บทที่ 6
ผลการทดลองใช้วิธีการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ ซึ่งประยุกต์ใช้วิธีการจัดกลุ่มข้อมูลแบบลำดับชั้นและการให้ค่าน้ำหนัก

เนื้อหาในบทนี้เกี่ยวข้องกับ การนำเสนอผลการทดลองใช้วิธีการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ที่พัฒนาขึ้น ในโปรแกรมต้นฉบับเชิงโครงสร้างภาษาซีจำนวนสามโปรแกรม ตามที่ขั้นตอนที่อธิบายไว้ในบทที่ 5 โดยได้แบ่งการอธิบายออกเป็น 3 หัวข้อ ได้แก่ (1) ผลการคัดแยกส่วนประกอบเดิม (2) ผลการค้นหาความสัมพันธ์ระหว่างส่วนประกอบเดิม และ (3) ผลการจัดเรียงส่วนประกอบเดิม ซึ่งมีรายละเอียดดังนี้

6.1 ผลการคัดแยกส่วนประกอบเดิม

ผลที่ได้จากการทำงานในขั้นตอนนี้คือ รายการส่วนประกอบเดิม (Artifact List) ที่คัดแยกได้ ซึ่งก็คือ รายการส่วนประกอบเดิมของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน รายการส่วนประกอบเดิมของโปรแกรมแถวลำดับแถวคอยแบบผูกติด และรายการส่วนประกอบเดิมของโปรแกรมลงทะเบียนพนักงาน ดังแสดงในตารางที่ 6.1, 6.2 และ 6.3 ตามลำดับ

ตารางที่ 6.1 รายการส่วนประกอบเดิมของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน

Artifact	Abbrev.
struct stack	STK
struct queue	QUE
function initStack	INS
function initQueue	INQ
function isEmptyStack	IES
function isEmptyQueue	IEQ
function push	PSH
function enQueue	ENQ
function pop	POP
function enQueue	DEQ

ตารางที่ 6.2 รายการส่วนประกอบเดิมของโปรแกรมแถวลำดับแถวคอยแบบผูกติด

Artifact	Abbrev.
struct stack	STK
struct queue	QUE
function initStack	INS
function initQueue	INQ
function isEmptyStack	IES
function isEmptyQueue	IEQ
function push	PSH
function enQueue	ENQ
function pop	POP
function enQueue	DEQ

ตารางที่ 6.3 รายการส่วนประกอบเดิมของโปรแกรมลงทะเบียนพนักงาน

Artifact	Abbrev.
variable max_pos	MAX
variable cur_pos	CUR
variable head_ptr	HDP
struct list	LIST
struct employee	EMP
function add_pos	ADP
function clear_pos	CLP
function check_pos	CHP
function enroll	ENR
function dismiss	DSM
function init	INI
function add	ADD
function del	DEL
function search	SRC
function print	PRN

6.2 ผลการค้นหาความสัมพันธ์ระหว่างส่วนประกอบเดิม

เนื่องจากการทำงานในขั้นตอนนี้แบ่งออกเป็น 2 ขั้นตอนย่อย จึงได้แบ่งการแสดงผลออกเป็น 2 กลุ่ม คือ (1) ผลการกำหนดนิยามความสัมพันธ์ และ (2) ผลการตรวจสอบความสัมพันธ์ ซึ่งมีรายละเอียดดังนี้

1) ผลการกำหนดนิยามความสัมพันธ์

ผลที่ได้จากการทำงานในขั้นตอนนี้คือ นิยามความสัมพันธ์เฉพาะเจาะจงที่เกิดจากการนำเอานิยามความสัมพันธ์ทั่วไปสามแบบไปประยุกต์ใช้กับส่วนข้อมูลทั้งหมดของโปรแกรม ซึ่งได้แก่ นิยามความสัมพันธ์เฉพาะเจาะจงของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน นิยามความสัมพันธ์เฉพาะเจาะจงของโปรแกรมแถวลำดับแถวคอยแบบผูกติด และนิยามความสัมพันธ์เฉพาะเจาะจงของโปรแกรมลงทะเบียนพนักงาน ดังแสดงในตารางที่ 6.4, 6.5 และ 6.6 ตามลำดับ

ตารางที่ 6.4 นิยามความสัมพันธ์เฉพาะเจาะจงของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน

