

**LECTURER: TAI LE QUY**

# **INTRODUCTION TO COMPUTER SCIENCE**

---

**Basic Concepts of Data Processing**

**1**

---

**Information Representation**

**2**

---

**Algorithms and Data Structures**

**3**

---

**Propositional Logic, Boolean Algebra and Circuit Design**

**4**

---

**Hardware and Computer Architectures**

**5**

---

**Networks and the Internet**

**6**

---

**Software**

**7**

---

**Computer Science as a Discipline**

---

**8**

**UNITS 2 & 3**

# **INFORMATION REPRESENTATION & ALGORITHMS AND DATA STRUCTURES**

## STUDY GOALS



- Understand the binary numbering system and how to convert decimal to binary.
- Learn about data storage sizes, such as kilobyte, megabyte, and terabyte.
- Master how graphics, documents, music, and other data are represented in memory.
- Understand about the different data types used in programming such as char, string, int, and float.
- Explore how computers perform error checking on stored and network data.

## STUDY GOALS

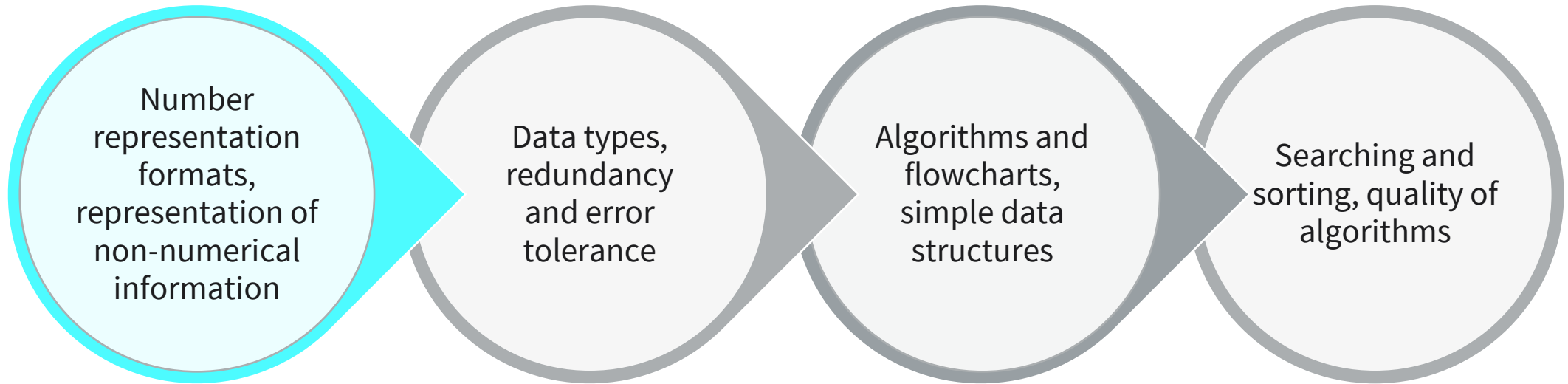


- Learn how the term algorithm is used in applications of computer science.
- Understand how to create flowcharts.
- Explore the details of data structures, such as arrays and stacks.
- Learn how sorting algorithms work, including quick and bubble sort.
- Master how the quality of different algorithms is evaluated.



1. What is the difference between analog and digital data?
2. Why does the “k” in “kg” stand for a different size than in “kB”?
3. What happens if you want to store the value of Pi (3.14) in an integer variable?

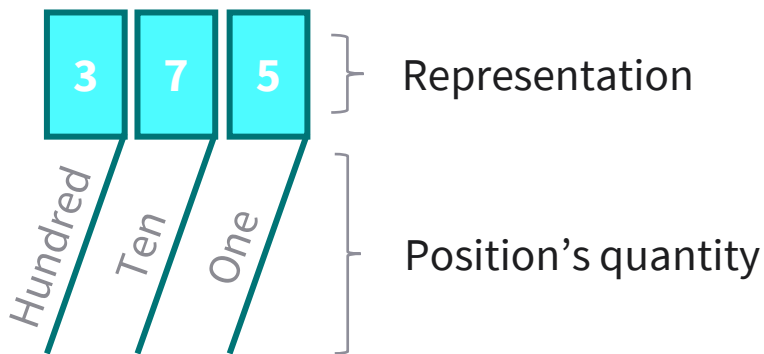
## INFORMATION REPRESENTATION



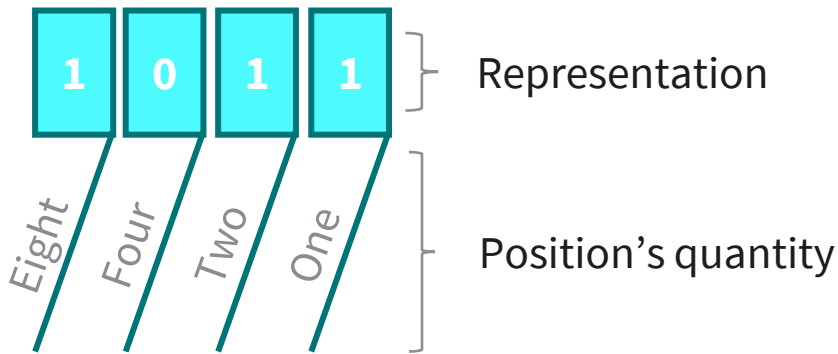


THE BASE TEN AND BINARY SYSTEMS

a. Base 10 system

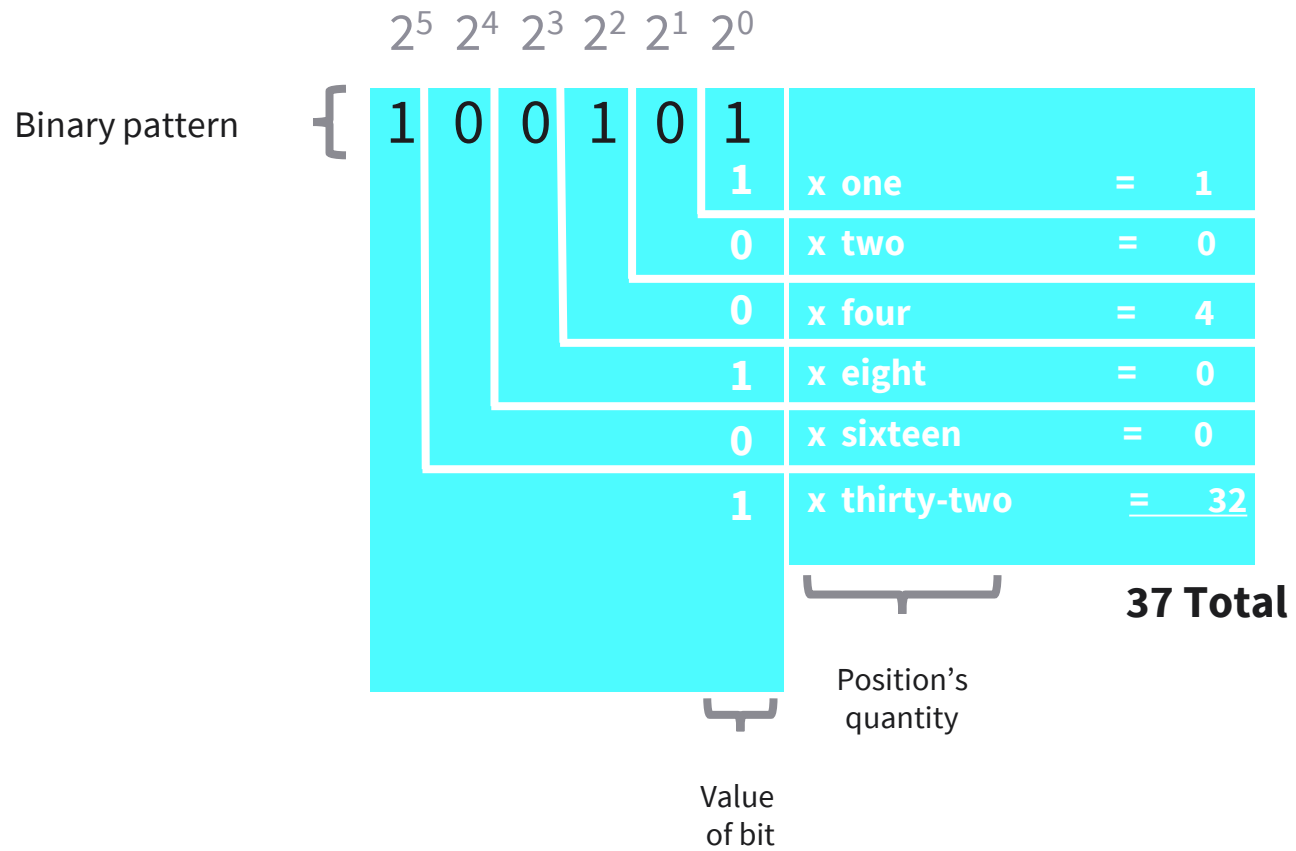


b. Base 2 system



$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
128	64	32	16	8	4	2	1	
1	1	0	1	1	1	1	0	= 222

## DECODING THE BINARY REPRESENTATION 100101



## ALGORITHM TO OBTAIN THE BINARY REPRESENTATION OF THE NUMBER 13

### Step 1

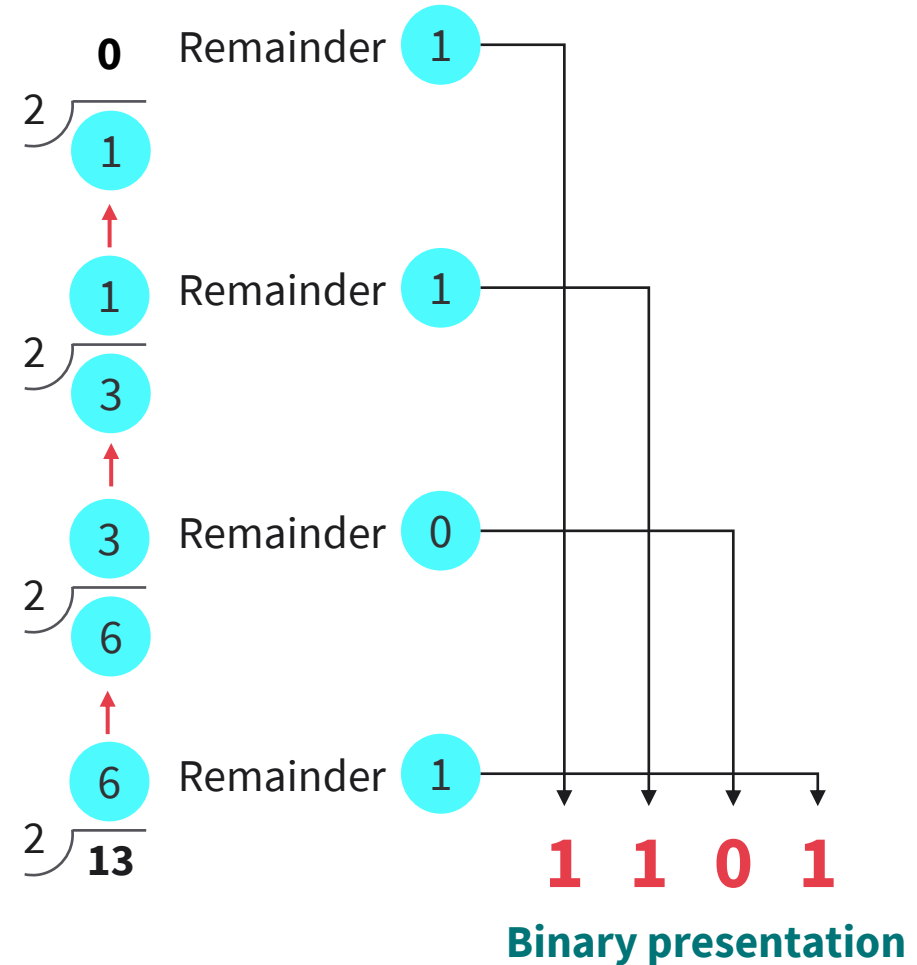
- Divide the value by 2 and record the remainder.

### Step 2

- As long as the quotient obtained is not 0, continue to divide the newest quotient by 2 and record the remainder.

### Step 3

- Now that the quotient of 0 has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.



