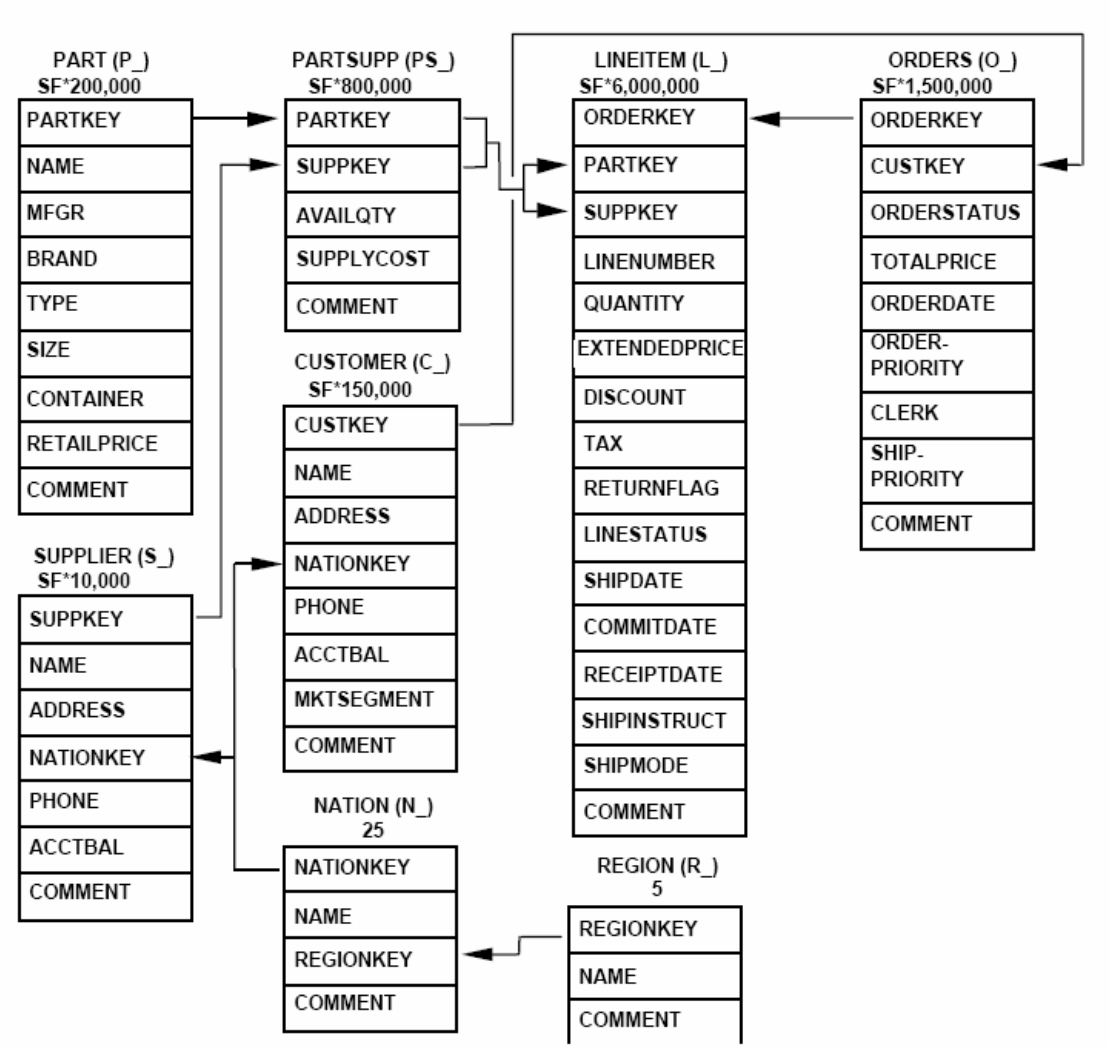
ภาคผนวก ก

ก.1 โครงสร้างของการวัดเปรียบเทียบสมรรถนะของ TPC-H

ข้อมูลทดสอบที่ใช้ในวิทยานิพนธ์นี้ ได้มาจากการวัดเปรียบเทียบสมรรถนะของ TPC-H ซึ่งเป็นตัววัดเปรียบเทียบสมรรถนะที่ใช้ในระบบสนับสนุนการตัดสินใจ (Decision Support) โดยต้องการทดสอบกับข้อมูลที่มีปริมาณมาก ประมวลผลการสอบถามที่เป็นแบบซับซ้อน ซึ่งฐานข้อมูลของการวัดเปรียบเทียบสมรรถนะของ TPC-H นี้ มีโครงสร้างฐานข้อมูลดังนี้



ก.1.1 โครงสร้างตาราง (Table Layouts) จากการวัดเปรียบเทียบสมรรถนะของ TPC-H

ฐานข้อมูลของการวัดเปรียบเทียบสมรรถนะของ TPC-H จะประกอบด้วยตารางเชิงความสัมพันธ์หลายตาราง เช่น ตาราง CUSTOMER จะเป็นตารางเชิงความสัมพันธ์ที่เก็บข้อมูลเกี่ยวกับลูกค้า ตาราง PART จะเป็นตารางเชิงความสัมพันธ์ที่เก็บข้อมูลเกี่ยวกับชิ้นส่วนของสินค้า โดยมีโครงสร้างตารางดังนี้

โครงสร้างตาราง CUSTOMER

ชื่อแอทริบิวต์	ชนิดข้อมูล	หมายเหตุ
C_CUSTKEY	identifier	SF*150,000 are populated
C_NAME	variable text, size 25	
C_ADDRESS	variable text, size 40	
C_NATIONKEY	identifier	Foreign key reference to N_NATIONKEY
C_PHONE	fixed text, size 15	
C_ACCTBAL	decimal	*8-byte decimals*
C_MKTSEGMENT	fixed text, size 10	
C_COMMENT	variable text, size 117	

คีย์หลักแบบผสม (Compound Primary Key) : C_CUSTKEY

ขนาด (ไบต์ต่อเรคอร์ด) :

223

ขนาดตาราง (ไบต์ต่อ SF=1) :

33,450,000

โครงสร้างตาราง PART

ชื่อแอทริบิวต์	ชนิดข้อมูล	หมายเหตุ
P_PARTKEY	identifier	SF*200,000 are populated
P_NAME	variable text, size 55	
P_MFGR	fixed text, size 25	
P_BRAND	fixed text, size 10	
P_TYPE	variable text, size 25	
P_SIZE	integer	*4-byte integers*
P_CONTAINER	fixed text, size 10	
P_RETAILPRICE	decimal	*8-byte decimals*
P_COMMENT	variable text, size 23	

คีย์หลัก (Primary Key) :

P_PARTKEY

ขนาด (ไบต์ต่อเรคอร์ด) :

164

ขนาดตาราง (ไบต์ต่อ SF=1) :

32,800,000

ก.2 ตัวอย่างการสอบถามของการวัดเปรียบเทียบสมรรถนะของ TPC-H

ในการวัดเปรียบเทียบสมรรถนะของ TPC-H จะมีแบบสอบถามข้อมูลที่ซับซ้อนมาให้ ซึ่งมีทั้งที่เป็นการสอบถามข้อมูลแบบค่าเท่ากัน แบบความเป็นสมาชิก แบบช่วงข้างเดียว และแบบช่วง 2 ข้าง แต่เนื่องจากในงานวิทยานิพนธ์นี้เกี่ยวข้องกับการสอบถามแบบค่าเท่ากันและแบบความเป็นสมาชิก จึงได้ทำการเลือกเฉพาะแบบสอบถามที่เป็นแบบค่าเท่ากันและแบบความเป็นสมาชิก ดังตัวอย่างในตาราง

