10301222 โครงสร้างข้อมูลและอัลกอริทึม

Chapter 3

อาเรย์

ผศ.ดร. ปวีณ เขื่อนแก้ว

สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยแม่โจ้

Array

A collection of homogenous, order and finite set of elements

- Homogenous: เหมือนกัน เป็นข้อมูลชนิดเดียวกัน

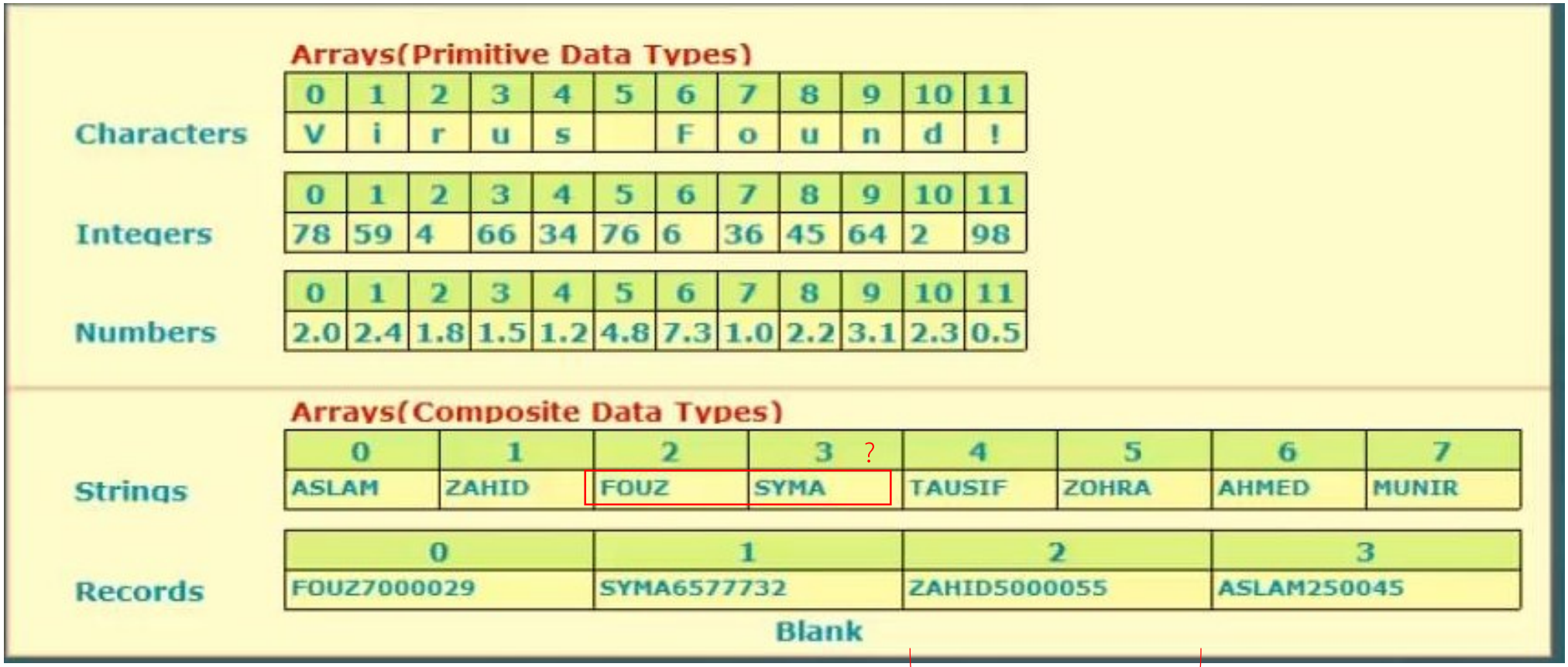
  จะเป็นข้อมูลแบบ primitive หรือ composite ก็ได้

- Order: ข้อมูลมีการเรียงเป็นลำดับ (sequence)

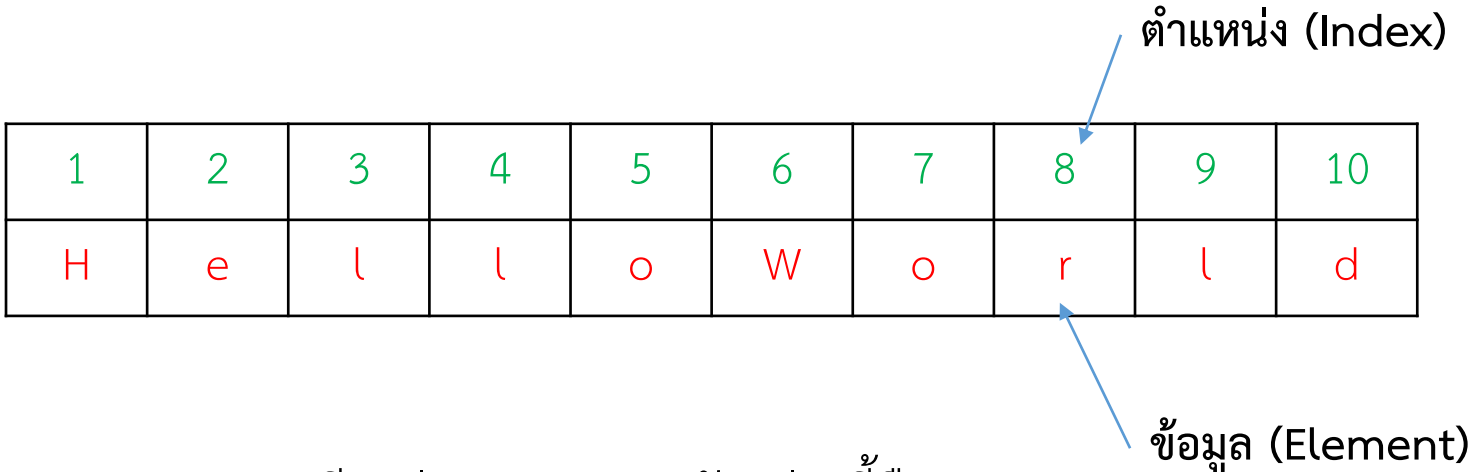
- Finite: จำนวนข้อมูลมีขนาดที่แน่นอน เปลี่ยนแปลงไม่ได้