Definition	Description
RSTK	Return struct stack
ASTK	Has struct stack argument
USTK	Use struct stack
RQUE	Return struct queue
AQUE	Has struct queue argument
UQUE	Use struct queue

ตารางที่ 6.5 นิยามความสัมพันธ์เฉพาะเจาะจงของโปรแกรมแถวลำดับแถวคอยแบบผูกติด

Definition	Description
RSTK	Return struct stack
ASTK	Has struct stack argument
USTK	Use struct stack
RQUE	Return struct queue
AQUE	Has struct queue argument
UQUE	Use struct queue

ตารางที่ 6.6 นิยามความสัมพันธ์เฉพาะเจาะจงของโปรแกรมลงทะเบียนพนักงาน

Definition	Description
RMAX	Return variable max_pos
AMAX	Has variable max_pos argument
UMAX	Use variable max_pos
RCUR	Return variable cur_pos
ACUR	Has variable cur_pos argument
UCUR	Use variable cur_pos
RHDP	Return variable head_ptr
AHDP	Has variable head_ptr argument
UHDP	Use variable head_ptr
RLIST	Return struct list
ALIST	Has struct list argument
ULIST	Use struct list
REMP	Return struct employee
AEMP	Has struct employee argument
UEMP	Use struct employee

2) ผลการตรวจสอบความสัมพันธ์

ผลที่ได้จากการทำงานในขั้นตอนนี้คือ ตารางความสัมพันธ์ที่แสดงความเกี่ยวเนื่องระหว่างส่วนคำสั่งกับส่วนข้อมูล ซึ่งได้แก่ ตารางความสัมพันธ์ของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน ตารางความสัมพันธ์ของโปรแกรมแถวลำดับแถวคอยแบบผูกติด และตารางความสัมพันธ์ของโปรแกรมลงทะเบียนพนักงาน ดังแสดงในตารางที่ 6.7, 6.8, 6.9 และ 6.10 ตามลำดับ

ตารางที่ 6.7 ตารางความสัมพันธ์ของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน

	RSTK	ASTK	USTK	RQUE	AQUE	UQUE
INS	T		T			
INQ				T		T
IES		T	T			
IEQ					T	T
PSH		T	T			
ENQ					T	T
POP		T	T			
DEQ					T	T

ตารางที่ 6.8 ตารางความสัมพันธ์ของโปรแกรมแถวลำดับแถวคอยแบบผูกติด

	RSTK	ASTK	USTK	RQUE	AQUE	UQUE
INS	T		T			
INQ				T		T
IES		T	T			
IEQ			T		T	T
PSH		T	T			
ENQ			T		T	T
POP		T	T			
DEQ					T	T

ตารางที่ 6.9 ตารางความสัมพันธ์ของโปรแกรมลงทะเบียนพนักงาน

	RMAX	AMAX	UMAX	RCUR	ACUR	UCUR	RHDP	AHDP	UHDP
ADP			T						
CLP			T						
CHP			T			T			
ENR						T			
DSM						T			
INI									T
ADD									T
DEL									
SRC									T
PRN									T

ตารางที่ 6.10 ตารางความสัมพันธ์ของโปรแกรมลงทะเบียนพนักงาน (ต่อ)

	RLIST	ALIST	ULIST	REMP	AEMP	UEMP
ADP						
CLP						
CHP						
ENR						T
DSM						
INI			T			
ADD			T		T	
DEL		T	T			
SRC	T		T			T
PRN			T			T

6.3 ผลการจัดเรียงส่วนประกอบเดิม

เนื่องจากการทำงานในขั้นตอนนี้แบ่งออกเป็นส่วนย่อยหลายส่วน จึงได้แบ่งการแสดงผลออกเป็นกลุ่มๆ ซึ่งมีรายละเอียดดังนี้

1) ผลการสร้างเมตริกซ์ข้อมูล

ผลที่ได้จากการทำงานในขั้นตอนนี้คือ เมตริกซ์ข้อมูลที่ได้จากการแปลงตารางความสัมพันธ์ ซึ่งได้แก่ เมตริกซ์ข้อมูลของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน เมตริกซ์ข้อมูลของโปรแกรมแถวลำดับแถวคอยแบบผูกติด และเมตริกซ์ข้อมูลของโปรแกรมลงทะเบียนพนักงาน ดังแสดงในตารางที่ 6.11, 6.12 และ 6.13 ตามลำดับ

