ครั้งที่ 7

โครงสร้างแฟ้มข้อมูล

โครงสร้างแฟ้มข้อมูล

Relational Model

เป็นโมเดลที่ข้อมูลจะถูกอธิบายโดย n-tuple ของ คุณสมบัติ (Attribute Values)

Hierarchical Approach

เป็นวิธีการที่ข้อมูลจะถูกแสดงในรูปของลำดับชั้นของข้อมูล

Network Approach

เป็นวิธีการที่ข้อมูลถูกเชื่อมโยงเป็นข่ายงาน ในแต่ละลิงค์ที่เชื่อมโยงระหว่างสองไอเท็มจะเป็นไปตามเงื่อนไขที่กำหนดเช่นมีคุณสมบัติเหมือนกัน

Hsiao และ Harary

กำหนดให้ A เป็นคุณสมบัติ(Attributes)

V เป็นเซตของค่า (Attribute Values)

R เป็นเรคอร์ด

ดังนั้น R เป็นซับเซตของ Cartesian Product AxV

ซึ่งแต่ละคุณสมบัติจะมีเพียงหนึ่งค่าเท่านั้น อยู่ในรูปของเซตของ Ordered Pairs ในรูปแบบของ (Attribute,Value)

ตัวอย่าง เรคอร์ด

R =((K1,X1), (K2,X2),…,(Kn,Xn))

โดย K หมายถึงคีย์เวิร์ด(Keyword) หรือคุณสมบัติ(Attribute)
X หมายถึงค่า(Value) หรือ ค่าน้ำหนัก(Numerical Weight)

ในบางกรณีที่เอกสารจะถูกกำหนดให้อยู่ในรูปแบบของ

R = {Kt1, Kt2, Kt3,…, Ktn}

โดย เอกสารจะถูกทดสอบว่ามีคีย์เวิร์ดนั้นอยู่ในเอกสารหรือไม่ในกรณีที่คีย์เวิร์ดมีอยู่ในเอกสารจะมีค่า X เท่ากับ 1 แต่ถ้าไม่มีคีย์เวิร์ดในเอกสารค่าของ X จะมีค่าเท่ากับ 0

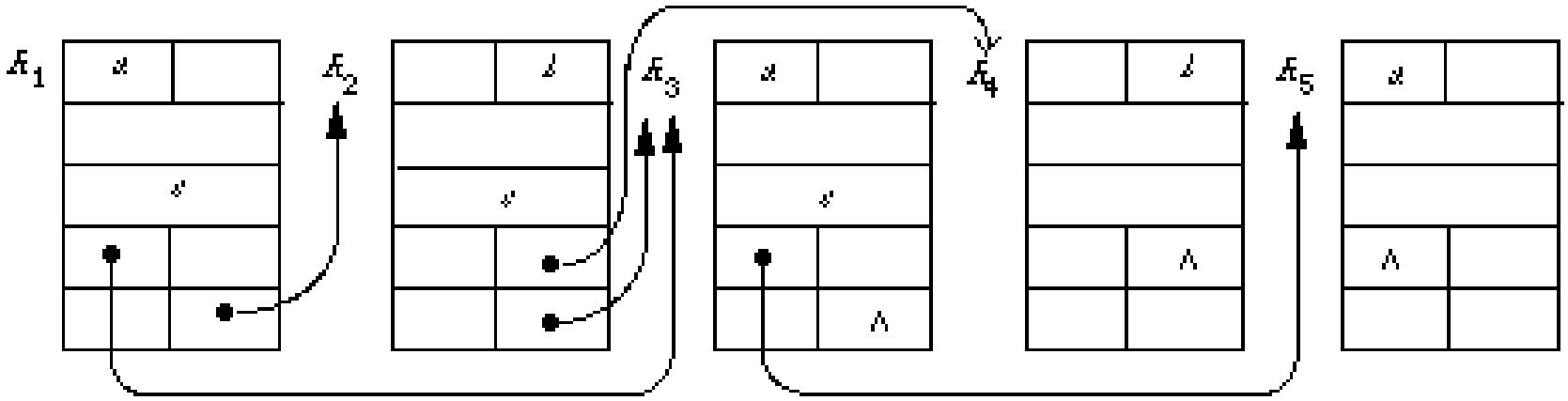
ตัวอย่าง เรคอร์ดที่ประกอบด้วยรายการข้อมูลที่เกี่ยวข้อง

$R =$

R_1	R_2
R_3	
R_4	
F_1	F_2
F_3	F_4

เรคอร์ดหนึ่งประกอบด้วยหลายฟิลด์แต่ละฟิลด์จะเก็บค่าคุณสมบัติ(Attribute)
ที่กำหนดความยาวของแต่ละฟิลด์ไม่จำเป็นต้องมีขนาดคงที่
และอาจจะมีบางฟิลด์ที่เก็บตำแหน่งที่อยู่(address)

การใช้พอยเตอร์ในการเชื่อมโยงเรคอร์ด



กำหนดให้ L เป็นลิสต์ของเรคอร์ดที่เกี่ยวข้องกับคีย์เวิร์ด K

K-list เป็นเซตของเรคอร์ดที่มี K อยู่

จากรูป

a-list เป็นลิสต์ของ R1,R3,R5

b-list เป็นลิสต์ของ R2,R4

c-list เป็นลิสต์ของ R1,R2,R3

ไดเรกทอรี่(Directory)ของแฟ้มข้อมูล

กำหนดให้ F เป็นแฟ้มข้อมูลหนึ่งที่มีเรคอร์ดที่บรรจุ

คีย์เวิร์ดที่แตกต่างกัน m ตัวได้แก่ K1,K2,…,Km

Ni เป็นจำนวนเรคอร์ดที่บรรจุคีย์เวิร์ด Ki

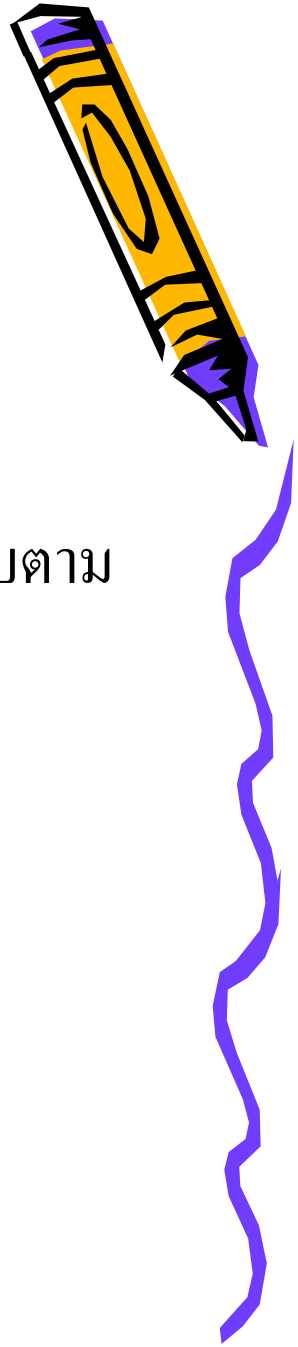
hi เป็นจำนวนของ Ki-Lists ใน F

Aij เป็นตำแหน่งเริ่มต้นของ Kj-list

ดังนั้น เราสามารถนิยามไดเร็กทอรี่ได้ว่า เป็นชุดของ

{Ki, Ni, Hi, Ai1, Ai2,…,Aihi}

Sequential File



- ไม่มีไดเร็กตอรี่และพอยเตอร์สำหรับเชื่อมโยง
- โดยทั่วไปการจัดเก็บหรือการสืบค้นข้อมูลจะกระทำเรียงลำดับตามคีย์
- เรียงตามลำดับที่เก็บในหน่วยบันทึกข้อมูล
- กรณีคีย์ซ้ำสามารถเลือกคีย์ที่สองมาจัดเรียงลำดับข้อมูล

Sequential File

ข้อดี

- ง่ายต่อการสร้าง และมีโครงสร้างแบบง่าย
- สามารถประมวลผลเรคอร์ดถัดไปได้อย่างรวดเร็ว
- ใช้ทรัพยากรได้อย่างมีประสิทธิภาพ เหมาะสำหรับงานที่ต้องใช้ข้อมูลทุกเรคอร์ด
- ใช้ได้กับหน่วยความจำสำรองที่มีราคาถูกเช่น เทปแม่เหล็ก