## HEXADECIMAL NOTATION

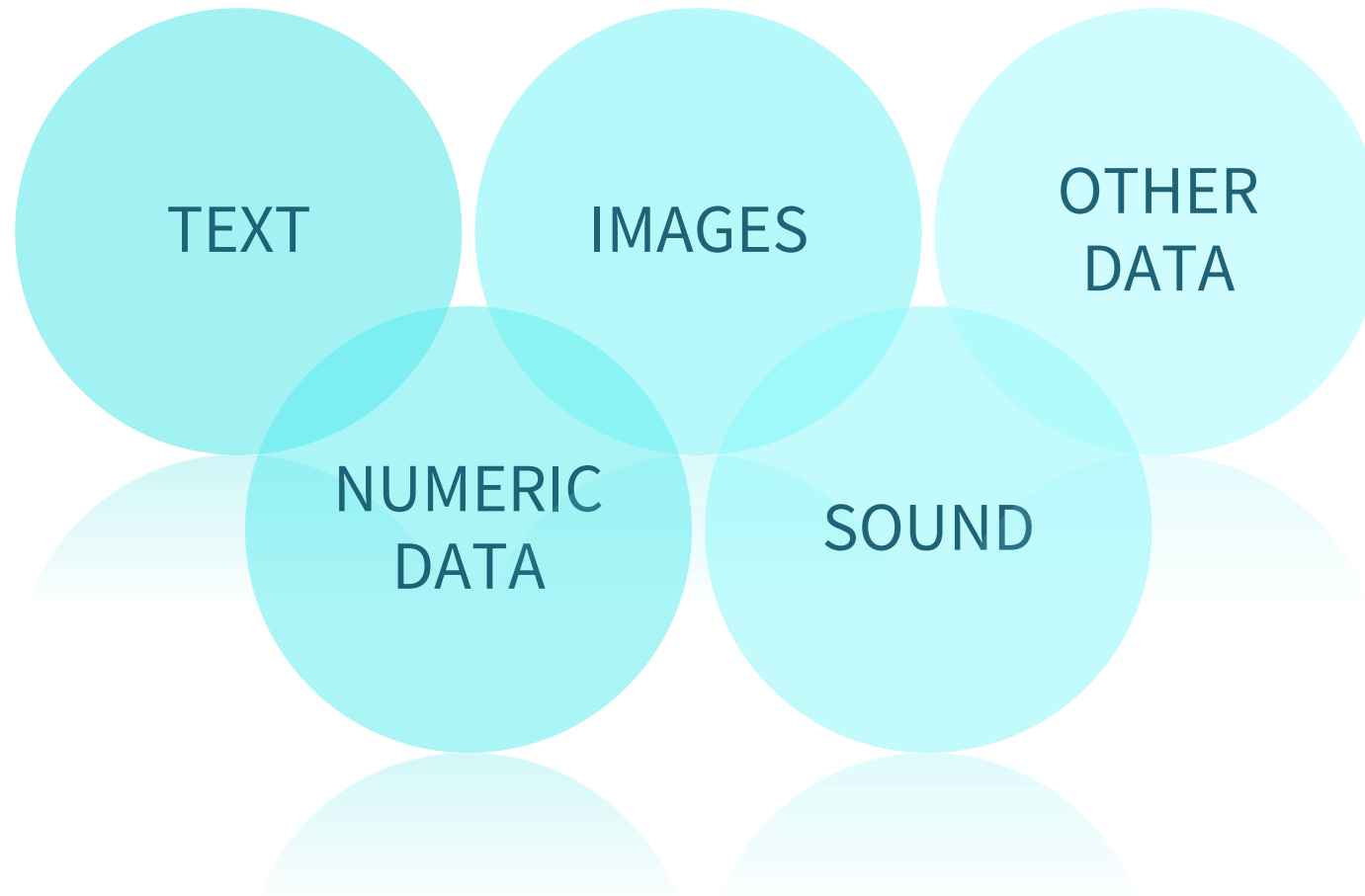
- a shorthand notation for long bit patterns
- divides a pattern into groups of four bits each
- represents each group by a single symbol

Example: **10110101** becomes **0xB5**

Dec	Hex	Bin
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	a	1010
11	b	1011
12	c	1100
13	d	1101
14	e	1110
15	f	1111

## REPRESENTING INFORMATION AS BIT PATTERNS

- Many kinds of information can be encoded as bit patterns.
- Systems for encoding information have been established for:



## REPRESENTING IMAGES

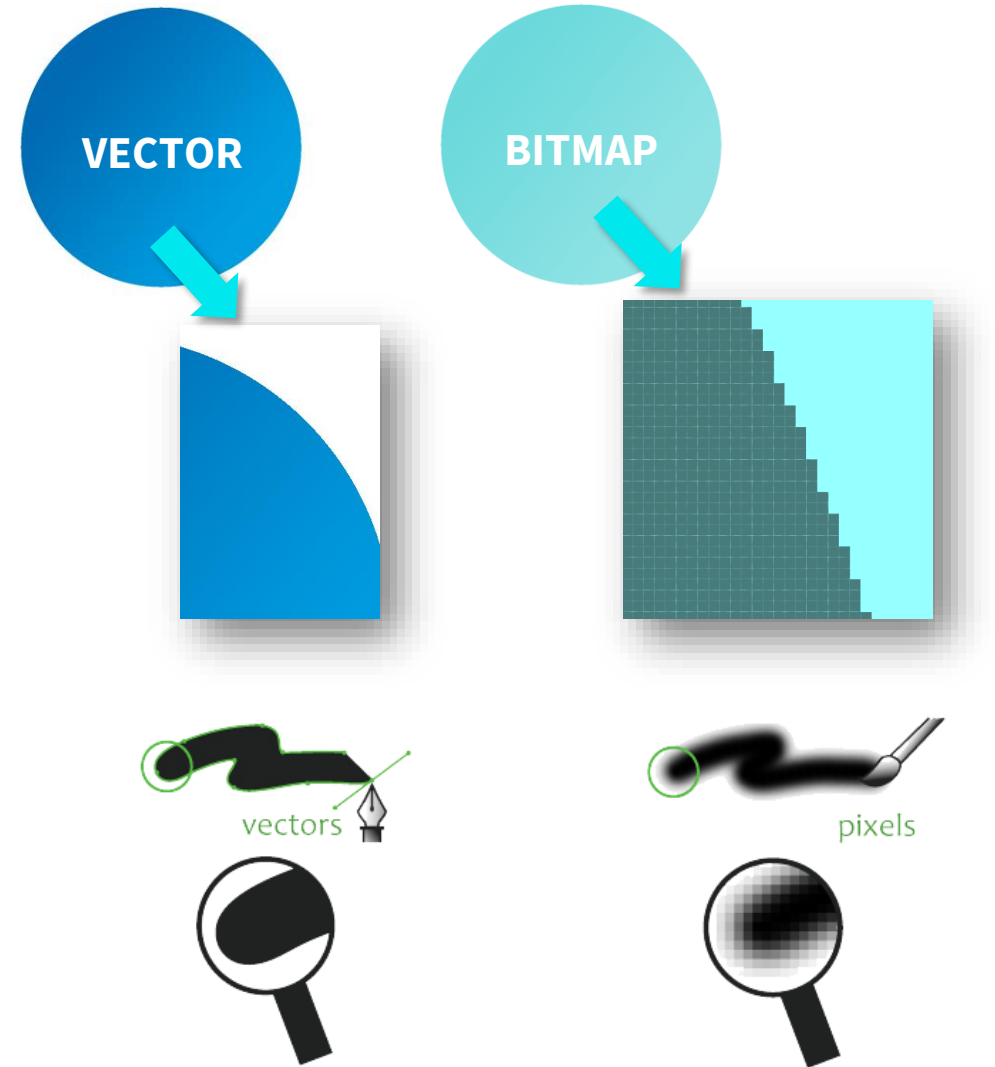
### Bit map techniques

Pixel:

- “picture element” represents one dot
- RGB: Red, Green, and Blue components
- luminance and chrominance
- problems with scaling up images

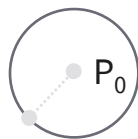
### Vector techniques

- represent images with geometric structures
- scalable
- TrueType and PostScript

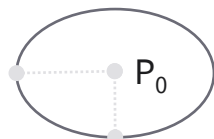


VECTOR ELEMENTS

Primitive Graphs



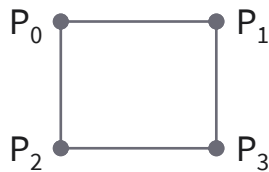
Circle



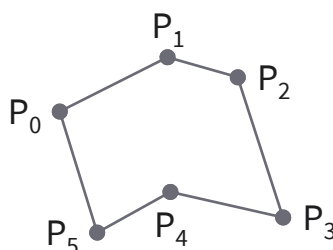
Ellipse



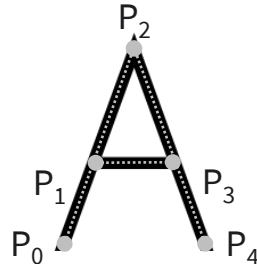
Path



Rectangle

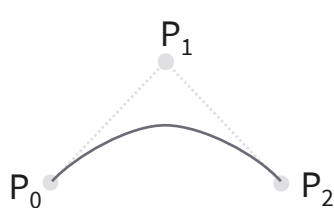


Polygon

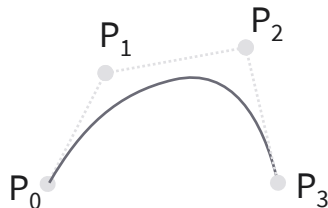


Text

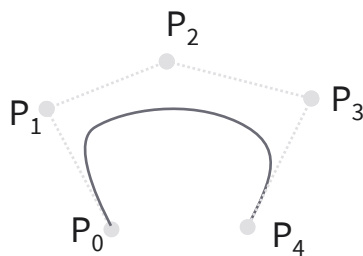
Graphs with control polygons



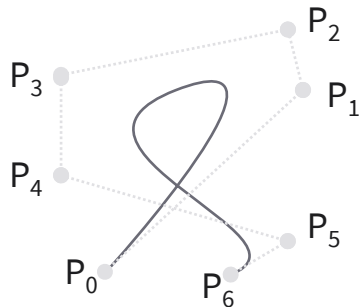
quadratic



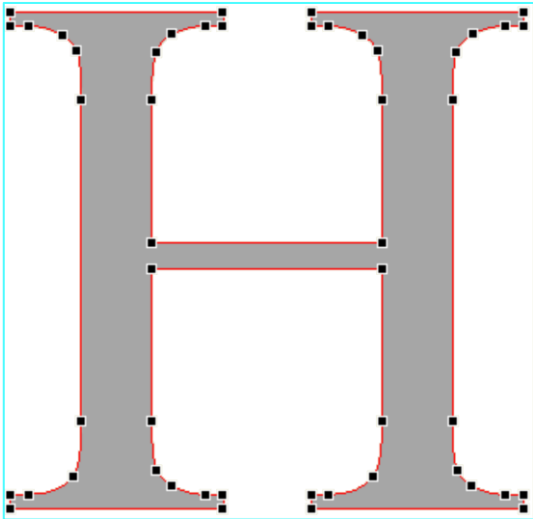
cubic



quartic



sextic



Character made out of Bezier elements

RGB COLOR SPACE



	Print Channel	Relative Channel
R		
G		
B		

The **RGB** color space is an additive color space using a mixture of red, green and blue, according to the color sensors of the human eye.



CMYK COLOR SPACE



**CMYK** is a subtractive color model, mixing colors cyan, magenta and yellow.  
It is the basis for four-color printing systems, which use a key plate for black areas.  
Systems based on CMYK are device dependent and need a color profile to reproduce colors exactly.

	Print Channel	Relative Channel
C		
M		
Y		
K		

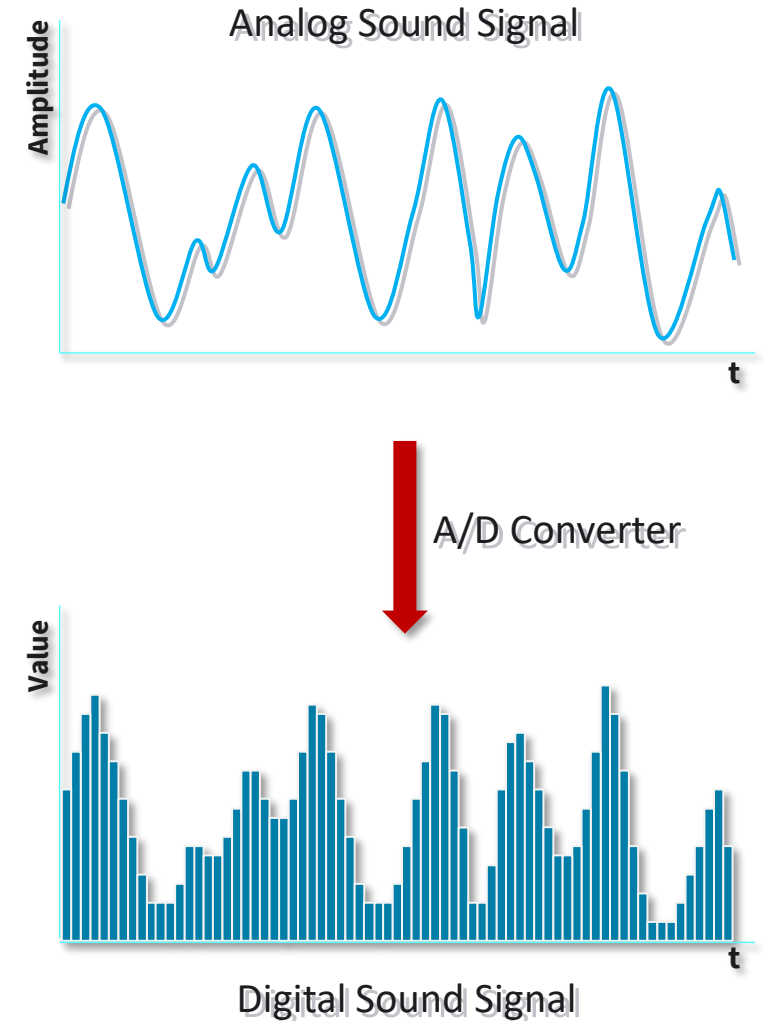
## REPRESENTING SOUND

The conversion of analog sound data to digital data is called **digitization**.

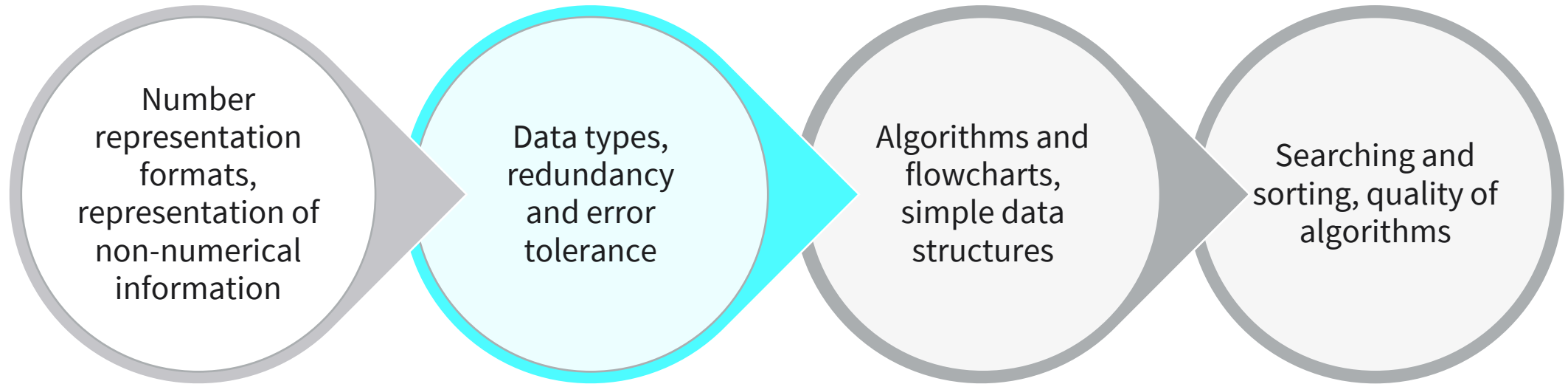
The analog sound signals are sampled rapidly over time, storing the value of the amplitude in a binary number.