ตารางที่ 6.11 เมตริกซ์ข้อมูลของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน

INS	0.1	0.0	0.1	0.0	0.0	0.0
INQ	0.0	0.0	0.0	0.1	0.0	0.1
IES	0.0	0.1	0.1	0.0	0.0	0.0
IEQ	0.0	0.0	0.0	0.0	0.1	0.1
PSH	0.0	0.1	0.1	0.0	0.0	0.0
ENQ	0.0	0.0	0.0	0.0	0.1	0.1
POP	0.0	0.1	0.1	0.0	0.0	0.0
DEQ	0.0	0.0	0.0	0.0	0.1	0.1

ตารางที่ 6.12 เมตริกซ์ข้อมูลของโปรแกรมแถวลำดับแถวคอยแบบผูกติด

INS	0.1	0.0	0.1	0.0	0.0	0.0
INQ	0.0	0.0	0.0	0.1	0.0	0.1
IES	0.0	0.1	0.1	0.0	0.0	0.0
IEQ	0.0	0.0	0.1	0.0	0.1	0.1
PSH	0.0	0.1	0.1	0.0	0.0	0.0
ENQ	0.0	0.0	0.1	0.0	0.1	0.1
POP	0.0	0.1	0.1	0.0	0.0	0.0
DEQ	0.0	0.0	0.0	0.0	0.1	0.1

ตารางที่ 6.13 เมตริกซ์ข้อมูลของโปรแกรมลงทะเบียนพนักงาน

ADP	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
CLP	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
CHP	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ENR	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
DSM	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
INI	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
ADD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0
DEL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
SRC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0
PRN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0

2) ผลการจัดเรียงส่วนประกอบเดิม

ผลที่ได้จากการทำงานในขั้นตอนนี้คือ เมตริกซ์ความแตกต่างที่ถูกสร้างขึ้นระหว่างการรวมคลัสเตอร์ แต่เนื่องจากเมตริกซ์ดังกล่าวมีเป็นจำนวนมาก จึงได้แยกเอาเมตริกซ์เหล่านั้นออกไปแสดงในภาคผนวก ข

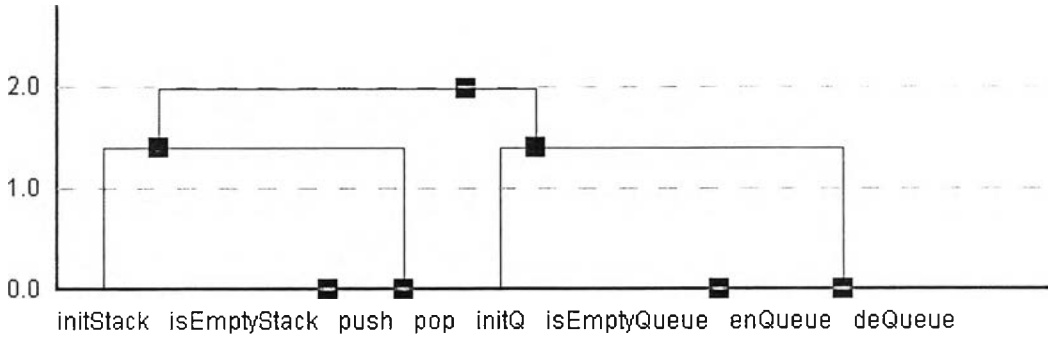
3) ผลการสร้างเดนโดแกรม

ผลที่ได้จากการทำงานในขั้นตอนนี้คือ เดนโดแกรมที่แสดงลำดับการรวมกันของคลัสเตอร์ต่างๆ ซึ่งสามารถแจกแจงตามชนิดของโปรแกรมและค่าน้ำหนักที่ใช้ทดลองได้ดังนี้

a. โปรแกรมแถวลำดับแถวคอยแบบแยกส่วน

i. ค่าน้ำหนัก -1%, -5%, -25% และ -75%

ผลการทดลองในกลุ่มนี้แสดงให้เห็นว่า วิธีการจัดกลุ่มข้อมูลฯ สามารถจัดเรียงส่วนคำสั่งเป็นคลัสเตอร์ที่มีลักษณะเป็นกลุ่มสมบูรณ์ได้ครบทั้ง 2 คลัสเตอร์ (ซึ่งได้แก่ คลัสเตอร์ที่ประกอบด้วย initStack, isEmptyStack, push, pop และคลัสเตอร์ที่ประกอบด้วย initQ isEmptyQueue, enQueue, deQueue[4]) เหมือนกันทุกค่าน้ำหนัก ไม่ว่าจะเป็น -1%, -5%, -25% หรือ -75% ดังแสดงในเดนโดแกรมรูปที่ 6.1 สำหรับกรณีค่าน้ำหนัก -1% และรูปที่ ค.1, ค.2 และ ค.3 ในภาคผนวก ค สำหรับกรณีอื่นๆ ที่เหลือ