- เป็นโครงสร้างข้อมูลแบบเชิงเส้น (linear)

ตัวอย่างการใช้ Array



ตัวอย่างการใช้ Array



ตำแหน่งทั้งหมดของ index เรียกว่า index set ตัวอย่างนี้คือ

Index set ={1,2,3,4,5,6,7,8,9,10}

ขอบเขตของตำแหน่งเรียกว่า range set ตัวอย่างนี้คือ 1 - 10

Index ที่ค่าน้อยที่สุดเรียกว่า Lower bound , Index ที่ค่าสูงที่สุดเรียกว่า Upper bound

ขนาดของ Array = Upper bound – Lower bound +1

ตัวอย่างการใช้ Array

X

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o	r	l	d

การเข้าถึงข้อมูล จะเข้าถึงโดยการอ้าง index

ในทางคณิตศาสตร์ นิยมระบุตำแหน่งของ Array ด้วยตัวห้อย (subscript notation)

$$X_1, X_2, X_3, \ldots\ldots X_n$$

$$X_6 = \text{‘W’}$$

ในภาษาคอมพิวเตอร์นิยมใช้วงเล็บเพื่อระบุตำแหน่ง

C, C++, Java ใช้ [ ] ตัวอย่างเช่น X[6] , X[7] , X[8]

Fortran , BASIC ใช้ ( ) ตัวอย่างเช่น X(6) , X(7) , X(8)

แต่ละภาษา จะมี lower bound ไม่เหมือนกัน

C, C++, Java ตำแหน่ง lower bound จะเป็น 0

x

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o	W	o	r	l	d

ตัวอย่างนี้ Range set คือ 0 - 9

BASIC , Visual BASIC ตำแหน่ง lower bound สามารถกำหนดได้เอง

x

-5	-4	-3	-2	-1	0	1	2	3	4
H	e	l	l	o	W	o	r	l	d

ตัวอย่างนี้ Range set คือ -5 - 4

เริ่มต้นใช้งาน Array

- ก่อนใช้งาน จะต้องทำการประกาศ Array ก่อนเสมอ

- Compiler จะทำการจองพื้นที่หน่วยความจำสำหรับใช้งานเป็น Array

- ขนาดของพื้นที่หน่วยความจำจะขึ้นอยู่กับ ชนิดข้อมูล และขนาดของ Array

ตัวอย่างการประกาศ Array ในภาษา C

```
Int X[100];
```

Range ของ X คือ 0 ถึง 99
ขนาด ของ X คือ 100

ขนาดของพื้นที่หน่วยความจำ = ขนาดของ Array X ขนาดของชนิดข้อมูล

Array

ขนาดของพื้นที่หน่วยความจำ = ขนาดของ Array X ขนาดของชนิดข้อมูล

ตัวอย่างเช่น

char X[100]; ขนาด ของ X คือ 100

ขนาดของ char คือ 1 byte

ขนาดของ X ในหน่วยความจำ = 100 X 1 = 100 bytes

long X[100];

ขนาดของ long คือ 4 bytes

ขนาดของ X ในหน่วยความจำ = 100 X 4 = 400 bytes

double X[100];

ขนาดของ double คือ 8 bytes

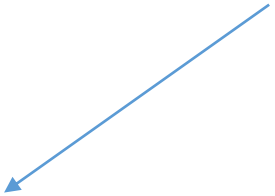
ขนาดของ X ในหน่วยความจำ = 100 X 8 = 800 bytes

Array ในหน่วยความจำ

ข้อมูลของ Array จะเรียงต่อกันในหน่วยความจำติดกันเป็นผืนเดียว โดยตำแหน่งแรกอาจเริ่มที่ตำแหน่งไหนก็ได้

ตำแหน่งเริ่มต้นของ Array เรียกว่า Base Address

char X[]={1,2,3,4,5,6};



X

1000	1001	1002	1003	1004	1005
1	2	3	4	5	6

Base Address คือ 1000

long X[]={1,2,3,4,5,6};

X

1000	1004	1008	1012	1016	1020
1	2	3	4	5	6

Base Address คือ 1000

X

1000	1004	1008	1012	1016	1020
0001	0002	0003	0004	0005	0006

แต่ละ element จะใช้พื้นที่จนเต็ม

การหา Address ในหน่วยความจำจาก index

Address X[j] = Base Address + ขนาดข้อมูล x j

ตัวอย่างเช่น double X[100]

ตำแหน่ง X[91] จะอยู่ที่ Address ใด ? เมื่อกำหนดให้ Base Address คือ 1234

ขนาดของ Double คือ 8

j = 91

Address X[91] = 1234+ 8 x 91

= 1962

การหาตำแหน่งข้อมูล สามารถคำนวณได้ ดังนั้นการเข้าถึงข้อมูล Array ในตำแหน่งต่าง ๆ จึงกระทำได้เร็วมาก

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o	W	o	r	l	d

เร็วมากในกรณีนี้ หมายถึงมีความซับซ้อน (time complexity) เป็นเท่าใด

O(1)

