บทที่ 3

การออกแบบโครงสร้างข้อมูลทรัยแถวคู่

โครงสร้างข้อมูลของอาร์เรย์คู่ (BASE และ CHECK)

จากแนวความคิดเดิมของ Aoe [4][5][6][7] ใน BASE ของทรัยแถวคู่ใช้ค่าบวกเป็นการบอกให้ไปสืบค้นตัวอักษรตัวต่อไปที่สมาชิกใดของ BASE และใช้ค่าลบบอกให้ไปสืบค้นตัวอักษรที่เหลือโดยเริ่มต้นที่สมาชิกใดใน TAIL พบว่าในการสืบค้นที่อาร์เรย์คู่จะมาสิ้นสุดใน TAIL เสมอ โดยใช้ '$' เป็นการแสดงการสิ้นสุดสตริงใน TAIL ดังนั้นจึงปรับปรุงเพิ่มตัวบ่งชี้ (flag) เพิ่มเติม โดยการสืบค้นจะตรวจสอบตัวบ่งชี้นี้ว่าเป็นจุดสิ้นสุดหรือยัง แทนที่ต้องสืบค้นเข้าไปหา '$' ใน TAIL ดังนั้นการสืบค้นคีย์สิ้นสุดในอาร์เรย์คู่ และยังเป็นการประหยัดเนื้อที่ใน TAIL เพราะไม่ต้องเก็บ '$' ไว้ใน TAIL ตัวบ่งชี้นี้จะเก็บเป็นส่วนหนึ่งของค่าใน BASE ดังนั้นค่าใน BASE จะมีตัวบ่งชี้ที่แสดงความหมายอยู่ 3 ความหมายคือ

BASE_NEXT	สืบค้นตัวอักษรถัดไปใน BASE
TAIL_NEXT	สืบค้นตัวอักษรที่เหลือโดยเริ่มต้นที่ TAIL ตำแหน่งใด
END_NEXT	คีย์ที่สืบค้นสิ้นสุดในอาร์เรย์คู่แล้ว ไม่ต้องสืบค้นต่อใน TAIL

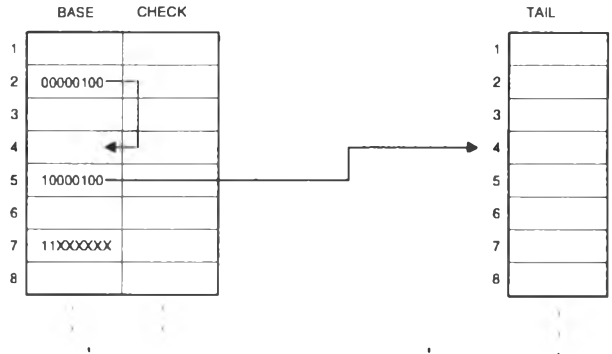
นั่นคือแทนที่จะใช้ค่าบวกลบใน BASE แสดงการสืบค้น ใช้ค่า 2 บิตแรกของ BASE เพื่อแสดง ค่าดังกล่าวข้างต้น โดยให้

BASE_NEXT	คือ 00
TAIL_NEXT	คือ 10
END_NEXT	คือ 11

ส่วนบิตที่เหลือจาก 2 บิตแรกแสดงถึงตำแหน่งของแต่ละโครงสร้างข้อมูลว่าให้ไปที่ใดตาม ค่าของ 2 บิตแรก

BASE_NEXT	บิตที่เหลือบอกตำแหน่งของ BASE ที่ต้องไปสืบค้นตัวอักษรต่อไป
TAIL_NEXT	บิตที่เหลือบอกตำแหน่งเริ่มต้นของ TAIL ที่ต้องสืบค้นตัวอักษรที่เหลือ

END_NEXT บิตที่เหลือไม่ได้ใช้



รูปที่ 3.1 แสดง BASE และ TAIL ที่ปรับปรุงแล้ว

ในรูปที่ 3.1 แสดงตัวอย่างส่วน BASE ที่ปรับปรุงแล้ว ที่ BASE[2] ไปสืบค้นอักษรตัวต่อไปที่ BASE[4], ที่ BASE[5] ไปสืบค้นตัวอักษรต่อไปที่ TAIL[4] และที่ BASE[7] พบคีย์ที่สืบค้นมีในทรัยแถวคู่