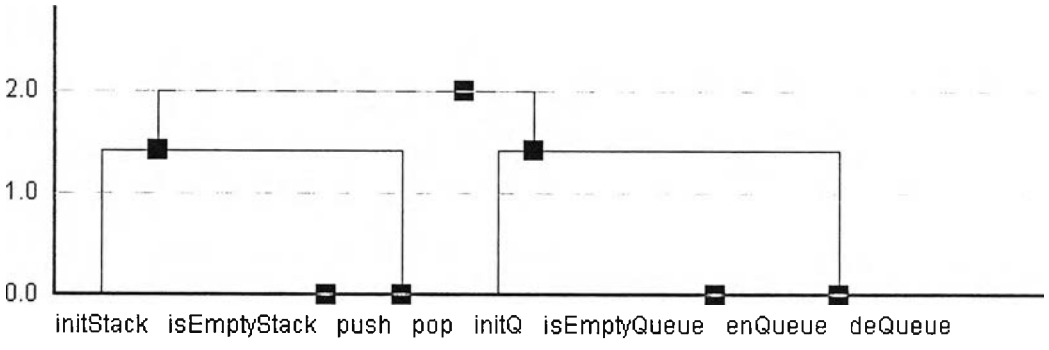
รูปที่ 6.1 เดนโดแกรมของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน
จากการใช้วิธีการจัดกลุ่มข้อมูลฯ ที่ค่าน้ำหนัก –1%

ii. ค่าน้ำหนัก +1%, +5%, +25% และ +75%

ผลการทดลองในกลุ่มนี้แสดงให้เห็นว่า วิธีการจัดกลุ่มข้อมูลฯ สามารถจัดเรียงส่วนคำสั่งเป็นคลัสเตอร์ที่มีลักษณะเป็นกลุ่มสมบูรณ์ได้ครบทั้ง 2 คลัสเตอร์ (ซึ่งได้แก่ คลัสเตอร์ที่ประกอบด้วย initStack, isEmptyStack, push, pop และคลัส

[4] ในการอธิบายเดนโดแกรมจำเป็นต้องเรียกฟังก์ชันโดยใช้ชื่อเต็ม (ซึ่งแตกต่างจากการอธิบายเมตริกซ์) ทั้งนี้เพื่อให้สอดคล้องกับชื่อในแผนภาพซึ่งวาดโดยเครื่องมือระบุวัตถุซอฟต์แวร์ฯ

เตอร์ที่ประกอบด้วย initQ isEmptyQueue, enQueue, deQueue) เหมือนกันทุกค่าน้ำหนัก ไม่ว่าจะเป็น +1%, +5%, +25% หรือ +75% ดังแสดงในเดนโดแกรมรูปที่ 6.2 สำหรับกรณีค่าน้ำหนัก +1% และรูปที่ ค.4 ค.5 และ ค.6 ในภาคผนวก ค สำหรับกรณีอื่นๆ ที่เหลือ