การสอบถาม	ตาราง	แอทริบิวต์	เงื่อนไขการสอบถาม	ลักษณะการสอบถาม
Q3	customer	c_mktsegment	c_mktsegment = ":1"	equality query
Q8	part	p_type	p_type = ":3"	equality query
Q16	part	p_size	p_size in (:3, :4, :5, :6, :7, :8, :9, :10)	IN-lists
Q19	part	p_brand p_container	p_brand = ":1" p_container in ("SM CASE", "SM BOX", "SM PACK", "SM PKT")	equality query IN-lists

ก.3 ตัวอย่างการกรองข้อมูลเฉพาะแอทริบิวต์ที่จะนำมาสร้างดัชนี

ข้อมูลทดสอบที่นำมาจากการวัดเปรียบเทียบสมรรถนะของ TPC-H จะอยู่ในรูปของ Flat File ในการทำดัชนีนั้น เราจะต้องทำการแปลงข้อมูลเหล่านั้นให้อยู่ในรูปที่พร้อมจะทำดัชนี ดังนั้นเราจึงต้องเลือกเฉพาะแอทริบิวต์ที่เราสนใจมาทำดัชนี โดยการใช้คำสั่ง awk บนระบบปฏิบัติการ Linux ดังนี้

- เลือกแอทริบิวต์ c_mktsegment (มีคาร์ดินอลิตี้ เท่ากับ 5) ในแฟ้มข้อมูล customer.tbl โดยการใช้คำสั่ง awk -F\| '{print $7}' customer.tbl > c_out07
- เลือกแอทริบิวต์ p_brand (มีคาร์ดินอลิตี้ เท่ากับ 25) ในแฟ้มข้อมูล part.tbl โดยการใช้คำสั่ง awk -F\| '{print $4}' part.tbl > p_out04
- เลือกแอทริบิวต์ p_size (มีคาร์ดินอลิตี้ เท่ากับ 50) ในแฟ้มข้อมูล part.tbl โดยการใช้คำสั่ง awk -F\| '{print $6}' part.tbl > p_out06

ก.4 ตัวอย่างการสอบถามข้อมูลแบบซับซ้อน (Complex Queries) และการแก้ปัญหาโดยใช้ดัชนีแบบบิตแมป

ในการดึงข้อมูลหรือสารสนเทศจากคลังข้อมูลขึ้นมาใช้งานแต่ละครั้งมักเป็นการสอบถามข้อมูลแบบซับซ้อนและแบบทันทีทันใด ซึ่งทำให้การค้นหาข้อมูลใช้เวลามากขึ้น ดังนั้นเราสามารถเพิ่มประสิทธิภาพในการค้นหาข้อมูลให้รวดเร็วขึ้นได้ โดยการใช้ดัชนีแบบบิตแมปมาแก้ปัญหาลดความซับซ้อนของการสอบถามเหล่านั้นลง ให้อยู่ในรูปของการสอบถามที่กระชับและง่ายขึ้น โดยการใช้คุณสมบัติของการดำเนินการตรรกะระดับบิตก่อนเข้าถึงข้อมูลจริง ดังตัวอย่าง ก-1 และ ก-2

ตัวอย่าง ก-1

```
SELECT T.year, T.month, L.state_or_province, I.item_name,
        SUM(dollars_sold)
FROM location L,
     sales S,
     items I,
     time T
WHERE S.item_key=I.item_key
      AND S.time_key=T.time_key
      AND T.year in (2003, 2004)
      AND T.month in (1-05-2003, 1-07-2003, 1-09-2003,
          1-11-2003, 1-01-2004, 1-03-2004)
      AND L.state_or_province in ("Phuket", "Songkhla", "Trang",
          "Krabi", "Chiang Mai")
      AND I.item_name = "milk"
GROUP BY T.year, T.month, L.city, I.item_name;
```

<u>การแก้ปัญหาโดยใช้ดัชนีแบบบิตแมป</u> โดยการ

- เลือกแอทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ ๆ ที่เป็นเงื่อนไขหลังอนุประโยค WHERE มาสร้างดัชนีแบบบิตแมป ได้แก่ แอทริบิวต์
  - T.year
  - T.month
  - L.state_or_province

- I.item_name

- ค้นหาข้อมูลจากดัชนีแบบบิตแมปที่สร้างขึ้นของแต่ละแอทริบิวต์ ได้แก่

1) การค้นหาข้อมูลแบบค่าเท่ากัน ได้แก่ I.item_name = “milk”

2) การค้นหาข้อมูลแบบความเป็นสมาชิก ได้แก่

2.1) T.year in (2003, 2004) โดยการค้นหาข้อมูล

2.1.1) T.year = “2003”

2.1.2) T.year = “2004”

2.1.3) นำผลลัพธ์จากข้อ 2.1.1) และ 2.1.2) มาดำเนินการตรรกะ OR ระดับบิตต่อบิต

2.2) T.month in (1-05-2003, 1-07-2003, 1-09-2003, 1-11-2003, 1-01-2004, 1-03-2004) โดยการค้นหาข้อมูล

2.2.1) T.month = “1-05-2003”

2.2.2) T.month = “1-07-2003”

2.2.3) T.month = “1-09-2003”

2.2.4) T.month = “1-11-2003”

2.2.5) T.month = “1-01-2004”

2.2.6) T.month = “1-03-2004”

2.2.7) นำผลลัพธ์จากข้อ 2.2.1) จนถึงข้อ 2.2.6) มาดำเนินการตรรกะ OR ระดับบิตต่อบิต

2.3) L.state_or_province in (“Phuket”, “Songkhla”, “Trang”, “Krabi”, “Chiang Mai”) โดยการค้นหาข้อมูล

2.3.1) L.state_or_province = “Phuket”

2.3.2) L.state_or_province = “Songkhla”

2.3.3) L.state_or_province = “Trang”

2.3.4) L.state_or_province = “Krabi”

2.3.5) L.state_or_province = “Chiang Mai”

2.3.6) นำผลลัพธ์จากข้อ 2.3.1) จนถึงข้อ 2.3.5) มาดำเนินการตรรกะ OR ระดับบิตต่อบิต

- นำผลลัพธ์จากข้อ 1), 2.1.3), 2.2.7) และ 2.3.6) มาดำเนินการตรรกะ AND ระหว่างบิตแมปเวกเตอร์ ในระดับบิตต่อบิต

ตัวอย่าง ก-2

```
SELECT B.branch_name, L.state_or_province, T.year, T.month,
        SUM(units_sold)
FROM    branch B,
        location L,
        time T,
        sales S,
WHERE S.branch_key = B.branch_key
        AND S.location_key = L.location_key
        AND S.time_key = T.time_key
        AND T.year in (2003, 2004)
        AND T.month in (2, 4, 6, 8, 10, 12)
        AND C_state_or_province = "Phuket"
GROUP BY L.state_or_province
```