โครงสร้างแฟ้มข้อมูลของอาร์เรย์คู่ (BASE และ CHECK)

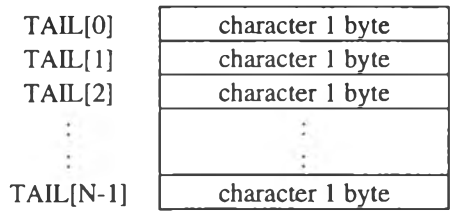
เนื่องจากพจนานุกรมมีจำนวนคำศัพท์มาก จึงเตรียมขนาดของค่า BASE และ CHECK ไว้ 4 ไบต์ โดยกัน 2 บิตแรกไว้ นั่นคือสามารถแสดงสถานะของเครื่องเปรียบแบบ (Pattern Matching Machine) ได้ 2^{30} คือ 1 G สถานะ และเก็บอาร์เรย์ของ BASE และ CHECK สลับกันไปไว้ในแฟ้มข้อมูลนามสกุล ".DA" ดังรูปที่ 3.2

BASE[0]	long integer 4 bytes
CHECK[0]	long integer 4 bytes
BASE[1]	long integer 4 bytes
CHECK[1]	long integer 4 bytes
BASE[2]	long integer 4 bytes
CHECK[2]	long integer 4 bytes
:	:
:	:
BASE[N-1]	long integer 4 bytes
CHECK[N-1]	long integer 4 bytes

รูปที่ 3.2 แสดง BASE และ CHECK ที่เก็บเป็นแฟ้มข้อมูล

โครงสร้างแฟ้มข้อมูลของ TAIL

ใน TAIL เก็บเป็นอาร์เรย์ของตัวอักษรขนาด 1 ไบต์ คือเก็บเป็นสตริงไว้ในแฟ้มข้อมูลนามสกุล ".TL" ดังรูปที่ 3.3



รูปที่ 3.3 แสดง TAIL ที่เก็บเป็นแฟ้มข้อมูล

ค่าคงที่ที่ใช้ในโปรแกรม

ในโปรแกรมได้กำหนดค่าคงที่ต่างๆ โดยใช้ #define ในภาษาซีไว้ในแฟ้ม "datdef.h" ซึ่งค่าคงที่ต่างๆ มีดังต่อไปนี้

1. ค่าคงที่ทั่วไป

```
#define   TRUE       1
#define   FALSE      0
#define   FOUND      0
#define   NOTFOUND   1
```

MAXTRIE จำนวนทรัยแถวคู่ที่มากที่สุดที่สามารถใช้ได้ในคราวเดียวกัน

```
#define   MAXTRIE    10
```

MAXFIELDLEN จำนวนตัวอักษรที่มากที่สุดของคีย์ที่นำมาสืบค้น

```
#define   MAXFIELDLEN        255
```

NUM_CHR_SET จำนวนสมาชิกทั้งหมดของเซ็ตของการผ่าน ในที่นี้ใช้ตัวอักษรขนาด 1 ไบต์

แทน 1 การผ่าน

```
#define    NUM_CHR_SET        256
```

TERMINATOR ตัวอักษรที่ใช้บอกจุดสิ้นสุดของสตริงในทรัยแถวคู่ คือ '#' หรือ endmarker นั่นเอง

```
#define    TERMINATOR         (NUM_CHR_SET - 1)
```

BASE_NEXT บิตที่เหลือจาก 2 บิตแรกบอกตำแหน่งของ BASE ที่ต้องไปสืบค้นตัวอักษรต่อไป

```
#define    BASE_NEXT  '\x00'
```

TAIL_NEXT บิตที่เหลือจาก 2 บิตแรกบอกตำแหน่งเริ่มต้นของ TAIL ที่ต้องไปสืบค้นตัวอักษรที่เหลือ

```
#define    TAIL_NEXT '\x80'
```

END_NEXT คีย์ที่สืบค้นสิ้นสุดใน BASE และ CHECK แล้ว ไม่ต้องสืบค้นต่อใน TAIL

```
#define    END_NEXT  '\xC0'
```