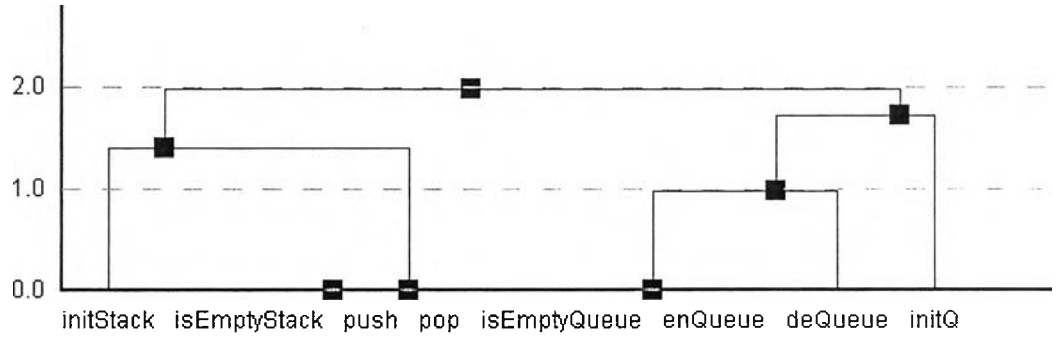


รูปที่ 6.2 เดนโดแกรมของโปรแกรมแถวลำดับแถวคอยแบบแยกส่วน
จากการใช้วิธีการจัดกลุ่มข้อมูลฯ ที่ค่าน้ำหนัก +1%

b. โปรแกรมแถวลำดับแถวคอยแบบผูกติด

i. ค่าน้ำหนัก -1%, -5%, -25% และ -75%

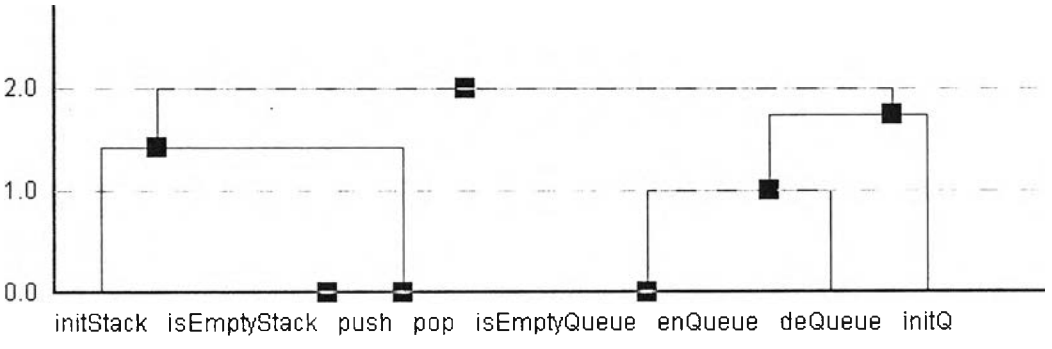
ผลการทดลองในกลุ่มนี้แสดงให้เห็นว่า วิธีการจัดกลุ่มข้อมูลฯ สามารถจัดเรียงส่วนคำสั่งเป็นคลัสเตอร์ที่มีลักษณะเป็นกลุ่มสมบูรณ์ได้ครบทั้ง 2 คลัสเตอร์ (ซึ่งได้แก่ คลัสเตอร์ที่ประกอบด้วย initStack, isEmptyStack, push, pop และคลัสเตอร์ที่ประกอบด้วย initQ isEmptyQueue, enQueue, deQueue) เหมือนกันทุกค่าน้ำหนัก ไม่ว่าจะเป็น -1%, -5%, -25% หรือ -75% ดังแสดงในเดนโดแกรมรูปที่ 6.3 สำหรับกรณีค่าน้ำหนัก -1% และรูปที่ ค.7, ค.8 และ ค.9 ในภาคผนวก ค สำหรับกรณีอื่นๆ ที่เหลือ



รูปที่ 6.3 เดนโดแกรมของโปรแกรมแถวลำดับแถวคอยแบบผูกติด
จากการใช้วิธีการจัดกลุ่มข้อมูลฯ ที่ค่าน้ำหนัก -1%

ii. ค่าน้ำหนัก +1%, +5%, +25% และ +75%

ผลการทดลองในกลุ่มนี้แสดงให้เห็นว่า วิธีการจัดกลุ่มข้อมูลฯ สามารถจัดเรียงส่วนคำสั่งเป็นคลัสเตอร์ที่มีลักษณะเป็นกลุ่มสมบูรณ์ได้ครบทั้ง 2 คลัสเตอร์ (ซึ่งได้แก่ คลัสเตอร์ที่ประกอบด้วย initStack, isEmptyStack, push, pop และคลัสเตอร์ที่ประกอบด้วย initQ isEmptyQueue, enQueue, deQueue) เหมือนกันทุกค่าน้ำหนัก ไม่ว่าจะเป็น +1%, +5%, +25% หรือ +75% ดังแสดงในเดนโดแกรมรูปที่ 6.4 สำหรับกรณีค่าน้ำหนัก +1% และรูปที่ ค.10, ค.11 และ ค.12 ในภาคผนวก ค สำหรับกรณีอื่นๆ ที่เหลือ