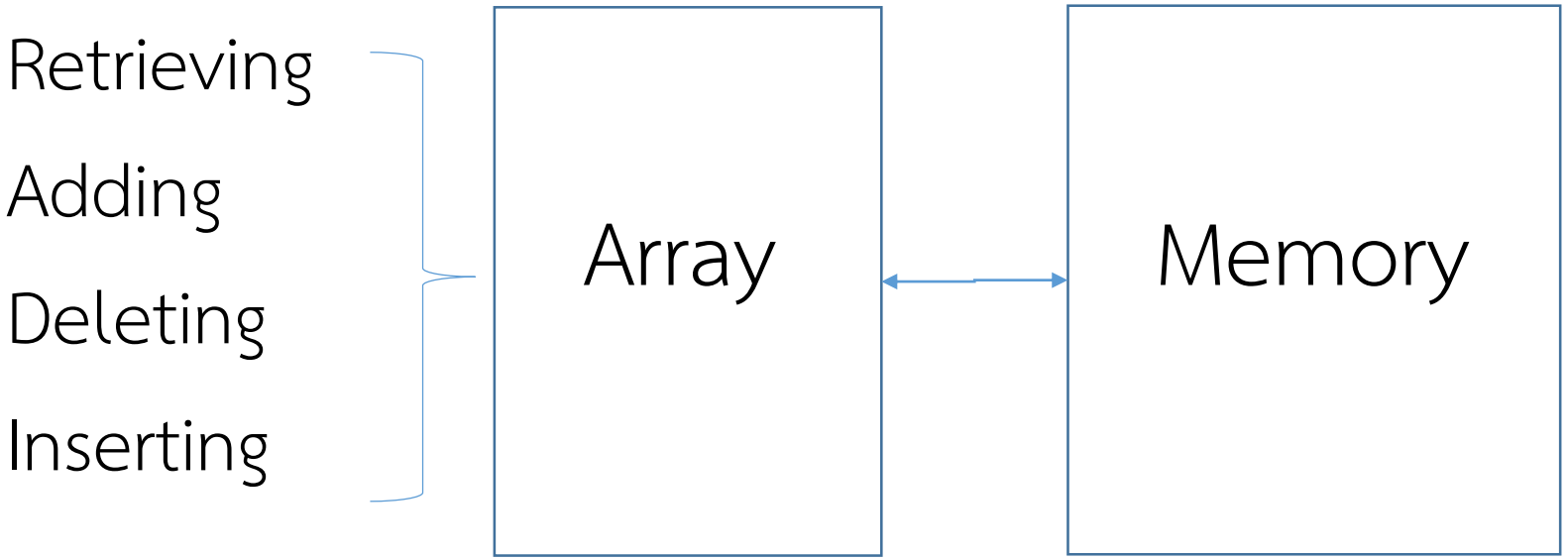
ไม่มีอะไรจะเร็วกว่านี้แล้ว

Array จะมีขนาดเท่าใด ก็ใช้เวลาในการเข้าถึงข้อมูลเท่ากัน

Array: Operation

การกระทำของ Array (Operation)

- ค้นคืน (Retrieving)
- เพิ่ม (Adding)
- ลบ (Deleting)
- แทรก (Inserting)



ค้นคืน (Retrieving)

ค้นคืน คือ การเข้าถึงข้อมูล

การเข้าถึงข้อมูล จะต้องรู้ตำแหน่ง index ของข้อมูลที่ต้องการเข้าถึง

ตำแหน่ง index สามารถคำนวณได้ทันที

ข้อมูลสามารถเข้าถึงได้โดยตรง (Direct Access)

การค้นคืนมีความซับซ้อนทางเวลาเป็น..

O(1)

เพิ่ม (Adding)

การเพิ่มคือการใส่ข้อมูลลงไปใน ตำแหน่งที่ต้องการ

หรือเรียกว่า การกำหนดค่า (Assigning)

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o	W	o	r	l	d

หากตำแหน่งที่ต้องการกำหนดค่ามีข้อมูลเดิมอยู่ ข้อมูลเดิมจะถูกเขียนทับ

การเพิ่มข้อมูลมีความซับซ้อนทางเวลาเป็น..

$$O(1)$$

ลบ (Deleting)

คือการลบข้อมูลในตำแหน่ง index ที่ต้องการออกไป

ข้อมูลที่อยู่ในตำแหน่งถัดไปจะถูกเลื่อนเข้ามาแทนที่ (เลื่อนขวามาซ้าย)

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o	r	l	d

ลบตำแหน่งที่ 8

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o		l	d

ลบ

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o	l		d

เลื่อน

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o	l	d	

เลื่อน

ลบ (Deleting)

Array มีขนาดเล็กลงเท่าใด ?

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o	l	d	

เท่าเดิม เพราะพื้นที่ถูกจองไว้ตั้งแต่แรกแล้ว

ขนาดของ Array เปลี่ยนแปลงไม่ได้ ไม่งั้นจะผิดนิยาม

ลบ (Deleting)

การลบจะต้องเสียเวลาในการเลื่อนข้อมูล

เวลาในการเลื่อนจะแตกต่างกันไป ตามสถานการณ์

แย่ที่สุด (worse case)

ลบ Index ตัวแรก

ความซับซ้อนทางเวลาจะเป็น

O(N-1) หรือ O(N)

ดีที่สุด (best case)

ลบ Index ตัวสุดท้าย

ความซับซ้อนทางเวลาจะเป็น

O(1) เพราะไม่ต้องเลื่อนข้อมูล

แทรก (Inserting)

การเพิ่มคือการใส่ข้อมูลเพิ่มลงไปใน ตำแหน่งที่ต้องการ โดยไม่ทับข้อมูลเดิม

ข้อมูลเดิมจะถูกขยับไปทางด้านขวาก่อนเพิ่มข้อมูลใหม่ลงไป

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o	l	d	

เพิ่ม r ลงใน ตำแหน่งที่ 8

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o	l		d

เลื่อน

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o		l	d

เลื่อน

1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	W	o	r	l	d

เพิ่ม

แทรก (Inserting)

การแทรกต้องเสียเวลาในการเลื่อนข้อมูล

เวลาในการเลื่อนจะแตกต่างกันไป ตามสถานการณ์

แย่ที่สุด (worse case)

แทรก Index ตัวแรก

ความซับซ้อนทางเวลาจะเป็น

O(N-1) หรือ O(N)

ดีที่สุด (best case)

แทรก Index ตัวสุดท้าย

ความซับซ้อนทางเวลาจะเป็น

O(1) เพราะไม่ต้องเลื่อนข้อมูล

แทรก (Inserting)

จะแทรกข้อมูลเกินขนาด Array ไม่ได้

จะรู้ได้อย่างไรว่า ข้อมูลเกินหรือยัง เพราะขนาดของ Array มันคงที่ตลอดเวลา ?