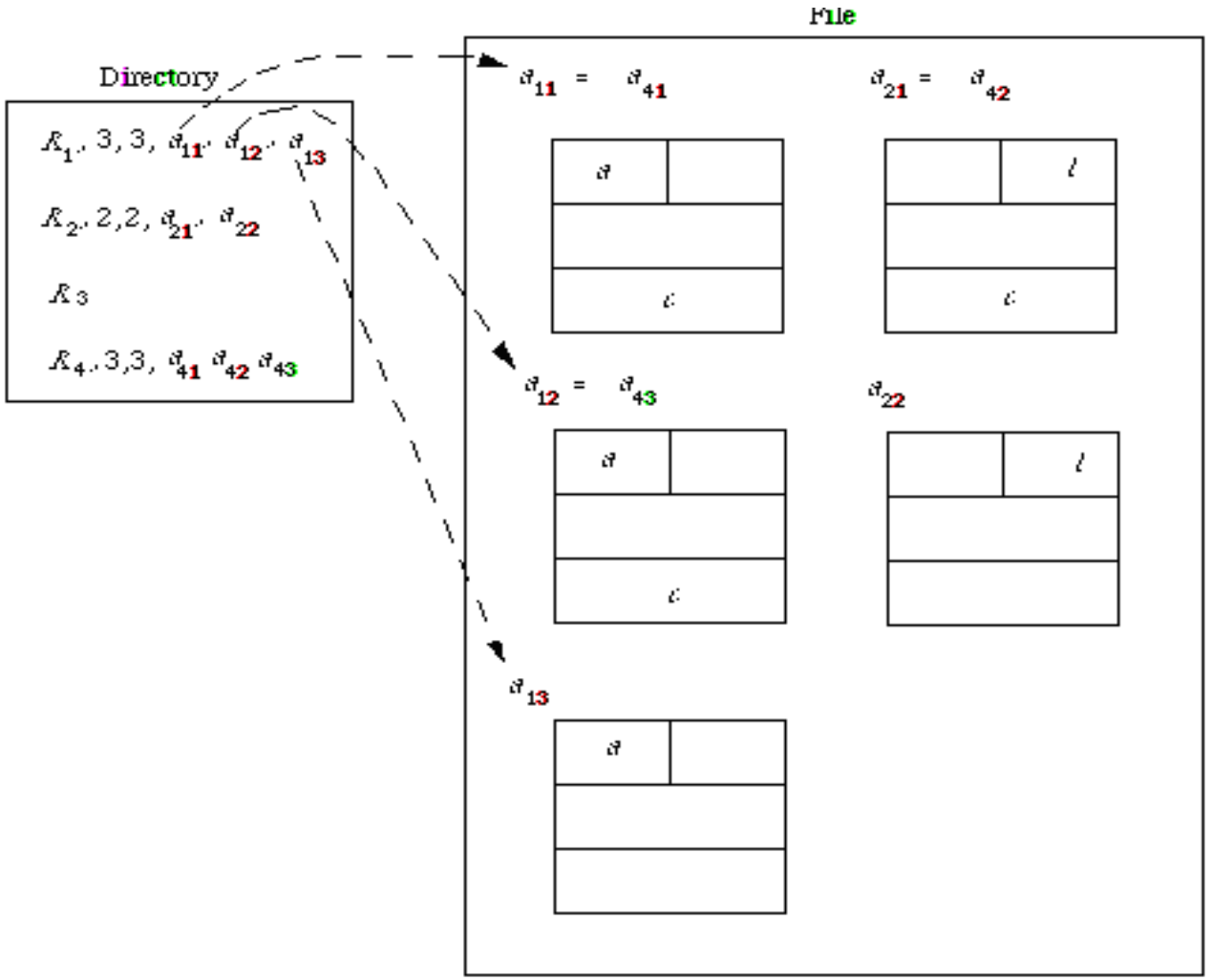
ข้อเสีย

- แทรกหรือแก้ไขข้อมูลในแฟ้มข้อมูลได้ยาก และเสียเวลามาก
- การสืบค้นข้อมูลที่ต้องการนั้น ต้องอ่านข้อมูลตั้งแต่เรคอร์ดแรกไปตามลำดับไม่เหมาะกับการประมวลผลแบบสุ่ม (Random)
- ต้องนำเรคอร์ดมาจัดเรียงลำดับก่อน (sort)
- ข้อมูลในแฟ้มข้อมูลไม่ทันสมัยหรือทันต่อเหตุการณ์ทุกขณะ

Invert File

- โครงสร้างแฟ้มข้อมูลที่ทุกๆลิสต์บรรจุเพียง 1 เรคอร์ด ดังนั้นทุกๆ K-list จะมีเพียง 1 เรคอร์ด
- ไดเร็กทอรี่ของแฟ้มข้อมูลจะมี Ni = hi สำหรับทุกๆ i ซึ่ง จำนวนเรคอร์ดที่บรรจุ Ki จะมีค่าเท่ากับจำนวนของ Ki-list

ไดเร็คทอรี่ของ invert File



ตัวอย่าง

สมมุติว่าเอกสารมีคีย์เวิร์ดดังนี้

D1 = (K1 ,K2, K3)

D2 = (K2, K3, K5)

D3 = (K1, K3, K4, K5)

D4 = (K1, K2, K3, K4)

Inverted file

จะเก็บข้อมูลในลักษณะนี้

K1 = (D1,D3, D4)

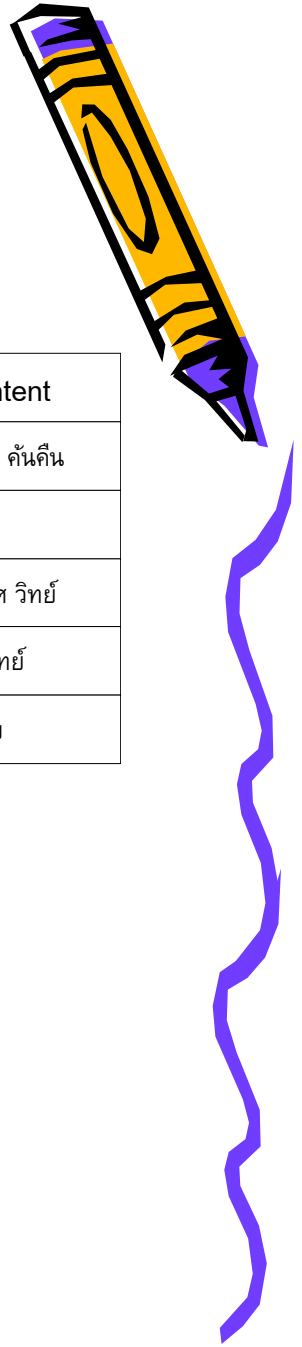
K2 = (D1, D2, D4)

K3 = (D1, D2, D3, D4)

K4 = (D3, D4)

K5 = (D2, D3)

ตัวอย่าง



ดัชนี	Keyword_id
อัลกอริธึม	1
สารสนเทศ	2
ค้นคืน	3
วิทย์	4
รูปแบบ	5

Keyword_id	DocId	Freq
1	1	1
2	1	1
3	1	1
3	2	1
4	2	1
1	3	1
2	3	1
4	3	1
5	4	1
3	4	1

DocId	Document Content
1	อัลกอริธึม สารสนเทศ ค้นคืน
2	ค้นคืน วิทย์
3	อัลกอริธึม สารสนเทศ วิทย์
4	รูปแบบ ค้นคืน วิทย์
5	วิทย์ อัลกอริทึม

กำหนด Vector File เป็นดังนี้

docs	*t1*	*t2*	*t3*	RSV=Q.Di
D1	1	0	1	4
D2	1	0	0	1
D3	0	1	1	5
D4	1	0	0	1
D5	1	1	1	6
D6	1	1	0	3
D7	0	1	0	2
D8	0	1	0	2
D9	0	0	1	3
D10	0	1	1	5
D11	1	0	1	3
Q	1	2	3	
	q1	*q2*	*q3*	

Invert file คือ Vector File
ที่ invert แถวให้เป็นคอลัมภ์
และเปลี่ยนคอลัมภ์ให้เป็นแถวดังนี้

Terms	D1	D2	D3	D4	D5	D6	D7	…
t1	1	1	0	1	1	1	0	
t2	0	0	1	0	1	1	1	
t3	1	0	1	0	1	0	0	

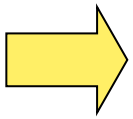
ตัวอย่าง

สมมุติว่า มีเอกสาร DOC1 และ DOC2 ดังนี้

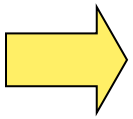
DOC1 ประกอบด้วย Now is the time for all good men to come to the aid of their country

DOC2 ประกอบด้วย It was a dark and stormy night in the country manor. The time was past midnight

Term	Doc #
now	1
is	1
the	1
time	1
for	1
all	1
good	1
men	1
to	1
come	1
to	1
the	1
aid	1
of	1
their	1
country	1
it	2
was	2
a	2
dark	2
and	2
stormy	2
night	2
in	2
the	2
country	2
manor	2
the	2
time	2
was	2
past	2
midnight	2



a	2
aid	1
all	1
and	2
come	1
country	1
country	2
dark	2
for	1
good	1
in	2
is	1
it	2
manor	2
men	1
midnight	2
night	2
now	1
of	1
past	2
stormy	2
the	1
the	1
the	2
the	2
their	1
time	1
time	2
to	1
to	1
was	2
was	2



Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
good	1	1
in	2	1
is	1	1
it	2	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	2	1
stormy	2	1
the	1	2
the	2	2
their	1	1
time	1	1
time	2	1
to	1	2
was	2	2

Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
good	1	1
in	2	1
is	1	1
it	2	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	2	1
stormy	2	1
the	1	2
the	2	2
their	1	1
time	1	1
time	2	1
to	1	2
was	2	2

Dictionary

Term	N docs	Tot Freq
a	1	1
aid	1	1
all	1	1
and	1	1
come	1	1
country	2	2
dark	1	1
for	1	1
good	1	1
in	1	1
is	1	1
it	1	1
manor	1	1
men	1	1
midnight	1	1
night	1	1
now	1	1
of	1	1
past	1	1
stormy	1	1
the	2	4
their	1	1
time	2	2
to	1	2
was	1	2

