \frac{\text{cut size}}{\min\{|S|, |T|\}} = \frac{2}{3}$$

NOTE:

cut size = # number of edges pass through $C_2$

for $C_2 \rightarrow$ ae, dg

cut size $C_2 = 2$

for $C_1 \rightarrow$ c h. only 1

cut size $C_1 = 1$

$\min\{|S|, |T|\}$ = (number of vertices through 1 cut and other cut.

$\min \Rightarrow \begin{cases} C_1 \text{ left has 5 vertices} \\ C_1 \text{ right has 3 vertices} \end{cases}$

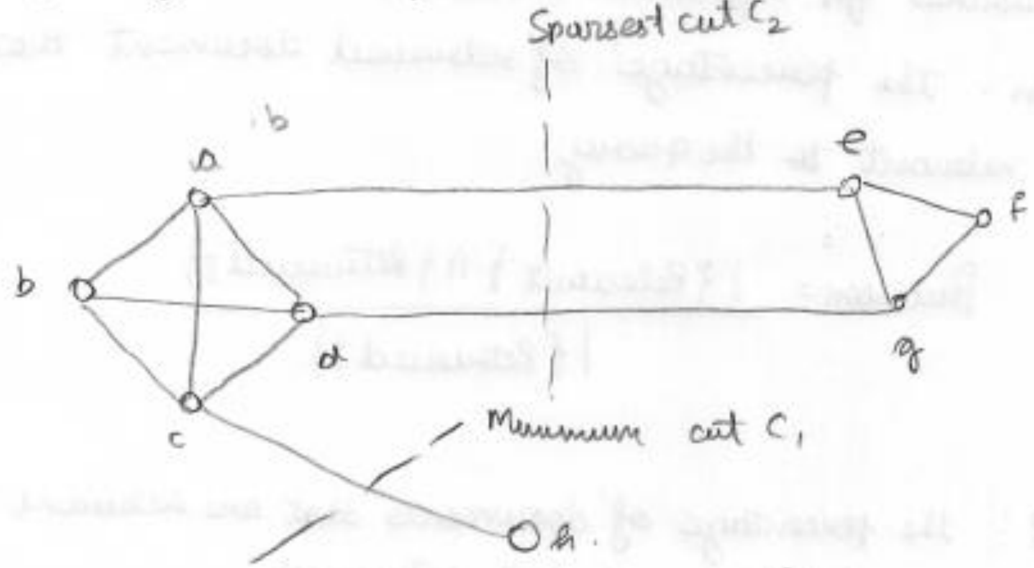
$\min(5, 3) = 3$

Sparcity is smaller for $C_2$ so $C_2$ is sparsest cut

Example. Graph G has two clusters : {a, b, c, d}, {e, f, g} and an outlier vertex, h.

Sol : cut $C_1$ = ({a, b, c, d, e, f, g}, {h})

cut set $C_1$ is {(c, h)}, cut size of $C_1$ is 1 and sparsity of $C_1 = \frac{1}{1} = 1$. $C_1$ is a minimum cut




cut $C_2$ = ({a, b, c, d, h}, {e, f, g})

cut $C_2$ is {(a, e), (d, g)}; cut size of $C_2$ is 2 and sparsity of $C_2$ is $2/3 = 0.67$, $C_2$ is a sparsest cut

☆ Graph Clustering Challenges

→ A minimum cut does not yield a good clustering

→ A sparsest cut leads to a good clustering, but the sparsest cut problem is NP-hard.

# Text Mining

→ Text databases (document databases)

eg news articles, research papers, books, digital libraries, e-mail messages, Web pages etc

→ Information retrieval (IR) → difficult using traditional methods (They can only handle exact search but not able to handle approximate search).

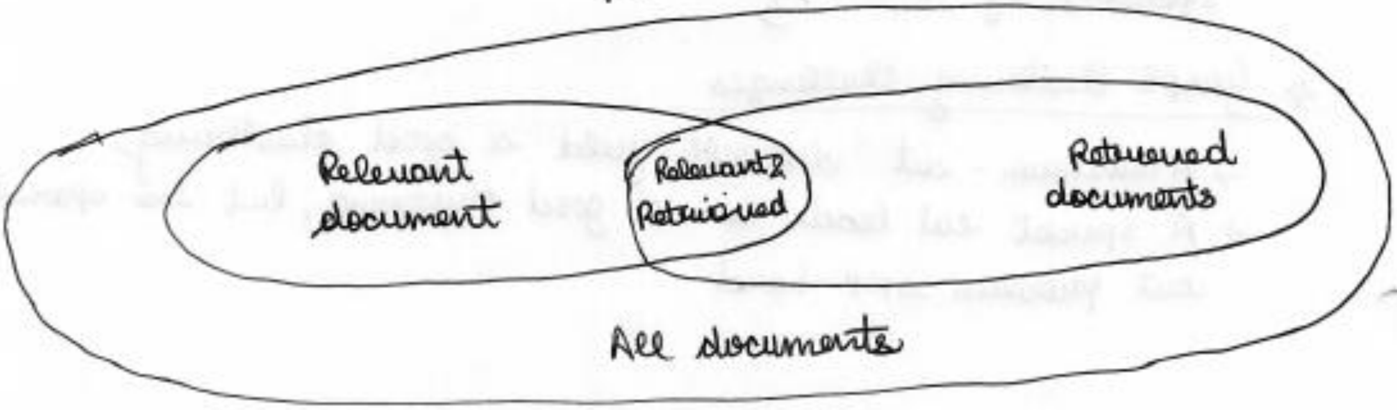
→ Basic measures for Information Retrieval

1.) Precision: the percentage of retrieved document that are infact relevant to the query.

$$\text{Precision} = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{|\{\text{Retrieved}\}|}$$

2.) Recall: the percentage of documents that are relevant to the query and were, in fact, retrieved

$$\text{Recall} = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{|\{\text{Relevant}\}|}$$



For ideal condition we need high precision and high recall

## ☆ Keyword Based Retrieval

A document is represented by a set of keywords.

Major difficulties of the keyword-based model

- Synonymy :- A keyword T (eg repair) does not appear anywhere in document, even though the document is closely related to T (eg. the document is about maintenance)
- Polysemy : The same word means different things (eg data mining vs coal mining)
- term frequency $tf(t,d)$ is the number of times the term $t$ occurs in the document $d$.

Then $\boxed{tf\text{-}idf(t,d) = tf(t,d) \times idf(t)}$

→ The inverse document frequency $idf(t) = \log\left(\frac{N}{df(t)}\right)$

where $df(t)$ is the document frequency of $t$, which is the number of document containing $t$.