MARK_TAIL เป็นค่าที่ไว้ทำการเปลี่ยนค่าใน BASE เป็นชนิด TAIL_NEXT โดยการ OR

```
#define    MARK_TAIL  0x80000000L
```

MARK_END เป็นค่าที่ไว้ทำการเปลี่ยนค่าใน BASE เป็นชนิด END_NEXT โดยการ OR

```
#define    MARK_TAIL  0xC0000000L
```

2. ข้อผิดพลาด

NOERROR ไม่เกิดข้อผิดพลาด

```
#define    NOERROR    0
```

NOWORKAREA ไม่มีพื้นที่สำหรับทรัยแถวคู่ที่เปิดใหม่ เนื่องจากมีการเปิดใช้ทรัยแถวคู่เกิน MAXTRIE

```
#define NOWORKAREA         2
```

DAOPENERROR เกิดข้อผิดพลาดในการเปิดแฟ้ม BASE และ CHECK

#define DAOPENERROR 3

TLOPENERROR เกิดข้อผิดพลาดในการเปิดแฟ้ม TAIL

#define TLOPENERROR 4

FDERROR file descriptor ที่ได้จากการเปิดทรัยแถวคู่ มีค่าน้อยกว่า 0 หรือมีค่าเกินกว่า MAXTRIE - 1

#define FDERROR 7

FDUNUSED ทรัยแถวคู่ที่ใช้มีสถานะในโครงสร้างข้อมูลที่ใช้อ้างอิงทรัยแถวคู่นี้เป็น UNUSED

#define FDUNUSED 8

DATAEXIST คีย์ที่ใช้เพิ่มเข้าไปในทรัยแถวคู่มีอยู่แล้วในทรัยแถวคู่นั้น

#define DATAEXIST 10

NULLKEYNOTALLOW คีย์ที่ใช้เพิ่มลบและสืบค้นในทรัยแถวคู่ไม่มีตัวอักษรใดๆในคีย์

#define NULLKEYNOTALLOW 11

DATANOTEXIST คีย์ที่ใช้สืบค้นหรือลบในทรัยแถวคู่ไม่ปรากฏในทรัยแถวคู่

#define DATANOTEXIST 12

โครงสร้างข้อมูลที่ใช้ในโปรแกรม

TEXT โครงสร้างข้อมูลชนิดตัวอักษรเก็บค่า ASCII ตั้งแต่ 0 ถึง 255 ใช้เก็บการผ่าน (transition)

typedef unsigned char TEXT;

NODE โครงสร้างข้อมูลชนิด integer ขนาด 4 ไบต์ เก็บตำแหน่งของ BASE และ CHECK

typedef long NODE;

TAILPOS โครงสร้างข้อมูลชนิด integer ขนาด 4 ไบต์ เก็บตำแหน่งของ TAIL

```
typedef     long     TAILPOS
```

TL โครงสร้างข้อมูลที่ใช้อ้างอิง TAIL โดยมี fpTail เก็บตัวชี้แฟ้ม (File Pointer) ของแฟ้ม TAIL ที่ใช้ และมี pos_ เก็บ ตำแหน่งสุดท้ายของ TAIL ที่ใช้

```
typedef   struct  {
          FILE      *fpTail;
          TAILPOS pos_;
} TL
```

DA โครงสร้างข้อมูลที่ใช้อ้างอิงทรัยแถวคู่ โดยมี fpBase เก็บตัวชี้แฟ้ม (File Pointer) ของแฟ้ม BASE และ CHECK ที่ใช้ มี da_size_ เก็บขนาดของทรัยแถวคู่ มี base_type_ เก็บชนิดของ BASE ว่า BASE นั้นเป็น BASENEXT, ENDNEXT หรือ TAILNEXT และมี tl เก็บโครงสร้างข้อมูล TL ที่ใช้อ้างอิง TAIL

```
typedef   struct  {
          FILE      *fpBase;
          NODE      da_size_;
          char      base_type_;
          TL     tl;
} DA
```