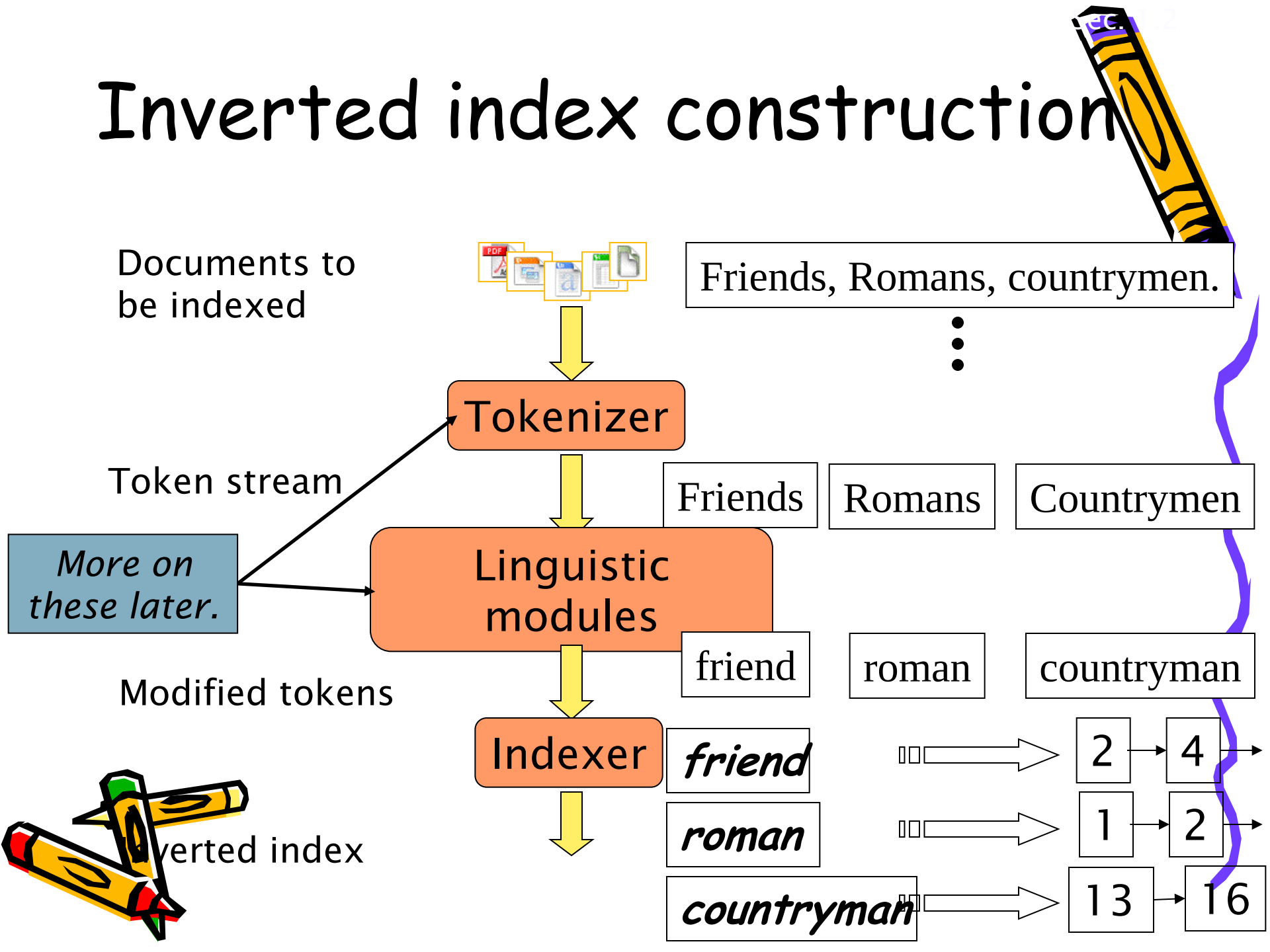
Postings

Doc #	Freq
2	1
1	1
1	1
2	1
1	1
1	1
2	1
2	1
1	1
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
1	1
1	1
2	1
2	1
1	2
2	2
1	1
1	1
2	1
1	2
2	2

Invert File

Term	docID
data	http://www-inst.eecs.berkeley.edu/~cs186
database	http://www-inst.eecs.berkeley.edu/~cs186
date	http://www-inst.eecs.berkeley.edu/~cs186
day	http://www-inst.eecs.berkeley.edu/~cs186
dbms	http://www-inst.eecs.berkeley.edu/~cs186
decision	http://www-inst.eecs.berkeley.edu/~cs186
demonstrate	http://www-inst.eecs.berkeley.edu/~cs186
description	http://www-inst.eecs.berkeley.edu/~cs186
design	http://www-inst.eecs.berkeley.edu/~cs186
desire	http://www-inst.eecs.berkeley.edu/~cs186
developer	http://www.microsoft.com
differ	http://www-inst.eecs.berkeley.edu/~cs186
disability	http://www.microsoft.com
discussion	http://www-inst.eecs.berkeley.edu/~cs186
division	http://www-inst.eecs.berkeley.edu/~cs186
do	http://www-inst.eecs.berkeley.edu/~cs186
document	http://www-inst.eecs.berkeley.edu/~cs186
document	http://www.microsoft.com

Inverted index construction



Indexer steps: Token sequence

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

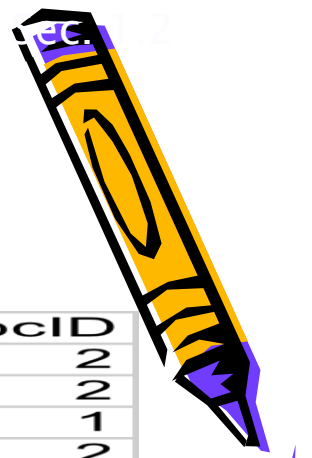
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Indexer steps: Sort

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

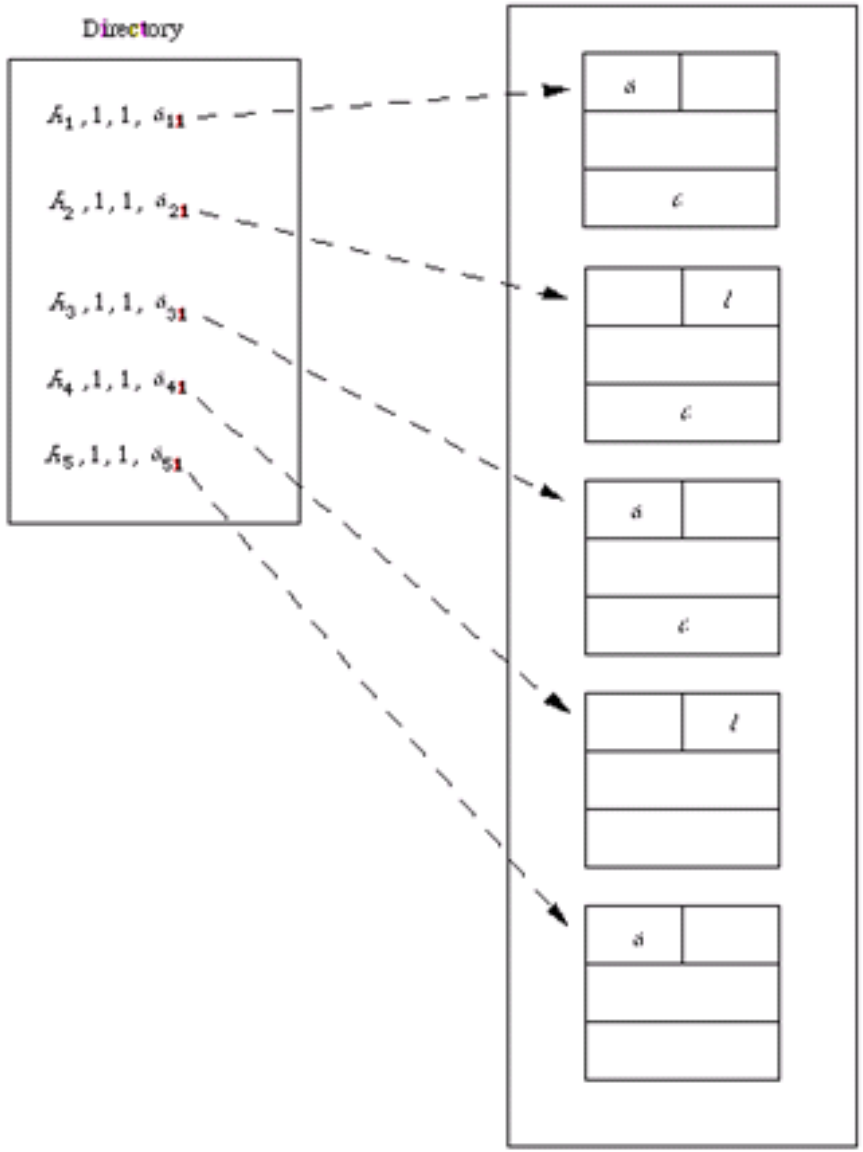
Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



Index-Sequential File(IS File)

- เป็นแฟ้มข้อมูลชนิด Inverted File ที่มีทุกคีย์เวิร์ด Ki ซึ่ง Ni = hi = 1 ซึ่งแต่ละเรคอร์ดจะมีเพียง 1 คีย์เวิร์ดเท่านั้น ซึ่งกลุ่มของเรคอร์ดเหล่านี้จะถูกจัดเรียงลำดับตามลำดับของคีย์