Then $tf\text{-}idf(t,d) = tf(t,d) \times idf(t)$

Run this tf-idf algorithm on all words / terms in every document in the database D. Rank the words/terms based on their tf-idf scores. Pick the top k unique words/terms (k is user determined) with the largest tf-idf scores and use these k words / terms as the keywords for the database. D.

## Vector Space Model for IR.

Each document d is represented by a k-dimensional vector where a dimension corresponds to a keyword / term t and the value in that dimension is tf-idf(t, d); if t does not occur in d, the value in that dimension is 0.

Similarity measure: measure the closeness of two document (vectors)

- Cosine distance:

$$sim(V_1, V_2) = \frac{V_1 \cdot V_2}{\|V_1\| \, \|V_2\|}$$

## Text mining Algorithm

Here we represent each document in the database D by a set of keywords. Two documents might have different (number of) keywords, depending on whether the keywords occur in the documents

Let X, Y be keywords:-

→ Support for rule X → Y: The number of documents in the database that contain both X and Y

→ Confidence for rule X → Y: The percentage of documents in the database containing X that also contain Y.

NOTE

| Tid | Item bought |
|---|---|
| 10 | Beer, Nuts, Diaper |
| 20 | Beer, coffe, Diaper |
| 30 | Beer, Diaper, Eggs |
| 40 | Nuts, Eggs, Milk |

→ Support (beer) = $\frac{3}{4} \times 100 = 75\%$.

Freq 1-itemset

Nuts = $\frac{2}{4} \times 100 = 50\%$.

Eggs = $\frac{1}{4} \times 100 = 25\%$.

Diaper = $\frac{3}{4} \times 100 = 75\%$

Freq 2-itemset

{Beer, diaper} = $\frac{3}{4} \times 100 = 75\%$.

Association rules that can introduced through it

Confidence, c : The conditional probability that a transaction containing X also contains Y.

$$C = \frac{sup(X \cap Y)}{sup(X)}$$

Support = $(X \cap Y)$

confidence for ~~P(Beer ∩ diaper) | Dia~~

$$P(Beer \mid Diaper) = \frac{P(Beer \cap Diaper)}{P(Diaper)}$$

$$= \frac{\frac{3}{4} \times 100}{\frac{3}{4}} = 100$$

---

Continued ...

Use Apriori algorithm to find these association rules, treating each document as a transaction and each keyword as an item in the transaction

For eg Database D

| DID | keyword |
|---|---|
| D1 | Computer, stock |
| D2 | Economy, Finance, stock |
| D3 | Computer, ~~Economy~~, Finance, Management |
| D4 | Computer, Economy, Finance, Stock |
| D5 | Economy, Finance, Management |
| D6 | Finance, Management, stock |
| D7 | Economy, Management, Stock |
| | |

Scan D $C_1 L_1$

| keywordset | Sup |
|---|---|
| {Computer} | 3 |
| {Economy} | 4 |
| {Finance} | 5 |
| {Management} | 4 |
| {Stock} | 5 |

⇨

| $C_2$ keywordset | Sup |
|---|---|
| ~~{Computer, Economy}~~ | ~~1~~ |
| {Computer, Finance} | 2 |
| ~~{Computer, Management}~~ | ~~1~~ |
| {Computer, Stock} | 2 |
| {Economy, Finance} | 3 |
| {Economy, Management} | 2 |
| {Economy, Stock} | 3 |
| {Finance, Management} | 3 |
| {Finance, Stock} | 3 |
| {Management, Stock) | 2 |

⇒

| $C_3$ 3keyword set keyword | |
|---|---|
| {Economy, Finance, stock} = 2 | |

Only {Economy, Finance, stock} satisfy 3keyword set with support. 2

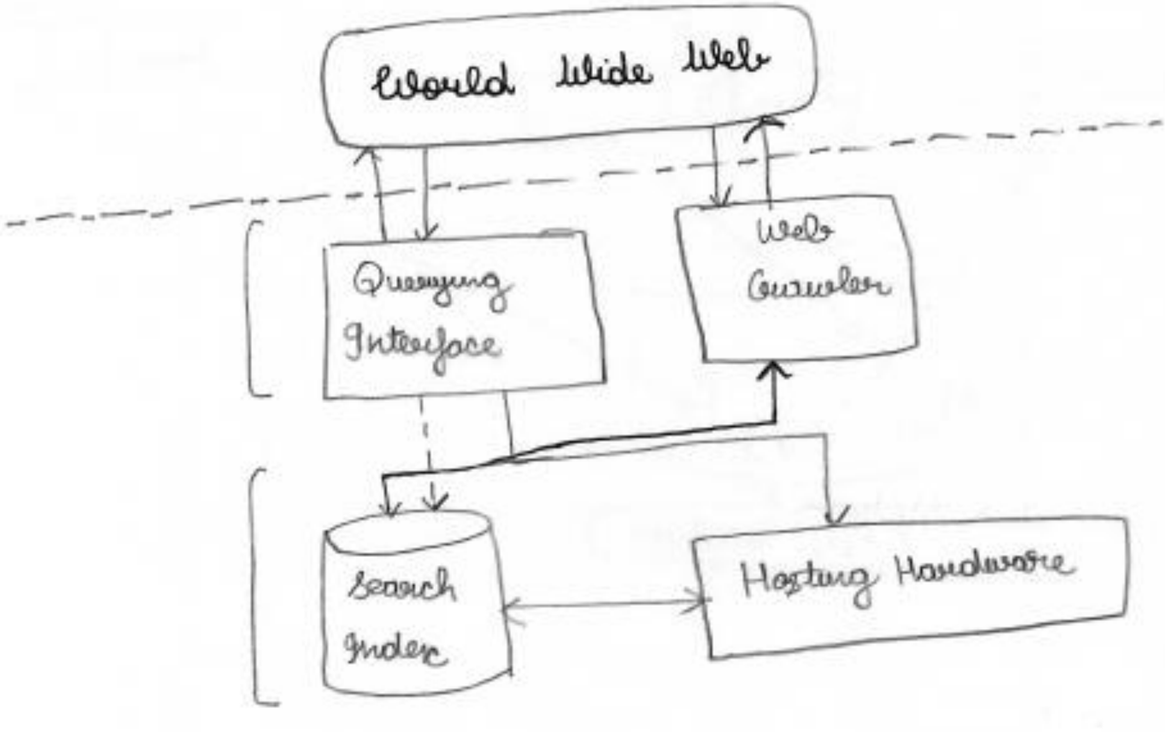
1.) Key Word Based Search Engines (Google, Yahoo)

1.) DBMS → search based on SQL attribute - value comparison

2.) Information Retrieval ⇒ search by topic or by keywords (recall, precision)

3.) Data mining ⇒ extraction of knowledge from data.