DAT โครงสร้างข้อมูลที่ใช้อ้างอิงทรัยแถวคู่และสถานะการใช้ โดย da เก็บโครงสร้างข้อมูลที่ใช้อ้างอิงทรัยแถวคู่ และ status เป็น USED หรือ UNUSED เพื่อแสดงว่าโครงสร้างข้อมูลนี้ถูกจองโดยทรัยแถวคู่ใดหรือไม่

```
typedef   struct  {
          DA     da;
          char   status;
} DAT
```

ARRAY_RESULT โครงสร้างข้อมูลอาร์เรย์ของ TEXT ขนาด MAXFIELDLEN
ใช้เก็บสตริงที่ได้จากฟังก์ชั่นการสืบค้นบางส่วน

```
typedef   TEXT ARRAY_RESULT[MAXFIELDLEN]
```

ฟังก์ชั่นการจัดการโครงสร้างข้อมูลทรัยแถวคู่

1. การเปิดแฟ้มอาร์เรย์คู่ (BASE กับ CHECK) และ TAIL ของทรัยแถวคู่

Open Double Array Trie File

int fopent(char *filename, char *mode)

filename : ชื่อแฟ้มของทรัยแถวคู่ที่ต้องการเปิด ไม่คำนึงถึงนามสกุลแฟ้ม

mode : โหมดของการเปิดแฟ้ม ในขั้นตอนนี้ยังไม่ได้ใช้จึงระบุเป็น NULL

Return : file descriptor ของโครงสร้างข้อมูล DAT ที่ใช้อ้างอิงทรัยแถวคู่ หรือ NULL หากเกิดข้อผิดพลาดขึ้น

ทรัยแถวคู่ 1 ชุดแบ่งเป็นแฟ้มอาร์เรย์คู่ (BASE กับ CHECK) (แฟ้มสกุล DA) หนึ่งแฟ้ม และแฟ้มของ TAIL (แฟ้มสกุล TL) หนึ่งแฟ้ม เมื่อต้องการใช้ทรัยแถวคู่ชุดใด จะต้องเปิดทรัยแถวคู่ชุดนั้นด้วยฟังก์ชั่นนี้ก่อน และในการอ้าง อิงทรัยแถวคู่ชุดนั้นในภายหลังจะใช้ file descriptor ที่ได้จากฟังก์ชั่นนี้แทน

2. การปิดแฟ้ม BASE กับ CHECK และ TAIL ของทรัยแถวคู่

Close Double Array Trie File

void fcloset(int fd)

fd : file descriptor ของโครงสร้างข้อมูล DAT ที่ใช้อ้างอิงทรัยแถวคู่

ฟังก์ชั่นนี้ทำการปิดแฟ้ม BASE กับ CHECK (แฟ้มสกุล DA) และแฟ้ม TAIL (แฟ้มสกุล TL) ของทรัย แถวคู่ที่อ้างอิงด้วย fd นอกจากปิดแฟ้มดังกล่าวแล้ว ยังเปลี่ยนสถานะของโครงสร้างข้อมูล DAT ที่ใช้อ้างอิงทรัย แถวคู่ที่ต้องการปิดนี้ให้เป็นสถานะว่าง เพื่อที่สามารถเปิดทรัยแถวคู่ชุดอื่นได้

3. การสืบค้นคีย์ในทรัยแถวคู่

Check Key in Double Array Trie

int fchkt(int fd, TEXT *key)

fd : file descriptor ของโครงสร้างข้อมูล DAT ที่ใช้อ้างอิงทรัยแถวคู่

key : สตริงใช้เป็นคีย์ในการสืบค้นทรัยแถวคู่

Return : FOUND หาก key ปรากฏอยู่ในทรัยแถวคู่
NOTFOUND หากไม่สามารถสืบค้น key ได้

ฟังก์ชั่นนี้ทำการสืบค้นสตริง key ในทรัยแถวคู่ที่อ้างอิงด้วย fd หากในทรัยแถวคู่มีคีย์ที่สืบค้นอยู่ จะ return FOUND และหากได้รับ NOTFOUND ให้ตรวจสอบตัวแปร errcode หาก errcode เป็น

DATANOTEXIST แสดง ว่ามีคีย์นั้นแล้วในทรัยแถวคู่ หาก errcode เป็นค่าอื่นแสดงว่าเกิดความผิดพลาดขึ้น