<u>การแก้ปัญหาโดยใช้ดัชนีแบบบิตแมป</u> โดยการ

- เลือกแอทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ ๆ ที่เป็นเงื่อนไขหลังอนุประโยค WHERE มาสร้างดัชนีแบบบิตแมป ได้แก่ แอทริบิวต์
  - T.year
  - T.month
  - C.state_or_province
- ค้นหาข้อมูลจากดัชนีแบบบิตแมปที่สร้างขึ้นของแต่ละแอทริบิวต์ ได้แก่

1) การค้นหาข้อมูลแบบค่าเท่ากัน ได้แก่ C_state_or_province = "Phuket"

2) การค้นหาข้อมูลแบบความเป็นสมาชิก ได้แก่

2.1) T.year in (2003, 2004) โดยการค้นหาข้อมูล

2.1.1) T.year = "2003"

2.1.2) T.year = "2004"

2.1.3) นำผลลัพธ์จากข้อ 2.1.1) และ 2.1.2) มาดำเนินการตรรกะ OR ระดับบิตต่อบิต

2.2) T.month in T.month in (2, 4, 6, 8, 10, 12) โดยการค้นหาข้อมูล

2.2.1) T.month = "2"

2.2.2) T.month = “4”

2.2.3) T.month = “6”

2.2.4) T.month = “8”

2.2.5) T.month = “10”

2.2.6) T.month = “12”

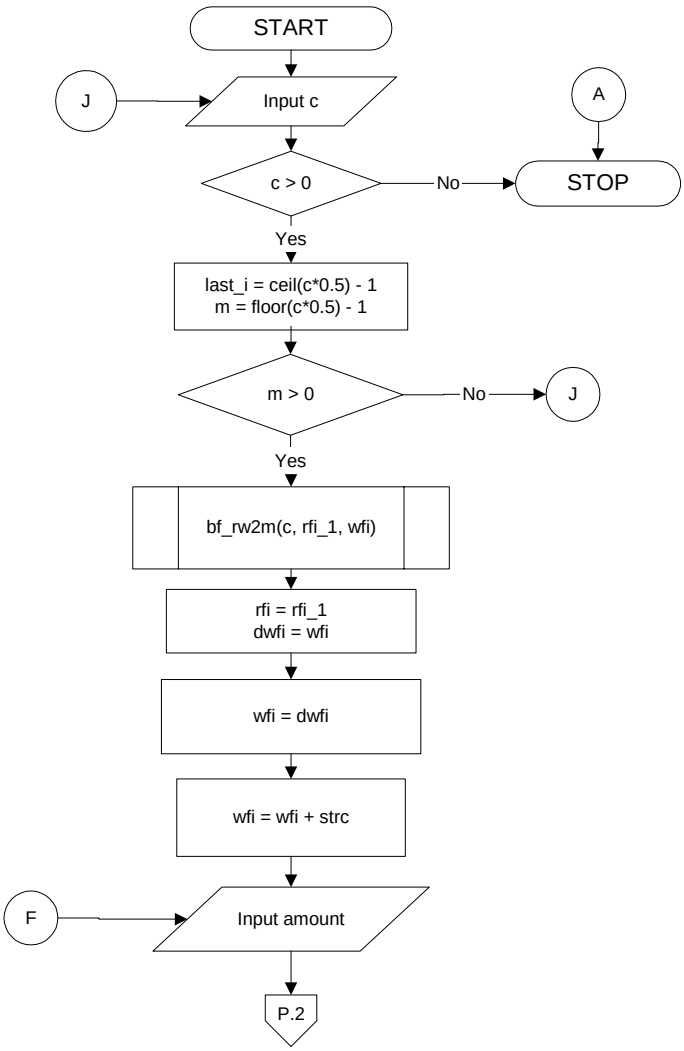
2.2.7) นำผลลัพธ์จากข้อ 2.2.1) จนถึงข้อ 2.2.6) มาดำเนินการตรรกะ OR ระดับบิตต่อบิต

- นำผลลัพธ์จากข้อ 1), 2.1.3) และ 2.2.7) มาดำเนินการตรรกะ AND ระหว่างบิตแมปเวกเตอร์ ในระดับบิตต่อบิต

ภาคผนวก ข

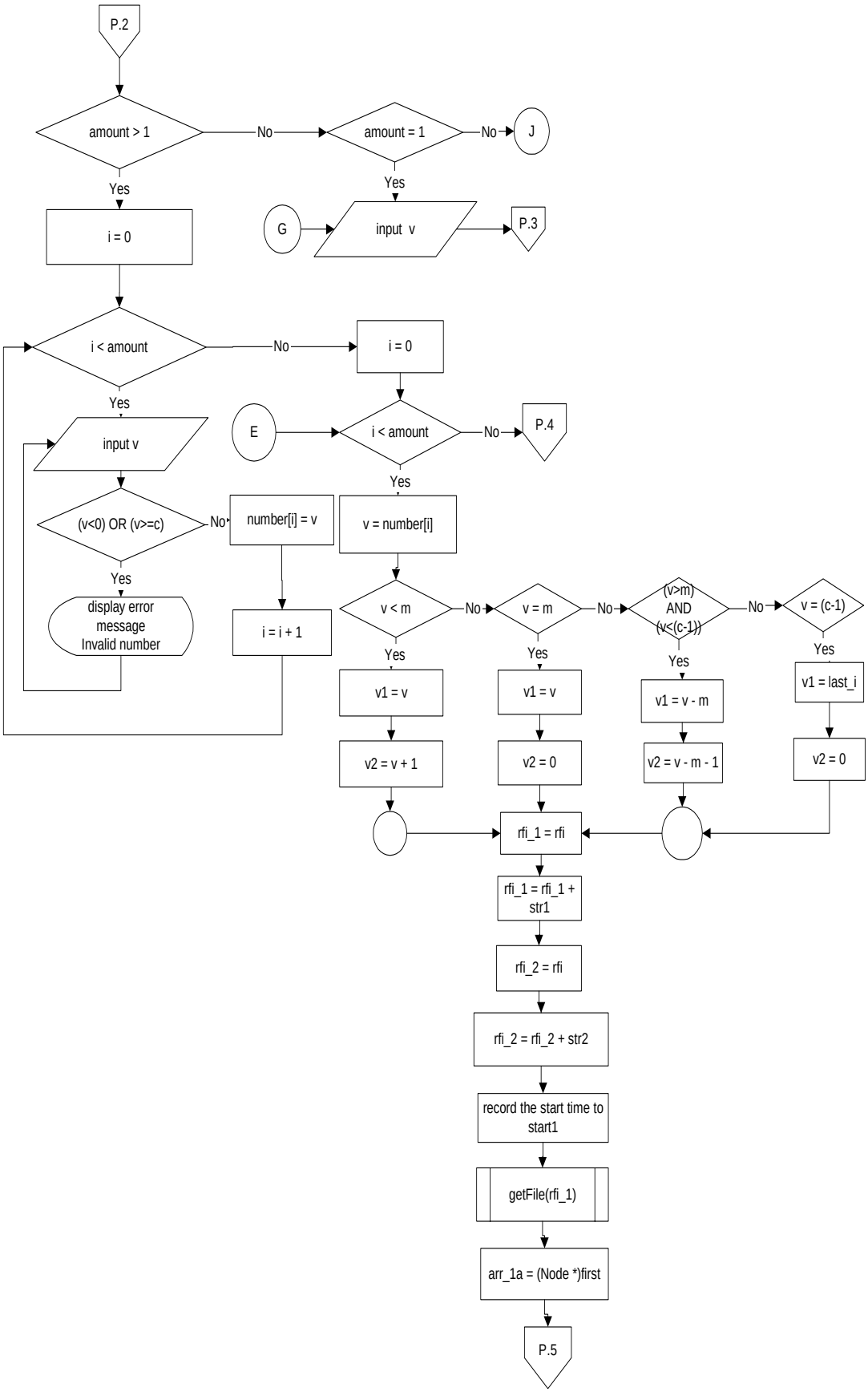
ข.1 แผนผังโปรแกรมการค้นหาข้อมูลของดัชนีบิตแมปแบบช่วง