Sampling techniques that record actual audio:

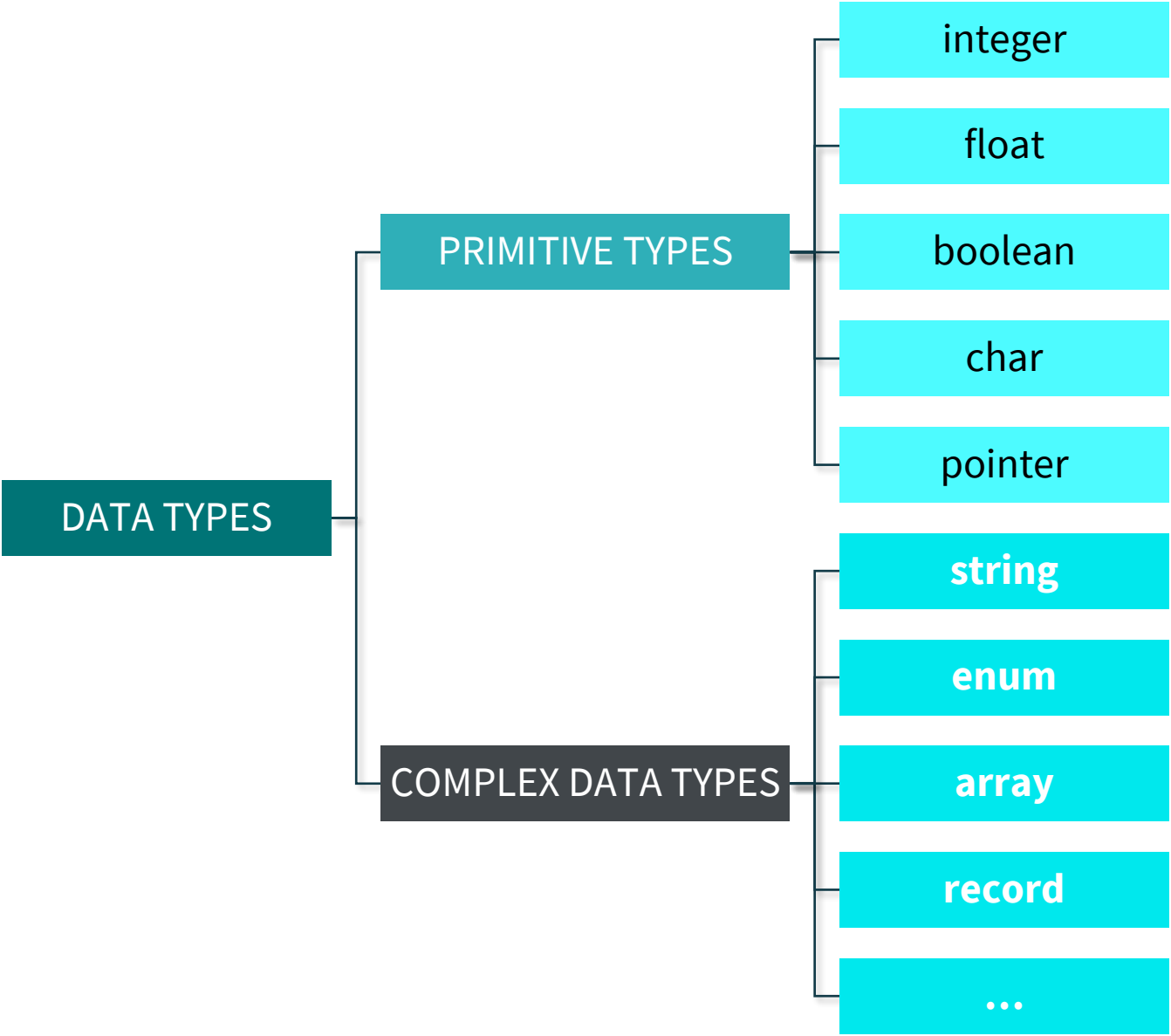
- long-distance telephone: **8000 samples/sec**
- CD sound: **44,100 samples/sec**



## INFORMATION REPRESENTATION



DATA TYPES



## PRIMITIVE TYPES: INTEGER AND CHAR

Programming languages often provide several sizes/ ranges

in C/C++/Java	<b>short</b>	(e.g., 2 bytes in Java)
	<b>int</b>	(e.g., 4 bytes in Java)
	<b>long</b>	(e.g., 8 bytes in Java)

Java has a **byte** type (1 byte)

in C/C++, **char** is considered an integer type

```
short shnum = 333;
```

```
long lnum = 888888;
```

```
sh = sh + 1;
```

```
int inum = 31126;
```

## PRIMITIVE TYPES: FLOATING-POINT

Again, languages often provide several sizes/ranges

in C/C++/Java                    **float**     (4 bytes in Java)

**double**   (8 bytes in Java)

C/C++ also have a **long double** type

Historically, floating-points have been stored in a variety of formats.

- same basic components: sign, fraction, exponent

In 1985, IEEE floating-point formats were standardized.

Other number types: decimal, fixed-point, rational, ...

```
float fnum = 333.221;
```

```
double dnum = 888E-7;
```

## PRIMITIVE TYPES: BOOLEAN

Introduced in ALGOL 60

C does have a **boolean** type, conditionals use zero (false) and nonzero (true)

C++ has **bool** type

- really just syntactic sugar, automatic conversion between **int** and **bool**

Java has **boolean** type

- no conversions between **int** and **bool**

Implementing booleans

- could use a sign bit, but not usually accessible
- use smallest easily-addressable unit (e.g., byte)

```
boolean b = False;    boolean b = 0;  
b = !b;               b = 1;
```

## PRIMITIVE TYPES: CHARACTER

Stored as numeric codes, e.g., ASCII (C/C++) or UNICODE (Java)

in C/C++, **char** is an integer type

- can apply integer operations, mix with integer values

```
char ch = 'A';          char ch = '8';  
ch = ch + 1;           int d = ch - '0';
```

in Java, **char** to **int** conversion is automatic

- but must explicitly cast **int** to **char**

```
char next = (char)(ch + 1);
```

### END OF LINE SEQUENCES

Windows end of line sequence: `\r\n`

Unix end of line sequence: `\n`

Mac end of line sequence: `\r`

### ASCII codes and their escape sequences:

ASCII Name	Description	C Escape Sequence
Nul	null byte	<code>\0 (zero)</code>
Bel	bel character	<code>\a</code>
bs	backspace	<code>\b</code>
ht	horizontal tab	<code>\t</code>
np	formfeed	<code>\f</code>
nl	newline	<code>\n</code>
cr	carriage return	<code>\r</code>

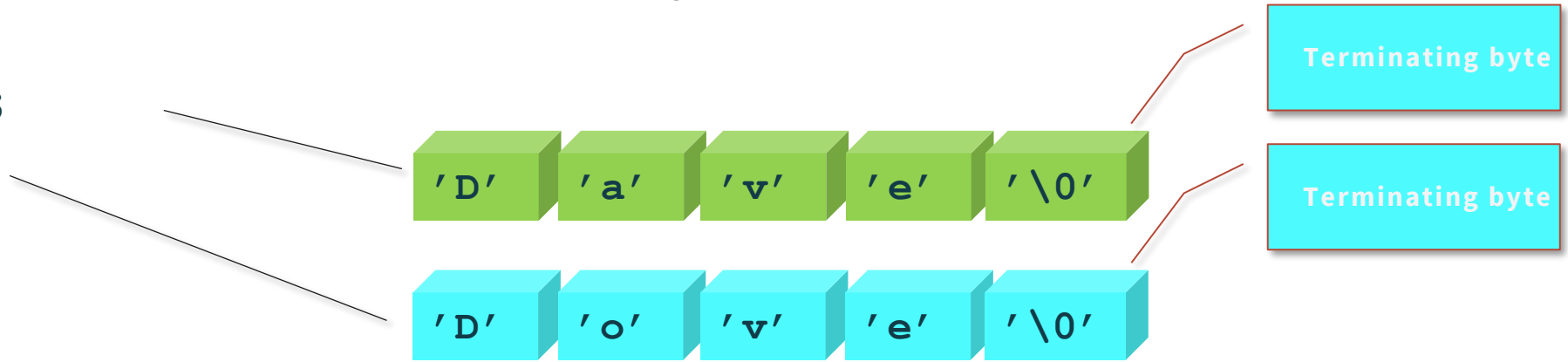


## COMPLEX TYPES: STRINGS

Can be a primitive type or can be a special kind of character array (e.g., Pascal, Ada, C)

- In C++ & Java, OOP can make the string type appear primitive
- Both are classes built on top of '\0'-terminated, C-style strings

```
String str = "Dave";  
str[1] = 'o';
```

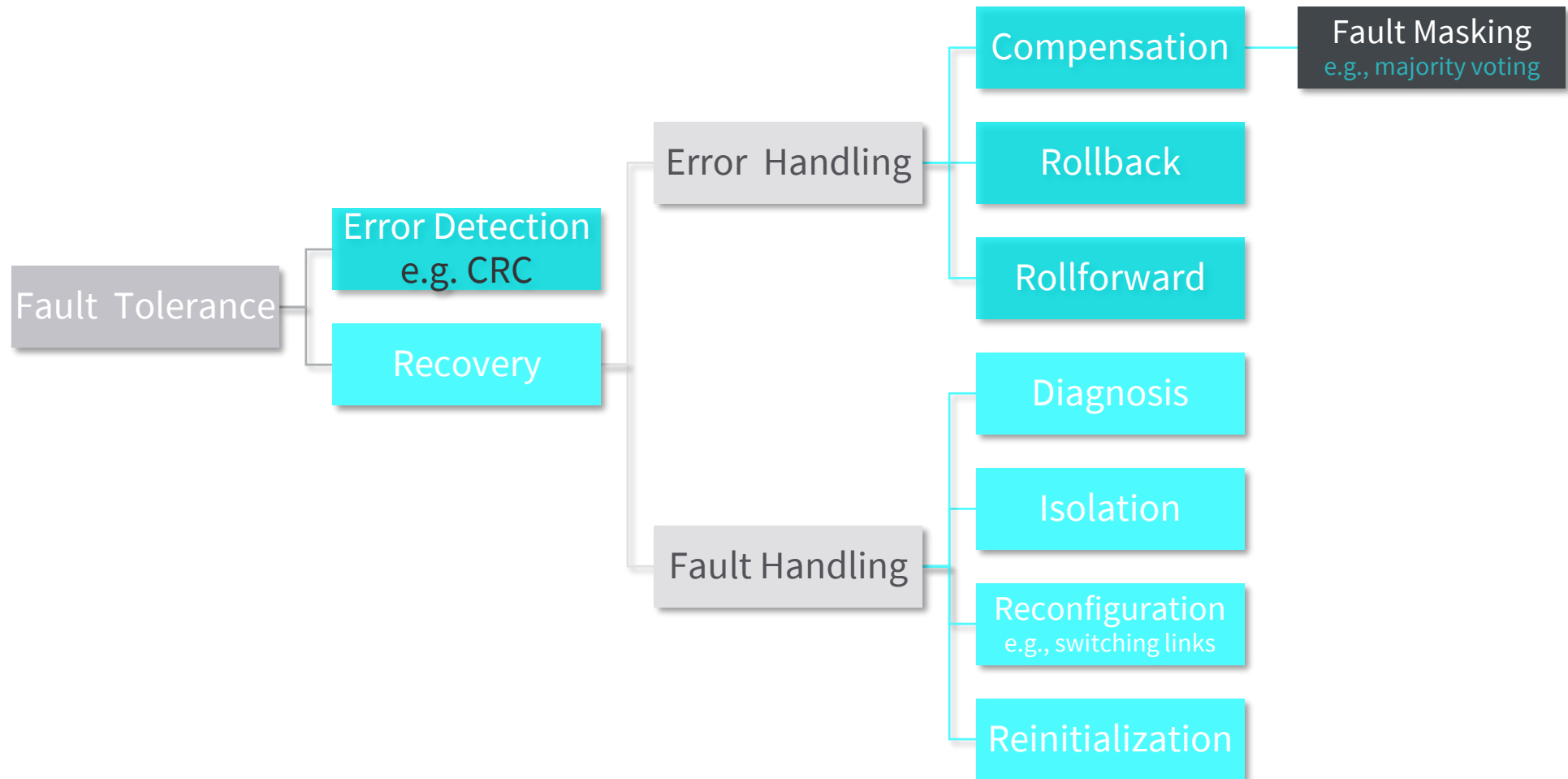


- Java strings are immutable – can't change individual characters, but can reassign an entire new value

```
str = str.substring(0, 1) + "el" + str.substring(2, 5);
```

**Reason: Structure sharing is used to save memory.**

## OVERVIEW OF FAULT TOLERANCE TECHNIQUES



**Redundancy** is a mandatory prerequisite for various fault tolerance mechanisms.

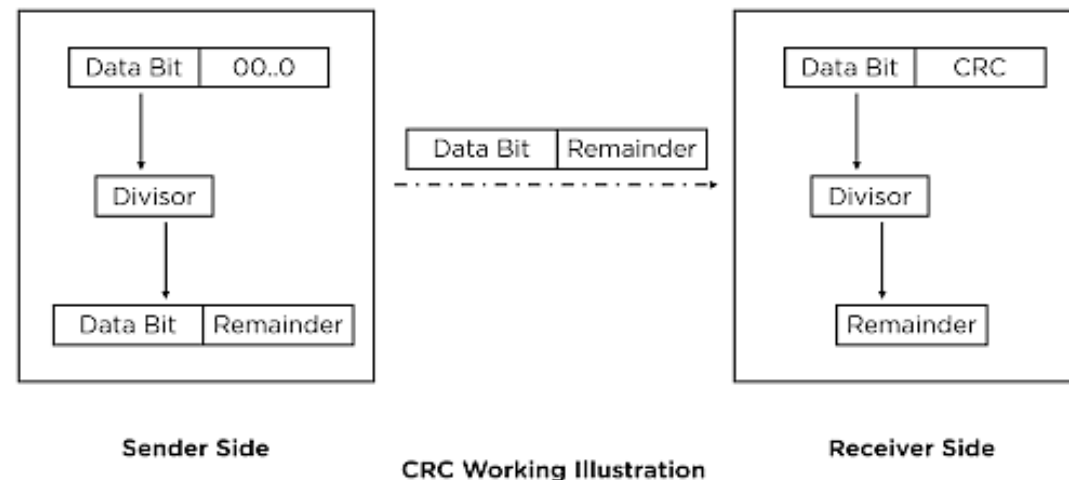
## CYCLIC REDUNDANCY CHECK

### Sender Side (CRC Generator and Modulo Division):

- Add the no. of zeroes to the data, calculated using  $k-1$  ( $k$  is the bits obtained through the polynomial equation)
- Applying the Modulo Binary Division to the data bit obtaining the remainder from the division
- Append the remainder to the end of the data bit and share it with the receiver.