## Architecture of Search Engine

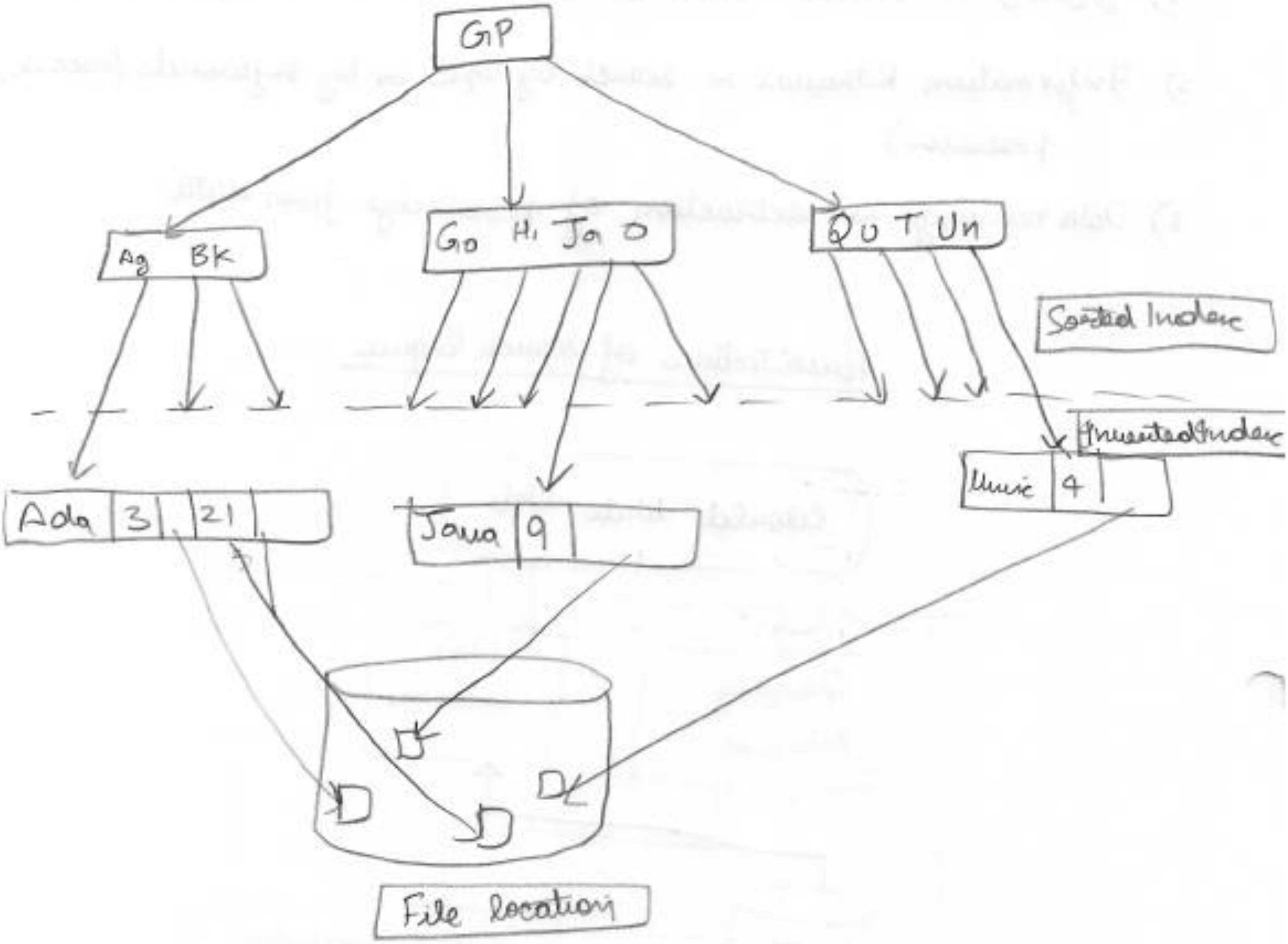



☆ Search Index

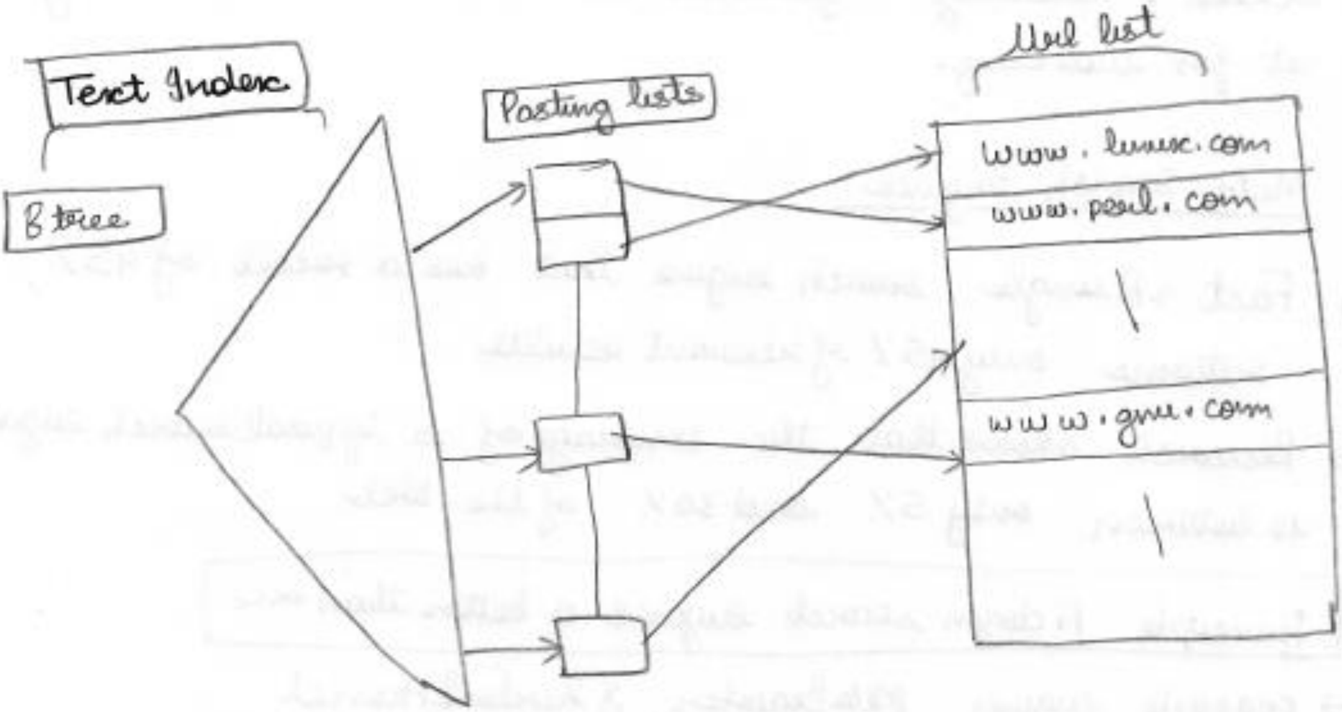
1) Inverted File
2.) Suffix Tree
3) Suffix Array
4) Signature File