ตอบ. ต้องสร้างตัวแปรมา track การเพิ่มหรือลด แล้วเช็คเอาจากตัวแปรนี้

สำรวจ (Traversing)

การสำรวจ หรือ เยี่ยมชม (Traversing, Visiting) คือการเข้าถึงข้อมูลใน element ต่าง ๆ อย่างน้อย 1 ครั้งจนครบ ทุก element เป็น operation ภายในของโครงสร้างข้อมูล ซึ่งนิยมใช้ในการ

หาค่าสูงสุด หาค่าน้อยสุด หาผลรวม หาค่าเฉลี่ย

ความซับซ้อนทางเวลาของการสำรวจ คือ ...

O(N)

ไม่มี best case เพราะยังไงก็ต้องผ่านทุก element

สรุป Worst case ของ Array

Operation	Complexity
Accessing	O(1)
Inserting	O(N)
Deleting	O(N)
Traversing	O(N)

Array 2 มิติ

Array 2 มิติ คือการนำเอา Array 1 มิติมารวมกลุ่มกัน

สามารถจะมองเป็น Array ที่อยู่ใน Array ก็ได้

มีลักษณะเป็นตาราง

ทุก element เป็นข้อมูลที่มีขนาดเท่ากัน และ Array มีขนาดที่แน่นอน

ต้องใช้ index จำนวน 2 ตัวในการระบุตำแหน่ง

X

80	20	70	55	32	70	76
40	55	56	80	0	27	14
55	68	72	35	44	21	0
60	75	62	75	0	100	85

Array 2 มิติ

ต้องใช้ index จำนวน 2 ตัวในการระบุตำแหน่ง

นิยมระบุตำแหน่งเป็นแบบ row และ column (แนวดิ่ง และ แนวนอน)และเริ่มนับตำแหน่งแรกเริ่มจาก 0

X[3][2]= 62

X

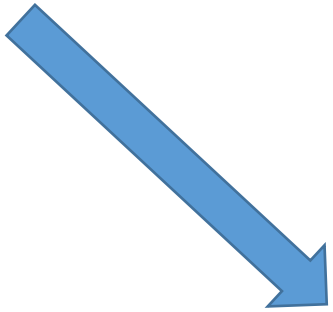
80	20	70	55	32	70	76
40	55	56	80	0	27	14
55	68	72	35	44	21	0
60	75	62	75	0	100	85

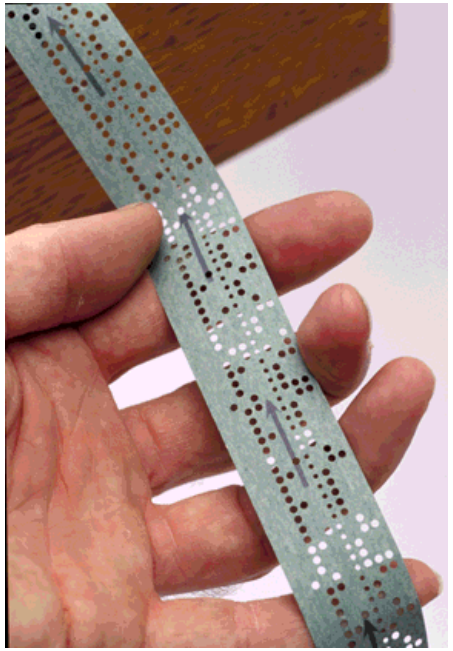
Array 2 มิติ

ถึงแม้จะมี 2 มิติ แต่ในการเก็บลงหน่วยความจำ ก็ต้องเก็บแบบ 1 มิติอยู่ดี

X

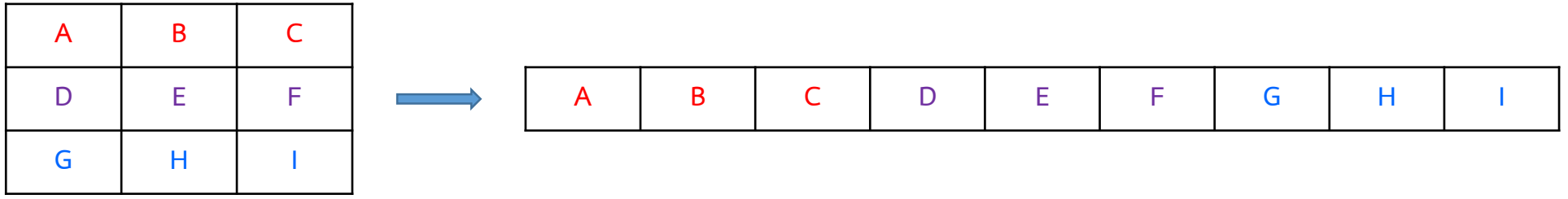
80	20	70	55	32	70	76
40	55	56	80	0	27	14
55	68	72	35	44	21	0
60	75	62	75	0	100	85





การบันทึกลงสู่หน่วยความจำ

วิธีที่ 1: แนวแถว (Row Major Storage) จะบันทึกทีละแถว



Address A[j][k]=Base Address + ขนาดข้อมูล x จำนวนแถว x j + k

j คือ row เริ่มจาก 0

K คือ column เริ่มจาก 0

การบันทึกลงสู่หน่วยความจำ

วิธีที่ 1: แนวแถว (Row Major Storage) จะบันทึกทีละแถว

char X[3][3];