### Receiver Side (Checking for errors in the received data):

- To check the error, perform the Modulo division again and check whether the remainder is 0 or not,
- If the remainder is 0, the data bit received is correct, without any errors.
- If the remainder is not 0, the data received is corrupted during transmission.



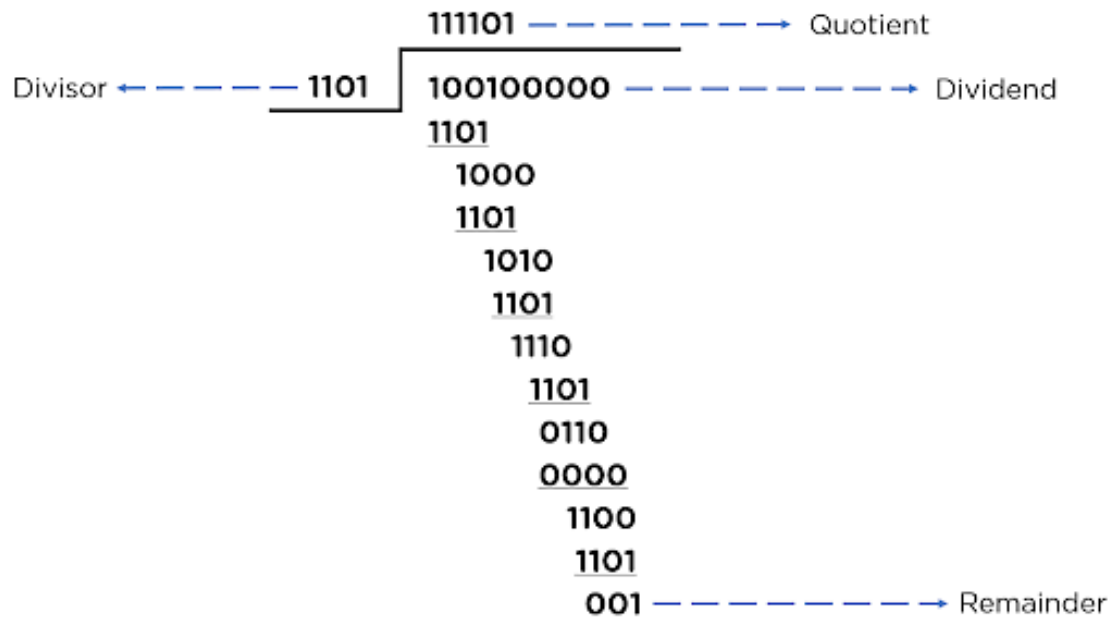
## CYCLIC REDUNDANCY CHECK

### Example of CRC:

Data: 100100

Divisor: 1101 ( $x^3 + x^2 + 1$ )

New data: 100100001

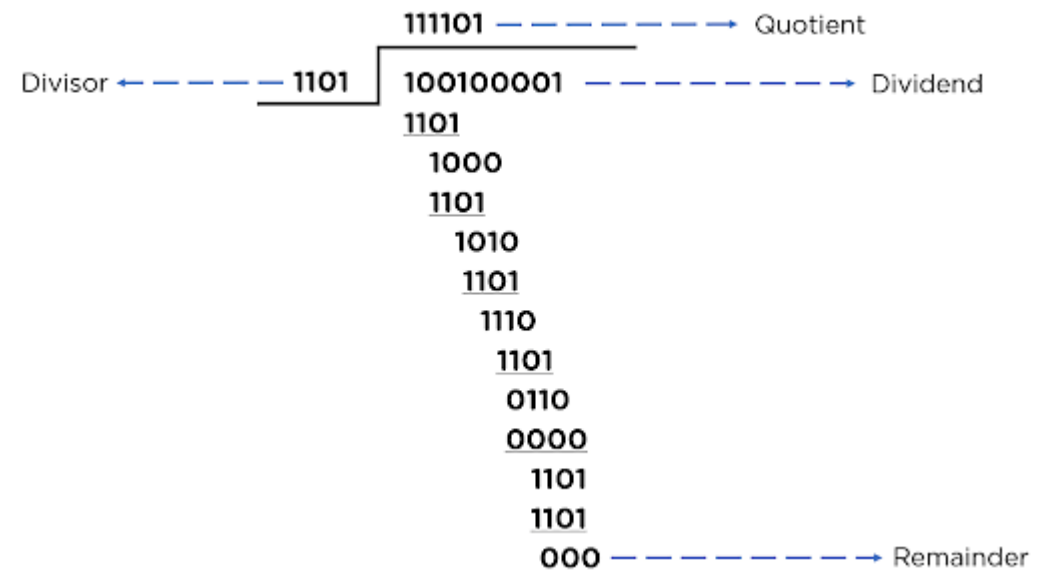


Sender side

Received data: 100100001

Divisor: 1101 ( $x^3 + x^2 + 1$ )

Remainder: 000 (error-free)



Receiver side

## REDUNDANCY CHECK

### The quick **brown** fox

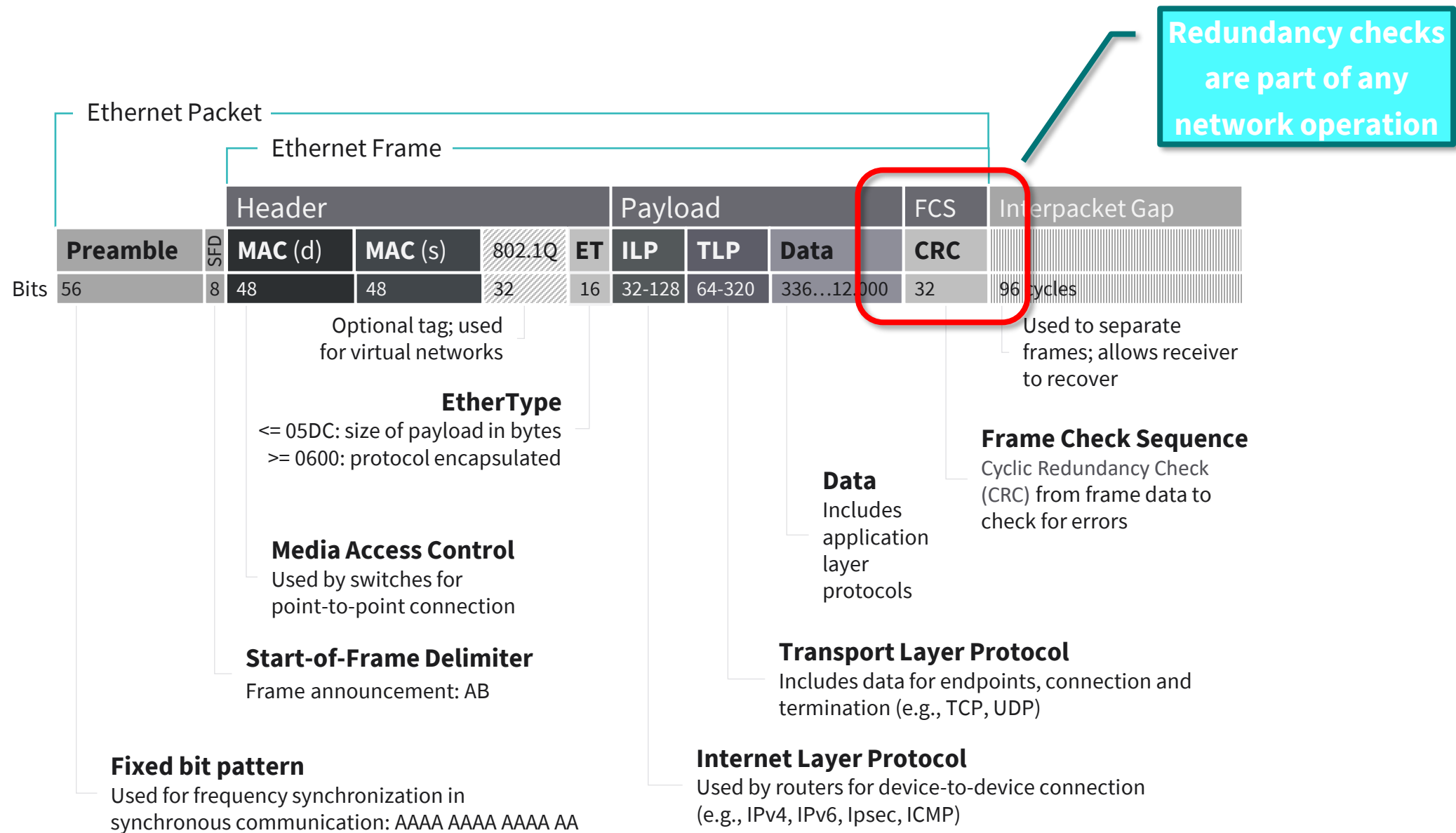
Algorithm	Result	Check	Poly	Init	RefIn	RefOut	XorOut
CRC-8	0x44	0xF4	0x07	0x00	FALSE	FALSE	0x00
CRC-8/CDMA2000	0x33	0xDA	0x9B	0xFF	FALSE	FALSE	0x00
CRC-8/DARC	0xE4	0x15	0x39	0x00	TRUE	TRUE	0x00
CRC-8/DVB-S2	0x22	0xBC	0xD5	0x00	FALSE	FALSE	0x00
CRC-8/EBU	0xA3	0x97	0x1D	0xFF	TRUE	TRUE	0x00
CRC-8/I-CODE	0xEA	0x7E	0x1D	0xFD	FALSE	FALSE	0x00
CRC-8/ITU	0x11	0xA1	0x07	0x00	FALSE	FALSE	0x55
CRC-8/MAXIM	0x95	0xA1	0x31	0x00	TRUE	TRUE	0x00
CRC-8/ROHC	0x1E	0xD0	0x07	0xFF	TRUE	TRUE	0x00
CRC-8/WCDMA	0x65	0x25	0x9B	0x00	TRUE	TRUE	0x00

### The quick **green** fox

Algorithm	Result	Check	Poly	Init	RefIn	RefOut	XorOut
CRC-8	0x41	0xF4	0x07	0x00	FALSE	FALSE	0x00
CRC-8/CDMA2000	0x8E	0xDA	0x9B	0xFF	FALSE	FALSE	0x00
CRC-8/DARC	0x74	0x15	0x39	0x00	TRUE	TRUE	0x00
CRC-8/DVB-S2	0xF3	0xBC	0xD5	0x00	FALSE	FALSE	0x00
CRC-8/EBU	0x2C	0x97	0x1D	0xFF	TRUE	TRUE	0x00
CRC-8/I-CODE	0xF6	0x7E	0x1D	0xFD	FALSE	FALSE	0x00
CRC-8/ITU	0x14	0xA1	0x07	0x00	FALSE	FALSE	0x55
CRC-8/MAXIM	0x2D	0xA1	0x31	0x00	TRUE	TRUE	0x00
CRC-8/ROHC	0x77	0xD0	0x07	0xFF	TRUE	TRUE	0x00
CRC-8/WCDMA	0xAA	0x25	0x9B	0x00	TRUE	TRUE	0x00