1) Inverted File

B - tree



# Index Structure of a Web Search Engine




## Techniques for Reducing Index Size

1.) Case folding :- converts everything to lower case
eg "Data Mining" → "data mining"

2.) Stemming :- reduces words to their morphological root.
eg. "compression" and "compressed" become "compress"

3.) Stop word removal :- removes common or semantically insignificant words
eg. "the", "a", "an" are removed.

Text compression- reduces the inverted file.

# Web Crawlers

These are programs the work continuously behind the screen, locating information on the Web and returning it for indexing.

## ✧ Meta Search Engines

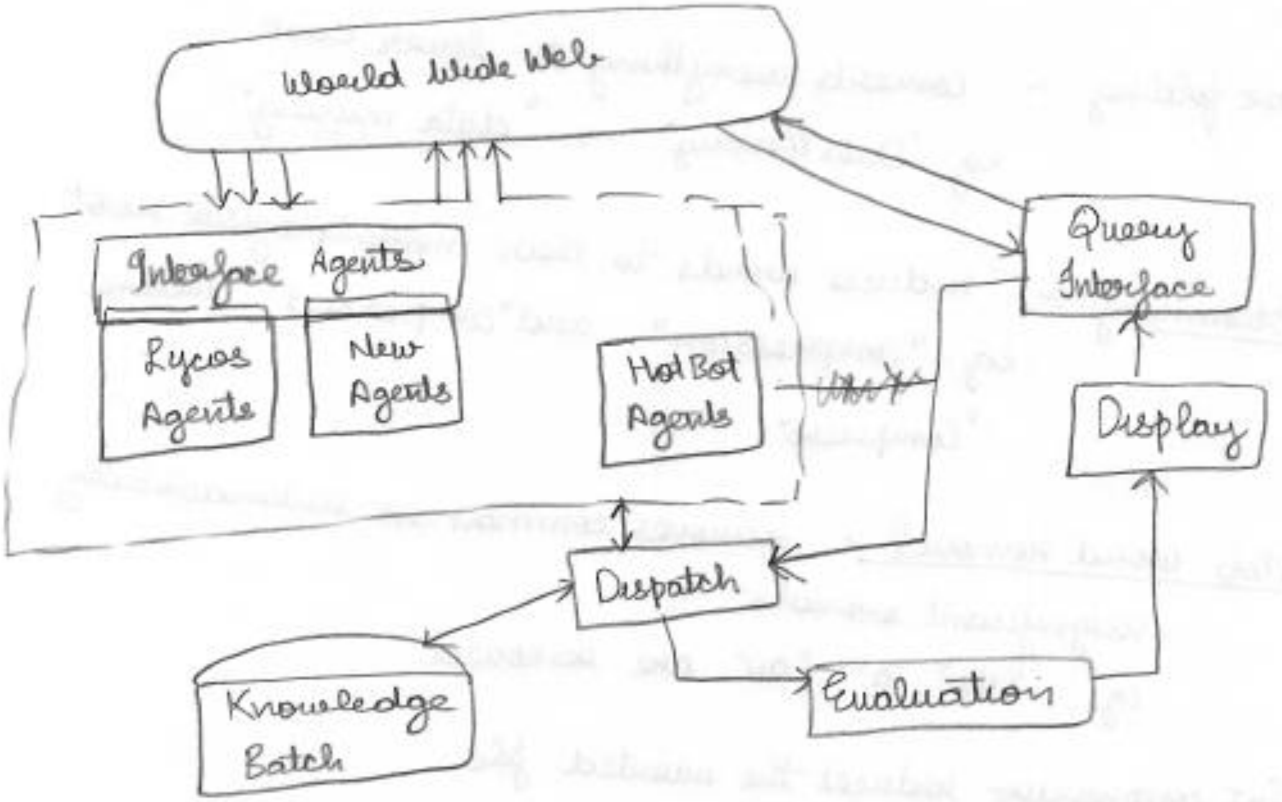
→ Fact → A single search engine that has a recall of 45%, returns only 45% of relevant results.

→ Research shows that the coverage of a typical search engine is between only 5% and 30% of the Web

| Principle: A dozen search engines is better than one. |
|---|

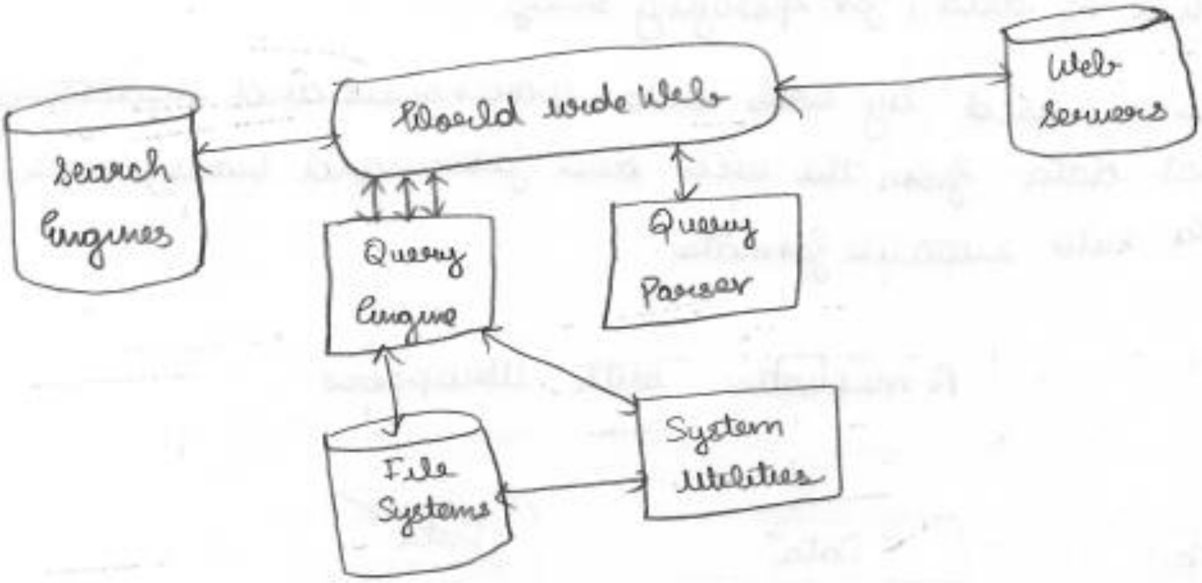
→ example engines: MetaCrawler, SherlockHound