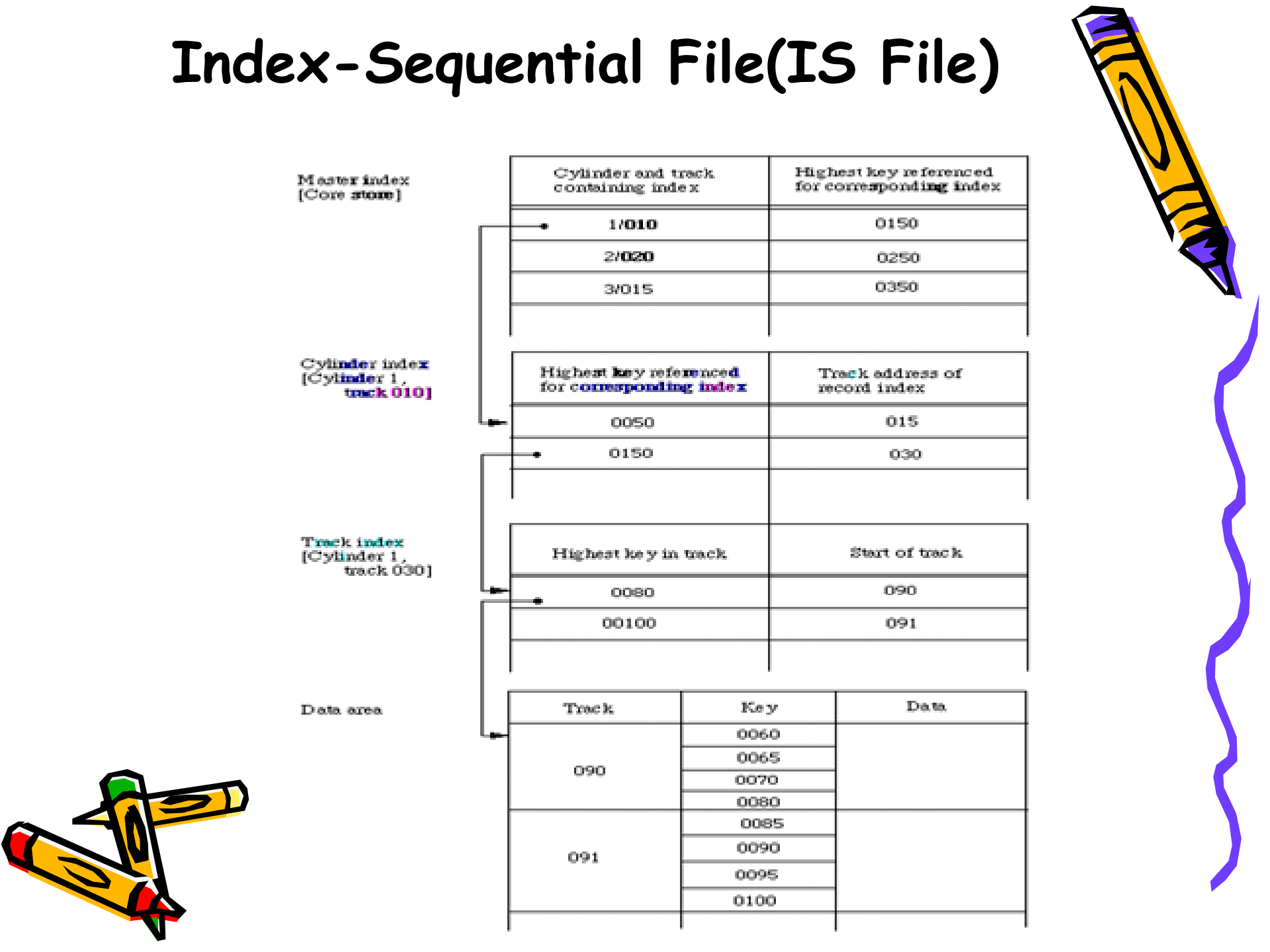
Index-Sequential File(IS File)



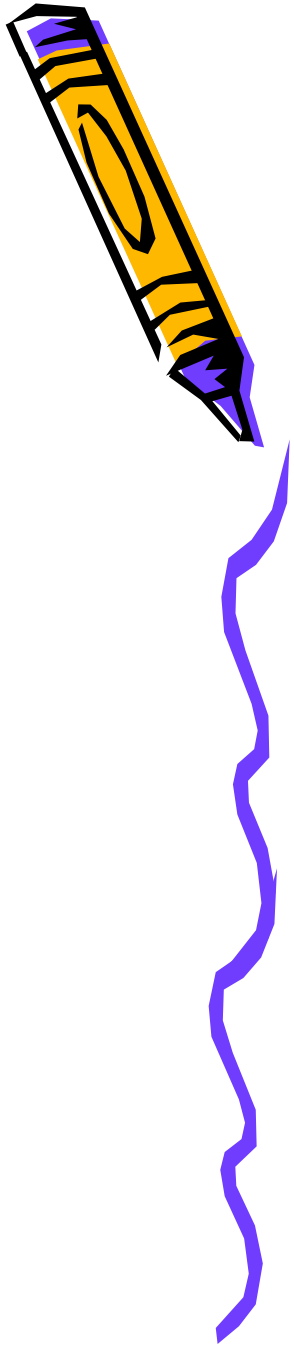
Index-Sequential File(IS File)

- เหมือนกับ Sequential File ที่มีดรรชนีเป็นลำดับชั้น โดยจะใช้หมายเลขเรคอร์ดเป็นคีย์
- ไดเร็กทอรี่ของแฟ้ม Index-Sequential File ที่ถูกสร้างบนเครื่องคอมพิวเตอร์ ประกอบด้วยดัชนี 3 ระดับได้แก่ Master Index , Cylinder index และ Track index

Index-Sequential File(IS File)



มัลติลิสต์(Multi-List)