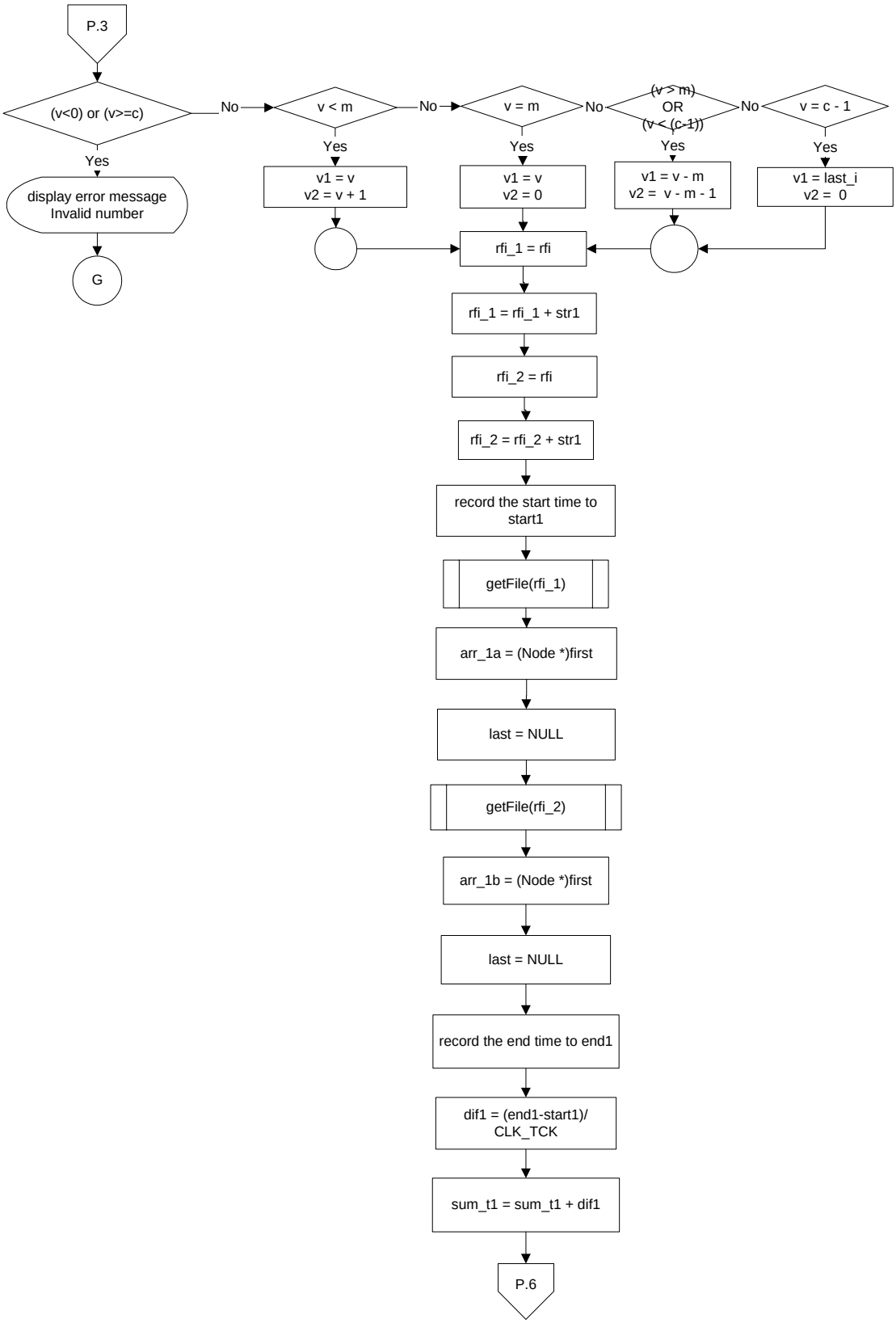
งานวิทยานิพนธ์นี้ได้ทำการประเมินเปรียบเทียบประสิทธิภาพในเรื่องของพื้นที่ที่ใช้ในการจัดเก็บดัชนีและเวลาที่ใช้ในการค้นหาข้อมูลของดัชนีบิตแมปแบบใหม่ที่คิดขึ้น ซึ่งเรียกว่า ดัชนีบิตแมปแบบกระจาย กับดัชนีบิตแมปแบบเดิมที่เคยมีอยู่ ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน แบบช่วง และแบบเข้ารหัส การประเมินประสิทธิภาพในส่วนของเวลาที่ใช้ในการค้นหาข้อมูล โดยการพัฒนาโปรแกรมขึ้นเพื่อจับเวลาในการค้นหาข้อมูลของดัชนีบิตแมปทั้ง 4 แบบ ตัวอย่างโปรแกรมในการค้นหาข้อมูลดังแผนภาพโปรแกรม ซึ่งเป็นโปรแกรมการค้นหาข้อมูลของดัชนีบิตแมปแบบช่วง