4. การเพิ่มคีย์ในทรัยแถวคู่

Put Key into Double Array Trie

int fputt(int fd, TEXT *key)

fd : file descriptor ของโครงสร้างข้อมูล DAT ที่ใช้อ้างอิงทรัยแถวคู่

key : สตริงที่เป็นคีย์เพิ่มเข้าไปในทรัยแถวคู่

Return : FOUND หาก key ปรากฏอยู่ในทรัยแถวคู่

NOTFOUND หากไม่สามารถเพิ่ม key เข้าในทรัยแถวคู่

ฟังก์ชั่นนี้ทำการเพิ่มสตริง key เข้าในทรัยแถวคู่ที่อ้างอิงด้วย fd หากในทรัยแถวคู่มีคีย์ที่สืบค้น อยู่ จะ return FOUND และหากได้รับ NOTFOUND ให้ตรวจสอบตัวแปร errcode หาก errcode เป็น DATANOTEXIST แสดงว่ามีคีย์นั้นแล้วในทรัยแถวคู่ หาก errcode เป็นค่าอื่นแสดงว่าเกิดความผิดพลาด ขึ้น

5. การลบคีย์ในทรัยแถวคู่

Delete Key from Double Array Trie

int fdelt(int fd, TEXT *key)

fd : file descriptor ของโครงสร้างข้อมูล DAT ที่ใช้อ้างอิงทรัยแถวคู่

key : สตริงที่เป็นคีย์ลบออกจากทรัยแถวคู่

Return : FOUND หาก key ปรากฏอยู่ในทรัยแถวคู่

NOTFOUND หากไม่สามารถลบ key จากทรัยแถวคู่

ฟังก์ชั่นนี้ทำการลบสตริง key ออกจากทรัยแถวคู่ที่อ้างอิงด้วย fd หากในทรัยแถวคู่มีคีย์ที่สืบค้นอยู่ จะ return FOUND และหากได้รับ NOTFOUND ให้ตรวจสอบตัวแปร errcode หาก errcode เป็น DATANOTEXIST แสดงว่ามีคีย์นั้นแล้วในทรัยแถวคู่ หาก errcode เป็นค่าอื่นแสดงว่าเกิดความผิดพลาดขึ้น

6. การสืบค้นคำที่ประกอบด้วยบางส่วนของคีย์แบบไปข้างหน้า

Partial Forward Search of Key in Double Array Trie

int ptfosrch(int fd, TEXT *key, int num, ARRAY_RESULT result[])

fd : file descriptor ของโครงสร้างข้อมูล DAT ที่ใช้อ้างอิงทรัยแถวคู่

key : สตริงที่เป็นคีย์ในการสืบค้นทรัยแถวคู่

num : จำนวนคำที่จะทำการสืบค้นด้วยบางส่วนของคีย์แบบไปข้างหน้า

result : อาร์เรย์ของสตริงที่ใช้เก็บคำที่สืบค้นแบบไปข้างหน้า

Return : จำนวนคำที่สืบค้นแบบไปข้างหน้า

ฟังก์ชั่นนี้ใช้สตริง key และบางส่วนของสตริง key เป็นคีย์สืบค้นหาคำศัพท์ในทรัยแถวคู่เป็นจำนวน num คำ ตัวอย่างเช่นให้ key = "afry" จะเริ่มต้นโดยใช้ "afry" เป็นคีย์ จะได้คำที่สืบค้นได้แบบไปข้างหน้าเป็น "afry", "afryz", "afryza" เป็นต้น จากนั้นตัดตัวอักษรท้ายสุดของ key ทิ้งได้คีย์ใหม่คือ "afr" นำไปสืบค้นต่อไป ข้างหน้าได้คำ "afrc", "afrdz" เป็นต้น ทั้งนี้จะทำการสืบค้นจนได้จำนวนคำเป็น num หรือสืบค้นจนหมดทรัยแถวคู่