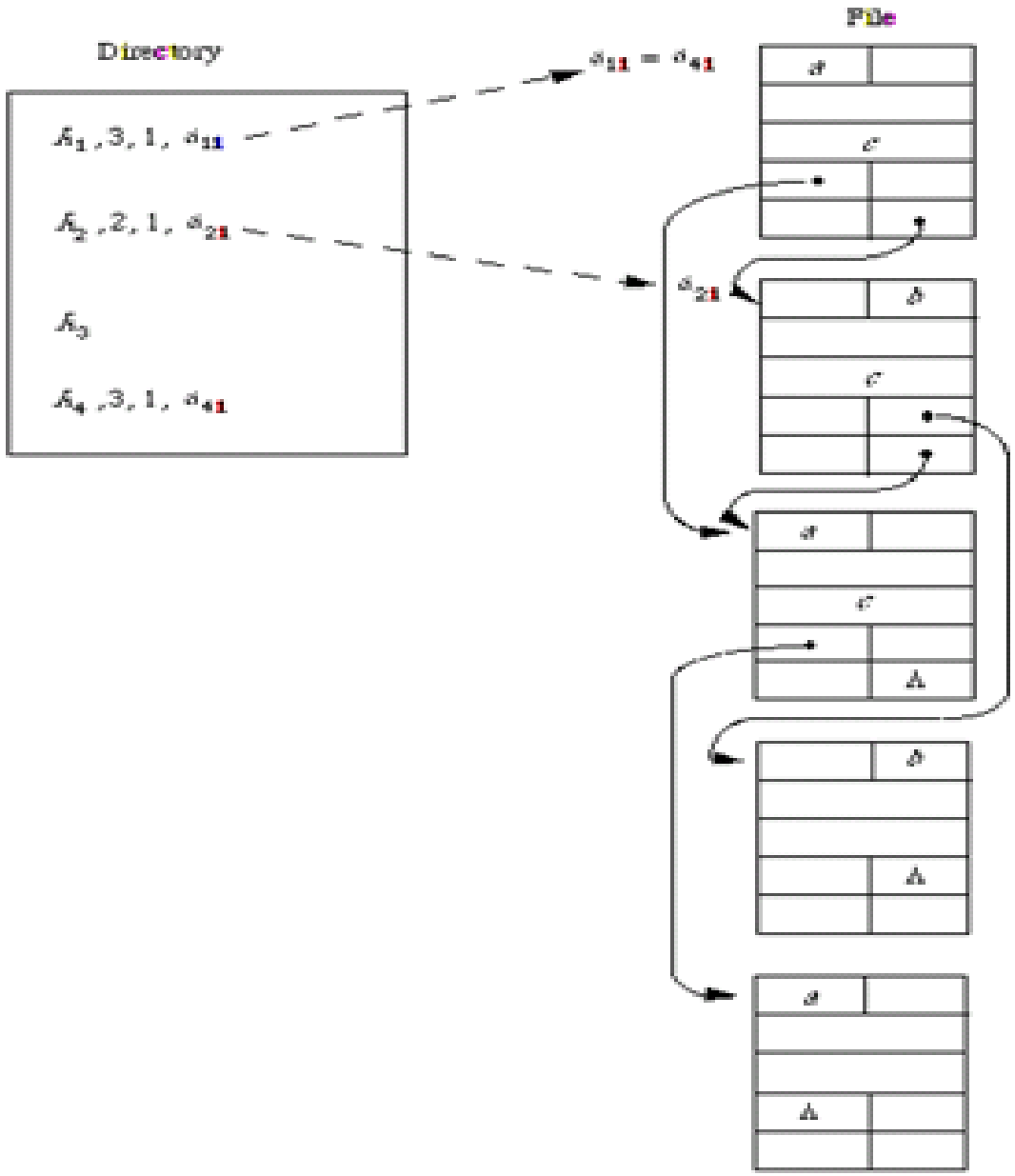


- เป็น Inverted File ที่มี 1 ลิสต์ต่อ 1 คีย์เวิร์ด

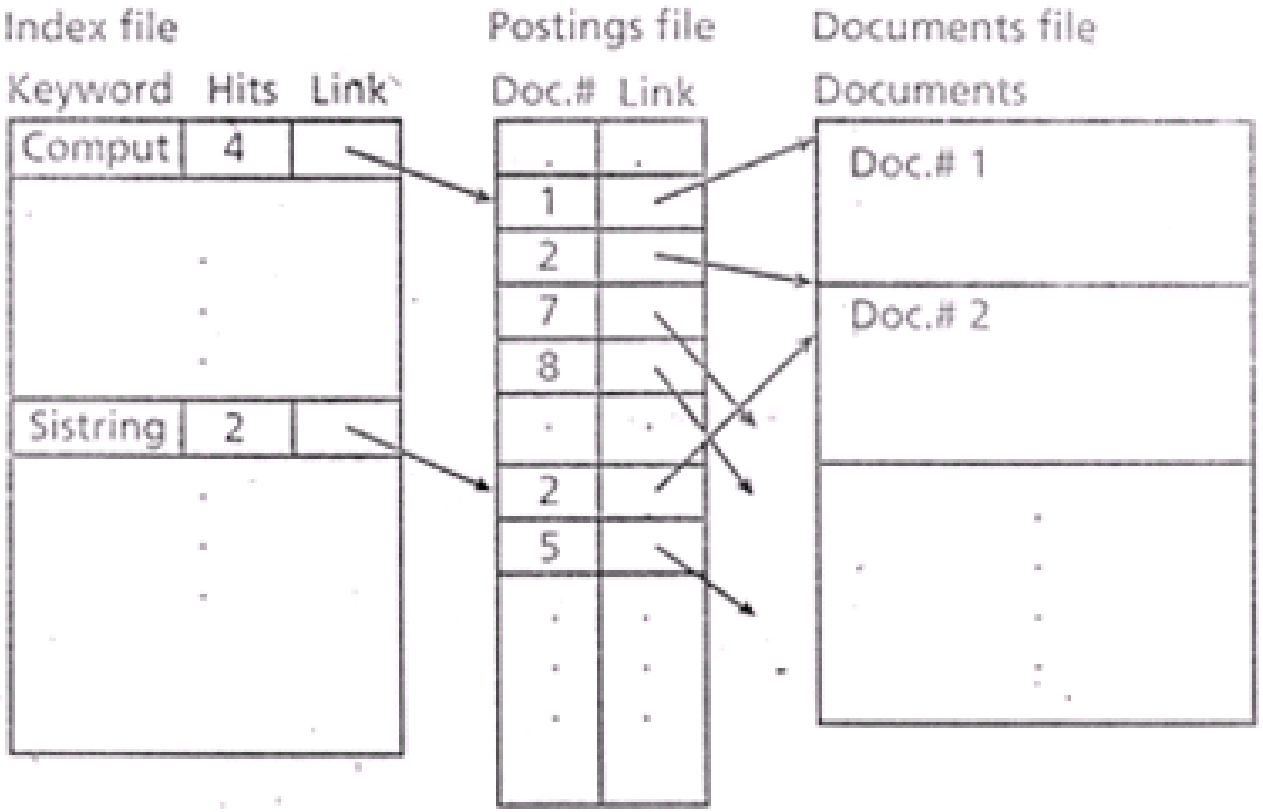
  ซึ่งไดเรคทอรี่จะมี hi = 1

  โดยไดเร็กทอรี่จะเก็บตำแหน่งที่อยู่ของเรคอร์ดแรกและมีการเชื่อมโยงเรคอร์ดที่มีคีย์เวิร์ด Ki ไว้ด้วยกัน

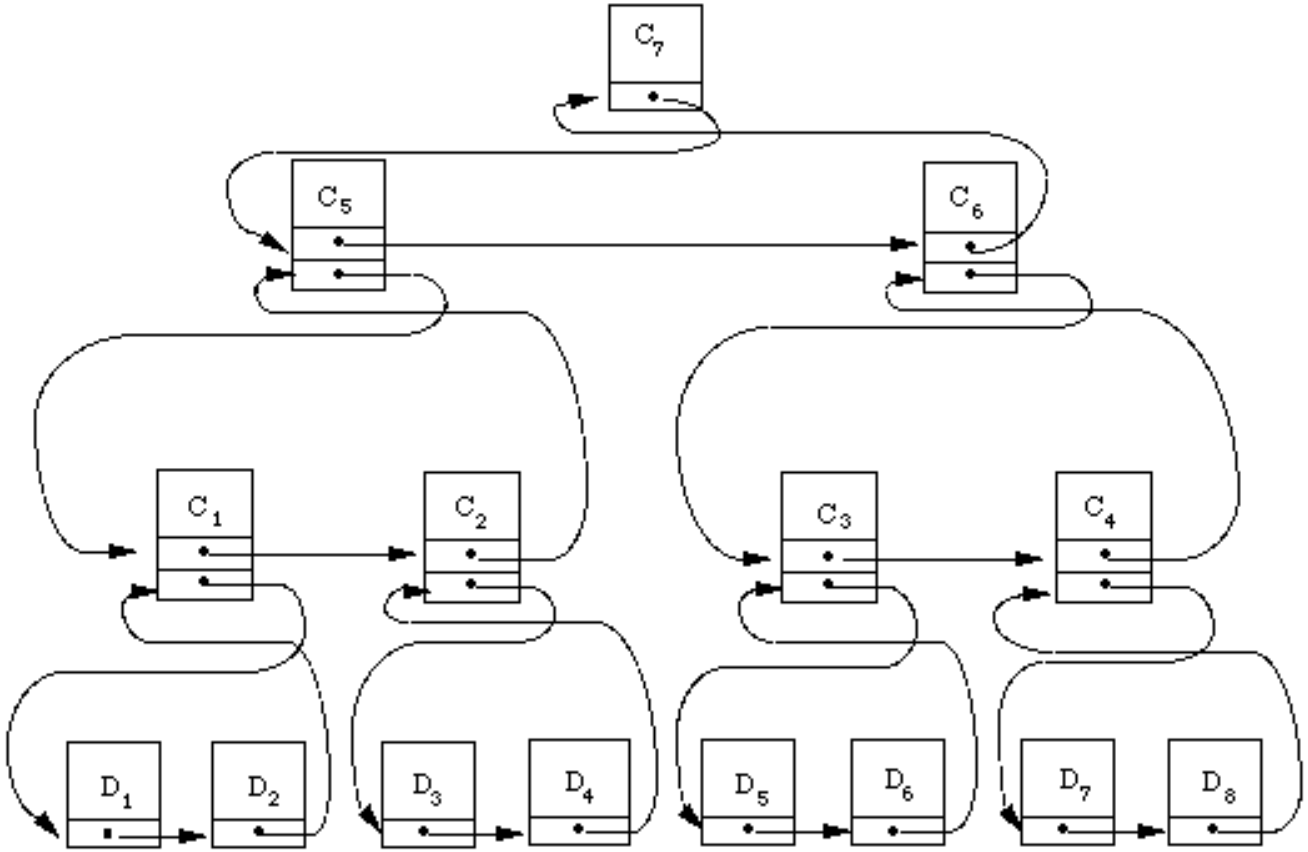
มัลติลิสต์(Multi-List)



Inverted file ที่ถูกสร้างโดยใช้วิธีการเรียงลำดับอาเรย์ (sorted array)