X

A	B	C
D	E	F
G	H	I

A	B	C	D	E	F	G	H	I

เมื่อกำหนด base address = 200

X[2][1] จะอยู่ที่ตำแหน่งใดในหน่วยความจำ

จำนวนแถว =3 char มีขนาด 1 byte

j=2, k=1

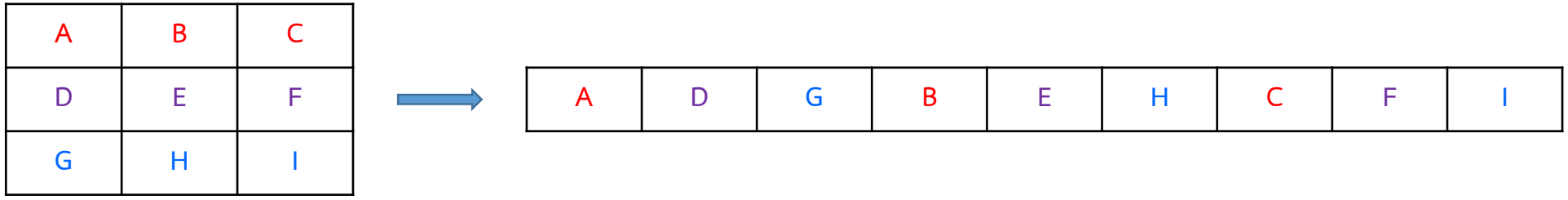
Address A[j][k] = Base Address + ขนาดข้อมูล x จำนวนแถว x j + k

Address A[j][k] = 200 + 1 x 3 x 2 + 1

Address A[j][k] = 207

การบันทึกลงสู่หน่วยความจำ

วิธีที่ 2: แนวสดมภ์ (Column Major Storage) จะบันทึกทีละสดมภ์



Address A[j][k]=Base Address + ขนาดข้อมูล x จำนวนสดมภ์ x k + j

j คือ row เริ่มจาก 0

K คือ column เริ่มจาก 0

การบันทึกลงสู่หน่วยความจำ

วิธีที่ 2: แนวสดมภ์ (Column Major Storage) จะบันทึกทีละสดมภ์

char X[3][3];

X

A	B	C
D	E	F
G	H	I

A	D	G	B	E	H	C	F	I

เมื่อกำหนด base address = 200

X[2][1] จะอยู่ที่ตำแหน่งใดในหน่วยความจำ

จำนวนแถว =3 char มีขนาด 1 byte

j=2, k=1

Address A[j][k]=Base Address + ขนาดข้อมูล x จำนวนสดมภ์ x k + j

Address A[j][k]= 200+ 1 x 3 x 1 + 2

Address A[j][k]= 205

ความซับซ้อนทางเวลาของ Array 2 มิติ

Operation	Complexity
Accessing	O(1)
Inserting	O(NM)
Deleting	O(NM)
Traversing	O(NM)

N คือ จำนวนแถว

M คือ จำนวนสดมภ์

การบันทึกลงสู่หน่วยความจำ

Row Major: C, C++, Objective-C, PL/I, Pascal

Column Major: Fortran

ไม่ใช่ทั้งสองแบบ : Java, Python

Array สำหรับบันทึกสายอักขระ

String คือ Array ของ char โดยมีรหัส null (\0) ปิดท้ายข้อความ

ขนาดของ string ต้องใหญ่กว่าความยาวข้อความ 1 ช่องเสมอ

char X[]=“hello”

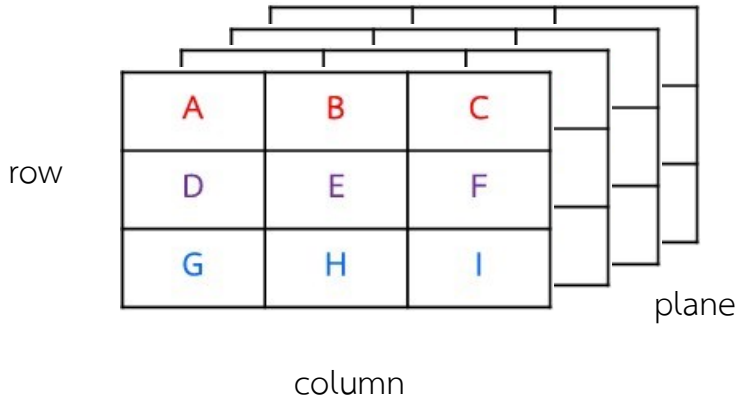
h	e	l	l	o	\0

สามารถสร้าง array ของ string ได้ โดยใช้ string 2 มิติ

h	e	l	l	o	\0
w	o	r	l	d	\0
m	j	u	\0	\0	\0

Array ขนาด 3 มิติ

คือ Array 2 มิติ ที่นำแต่ละมิติมาซ้อนกัน เป็นแผ่นๆ เรียกว่า plane



Array: Example

ตัวอย่างการใช้งาน Array

สายอักขระ

Strings Array: M

M[0,0]	M[0,1]	M[0,2]	M[0,3]	M[0,4]	M[0,5]	M[0,6]	M[0,7]	M[0,8]	M[0,9]
F	O	U	Z	\0	\0	\0	\0	\0	\0
M[1,0]	M[1,1]	M[1,2]	M[1,3]	M[1,4]	M[1,5]	M[1,6]	M[1,7]	M[1,8]	M[1,9]
S	Y	M	A	\0	\0	\0	\0	\0	\0
M[2,0]	M[2,1]	M[2,2]	M[2,3]	M[2,4]	M[2,5]	M[2,6]	M[2,7]	M[2,8]	M[2,9]
Z	A	H	I	D	\0	\0	\0	\0	\0
M[3,0]	M[3,1]	M[3,2]	M[3,3]	M[3,4]	M[3,5]	M[3,6]	M[3,7]	M[3,8]	M[3,9]
T	A	U	S	E	E	N	\0	\0	\0
M[4,0]	M[4,1]	M[4,2]	M[4,3]	M[4,4]	M[4,5]	M[4,6]	M[4,7]	M[4,8]	M[4,9]
E	L	I	Z	A	B	E	T	H	\0