Architecture of a meta Search Engine



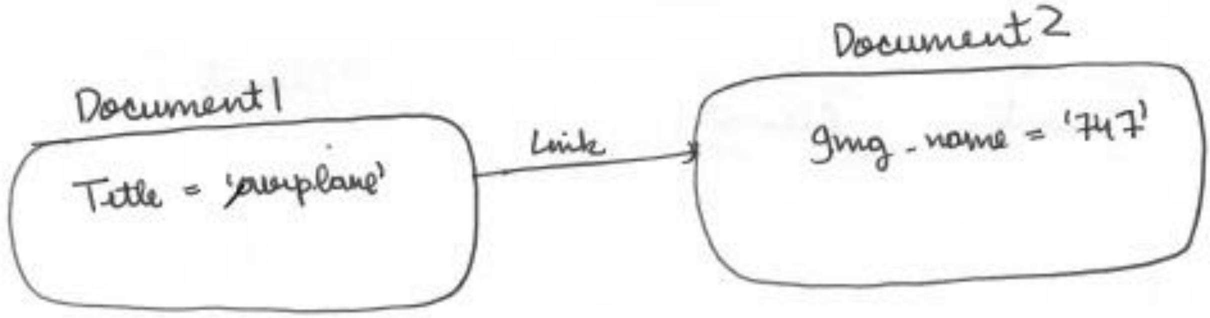
## 2.) Query Based Web Search Systems

System allows SQL like queries




Important Component of this is a parser that transforms the SQL query

### structure based Query



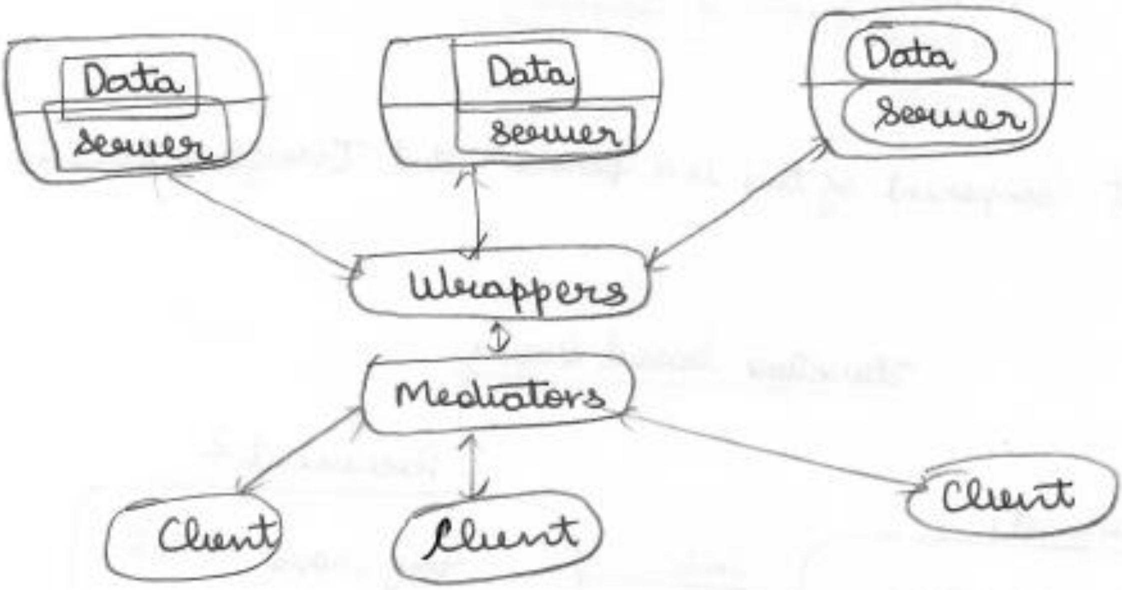
# Data Warehouse and Mediator

→ Data warehouses - provide a centralized location to store data and process queries

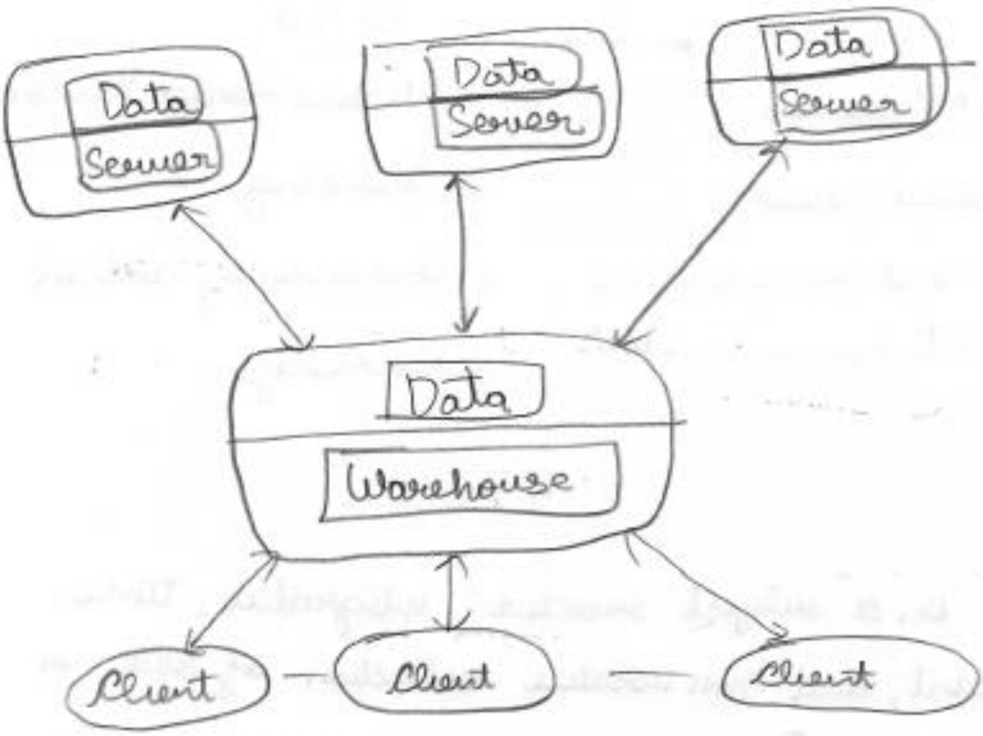
→ Mediators - provide a centralized location (with a small amount of data) for querying only

→ Wrappers - used by both data warehouses and mediators to extract data from the web and filter and transform the web data into suitable formats.