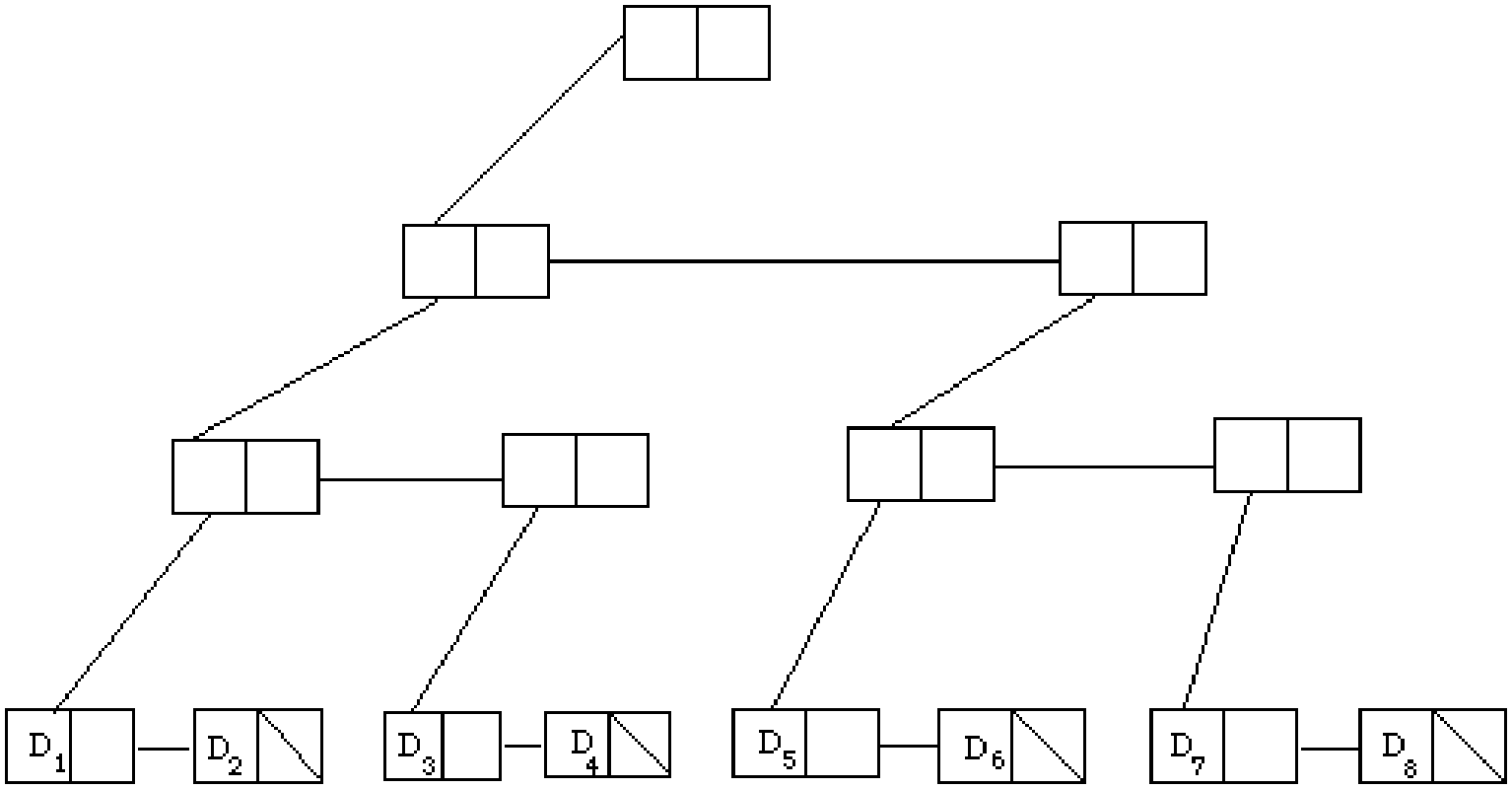
Ring Structure



ตอนต้นและตอนปลายของลิสต์จะต้องเป็นเรคอร์ดเดียวกัน

ประโยชน์ใช้ในการแบ่งชนิดของข้อมูล

Threaded List

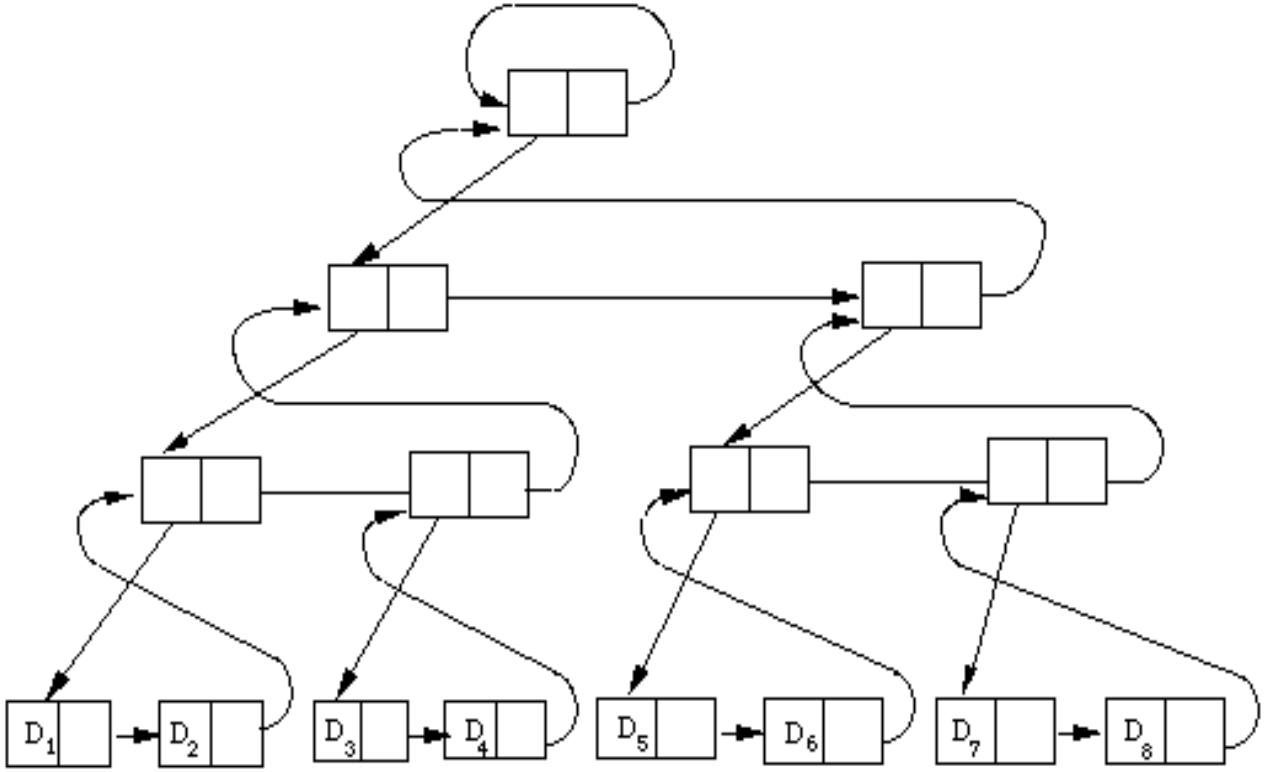


เป็นการแบ่งกลุ่มแบบง่ายๆ การแบ่งกลุ่มของเอกสารออกเป็น 4 กลุ่มคือ

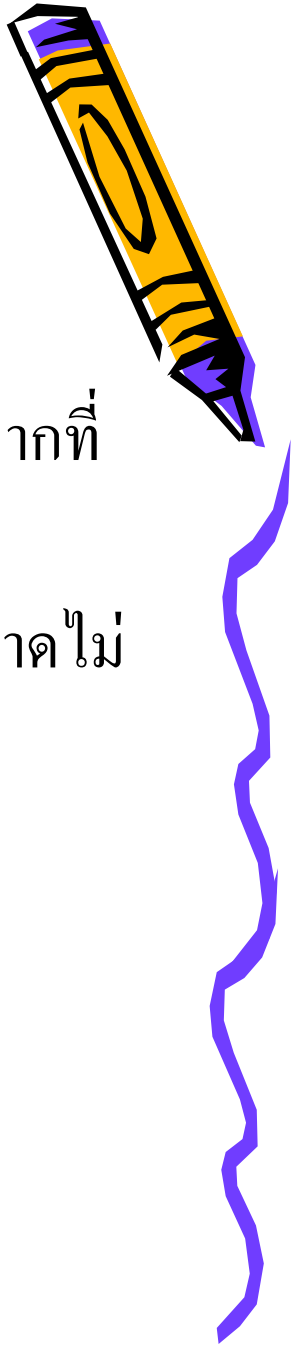
(D1,D2) , (D3,D4) , (D5,D6) , (D7,D8)

ซึ่งแต่ละลิสต์ย่อย(sublist)จะมีตัวชี้ตำแหน่ง 2 ตัวเท่านั้น

threaded list ที่สร้าง hierarchic classification

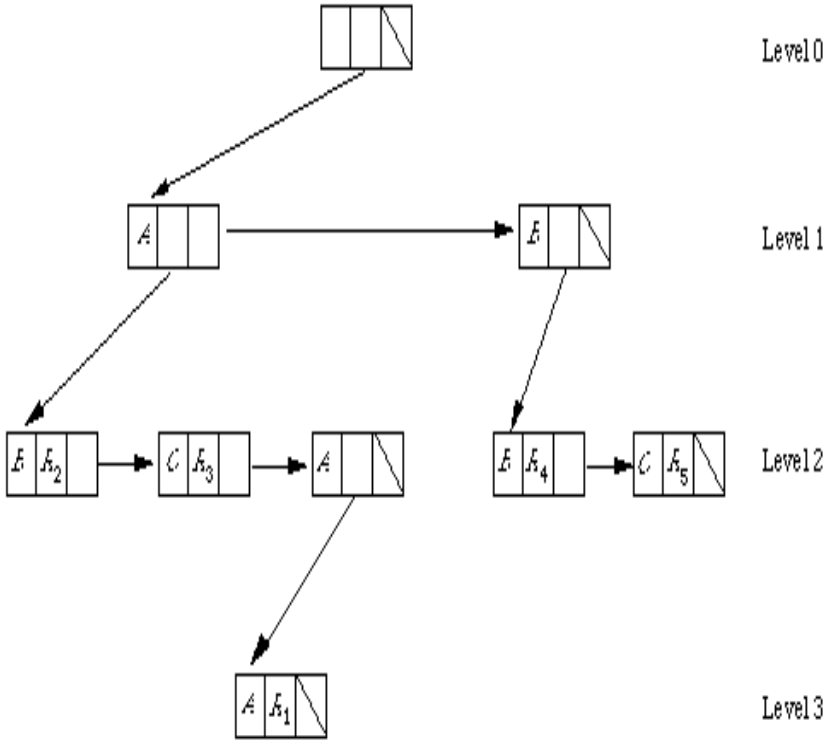


Double Chained Tree



- เป็นโครงสร้างของข้อมูลที่มีการปรับเปลี่ยนแต่ละโหนดโดยจากที่มีตัวชี้หรือ pointer 2 ตัวโดยมีการเพิ่มฟิลด์สัญลักษณ์ (Symbol) เข้าไปเพื่อนำไปใช้สำหรับการเก็บคีย์ที่มีขนาดไม่คงที่

Double Chained Tree



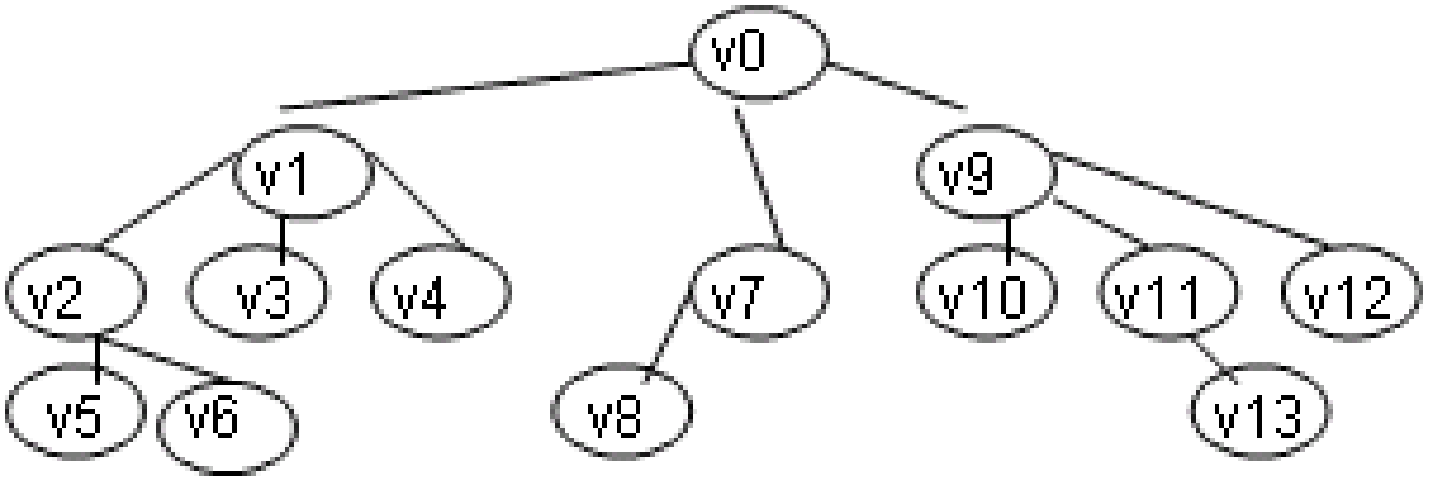
กำหนดให้ {A, B ,C } เป็นเซตของสัญลักษณ์ของคีย์