Array: Example

ตัวอย่างการใช้งาน Array

Matrix

	A	R	N	D	C	Q	E	G	H	1	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Array: Example

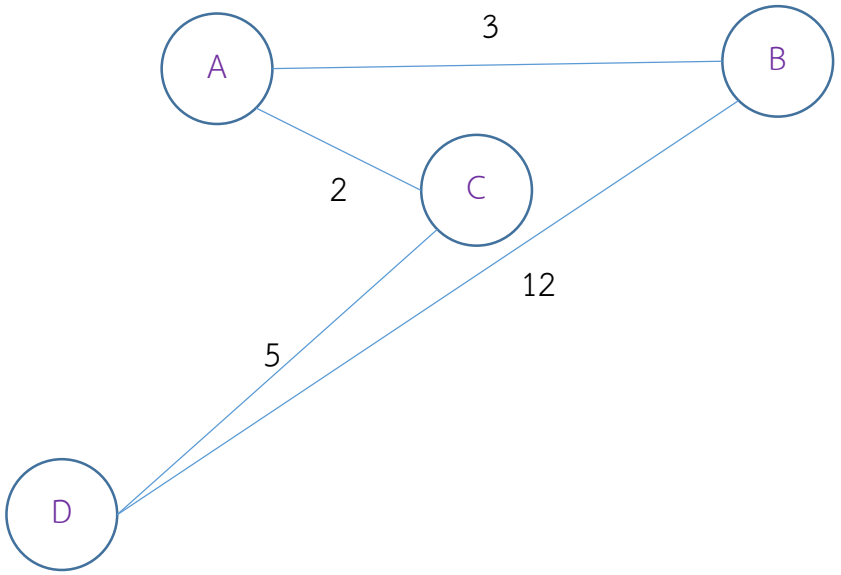
ตัวอย่างการใช้งาน Array

ภาพ

0.5020	0.4941	0.4902	0.4078	0.4824	0.3608	0.2941	0.3686	0.5451	0.6431	0.4431	0.1412	0.1294	0.1255	0.1373	0.1412	0.1255	0.1176	0.1216	0.1333	0.1333
0.5020	0.4980	0.4863	0.3059	0.4510	0.3922	0.2706	0.2863	0.4745	0.5137	0.2078	0.1176	0.1098	0.1137	0.1294	0.1373	0.1294	0.1255	0.1255	0.1451	0.1412
0.5020	0.4941	0.4784	0.2392	0.3137	0.3569	0.3804	0.3451	0.3490	0.3255	0.1373	0.1451	0.1373	0.1333	0.1373	0.1373	0.1333	0.1294	0.1294	0.1529	0.1412
0.5020	0.4902	0.4627	0.2863	0.2314	0.2902	0.4392	0.4235	0.2902	0.2392	0.2078	0.1882	0.1765	0.1608	0.1451	0.1451	0.1451	0.1451	0.1412	0.1412	0.1333
0.5059	0.4784	0.4353	0.3804	0.2902	0.2627	0.3451	0.3647	0.2549	0.1804	0.1922	0.1922	0.1882	0.1725	0.1529	0.1529	0.1647	0.1647	0.1569	0.1176	0.1216
0.5098	0.4706	0.4118	0.3922	0.4196	0.3608	0.2784	0.2784	0.2353	0.1373	0.1333	0.1647	0.1725	0.1686	0.1569	0.1608	0.1686	0.1647	0.1490	0.1059	0.1333
0.5098	0.4627	0.3961	0.3333	0.5137	0.5020	0.3255	0.2863	0.2627	0.1608	0.1569	0.1451	0.1569	0.1647	0.1569	0.1608	0.1647	0.1490	0.1255	0.1059	0.1529
0.5255	0.4588	0.5255	0.3843	0.3725	0.3765	0.3098	0.2667	0.2627	0.2078	0.1961	0.1725	0.1647	0.1608	0.1686	0.1686	0.1490	0.1098	0.0784	0.1216	0.1451
0.5725	0.5529	0.5922	0.3373	0.3020	0.2863	0.2235	0.1961	0.2078	0.1804	0.1922	0.2039	0.2196	0.2196	0.1961	0.1608	0.1333	0.1333	0.1412	0.1647	0.1294
0.6157	0.6431	0.6000	0.2941	0.2627	0.2745	0.2471	0.2314	0.2235	0.1804	0.1882	0.1765	0.1843	0.1804	0.1490	0.1176	0.1059	0.1255	0.1490	0.1647	0.1686
0.6392	0.6745	0.4902	0.2588	0.2275	0.2588	0.2588	0.2588	0.2549	0.2118	0.2353	0.2157	0.1961	0.1686	0.1569	0.1529	0.1569	0.1529	0.1490	0.1647	0.2471
0.6706	0.6510	0.3412	0.2157	0.1725	0.1804	0.1569	0.1608	0.1922	0.2000	0.2471	0.2118	0.2000	0.1882	0.1804	0.1765	0.1686	0.1569	0.1490	0.2314	0.3098
0.6980	0.5765	0.2392	0.1882	0.1765	0.1882	0.1333	0.1059	0.1255	0.1176	0.1490	0.1137	0.1451	0.1725	0.1686	0.1451	0.1412	0.1725	0.2039	0.3020	0.3804
0.6941	0.4510	0.1922	0.1647	0.2039	0.2431	0.1725	0.1176	0.1255	0.0902	0.0863	0.1176	0.1490	0.1765	0.1804	0.1725	0.1961	0.2549	0.3059	0.3647	0.4431
0.6627	0.3373	0.1686	0.1373	0.1882	0.2314	0.1451	0.0980	0.1412	0.1373	0.1451	0.1922	0.1804	0.1647	0.1608	0.1843	0.2235	0.2706	0.3020	0.4314	0.4588
0.5176	0.2039	0.1098	0.1373	0.2471	0.1608	0.1176	0.1451	0.1412	0.0392	0.2157	0.1608	0.1451	0.1529	0.1804	0.2000	0.2235	0.2824	0.3412	0.3686	0.4863
0.4314	0.2235	0.1765	0.1843	0.1529	0.1294	0.2000	0.0941	0.0784	0.1922	0.0902	0.1490	0.1333	0.1451	0.1804	0.2157	0.2431	0.3020	0.3608	0.4588	0.5451
0.7647	0.8157	0.8745	0.7216	0.4196	0.1137	0.0941	0.1529	0.1255	0.1333	0.0863	0.1333	0.1216	0.1373	0.1882	0.2314	0.2706	0.3294	0.3882	0.5020	0.5725
0.8627	0.8863	0.8588	0.9255	0.8902	0.6118	0.1255	0.0157	0.1608	0.0784	0.1412	0.1255	0.1176	0.1373	0.1922	0.2431	0.2863	0.3490	0.4078	0.4980	0.5725
0.9137	0.9059	0.8431	0.8353	0.8824	0.9255	0.5922	0.0510	0.0784	0.1059	0.1176	0.1216	0.1216	0.1451	0.1961	0.2392	0.2824	0.3529	0.4196	0.5216	0.6000
0.8627	0.8863	0.8745	0.8902	0.7843	0.8118	0.9255	0.4118	0.0275	0.1020	0.1059	0.1176	0.1255	0.1529	0.1961	0.2275	0.2667	0.3490	0.4275	0.5490	0.6235