A mediator with Wrappers



## A Data Warehouse Architecture




Data Warehouse → Integrate data from heterogenous data sources

### OLAP

(Online Analytical processing)

→ knowledge worker eg Data scientis, ML engineer

→ Decision support

→ Subject oriented

→ Historical data

→ Complex queries

→ Query optimization

→ 100GB

→ use to create models for Machine learning

### OLTP

(Online Transaction processing)

→ Clerk

→ Daily operations

→ Application oriented

→ Current data

→ Transactions eg Bank

→ Transaction processing

→ 100MB

| Warehouse | DBMS |
| --- | --- |
| OLAP | → OLTP |
| → Complex OLAP queries | → Access ~~Model~~ methods |
| → Multidimensional view | → Indexing |
| → Aggregation and summarization of data from heterogeneous sources | → Concurrency control |
| | → Recovery |

Data warehouse is a subject oriented integrated, time ~~management~~ variant, and non volatile collection of data in support of management's decision-making process.

---

## ✧ Mining the World-Wide Web.

Only a small portion of the information on the web is truly relevant or useful.

- 99% of the Web information is useless to 99% of Web users

Issue - How can we find high quality Web pages on a specified topic ?

A challenging task :

- The abundance problem
- limited coverage of the Web : hidden web sources, majority of data in DBMS.
- limited query interface based on keyword-oriented search
- limited customization to individual users.

## Web mining Taxonomy

1.) Web content mining -- automatic discovery of Web document content patterns eg Web page mining, analyzing text and graphic contents on the web.
2.) Web usage mining -- automatic discovery of web server access patterns eg General access pattern tracking, user access patterns from the large collection of access logs.
3.) Web structure mining -- automatic discovery of hypertext /linking structure patterns.

## 2.) Web usage mining

four processing stages of Web usage mining:

1.) usage data collection
2.) usage data preprocessing.
3.) usage pattern discovery.
4.) usage pattern analysis (eg construct a web log data cube and apply OLAP ~~[illegible]~~ operations)

Web servers store access request information in Web server access logs.
Access logs are like fingerprints characterizing Web servers.
For each browsing session to a Web server, entries are recorded.
Access logs - store client access information (date, Client IP, request URL, bytes transferred etc.)

eg Statistical Analysis
many log analysis tools use this technique to analyze site traffic including frequently accessed pages, average file size, daily traffic, the number of site visitors, access error reporting etc

The discovery of facts about a Website is potentially important in monitoring Web usage, security checking, performance tuning, and site improvement.

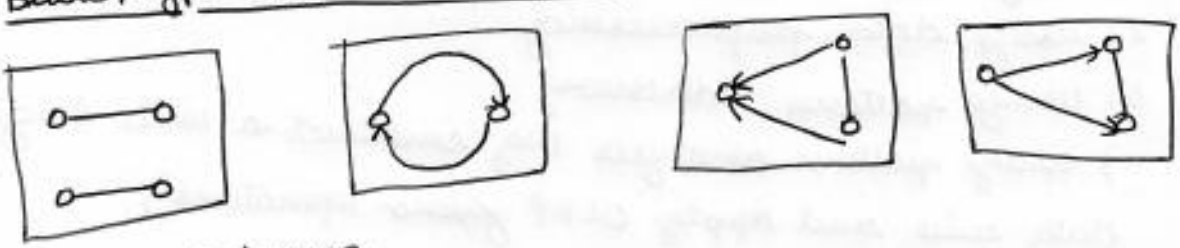
3.) Web structure mining

structure information on Web can be broadly classified as intra page and inter - page.

Inter page structure information can be analyzed by traversing hyperlinks and is often called Web linking structure

Intra page :→ refers to internal document structure of the actual web document in HTML or XML which is usually represented as a tree.

Basic Hyperlink Relationships



o → Web page

# Time Series

## ✧ Dynamic Time Wraping

Tow time series, Q and C, of length $n$ and $m$.

$Q = q_1, q_2, \dots, q_i, \dots, q_n.$

$C = c_1, c_2, \dots, c_j, \dots, c_m$

A wraping path $W$ is a contiguous set of matrix elements that defines a mapping between Q and C. The $k^{th}$ element of $W$ is defined as $w_k = (i, j)$.

$w_1, w_2, \dots, w_k, \dots w_K \quad \max(n, m) \le K < n+m-1$

the length of the wraping path

## ✧ Constraints of Wraping path

→ Boundary conditions : (from beginning to ending)

$w_1 = (1,1)$ and $w_K = (n, m)$

→ Continuity : (no jumps)

Given $w_k = (a, b)$ then $w_{k-1} = (a', b')$

where $a - a' \le 1$ and $b - b' \le 1$

→ Monotonicity : (can't go back in time)

Given $w_k = (a, b)$ then $w_{k-1} = (a', b')$,

$a - a' \ge 0$ and $b - b' \ge 0$

## Algorithm

1) Compute the distance $d(i, j)$.

$$d(q_i, c_j) = (q_i - c_j)^2;$$

2) Use dynamic programming to evaluate the cumulative distance $\gamma(i, j)$.

$$\gamma(i, j) = d(q_i, c_j) + \min(\gamma(i-1, j), \gamma(i, j-1), \gamma(i-1, j-1));$$

$q_i$ $c_j$

Base Case

$\gamma(0,0) = 0$ ; $\gamma(0,i) = \infty$ ; $\gamma(j,0) = \infty$ ;

$1 \le i \le m$, $1 \le j \le n$

---

## Dynamic Timing Warping Example

$Q = 0.2, 0.3, 0.2, 0.4$

$C = 0.2, 0.25, 0.3, 0.25, 0.4, 0.45$

| | 0.2 | 0.25 | 0.3 | 0.25 | 0.4 | 0.45 |
|---|---|---|---|---|---|---|
| 0.2 | 0 | 0.0025 | 0.01 | 0.0025 | 0.04 | 0.0625 |
| 0.3 | 0.01 | 0.0025 | 0 | 0.0025 | 0.01 | 0.0225 |
| 0.2 | 0 | 0.0025 | 0.01 | 0.0025 | 0.04 | 0.0625 |
| 0.4 | 0.04 | 0.0225 | 0.01 | 0.0225 | 0 | 0.0025 |