R1 ,R2, R3, R4, R5 เป็นเรคอร์ดที่เก็บบันทึก

สมมุติว่าคีย์แต่ละเรคอร์ดมีขนาดไม่เท่ากัน

R1	มีคีย์ว่า	AAA
R2	มีคีย์ว่า	AB
R3	มีคีย์ว่า	AC
R4	มีคีย์ว่า	BB
R5	มีคีย์ว่า	BC

Tree (ต้นไม้)

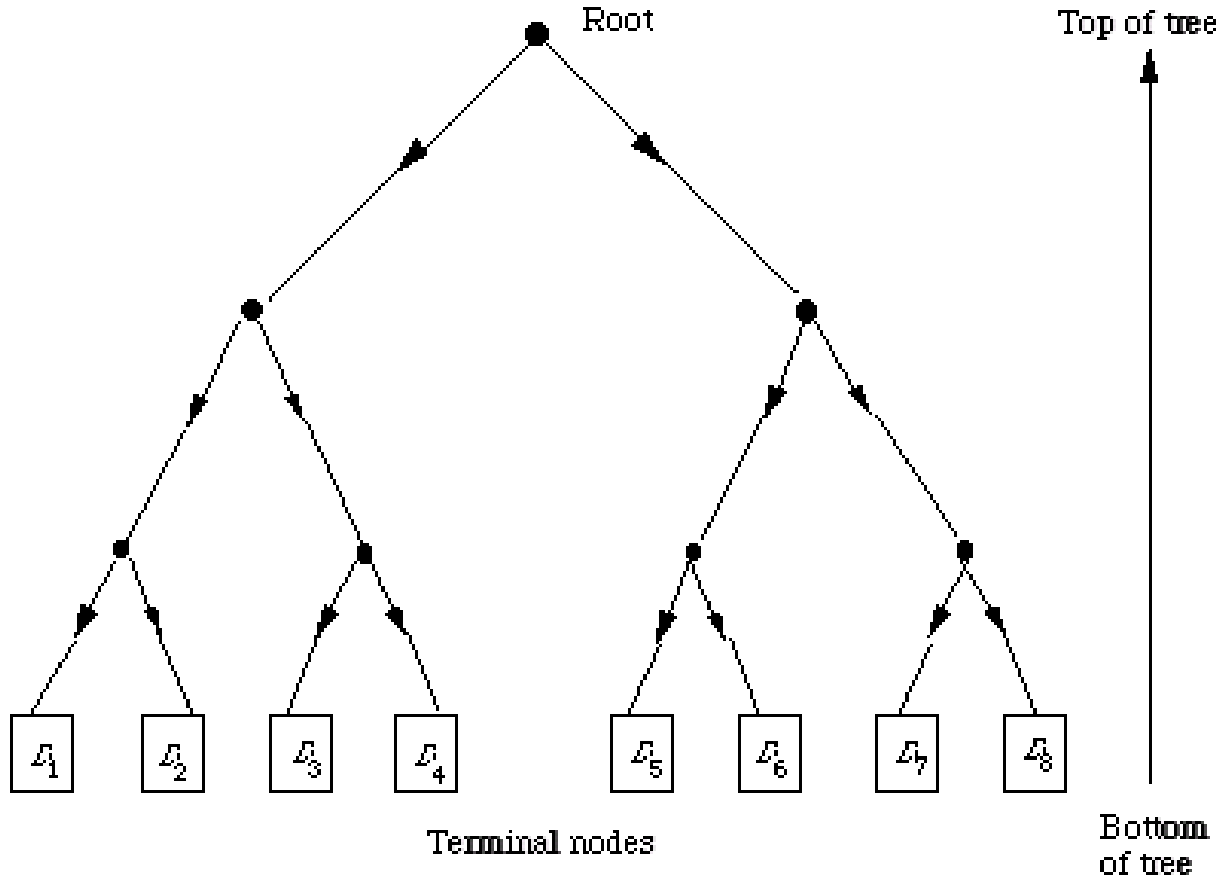


v0 เป็นราก(root node)

tree = {v0,v1,v2,v3,v4,v5,v6,v7,v8,v9,10,v11,v12,v13}

subtreeของ v0 คือ {v1,v2,v3,v4,v5,v6},{v7,v8},{v9,v10,v11,v12,v13}

โครงสร้างต้นไม้ที่แทน dendrogram



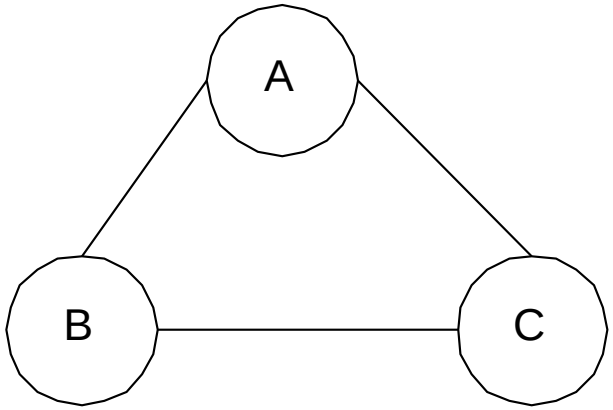
กราฟ (Graph)

กราฟประกอบด้วยเซตของโหนดและเอจ (edge) โดยที่โหนดเป็นข้อมูลเบื้องต้นของกราฟ และเอจ เป็นเส้นเชื่อม (link) ระหว่างโหนดในกราฟ ดังนั้นเราสามารถกล่าวอีกนัย คือกราฟ G ประกอบด้วย

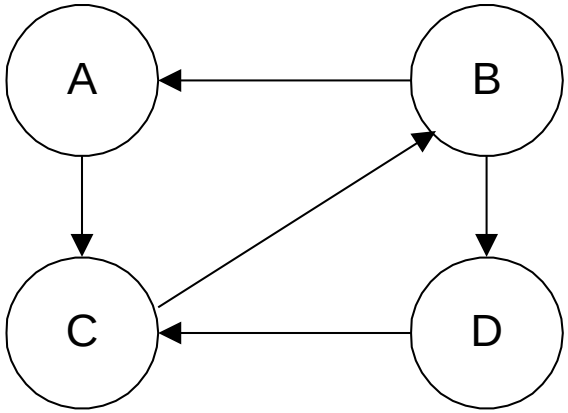
1. เซต V ของข้อมูลเรียกว่า โหนด (node) ,จุด (point) หรือ vertice
2. เซต E ของเส้นเชื่อม ซึ่งแต่ละเส้นเชื่อม e ใน E เทียบเท่ากับคู่หน่วยที่ไม่เรียงลำดับของ [u ,v] ของโหนดใน V ซึ่งสามารถแสดงได้โดย e = [u ,v]

บางครั้งเราแสดงส่วนของกราฟด้วยสมการ G = (V , E)