CRC calculation: <https://crccalc.com/>

ETHERNET PACKET



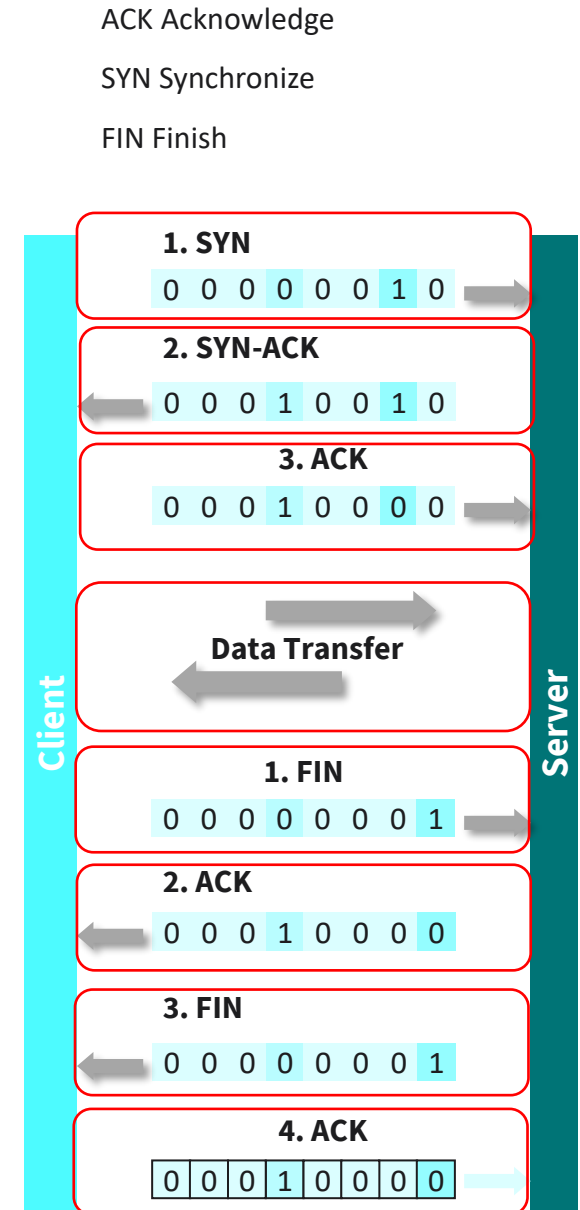
## TCP CONNECTION

### Connection establishment

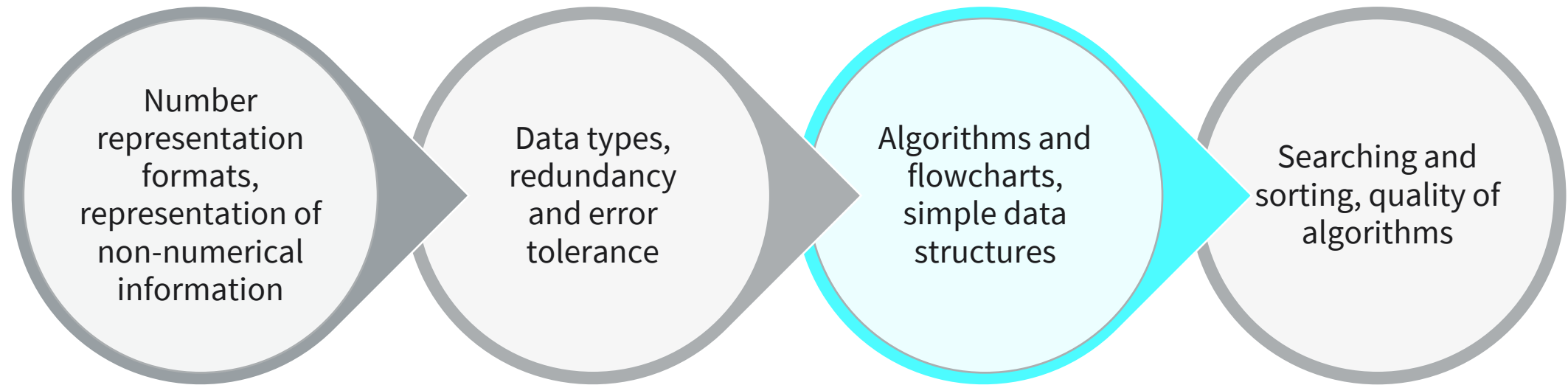
1. Client sends SYN segment (no data)  
SYN=1; SeqNo = initial sequence number (ISN)
2. Server assigns buffer and sends SYN-ACK segment (no Data)  
ACK=1; AckNo = Client-ISN+1; SeqNo = ServerISN
3. Client establishes connection, assigns buffer and sends ACK segment (no data)  
SYN=0; AckNo = Server-ISN+1

### Connection termination

1. Client sends FIN segment
2. Server acknowledges with ACK and waits for FIN
3. Server sends FIN segment and waits for ACK
4. Client sends ACK and empties buffer; server receives ACK and empties buffer



## ALGORITHMS AND DATA STRUCTURES





## THE CONCEPT OF AN ALGORITHM

Many researchers believe that every activity of the human mind is the result of an algorithm.

In mathematics and computer science, an algorithm (*/ˈælgərɪðəm/*) is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation.

Algorithms are always unambiguous and used as specifications for performing calculations, data processing, automated reasoning, and other tasks.

Obtain a basket of unshelled peas and an empty bowl.

As long as there are unshelled peas in the basket continue to execute the following steps:

- a. Take a pea from the basket.
- b. Break open the pea pod.
- c. Dump the peas from the pod into the bowl.
- d. Discard the pod.

## FORMAL DEFINITION OF ALGORITHM

An algorithm is an ordered set of unambiguous, executable steps that defines a terminating process.

The steps of an algorithm can be sequenced in different ways:

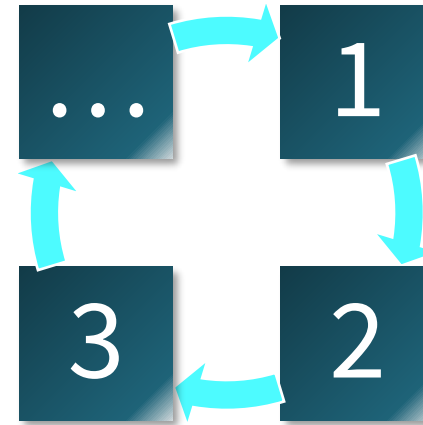
**Linear (1, 2, 3, ...)**



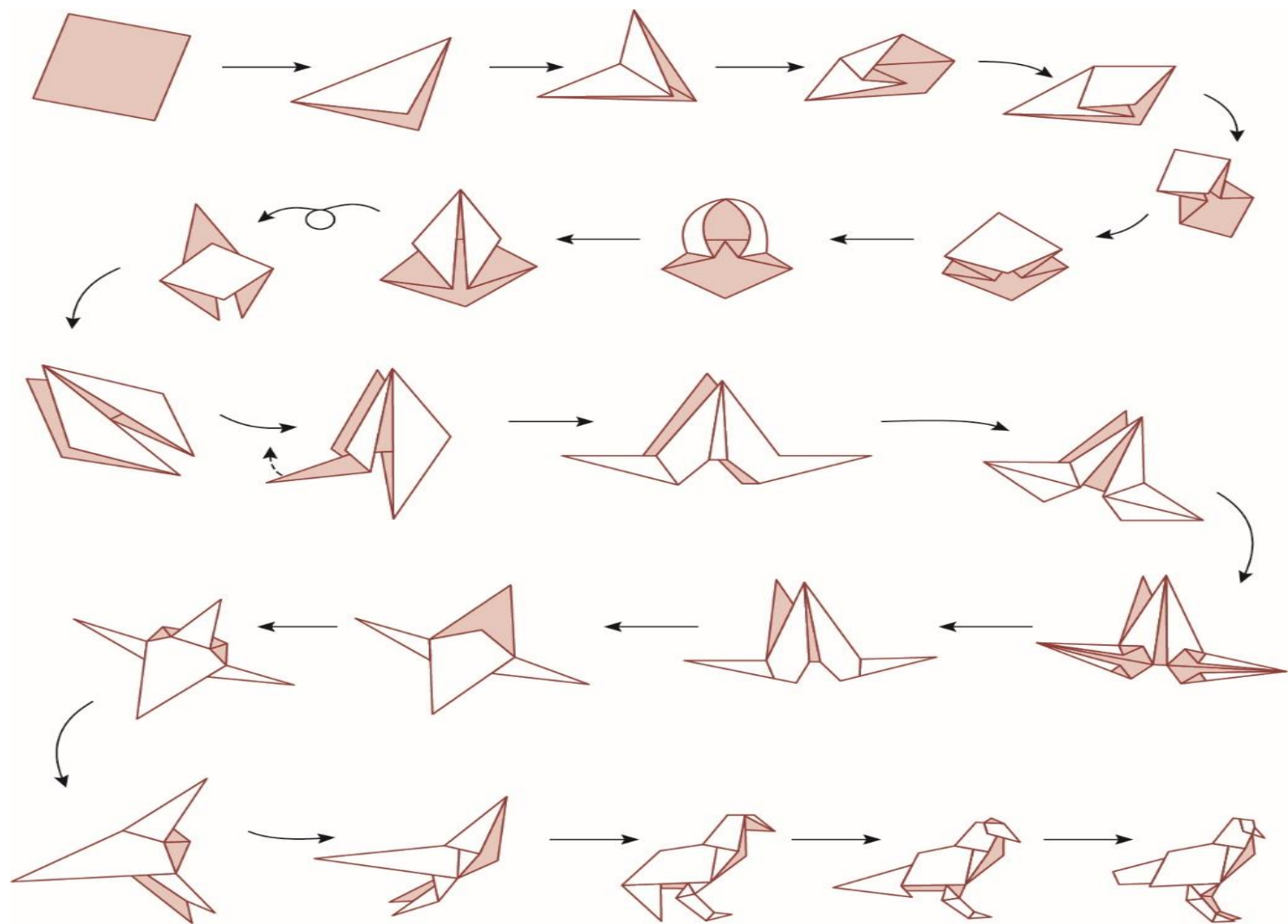
**Parallel (multiple processors)**



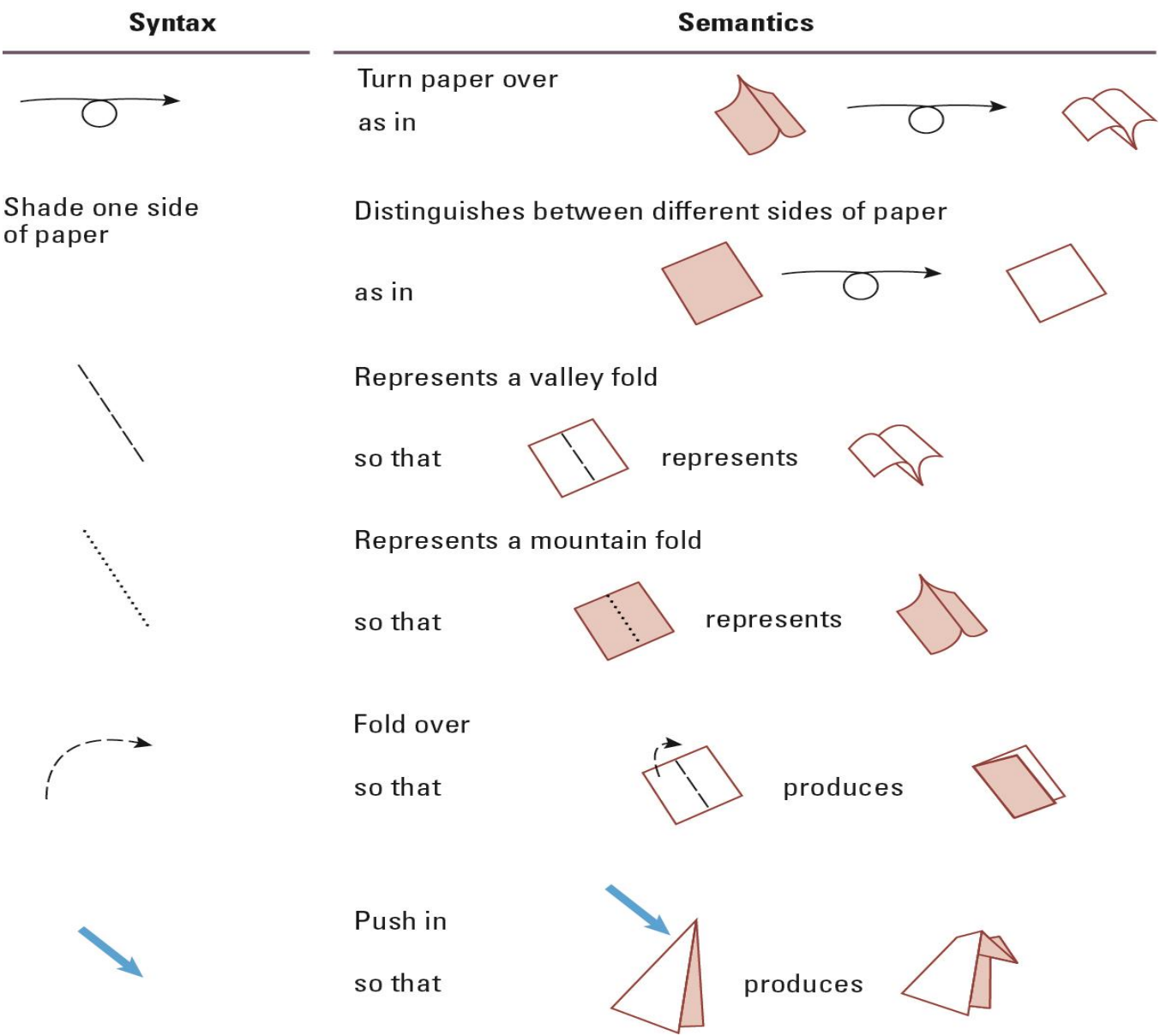
**Cause and Effect (circuits)**



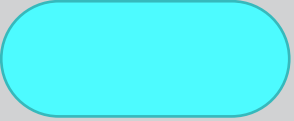

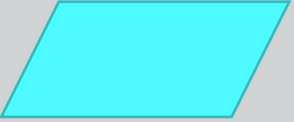
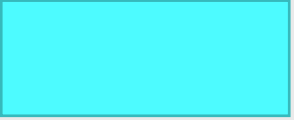
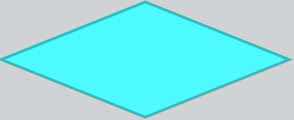

**ALGORITHM: FOLDING A BIRD FROM A SQUARE PIECE OF PAPER**



ORIGAMI PRIMITIVES



## PROGRAMMING FLOWCHARTS

Symbol	Name	Function
	start/end	An oval represents a start or end point.
	arrows	A line is a connector that shows relationships between the representative shapes.
	input or output	A parallelogram represents input or output.
	action or process	A rectangle represents an action or process.
	decision	A diamond indicates a decision.
	subroutine or myBlock	Indicates a sequence of actions that perform a specific task embedded within a larger process. The sequence of actions could be described in more detail on a separate flowchart.

A good flowchart should communicate a process clearly and effectively. When starting out, it's a good idea to focus on a few steps.