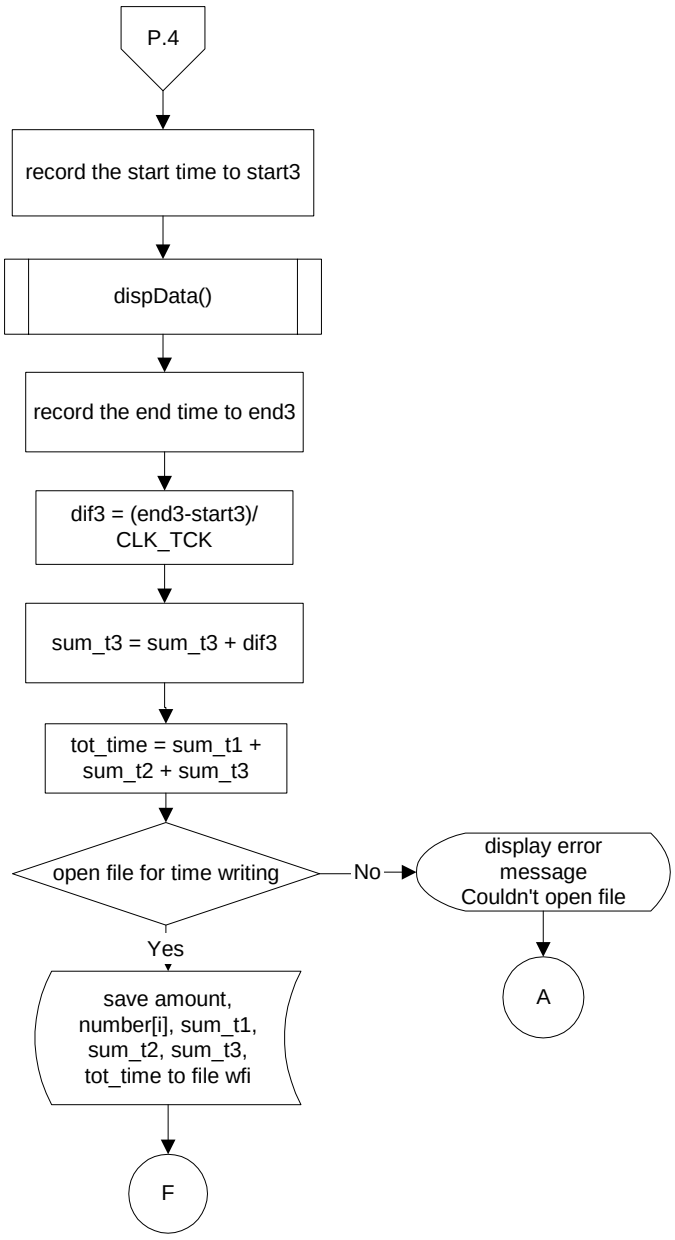


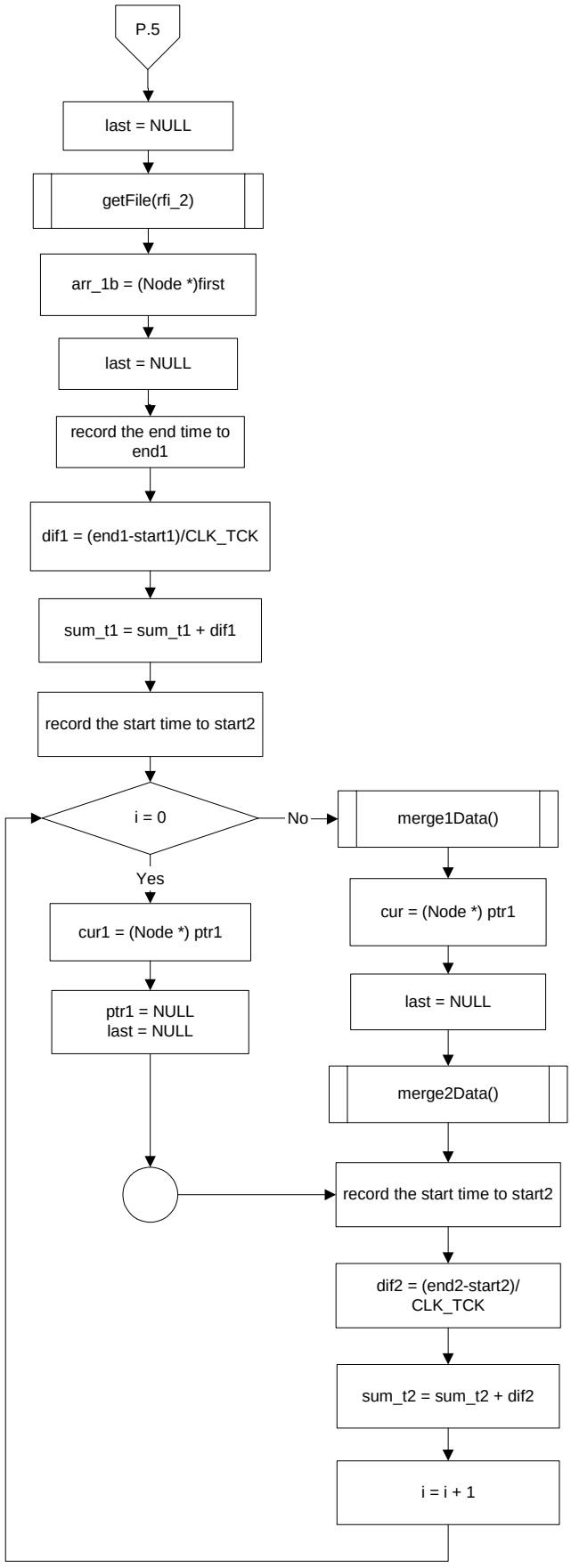
IChkbit_4m1.cpp

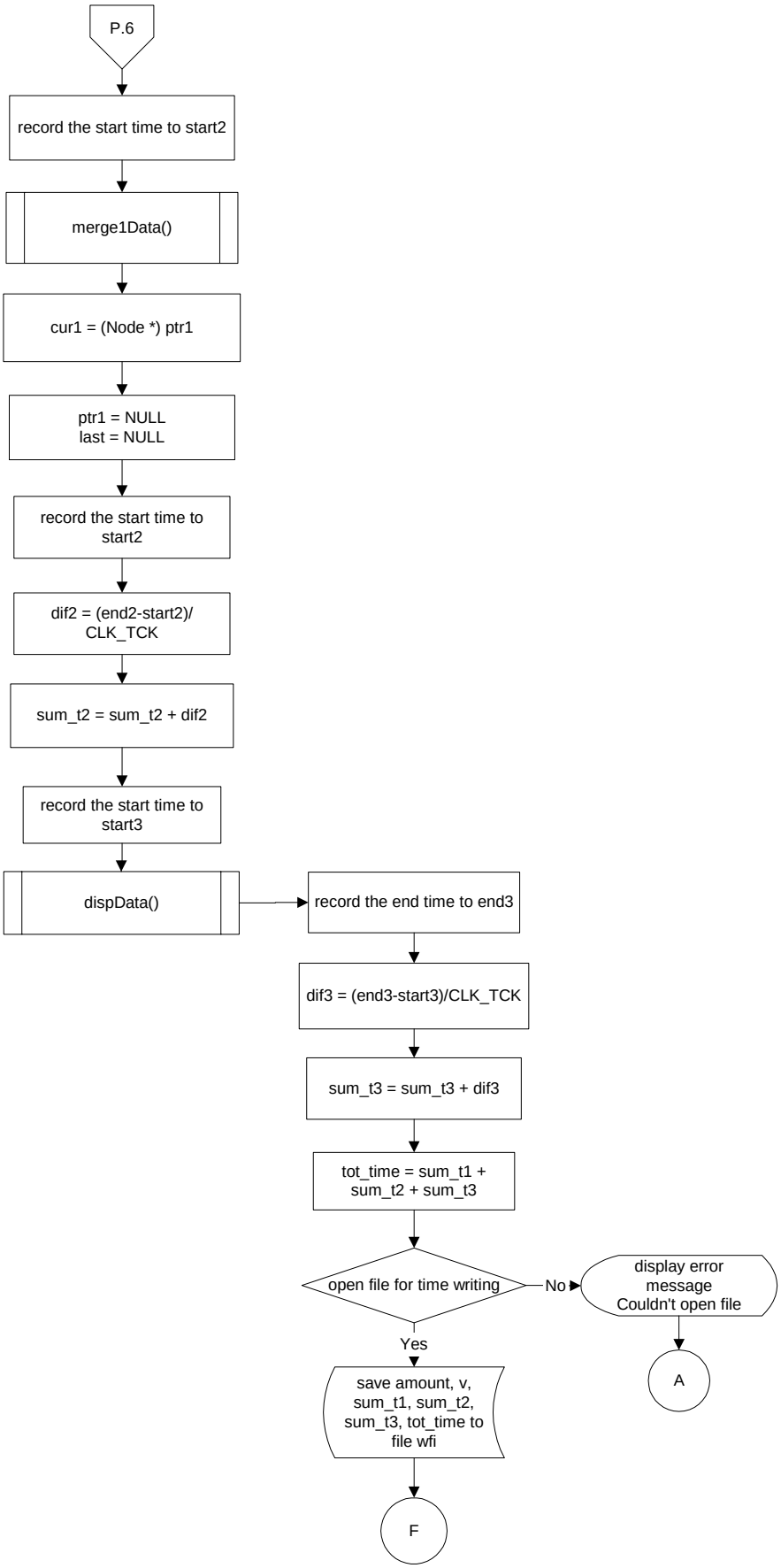
(For Interval Bitmap Index)