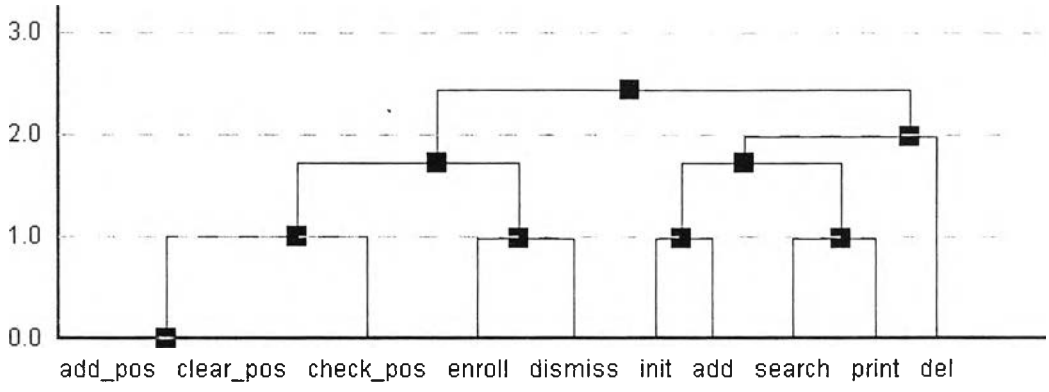


รูปที่ 6.4 เดนโดแกรมของโปรแกรมแถวลำดับแถวคอยแบบผูกติด
จากการใช้วิธีการจัดกลุ่มข้อมูลฯ ที่ค่าน้ำหนัก +1%

c. โปรแกรมแถวลงทะเบียนพนักงาน

i. ค่าน้ำหนัก –1%, -5%, -25%

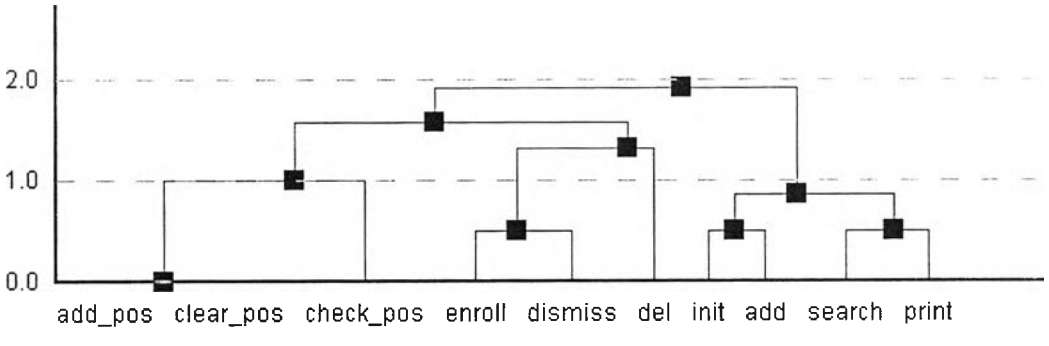
ผลการทดลองในกลุ่มนี้แสดงให้เห็นว่า วิธีการจัดกลุ่มข้อมูลฯ สามารถจัดเรียงส่วนคำสั่งเป็นคลัสเตอร์ที่มีลักษณะเป็นกลุ่มสมบูรณ์ได้ครบทั้ง 2 คลัสเตอร์ (ซึ่งได้แก่ คลัสเตอร์ที่ประกอบด้วย add_pos, clear_pos, check_pos, enroll, dismiss และคลัสเตอร์ที่ประกอบด้วย init, add, search, print, del) เหมือนกันทุกค่าน้ำหนัก ไม่ว่าจะเป็น -1%, -5% หรือ -25% ดังแสดงในเดนโดแกรมรูปที่ 6.5 สำหรับกรณีค่าน้ำหนัก -1% และรูปที่ ค.13 และ ค.14 ในภาคผนวก ค สำหรับกรณีอื่นๆ ที่เหลือ