7. การสืบค้นคำที่ประกอบด้วยบางส่วนของคีย์แบบย้อนหลัง

Partial Backward Search of Key in Double Array Trie

int ptbksrch(int fd, TEXT *key, int num, ARRAY_RESULT result[])

fd : file descriptor ของโครงสร้างข้อมูล DAT ที่ใช้อ้างอิงทรัยแถวคู่

key : สตริงที่เป็นคีย์ในการสืบค้นทรัยแถวคู่

num : จำนวนคำที่จะทำการสืบค้นด้วยบางส่วนของคีย์แบบย้อนหลัง

result : อาร์เรย์ของสตริงที่ใช้เก็บคำที่สืบค้นแบบย้อนหลัง

Return : จำนวนคำที่สืบค้นแบบย้อนหลัง

ฟังก์ชั่นนี้ใช้สตริง key และบางส่วนของสตริง key เป็นคีย์สืบค้นหาคำศัพท์ในทรัยแถวคู่เป็นจำนวน num คำ ตัวอย่างเช่นให้ key = "afry" จะเริ่มต้นโดยใช้ "afry" เป็นคีย์ จะได้คำที่สืบค้นได้แบบย้อนหลังเป็น "afrxabc", "afrx", "afrsabc" เป็นต้น จากนั้นตัดตัวอักษรท้ายสุดของ key ทิ้งได้คีย์ใหม่คือ "afr" นำไปสืบค้นต่อย้อนหลังได้คำ "afmabc", "afma", "afg" เป็นต้น ทั้งนี้จะทำการสืบค้นจนได้จำนวนคำเป็น num หรือสืบค้น จนหมดทรัยแถวคู่

โปรแกรมอรรถประโยชน์

1. การเพิ่มคำศัพท์เข้าไปในแฟ้มทรัยแถวคู่

APPTXT

โปรแกรมนี้ทำการเพิ่มระเบียนของคำในแฟ้มชนิดข้อความ คือมีปัดแคร่กั้นระหว่างระเบียนของคำเข้าไปในแฟ้มทรัยแถวคู่ หากไม่มีแฟ้มทรัยแถวคู่ที่ระบุชื่อ โปรแกรมจะสร้างแฟ้มทรัยแถวคู่ชื่อนั้นขึ้นมา และโปรแกรมจะแสดงผลของการเพิ่มคำแต่ละคำ โดยหากมีคำนั้นในแฟ้มทรัยแถวคู่อยู่แล้วจะแสดงข้อความ not inserted ท้ายคำนั้น แต่หากสามารถเพิ่มคำได้ จะแสดงข้อความ OK ท้ายคำนั้น

2. การแสดงค่าของ BASE, CHECK และ TAIL ของแฟ้มทรัยแถวคู่

DUMP

โปรแกรมนี้ทำการแสดงค่าของ BASE และ CHECK ในแฟ้มอาร์เรย์คู่ (แฟ้มสกุล DA) และค่าของ TAIL ในแฟ้ม TAIL (แฟ้มสกุล TL) หากไม่มีแฟ้มทรัยแถวคู่ที่ระบุชื่อ โปรแกรมจะสร้างแฟ้มทรัยแถว คู่ชื่อนั้นขึ้นมา โปรแกรม แสดงผลของ BASE และ CHECK ก่อน โดยแสดงผลเป็นหมายเลขโหนด, ค่าของ BASE ที่โหนดนั้น, ชนิดของ BASE โดยแสดงเป็น D หากเป็น ENDNEXT หรือ T หากเป็น TAILNEXT และสุดท้ายเป็นค่าของ TAIL ที่ โหนดนั้น โดยเรียงลำดับจากโหนดแรกถึงโหนดสุดท้าย จากนั้นจะแสดงผลแฟ้ม TAIL โดยแสดงผลเป็นตำแหน่ง และตัวอักษรของ TAIL ที่ตำแหน่งนั้นเรียง ลำดับจากตำแหน่งแรกจนถึงสุดท้าย

3. การลดขนาดของแฟ้ม TAIL

PACKTL

เนื่องจากการจัดการกับโครงสร้างข้อมูลทรัยแถวคู่ทำให้เกิดตำแหน่งว่างกระจายอยู่ใน TAIL จึงสามารถกระชับ TAIL เพื่อไม่ให้เกิดตำแหน่งว่างได้ โปรแกรมนี้มีไว้เพื่อลดขนาดของ TAIL ในแฟ้ม TAIL โดยระบุชื่อของแฟ้ม ทรัยแถวคู่ที่ต้องการลดขนาดของ TAIL