P.2
amount > 1
No
amount = 1
No
J
Yes
Yes
i = 0
G
input v
P.3
i < amount
No
i = 0
Yes
E
i < amount
No
P.4
input v
Yes
(v<0) OR (v>=c)
No
number[i] = v
v = number[i]
Yes
display error message Invalid number
i = i + 1
v < m
No
v = m
No
(v>m) AND (v<(c-1))
No
v = (c-1)
Yes
Yes
Yes
Yes
v1 = v
v1 = v
v1 = v - m
v1 = last_i
v2 = v + 1
v2 = 0
v2 = v - m - 1
v2 = 0
rfi_1 = rfi
rfi_1 = rfi_1 + str1
rfi_2 = rfi
rfi_2 = rfi_2 + str2
record the start time to start1
getFile(rfi_1)
arr_1a = (Node *)first
P.5

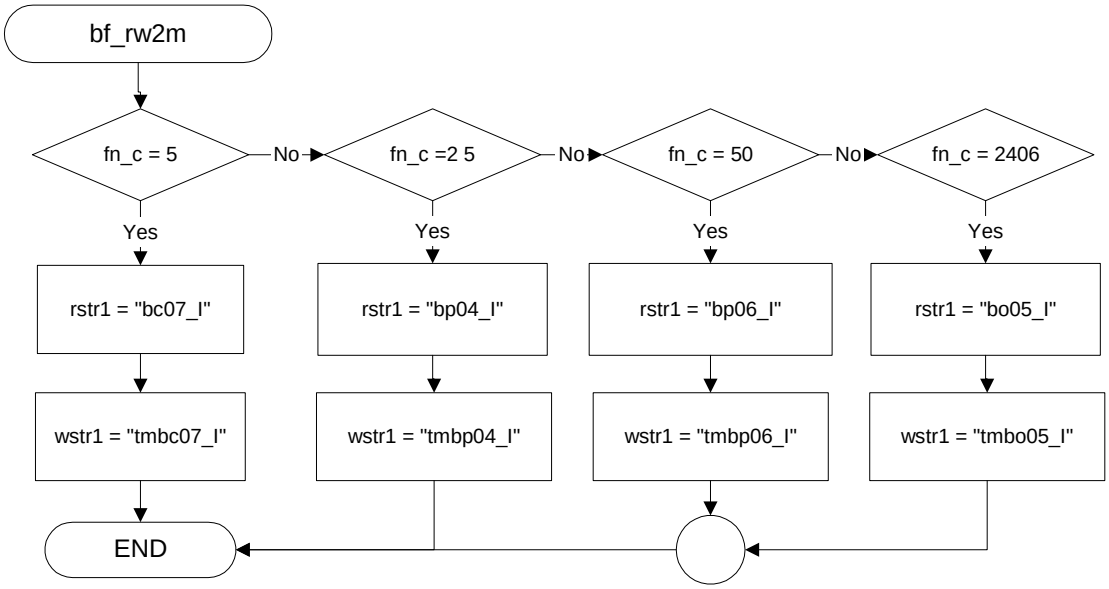

P.3
(v<0) or (v>=c)
No
v < m
No
v = m
No
(v > m)
OR
(v < (c-1))
No
v = c - 1
Yes
display error message
Invalid number
G
Yes
v1 = v
v2 = v + 1
Yes
v1 = v
v2 = 0
Yes
v1 = v - m
v2 = v - m - 1
Yes
v1 = last_i
v2 = 0
rfi_1 = rfi
rfi_1 = rfi_1 + str1
rfi_2 = rfi
rfi_2 = rfi_2 + str1
record the start time to start1
getFile(rfi_1)
arr_1a = (Node *)first
last = NULL
getFile(rfi_2)
arr_1b = (Node *)first
last = NULL
record the end time to end1
dif1 = (end1-start1)/ CLK_TCK
sum_t1 = sum_t1 + dif1
P.6


P.4
record the start time to start3
dispData()
record the end time to end3
dif3 = (end3-start3)/
CLK_TCK
sum_t3 = sum_t3 + dif3
tot_time = sum_t1 +
sum_t2 + sum_t3
open file for time writing
No
display error
message
Couldn't open file
A
Yes
save amount,
number[i], sum_t1,
sum_t2, sum_t3,
tot_time to file wfi
F

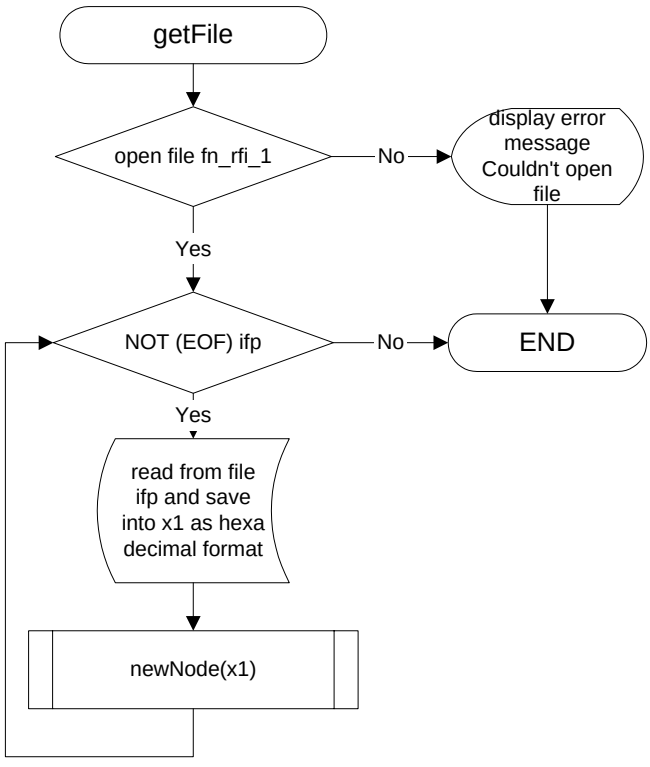

P.5
last = NULL
getFile(rfi_2)
arr_1b = (Node *)first
last = NULL
record the end time to end1
dif1 = (end1-start1)/CLK_TCK
sum_t1 = sum_t1 + dif1
record the start time to start2
i = 0
No
Yes
merge1Data()
cur = (Node *) ptr1
last = NULL
merge2Data()
cur1 = (Node *) ptr1
ptr1 = NULL
last = NULL
record the start time to start2
dif2 = (end2-start2)/ CLK_TCK
sum_t2 = sum_t2 + dif2
i = i + 1