รูปที่ 6.5 เดนโดแกรมของโปรแกรมลงทะเบียนพนักงาน
จากการใช้วิธีการจัดกลุ่มข้อมูลฯ ที่ค่าน้ำหนัก -1%

ii. ค่าน้ำหนัก –75%

ผลการทดลองในกลุ่มนี้แสดงให้เห็นว่า วิธีการจัดกลุ่มข้อมูลฯ ไม่สามารถจัดเรียงส่วนคำสั่งเป็นคลัสเตอร์ที่มีลักษณะเป็นกลุ่มสมบูรณ์ได้ ทั้งนี้เพราะคลัสเตอร์ del ถูกนำรวมเข้ากับคลัสเตอร์ enroll+dismiss ซึ่งทำให้เกิดคลัสเตอร์ที่ประกอบด้วย add_pos, clear_pos, check_pos, enroll, dismiss, del อันเป็นคลัสเตอร์ที่ไม่มีความหมาย ส่วนคลัสเตอร์ที่ประกอบด้วย init, add, search, print ซึ่งขาดคลัสเตอร์ del ไป ก็กลายเป็นคลัสเตอร์ที่ไม่มีความหมายเช่นเดียวกัน ดังแสดงในเดนโดแกรมรูปที่ 6.6

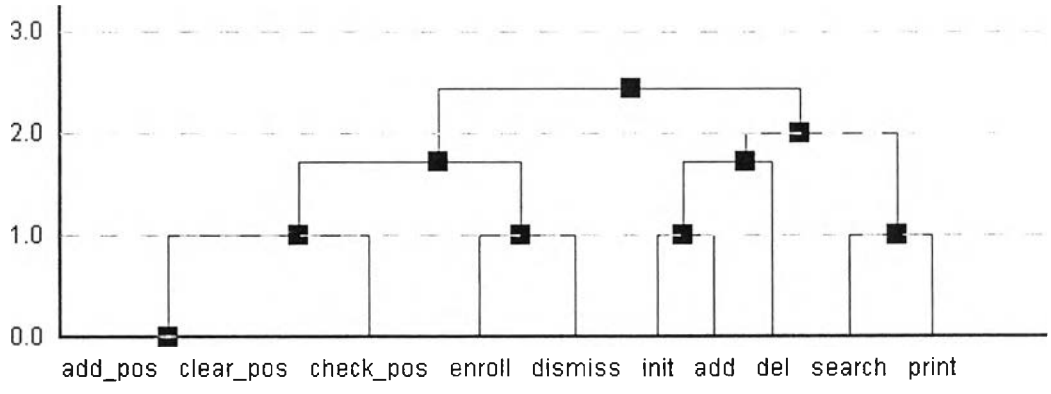


รูปที่ 6.6 เดนโดแกรมของโปรแกรมลงทะเบียนพนักงาน
จากการใช้วิธีการจัดกลุ่มข้อมูลฯ ที่ค่าน้ำหนัก -75%

iii. ค่าน้ำหนัก +1%, +5%, +25%, +75%

ผลการทดลองในกลุ่มนี้แสดงให้เห็นว่า วิธีการจัดกลุ่มข้อมูลฯ สามารถจัดเรียงส่วนคำสั่งเป็นคลัสเตอร์ที่มีลักษณะเป็นกลุ่มสมบูรณ์ได้ครบทั้ง 2 คลัสเตอร์ (ซึ่งได้แก่ คลัสเตอร์ที่ประกอบด้วย add_pos, clear_pos, check_pos, enroll, dismiss และคลัสเตอร์ที่ประกอบด้วย init, add, search, print, del) เหมือนกันทุก

ค่าน้ำหนัก ไม่ว่าจะเป็น +1%, +5%, +25% หรือ +75% ดังแสดงในเดนโดแกรมรูปที่ 6.7 สำหรับกรณีค่าน้ำหนัก +1% และรูปที่ ค.15, ค.16 และ ค.17 ในภาคผนวก ค สำหรับกรณีอื่นๆ ที่เหลือ ทั้งนี้มีข้อสังเกตว่า ลำดับการสร้างกลุ่มสมบูรณ์ init, add, search, print, del ในกรณีนี้แตกต่างจากลำดับการสร้างกลุ่มสมบูรณ์ในกรณีค่าน้ำหนัก -1%, -5% และ -25%



รูปที่ 6.7 เดนโดแกรมของโปรแกรมลงทะเบียนพนักงาน
จากการใช้วิธีการจัดกลุ่มข้อมูลฯ ที่ค่าน้ำหนัก +1%