$d(i,j)$

$d(i,j)$

Using $d(i,j)$ values calculated we further calculate $\gamma(i,j)$ values

| | | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| 1 | ∞ | 0 | 0.0025 | 0.0125 | 0.015 | 0.055 | 0.1175 |
| 2 | ∞ | 0.01 | 0.0025 | 0.0025 | 0.005 | 0.015 | 0.0375 |
| 3 | ∞ | 0.01 | 0.005 | 0.0125 | 0.005 | 0.045 | 0.0775 |
| 4 | ∞ | 0.05 | 0.0275 | 0.015 | 0.0275 | 0.005 | 0.0075 |

$\gamma(i,j)$

This is considered in backtrack as to get 0.0075 we need to use this minima 0.005.

optimal warping path is
(1,1), (2,2), (2,3), (3,4), (4,5), (4,6)

let us look at ~~γ(3,4)~~ $\gamma(3,4)$ = $d(q_3, c_4)$ + min ( $\gamma(2,4)$, $\gamma(3,3)$, $\gamma(2,3)$ )

$$\gamma(3,4) = d(q_3, c_4) + \min(\gamma(2,4), \gamma(3,3), \gamma(2,3))$$

Ex 1 $\gamma(3,4) = 0.0025 + \min(0.005, 0.0025, 0.0125)$

$$\boxed{\gamma(3,4) = 0.0025 + 0.0025 = 0.0050}$$

Ex.2 $\gamma(3,5) = \underbrace{0.04}_{d(q_3, c_5)} + \min(0.015, 0.005, 0.005)$

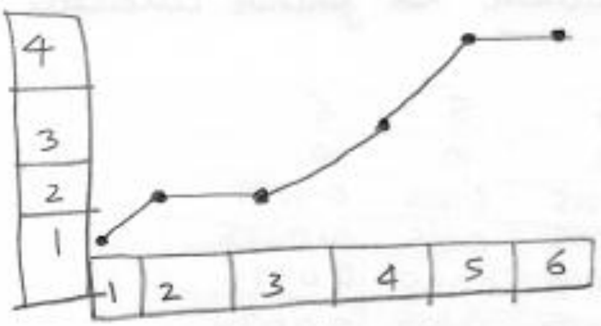
$= 0.04 + 0.005$

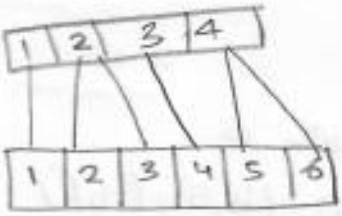
$$\boxed{\gamma(3,5) = 0.045}$$

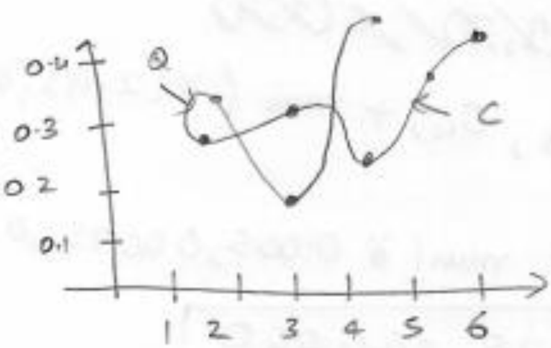
Similarly we can calculate the gamma

After finding all $\gamma$ we must backtrack to find optimal distance. We basically consider the minimum neighbour.

We get the path
(1,1), (2,2), (2,3), (3,4), (4,5), (4,6)


4
3
2
1
1 2 3 4 5 6


1 2 3 4
1 2 3 4 5 6


0.4
0.3
0.2
0.1
C
1 2 3 4 5 6

## Apriori - like Approach.

Based on an apriori if any length K pattern is not frequent in the database, its length (K+1) super pattern can never be frequent.

Frequent Pattern tree / FP greddy

Example database 6 transactions   Min Support = 3

| TID | Item |
|---|---|
| 100 | C, F, B, G |
| 200 | A, D, B, H, F |
| 300 | A, C, G, B |
| 400 | B, D, E, A |
| 500 | H, C, A, D, F |
| 600 | G, B, E, A, D |

$\xrightarrow[\text{DB}]{\text{Scan}}$

| Item | Support |
|---|---|
| A | 5 |
| B | 5 |
| C | 3 |
| D | 4 |
| E | 2 |
| F | 3 |
| G | 3 |
| H | 2 |

sort

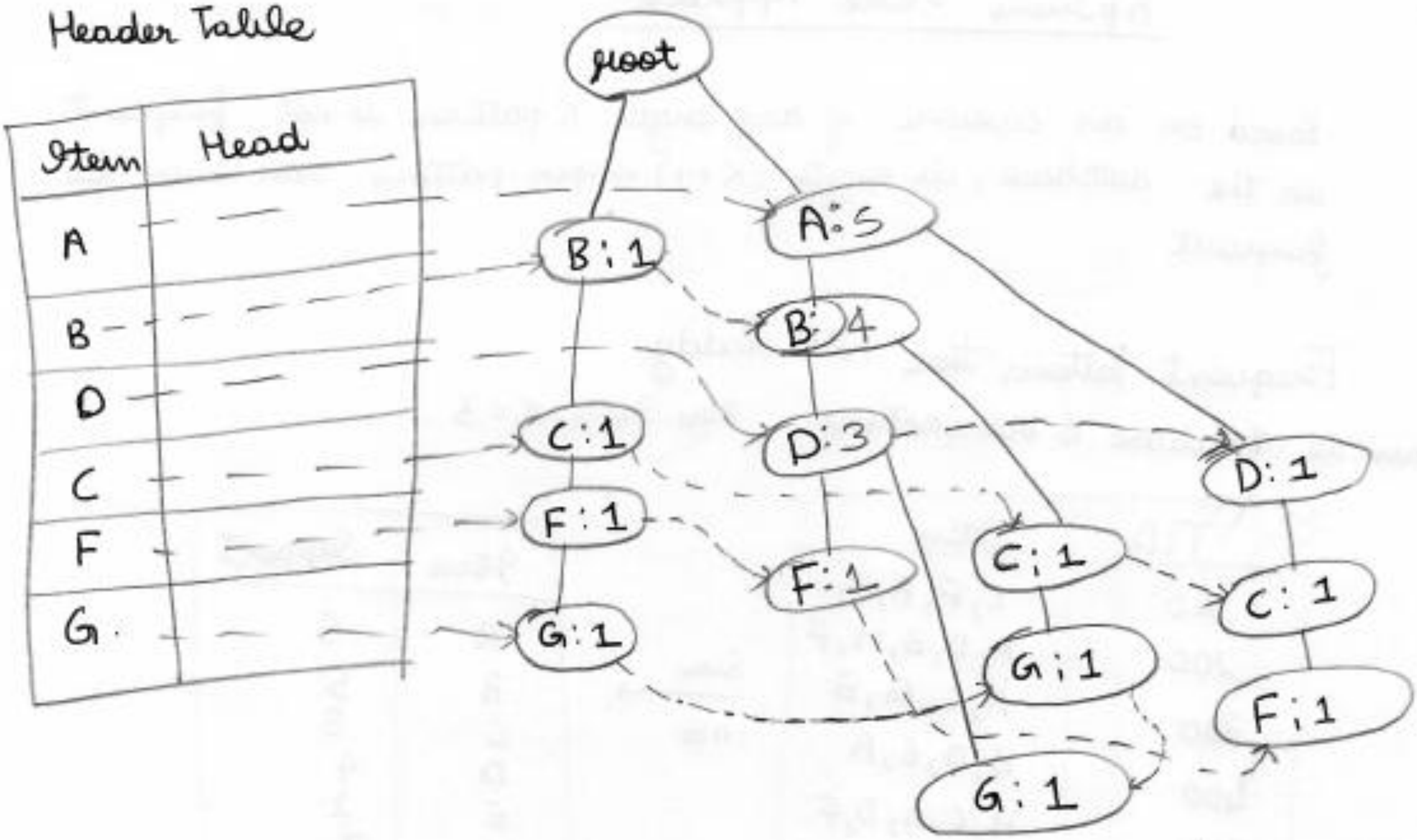
| Item | Support |
|---|---|
| A | 5 |
| B | 5 |
| D | 4 |
| C | 3 |
| F | 3 |
| G | 3 |