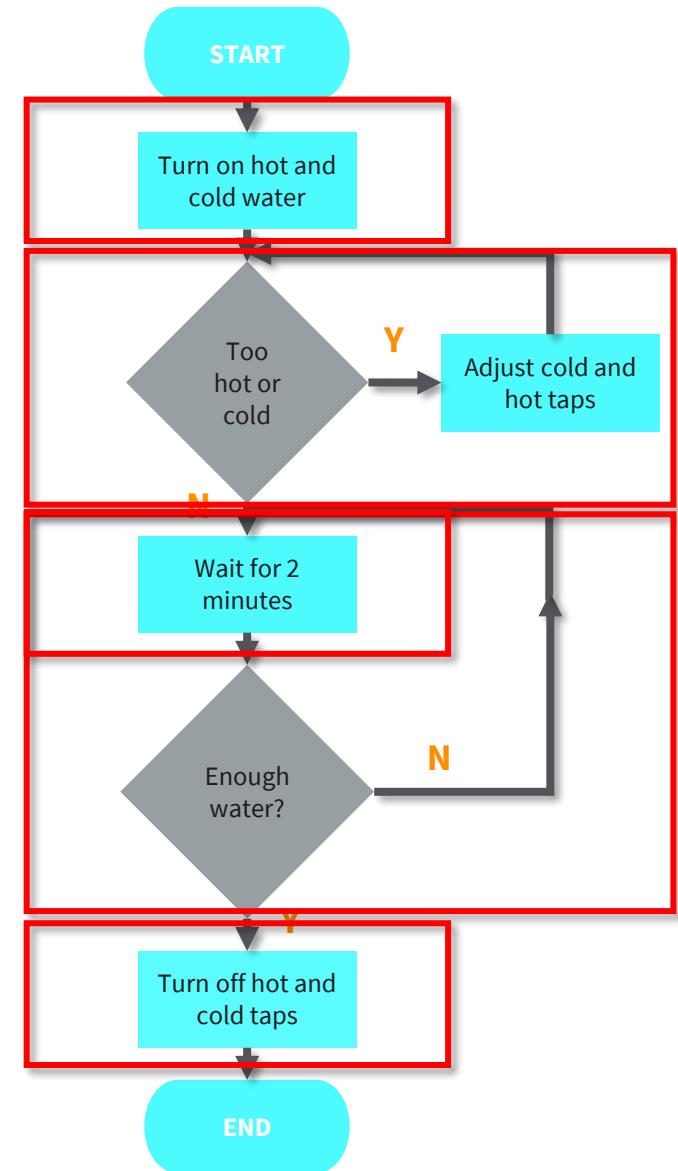
- Determine the purpose or function of the flowchart
- Add steps and connect them with arrows
- Add decisions or split paths
- Show any loops back to previous steps
- Share your flowchart

Most flowcharts should be built using only the Start/End and Action or Process symbols and should follow a very basic set of best practices.

Sticking to these primary flowchart symbols is the best way to ensure that your diagram will be easy to understand.

## ALGORITHM AND FLOWCHART

1. Turn on the hot and cold taps.
2. Is it too hot or cold? If it is, go to step 3, otherwise go to step 4.
3. Adjust the hot and cold taps and go back to step 2.
4. Wait for 2 minutes.
5. Is the bath full? If it is, go to step 7, otherwise go to step 6.
6. Turn off the hot and cold taps.
7. Finish.



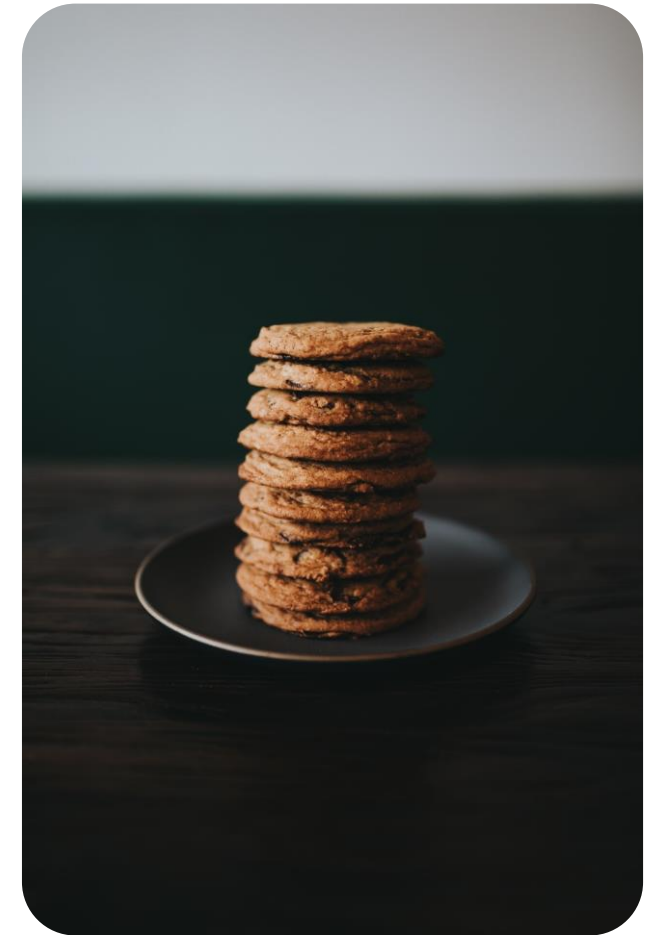
## STACKS AND LISTS

In computer science, a stack is an abstract data type that serves as a collection of elements, with two main principal operations:

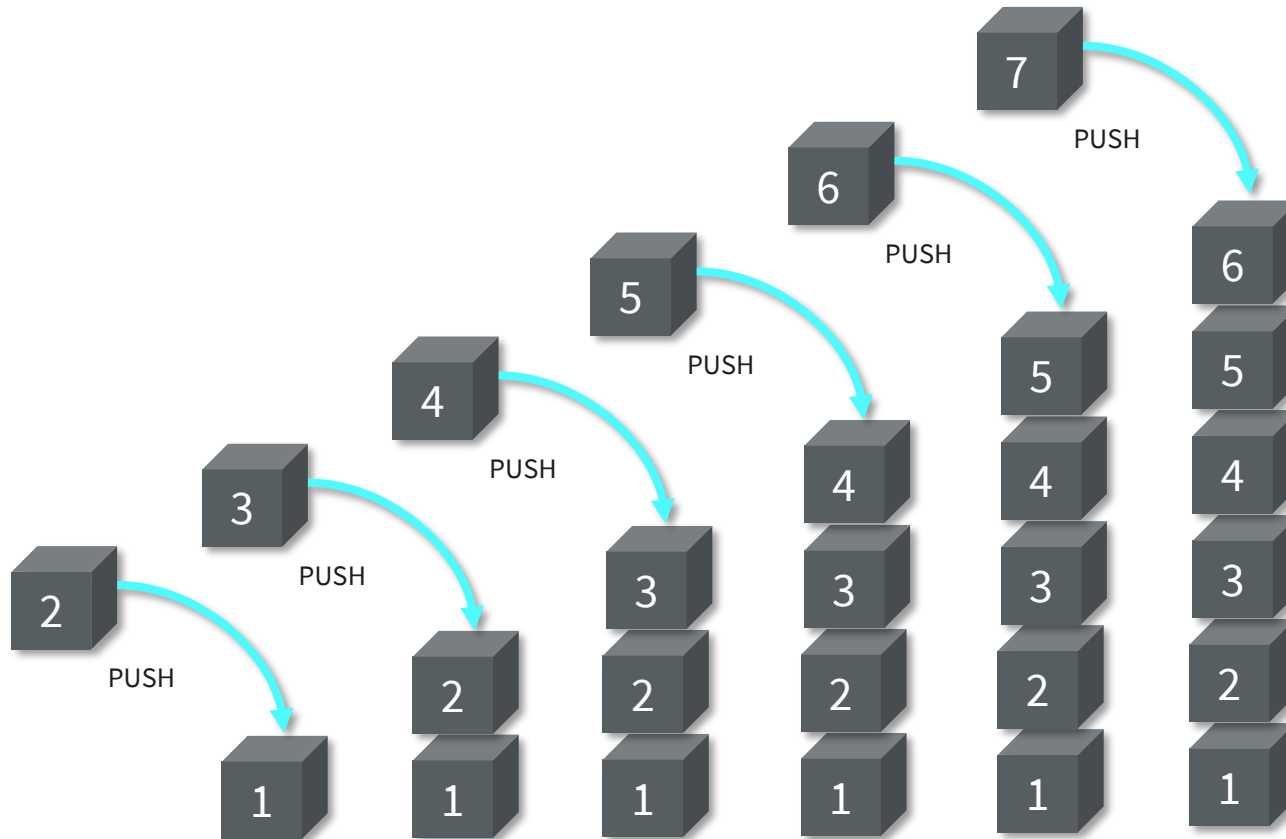
- **push**, which adds an element to the collection, and
- **pop**, which removes the most recently added element that was not yet removed.

A list can be made by using a stack.

Stack: Last in, first out (**LIFO**)



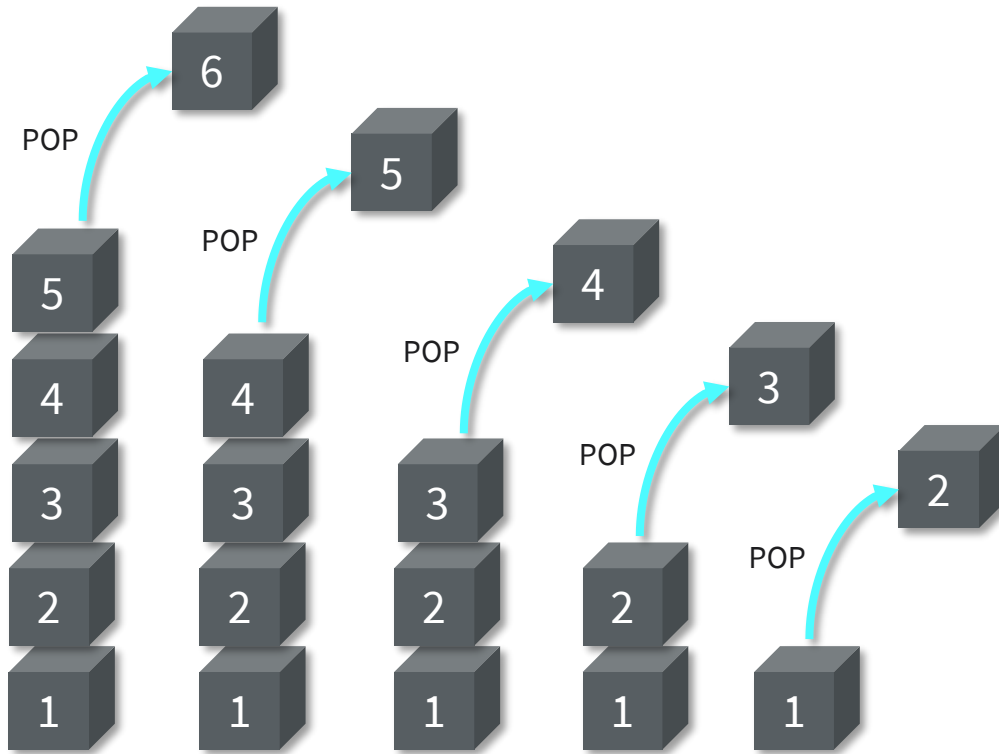
**Push** - adds an element to the collection





## USING STACKS

**Pop** - removes the most recently added element that was not yet removed



## COMPLEX TYPES: ARRAYS

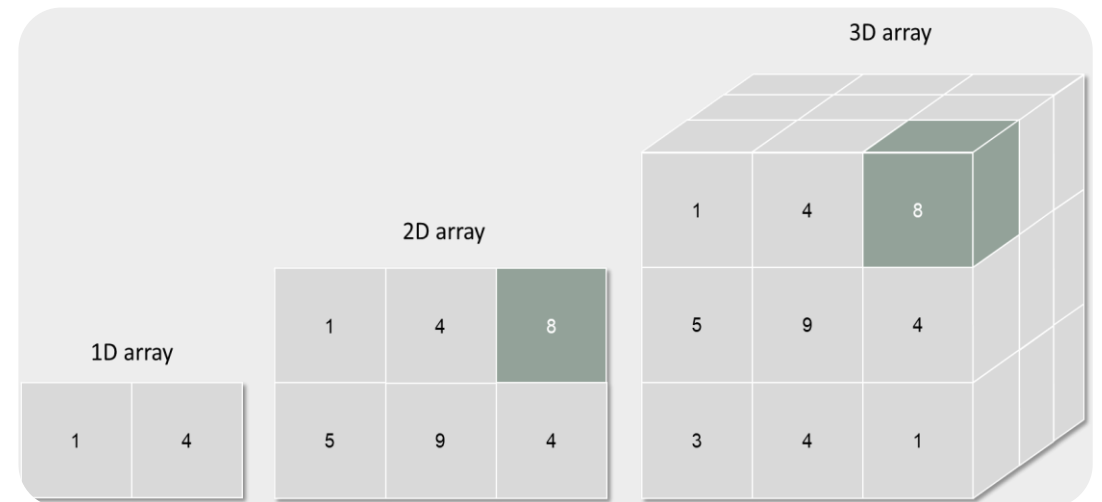
An array is a homogeneous aggregate of data elements that supports random access via indexing

Design issues:

- index type (C/C++ & Java only allow int, others allow any ordinal type)
- index range (C/C++ & Java fix low bound to 0, others allow any range)

Dimensionality:

- C/C++ & Java only allow 1-D, but can have array of arrays

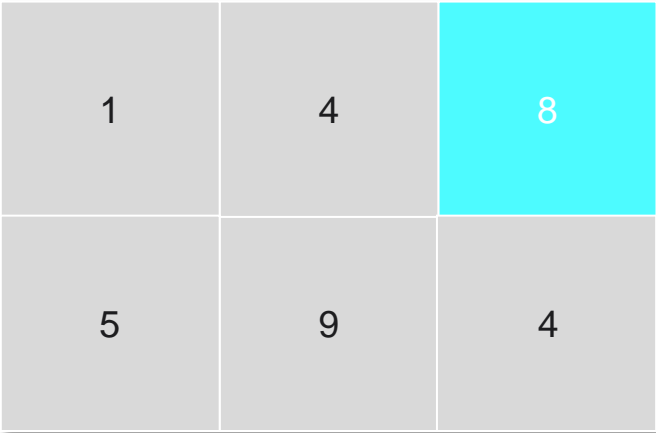


COMPLEX TYPES: ARRAYS

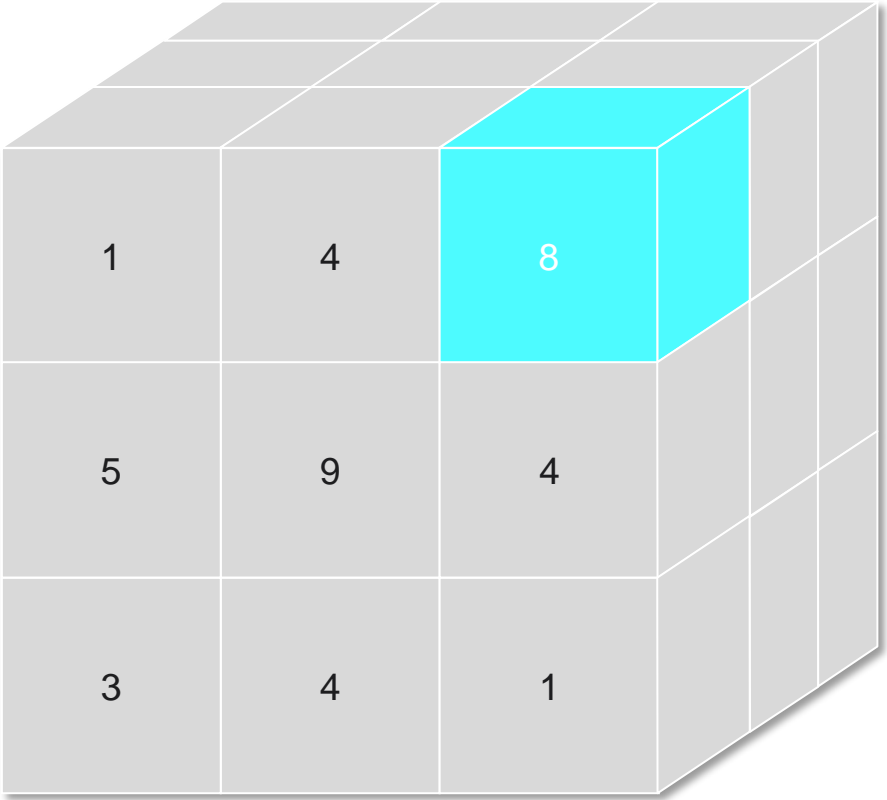
1D array



2D array



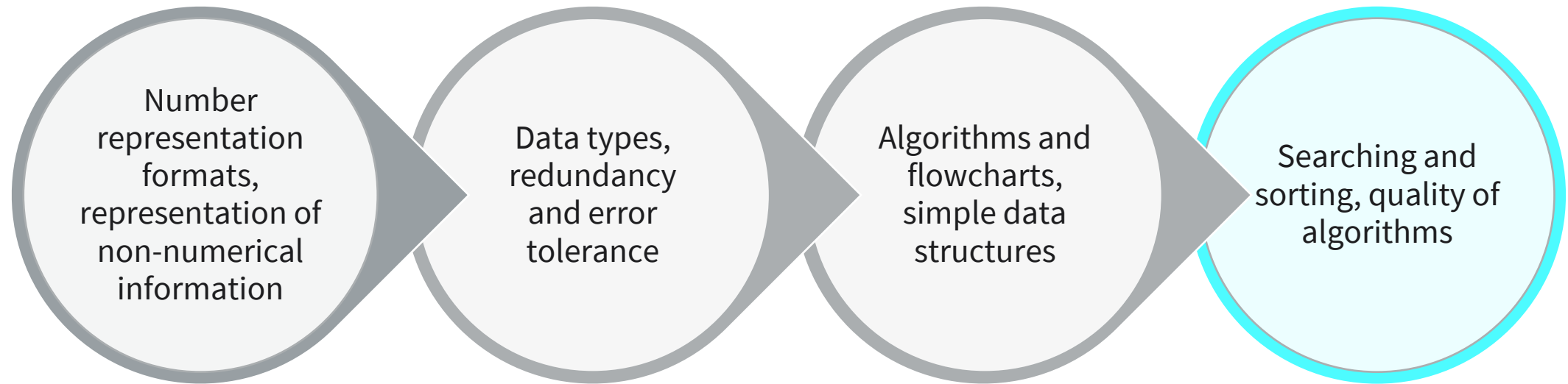
3D array



A string is a 1D array of characters



## ALGORITHMS AND DATA STRUCTURES

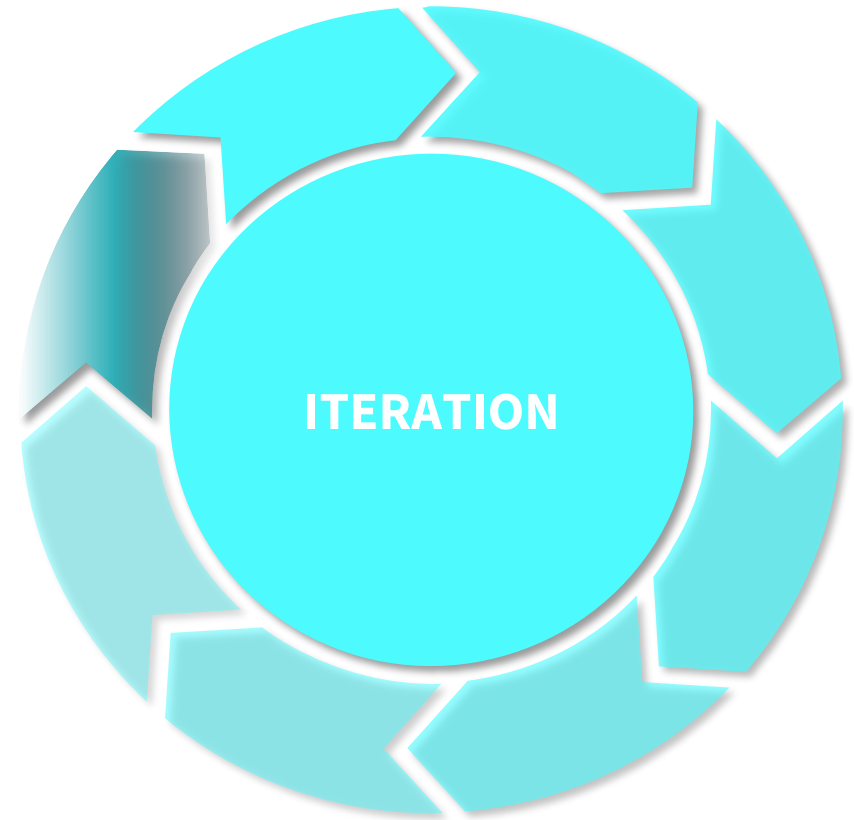


## ITERATIVE STRUCTURES

A collection of instructions repeated in a looping manner

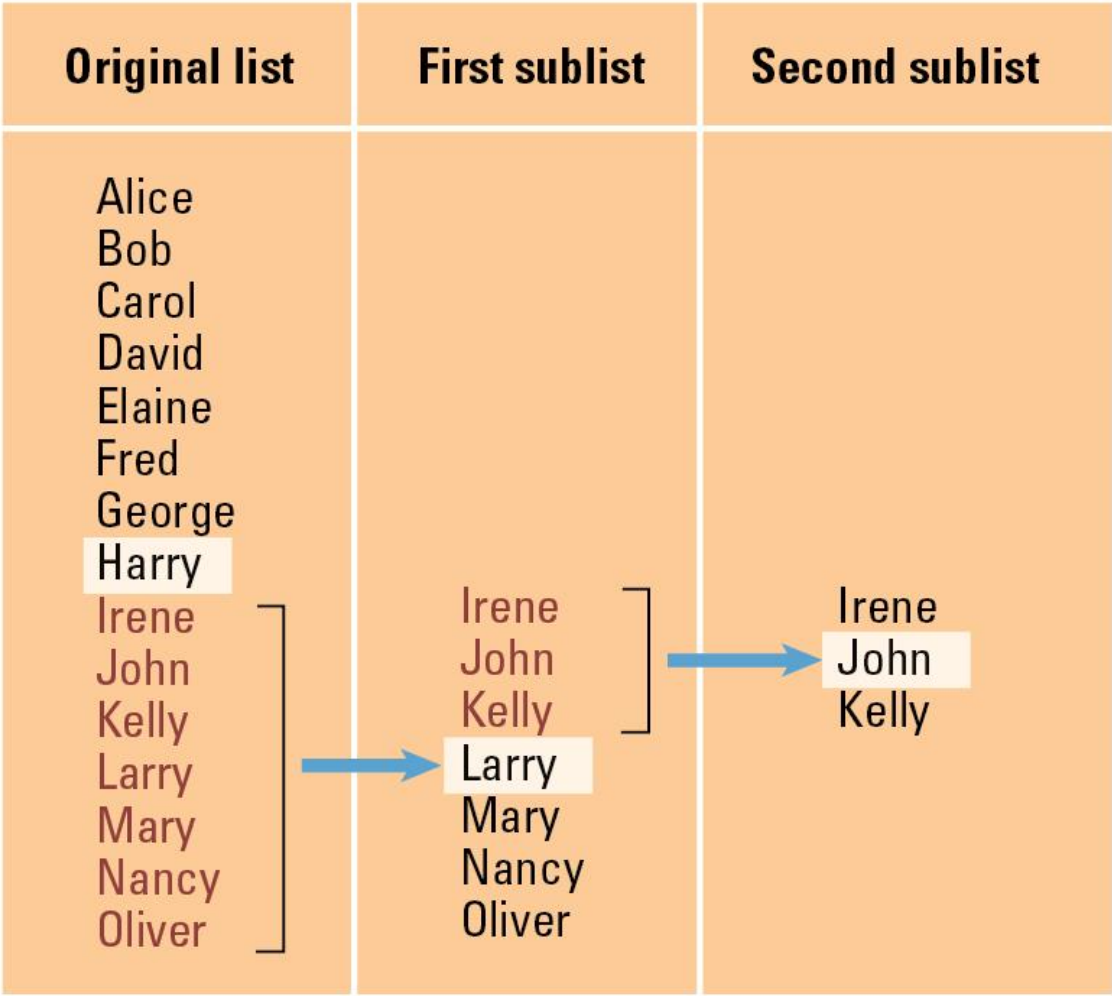
Examples include:

- **Sequential Search Algorithm**
- **Insertion Sort Algorithm**



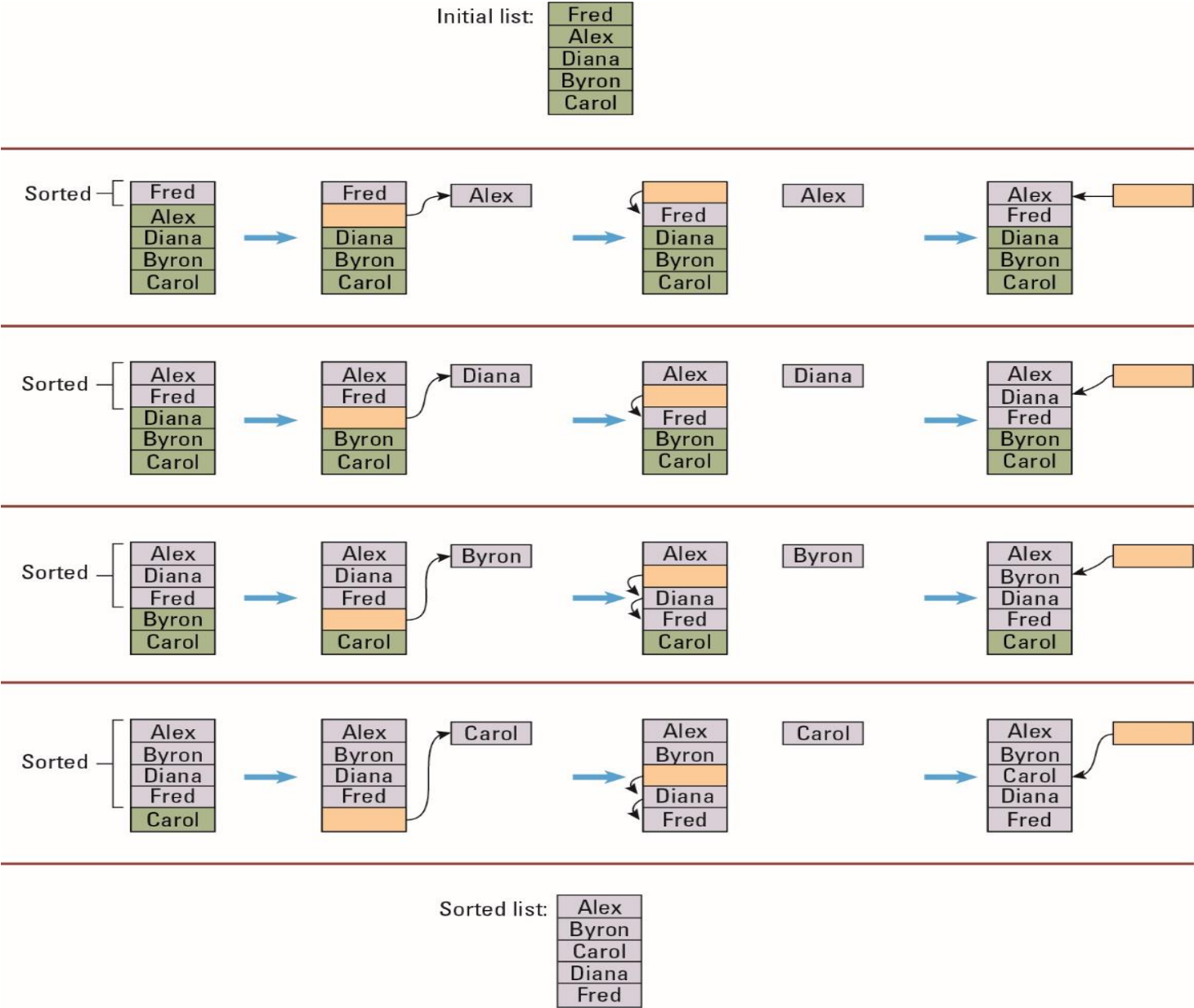
**BINARY SEARCHING**

Applying binary search strategy to search a list for the entry John



INSERTION SORT

Sorting the list Fred, Alex, Diana, Byron, and Carol alphabetically



## EFFICIENCY AND CORRECTNESS

The choice between **efficient and inefficient** algorithms can make the difference between a practical solution and an impractical one.