ตัวอย่างการใช้งาน Array

กราฟ



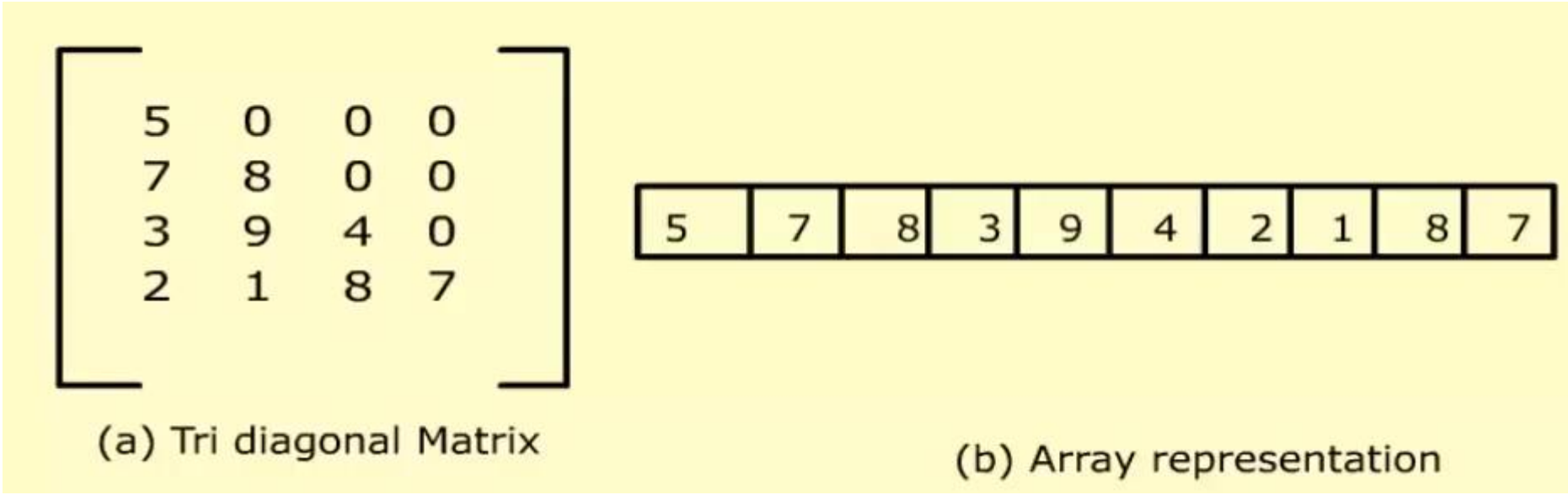
node

node	A	B	C	D
A	0	3	2	∞
B	3	0	∞	12
C	2	∞	0	5
D	∞	12	5	0

0, 3, 2, ∞, 3, 0, ∞, 12, 2, ∞, 0, 5, ∞, 12, 5, 0

Array: Example

Tri diagonal Matrix

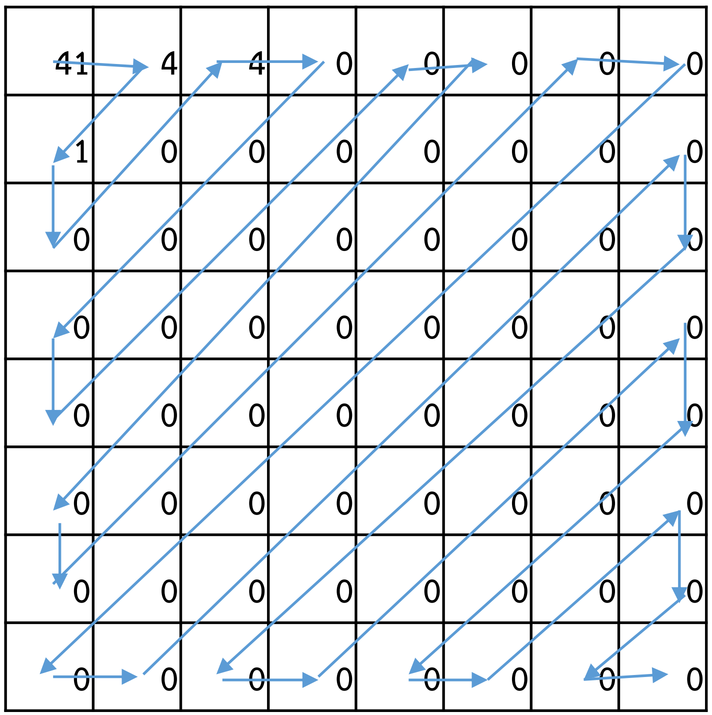


(a) Tri diagonal Matrix

(b) Array representation

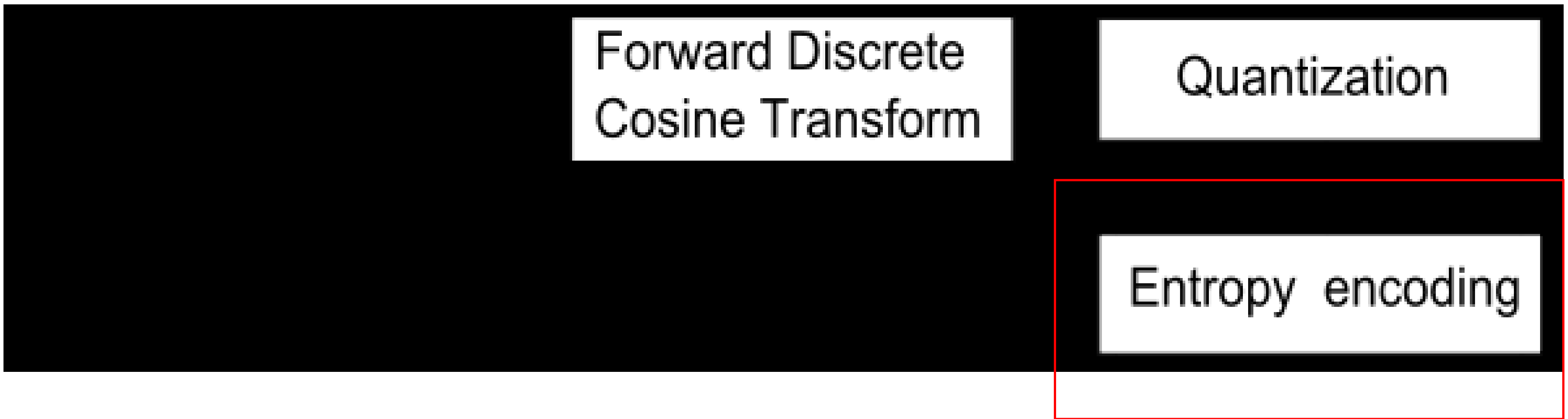
Array: Example

การกวาดแบบซิกแซ็กในการบีบอัดภาพแบบ jpeg



เพื่อให้ข้อมูลมีค่าเหมือนกันอยู่ติดกันให้ยาวที่สุด

41,4,1,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0



สรุป

ข้อดี

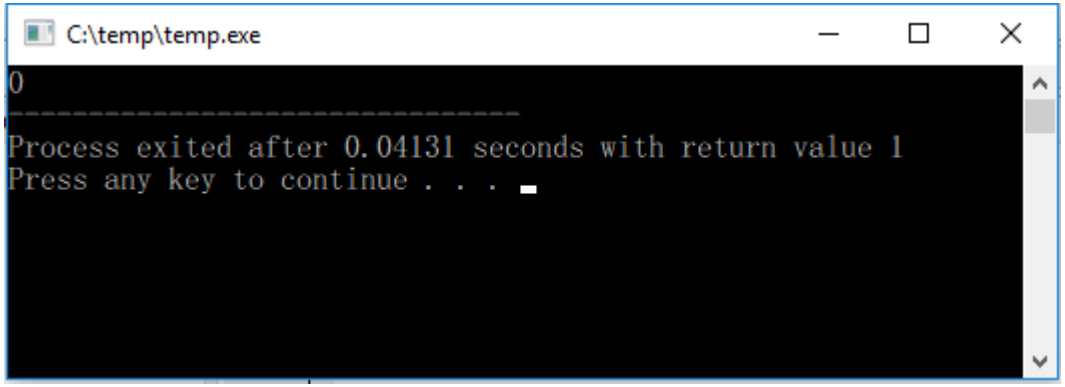
- เก็บข้อมูลได้หลายค่าด้วยตัวแปรเดียว
- เข้าถึงข้อมูลได้เร็ว O(1)
- ทุก element เข้าถึงได้โดยตรง (Direct Access)
- เขียนโปรแกรมได้ง่าย
- นิยมใช้คู่กับ for loop
- Array หลายมิติ มีโครงสร้างเป็นลำดับชั้นชัดเจน

สรุป

ข้อเสีย

- เปลี่ยนขนาดขณะรันไม่ได้ (runtime)
- หากใช้เก็บข้อความที่ยาวไม่เท่ากัน จะเปลืองพื้นที่
- แทรก และ ลบ ทำได้อย่างไม่มีประสิทธิภาพ
- ใช้กับข้อมูลขนาดใหญ่ไม่ได้
- บางภาษาไม่มีการตรวจสอบ range ตัวอย่างเช่น ภาษา C

```c
#include <stdio.h>
main()
{
    char X[]={1,2,3,4,5,6,7,8,9};
    printf("%d",X[10]);
}
```



Array: Example

การตรวจสอบ range ของ Array ในภาษา Java

```
1  class StringIndexDemo
2  {
3      public static void main(String[] args)
4      {
5          String myString = "Hello";
6          for(int i = 0; i <= myString.length(); i++)
7          {
8              System.out.println(myString.charAt(i));
9          }
10     }
11 }
```

```
C:\Windows\system32\cmd.exe
H
e
l
l
o
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String ind
ex out of range: 5
        at java.lang.String.charAt(String.java:686)
        at StringIndexDemo.main(StringIndexDemo.java:8)
```