P.6
record the start time to start2
merge1Data()
cur1 = (Node *) ptr1
ptr1 = NULL
last = NULL
record the start time to start2
dif2 = (end2-start2)/ CLK_TCK
sum_t2 = sum_t2 + dif2
record the start time to start3
dispData()
record the end time to end3
dif3 = (end3-start3)/CLK_TCK
sum_t3 = sum_t3 + dif3
tot_time = sum_t1 + sum_t2 + sum_t3
open file for time writing
No
display error message Couldn't open file
A
Yes
save amount, v, sum_t1, sum_t2, sum_t3, tot_time to file wfi
F

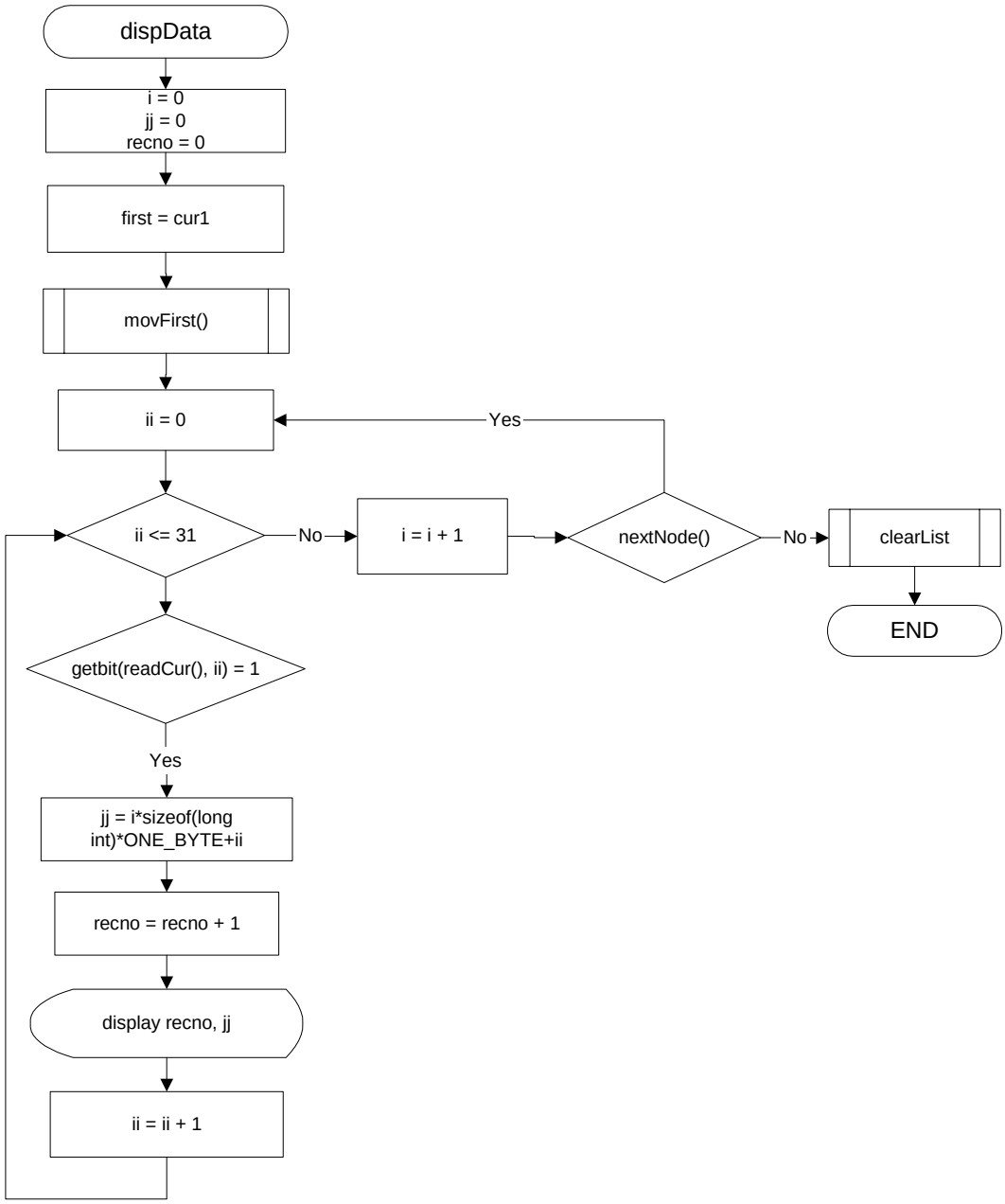
ข.2 แผนผังโปรแกรมย่อยการเลือกไฟล์ (bf_rw2m)



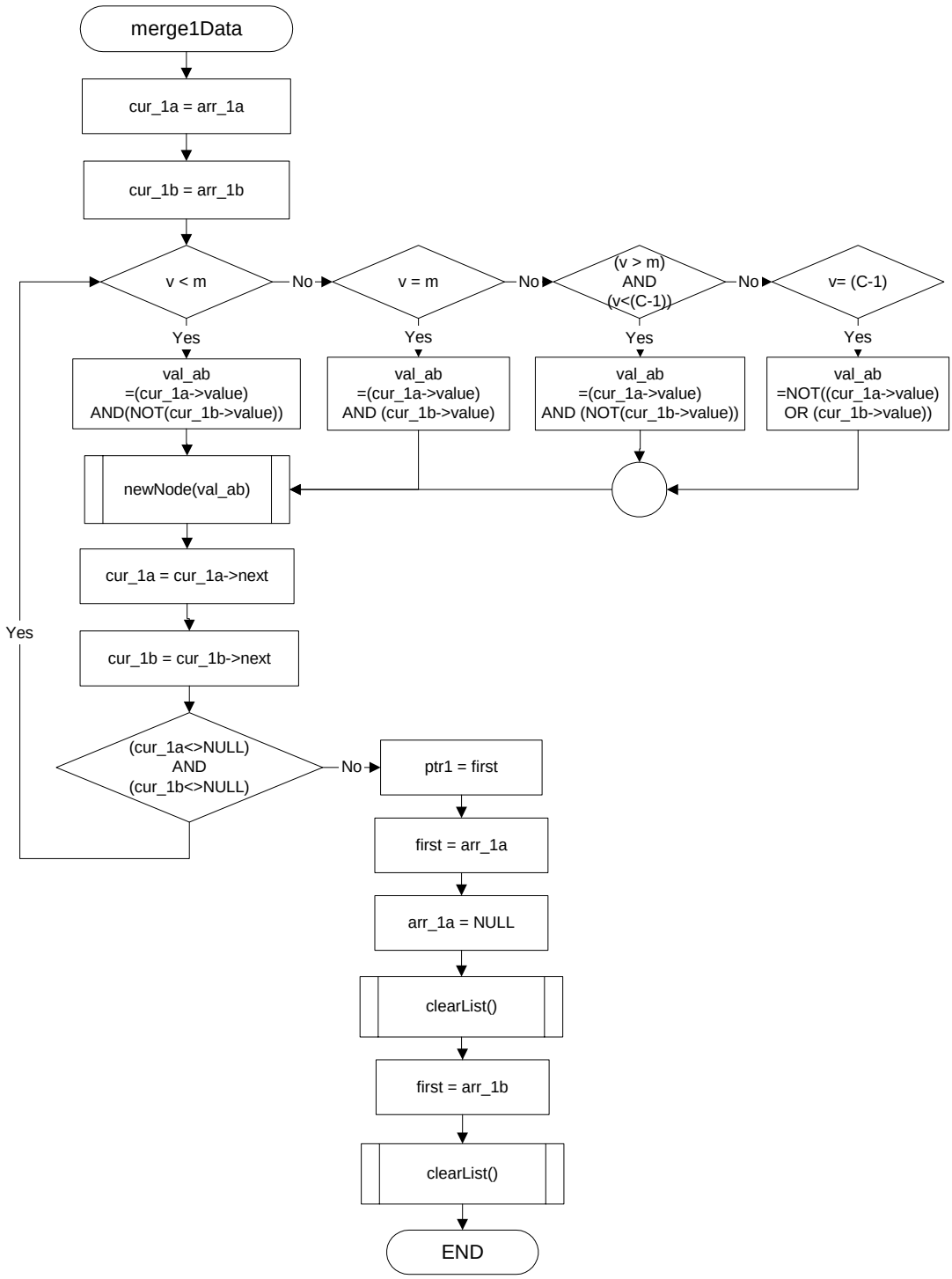
ข.3 แผนผังโปรแกรมย่อยการเปิดอ่านไฟล์ (getFile)