DB

| TID | Item | Frequent Items |
|---|---|---|
| 100 | C, F, B, G | B, C, F, G |
| 200 | A, D, B, H, F | A, B, D, F |
| 300 | A, C, G, B | A, B, C, G |
| 400 | B, D, E, A | A, B, D |
| 500 | H, C, A, D, F | A, D, C, F |
| 600 | G, B, E, A, D | A, B, D, G |

FP Growth Construction in next state

# FP Tree




- For item G, it derives a frequent pattern (G:3) and three path in the FP-tree.
  - ⟨B:1, C:1, F:1, G:1⟩
  - ⟨A:5, B:4, C:1, G:1⟩
  - ⟨A:5, B:4, D:3, G:1⟩

- G's conditional pattern base:
  {(B:1, C:1, F:1), (A:1, B:1, C:1), (A:1, B:1, D:1)}

- G's conditional FP-tree:
  - {(B:3)}|G.

- Frequent patterns: {(G:3), (BG:3)}

→ For item F, it derives a frequent pattern (F:3) and three paths in the FP tree:

$\langle B:1, C:1, F:1 \rangle$
$\langle A:5, B:4, D:3, F:1 \rangle$
$\langle A:5, D:1, C:1, F:1 \rangle$

- F's conditional pattern base:
  $\{(B:1, C:1), (A:1, B:1, D:1), (A:1, D:1, C:1)\}$

- F's conditional FP-tree:
  $- \emptyset$

- Frequent patterns: $\{(F:3)\}$

→ For item C, it derives a frequent pattern (C:3) and three paths in the FP tree

- $\langle B:1, C:1 \rangle$
- $\langle A:5, B:4, C:1 \rangle$
- $\langle A:5, D:1, C:1 \rangle$

- C's conditional pattern base:
  $\{(B:1), (A:1, B:1), (A:1, D:1)\}$

- C's conditional FP-tree:
  $\emptyset$

- Frequent Patterns: $\{(C:3)\}$.

→ For item D, it derives a frequent pattern (D : 4) and two paths in the FP-tree :

$\langle A:5, B:4, D:3 \rangle$

$\langle A:5, D:1 \rangle$

∘ D's conditional pattern base :

- {{A : 4, B : 3)} (A : 1) }

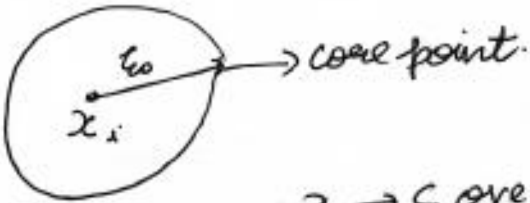
D's conditional FP-tree

{(A:4, B:3)} | D.

∘ Frequent patterns : {(D:4), (AD:4), (BD:3), (ABD:3)}.

# DBSCAN algorithm

① $\forall x_i \in D \rightarrow$ label each point as a core point, border point, noise point:




$$\begin{cases} x_1 \rightarrow \text{Core point} \\ x_2 \rightarrow \text{Border point} \\ x_3 \rightarrow \text{Noise point} \\ x_4 \rightarrow \text{Core point} \\ x_5 \rightarrow \text{Core point} \end{cases}$$

($Q_i$) = To get all these points we need to give range query $(x_i, D, \varepsilon_0)$

↕

These are implemented structures using (KD-trees.)

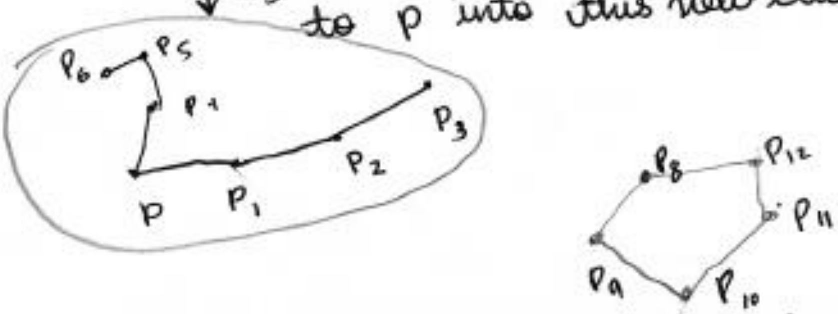
② remove all noise points from your data

↳ sparse regions ⇒ don't belong to any clusters.

③ For each core pt 'p' not assigned to a cluster.

a) create a new cluster with 'p'

b) Add all the points that are density connected to p into this new clusters.




4) each border pt → assign it to the nearest core pt's cluster

$S_i = \text{range Query}(x_i, D, \varepsilon_0)$

Kd tree this can be implemented