ตัวอย่างกราฟแบบไม่มีทิศทางและมีทิศทาง



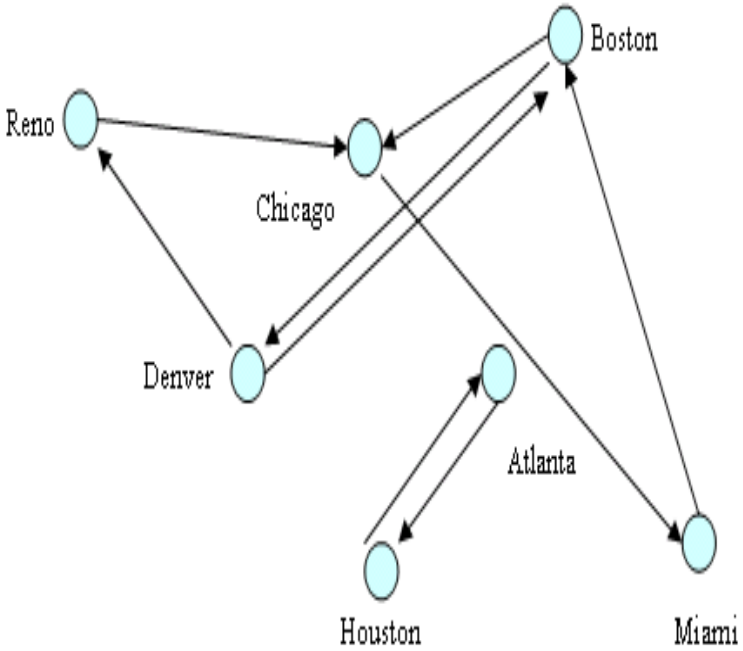
กราฟแบบไม่มีทิศทาง



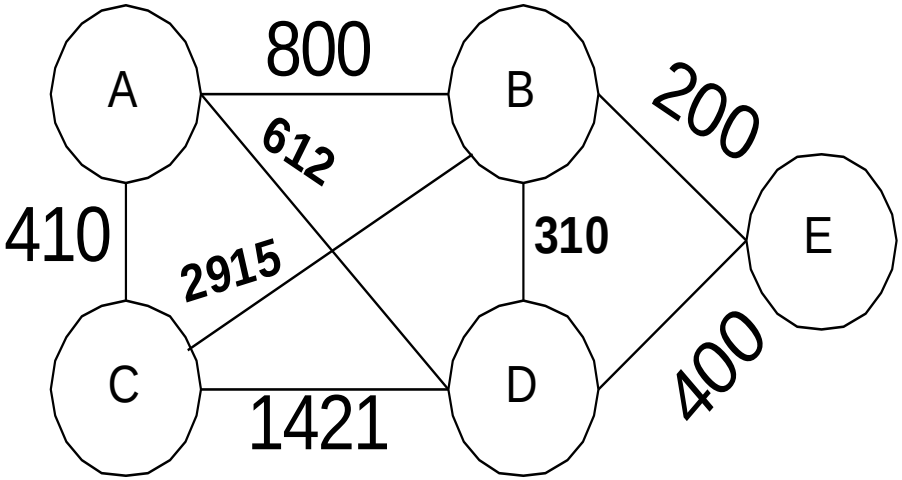
กราฟแบบมีทิศทาง

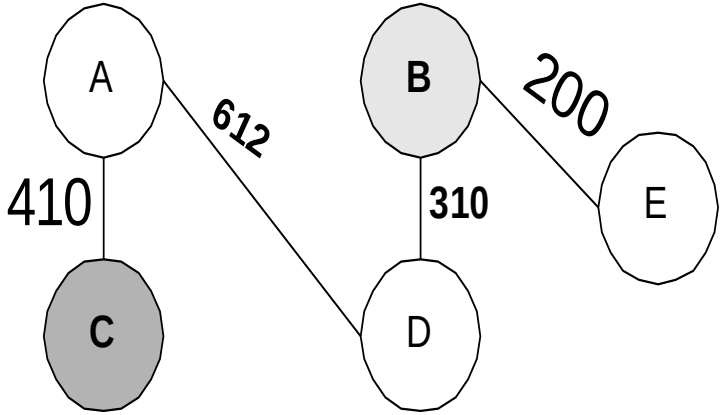
ตัวอย่างสายการบิน Friendly Airways มีเที่ยวบินประจำวัน 9 เที่ยว

103	Atlanta	to	Houston
104	Houston	to	Atlanta
201	Boston	to	Chicago
203	Boston	to	Denver
204	Denver	to	Boston
301	Denver	to	Reno
305	Chicago	to	Miami
308	Miami	to	Boston
402	Reno	to	Chicago



ตัวอย่างของข่ายงานและ Minimum Spanning





Direct File หรือScatter Storage หรือ Hash Addressing

- เทคนิคนี้เป็นการค้นหาแบบ Hashing ซึ่งใช้หลักการโดยมีค่าคีย์แบบหนึ่งในการค้นหาข้อมูล
- วิธีนี้จะแปลงค่าคีย์ให้เป็นค่าแอดเดรสของคีย์ โดยใช้ฟังก์ชั่นที่เรียกว่า Hashing function หรือ key-to-address transformation
- โดยใช้สัญลักษณ์ H(k) ซึ่ง K คือค่าคีย์ที่ใช้ในการค้นหาข้อมูล

Division method

- เป็นวิธีการหาร รูปแบบของวิธีการนี้คือ

$$H(k) = k \bmod m + 1$$

โดย K คือคีย์ที่ใช้ในการค้นหา

M คือตัวเลขจำนวนเต็มที่ใช้ในการหาร ส่วนมากเลือกเป็นเลข จำนวนเฉพาะที่ใกล้เคียงกับจำนวนข้อมูลมากที่สุด

ตัวอย่าง

หน่วยงานหนึ่งมีจำนวนสูงสุดไม่เกิน 100 เล่ม
แต่การกำหนดรหัสหนังสือเป็นตัวเลข 4 หลัก เช่น

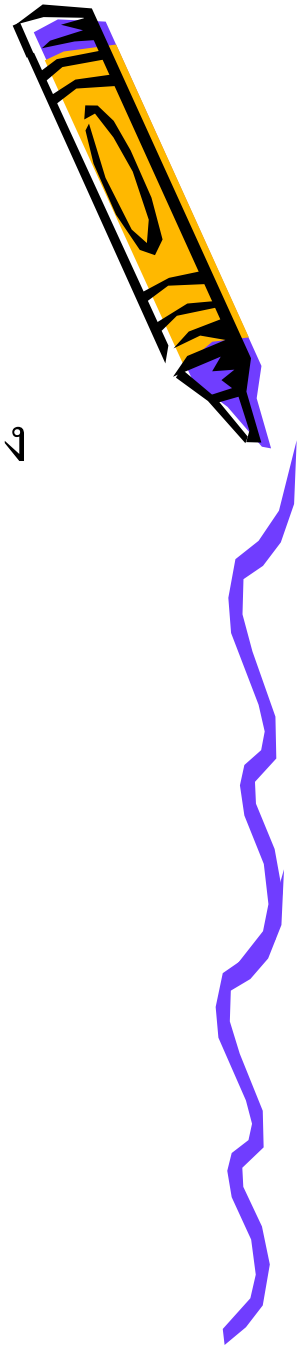
รหัสหนังสือ	*ชื่อหนังสือ*	*สำนักพิมพ์*
3205	Pascal Programming	ซีเอ็ดบุ๊ค
7148	C++	มหาวิทยาลัยรามคำแหง
2345	Programming Language	คุรุสภา
9524	JAVA	สยาม
..	.. ..	..

ตัวอย่าง

H(3205) = 3205 mod 97 + 1
= 5
H(7148) = 7148 mod 97 +1
= 68
H(2345) = 2345 mod 97 + 1
= 8

เมื่อนำค่าคีย์ซึ่งคือรหัสหนังสือมาแปลงเป็นตำแหน่งที่อยู่นั้น เมื่อนำมาหารด้วย 97 เศษที่ได้จะมีค่าระหว่าง 0 ถึง 96 ในกรณีที่เกิดการชนกันเราสามารถนำข้อมูลไปเก็บในตำแหน่งว่างถัดไปได้

Midsquare



เป็นวิธีการที่นำค่าคีย์มายกกำลังสอง แล้วนำค่าหลักที่อยู่ตรงกลางของผลลัพธ์มาเป็นค่าตำแหน่งของข้อมูล

$$H(k) = I$$

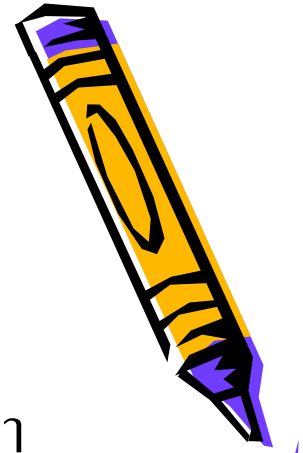
Midsquare

รหัสหนังสือ	ชื่อหนังสือ	สำนักพิมพ์
3205	Pascal Programming	ซีเอ็ดบุ๊ค

K	3205	7148	2345
K^2	10272025	51093904	5499025
H(k)	72	93	99

วิธีการนี้ในกรณีที่ค่าคีย์มีจำนวนหลักมากๆ การยกกำลังสองอาจจะส่งผลให้ยุ่งยากได้

Folding



นำค่าคีย์ที่มีจำนวนหลักมากๆ แบ่งออกเป็นส่วนๆ ส่วนแล้วนำมาพับรวมกัน ก็ได้

$$H(k) = k1 + k2 + k3 + \ldots + kr$$

วิธีนี้แบ่งค่าคีย์ เป็นส่วนๆ โดยที่แต่ละส่วนมีจำนวนหลักของเลขสำหรับชี้ตำแหน่งเท่ากัน ยกเว้นส่วนสุดท้าย ต่อจากนั้นนำส่วนต่างๆ มารวมกันแล้วตัดตัวทดออก

Folding

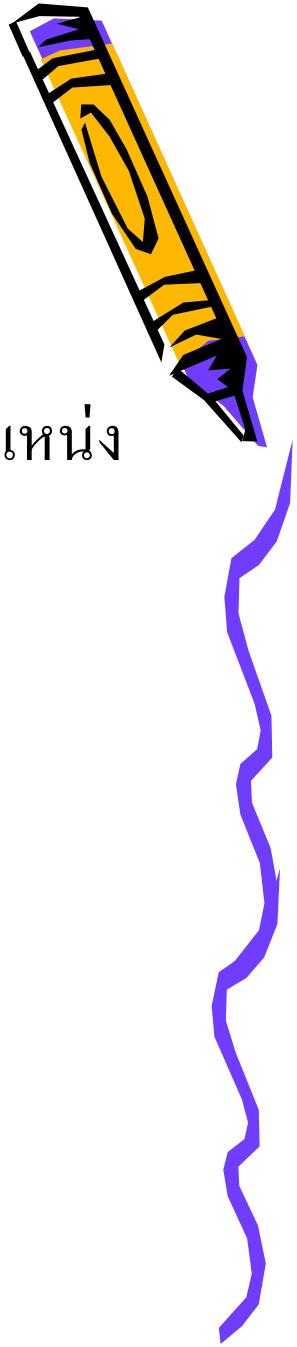
รหัสหนังสือ	ชื่อหนังสือ	สำนักพิมพ์
3205	Pascal Programming	ซีเอ็ดบุ๊ค

H(3205) = 32+05 = 37
H(7148) = 71+48 = 119 = 19
H(2345) = 23+45 = 68

หรืออาจกลับค่าส่วนที่สองก่อนการบวกก็ได้

H(3205) = 32+50 = 82
H(7148) = 71+84 = 155 = 55
H(2345) = 23+54 = 77

Open Hashing (Separate Chaining)



- เป็นการนำตำแหน่งโหนดของข้อมูลเก็บเชื่อมโยงกัน โดยตำแหน่งโหนดข้อมูลตัวแรกของลิสต์เก็บในโครงสร้างชนิดแถว

Open Hashing (Separate Chaining)