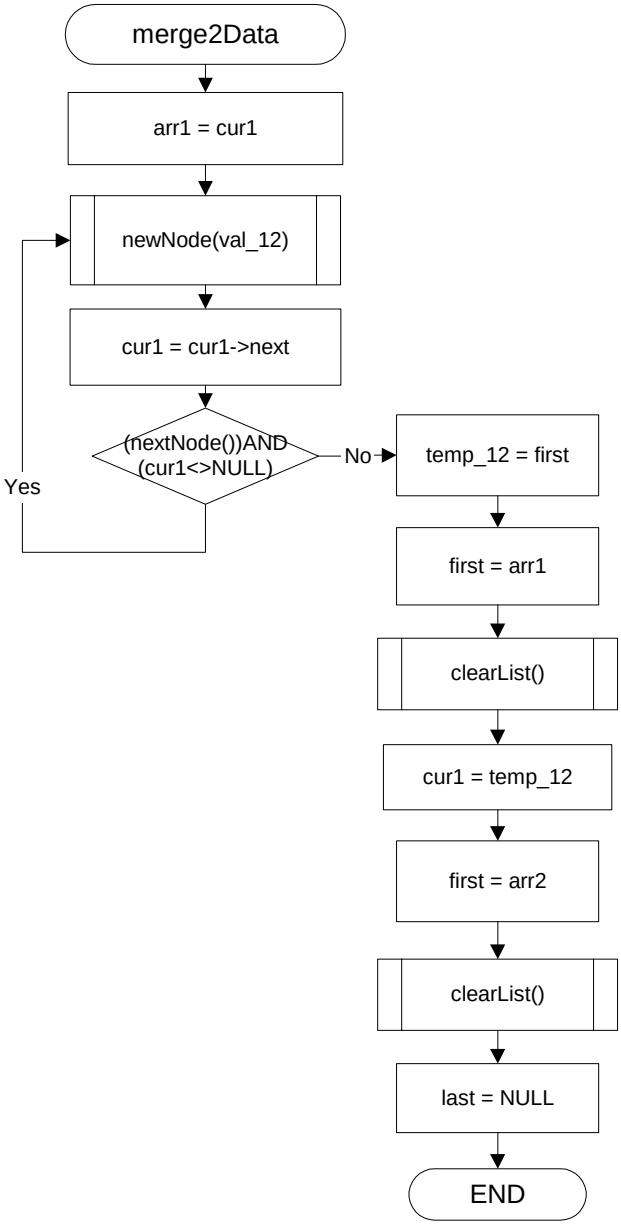
ข.4 แผนผังโปรแกรมย่อยการแสดงข้อมูล (dispData)



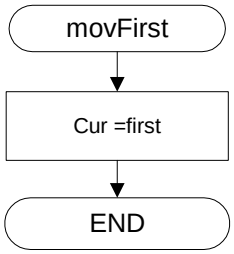
ข.5 แผนผังโปรแกรมย่อยการรวมบิตแมปเวกเตอร์ย่อยเข้าด้วยกัน (merge1Data)



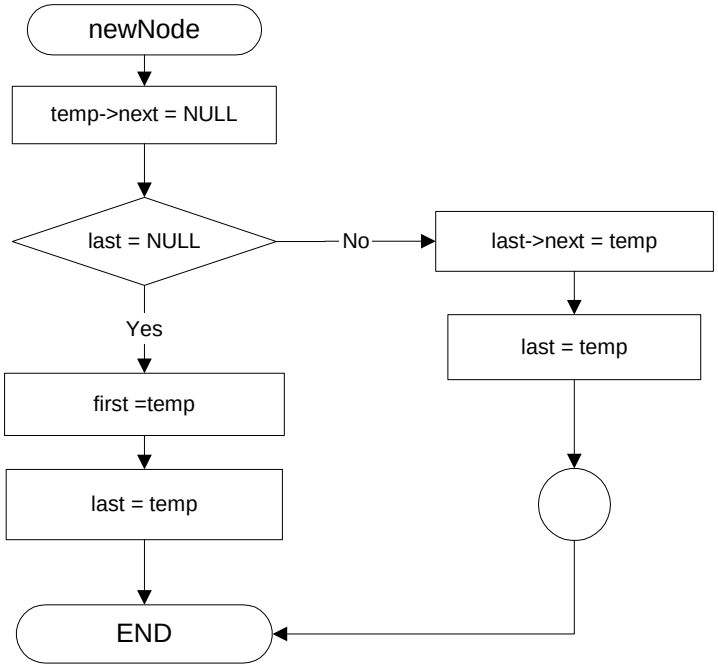
ข.6 แผนผังโปรแกรมย่อยการรวมบิตแมปเวกเตอร์เข้าด้วยกัน เพื่อค้นหาค่าแบบความเป็นสมาชิก (merge2Data)



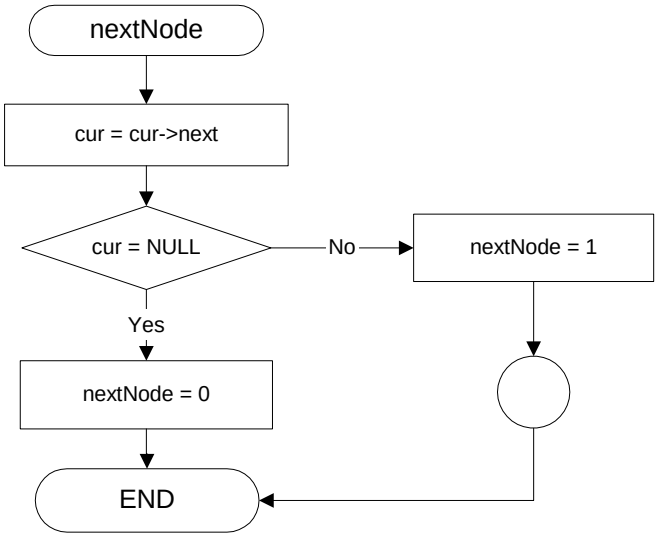
ข.7 แผนผังโปรแกรมย่อยการย้ายไปยังโหนดแรก (movFirst)



ข.8 แผนผังโปรแกรมย่อยการสร้างโหนดใหม่ (newNode)



ข.9 แผนผังโปรแกรมย่อยการย้ายไปยังโหนดถัดไป (nextNode)



ข.10 แผนผังโปรแกรมย่อยการเคลียร์ลิสต์ (clearList)