The **correctness of an algorithm** is determined by reasoning formally about the algorithm, not by testing its implementation.





## EFFICIENCY BIG O NOTATION (BIG THETA $\Theta$ NOTATION)

- measured as number of instructions executed
- example: insertion sort is in  $\Theta(n^2)$
- incorporates best, worst, and average case analysis

Comparisons made for each pivot					Sorted list
Initial list	1st pivot	2nd pivot	3rd pivot	4th pivot	
Elaine David Carol Barbara Alfred	1 Elaine David Carol Barbara Alfred	3 David Elaine 2 Carol Barbara Alfred	6 Carol David 5 Elaine 4 Barbara Alfred	10 Barbara Carol 9 David 8 Elaine 7 Alfred	Alfred Barbara Carol David Elaine

Applying the insertion sort in a worst-case situation

Big O Notation: The Big  $\Theta$  notation defines an upper bound of an algorithm, it bounds a function only from above.

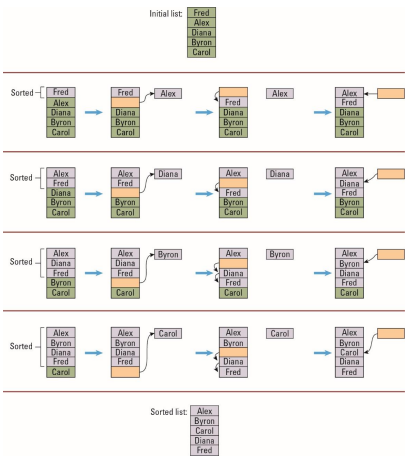
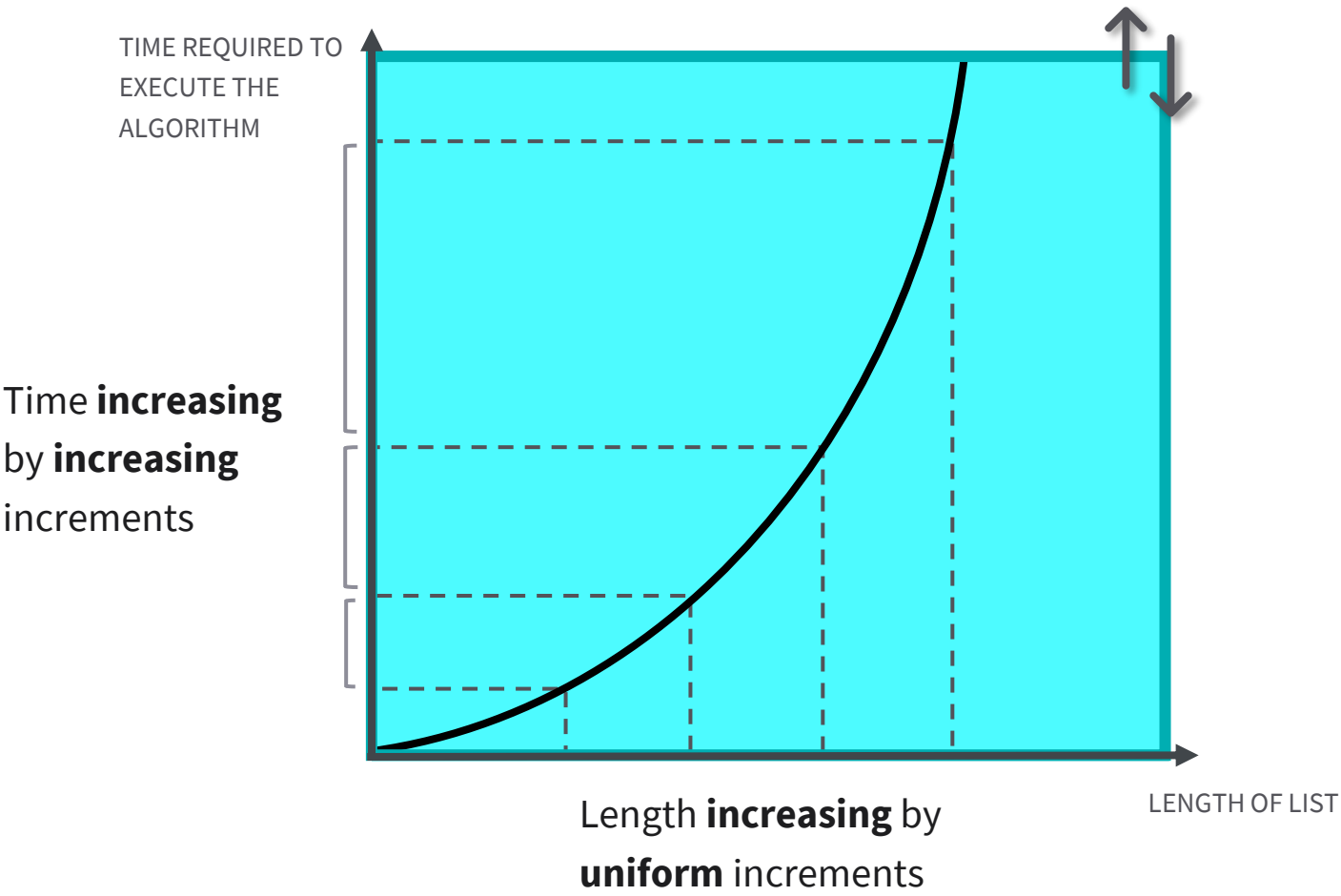
For example, consider the case of Insertion Sort. It takes linear time in best case and quadratic time in worst case. We can safely say that the time complexity of Insertion sort is  $O(n^2)$ . Note that  $\Theta(n^2)$  also covers linear time.

If we use  $\Theta$  notation to represent time complexity of Insertion sort, we have to use two statements for best and worst cases:

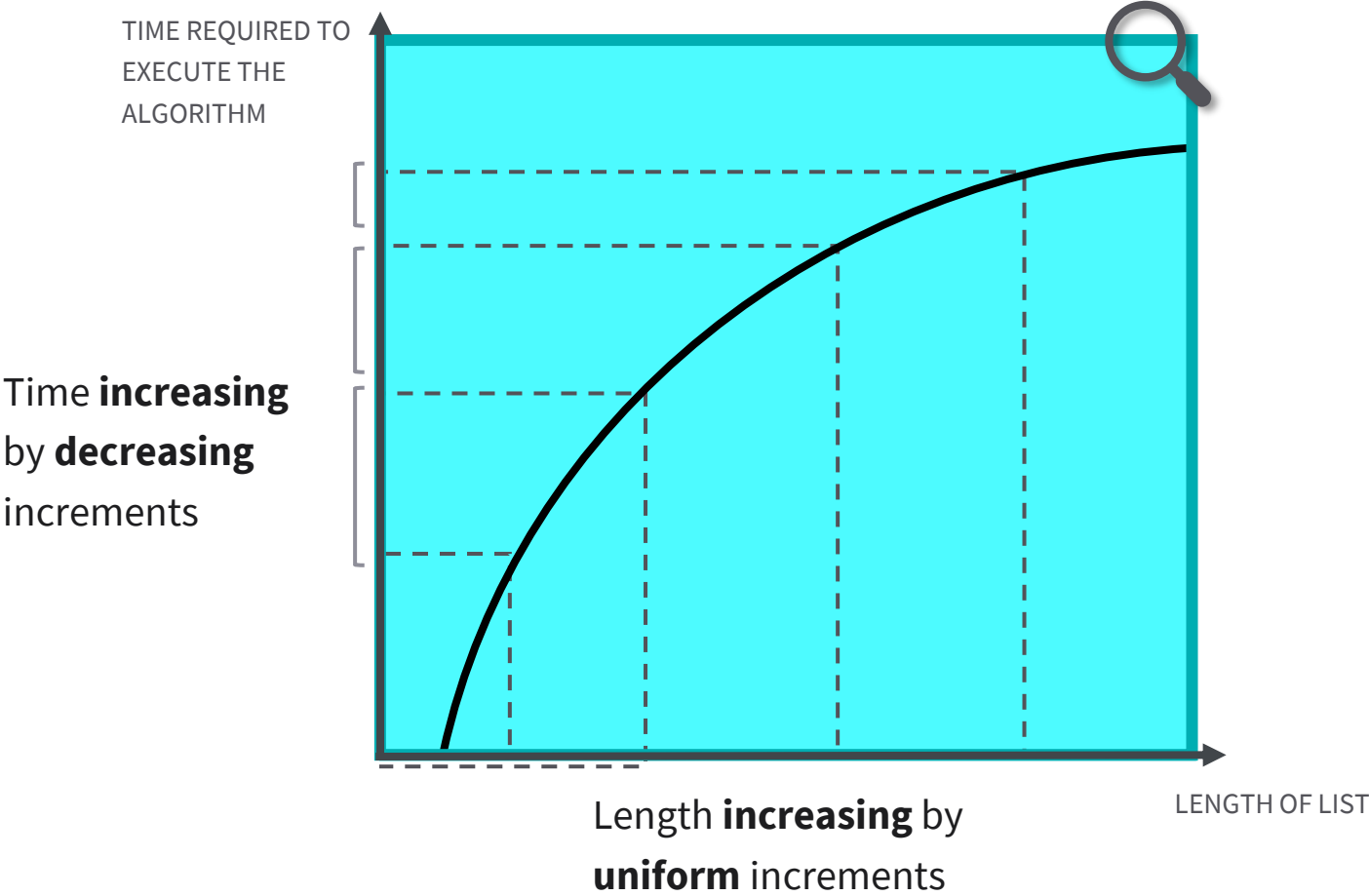
1. The worst case time complexity of Insertion Sort is  $\Theta(n^2)$ .
2. The best case time complexity of Insertion Sort is  $\Theta(n)$ .

The Big O notation is useful when we only have upper bound on time complexity of an algorithm. Many times, we easily find an upper bound by simply looking at the algorithm.

GRAPH OF THE WORST-CASE ANALYSIS OF THE INSERTION SORT ALGORITHM



GRAPH OF THE WORST-CASE ANALYSIS OF THE BINARY SEARCH ALGORITHM



Original list	First sublist	Second sublist
Alice Bob Carol David Elaine Fred George Harry Irene John Kelly Larry Mary Nancy Oliver	Irene John Kelly Larry Mary Nancy Oliver	Irene John Kelly

## REVIEW STUDY GOALS



- Understand the binary numbering system and how to convert decimal to binary.
- Learn about data storage sizes, such as kilobyte, megabyte, and terabyte.
- Master how graphics, documents, music, and other data are represented in memory.
- Understand about the different data types used in programming such as char, string, int, and float.
- Explore how computers perform error checking on stored and network data.

## REVIEW STUDY GOALS



- Learn how the term algorithm is used in applications of computer science.
- Understand how to create flowcharts.
- Explore the details of data structures, such as arrays and stacks.
- Learn how sorting algorithms work, including quick and bubble sort.
- Master how the quality of different algorithms is evaluated.

Session 2

# TRANSFER TASK



## Ages of Children Problem – Application of an Algorithm

Person A is charged with the task of determining the ages of B's three children.

**B tells A that the product of the children's ages is 36.**

*A replies that another clue is required, because there are too many choices.*

**B tells A the sum of the children's ages.**

*A replies that another clue is needed, because there are still too many choices.*

**B tells A that the oldest child plays the piano.**

*A is now able to tell B the ages of the three children.*

How old are the three children?

**TRANSFER TASK**  
**PRESENTATION OF THE RESULTS**

Please present your  
results.

The results will be  
discussed in  
plenary.





## SAMPLE SOLUTION



The first clue given A is that the product of the children's ages is 36. This means that the triple representing the three ages is one of those listed in Figure a.

The next clue is the sum of the desired triple. We are not told what this sum is, but we are told that this information is not enough for A to isolate the correct triple; therefore, the desired triple must be one whose sum appears at least twice in the table of Figure b. But the only triples appearing in Figure b with identical sums are (1,6,6) and (2,2,9), both of which produce the sum 13.

This is the information available to A at the time the last clue is given. It is at this point that we finally understand the significance of the last clue. It has nothing to do with playing the piano; rather it is the fact that there is an oldest child. This rules out the triple (1,6,6) and thus allows us to conclude that the children's ages are 2, 2, and 9.

### a. Triples whose product is 36

(1, 1, 36)	(1,6,6)
(1, 2, 18)	(2,2,9)
(1, 3, 12)	(2,3,6)
(1, 4, 9)	(3,3,4)

### b. Sums of triple from part (a)

$1 + 1 + 36 = 38$
$1 + 2 + 18 = 21$
$1 + 3 + 12 = 16$
$1 + 4 + 9 = 14$

$1 + 6 + 6 = 13$
$2 + 2 + 9 = 13$
$2 + 3 + 6 = 11$
$3 + 3 + 4 = 10$



1. Which of these is largest in terms of data size?
- a) 2000GB
  - b) 1TB
  - c) 50KB
  - d) 900MB



2. What is the binary number 11111111 in decimal?

- a) 128
- b) 256
- c) 127
- d) 255



3. What type of error checking attaches a fixed-length code to the end of a data file?
- a) checksum
  - b) parity
  - c) hexadecimal
  - d) ACK



4. The “terminator” symbol in a flowchart is used for the end of the process flow and also:
- a) decisions
  - b) the beginning
  - c) data entry
  - d) processing



5. Which of these types of algorithms is generally the most efficient?

- a) quadratic
- b) constant
- c) exponential
- d) cubic

## LIST OF SOURCES

Brookshear, G. & Bylow, D. (2011). *Computer science: An overview* (11th ed.). Pearson.

Dale, N. & Lewis, J. (2020). *Computer science illuminated* (7th ed.). Jones & Bartlett Learning.

Downey, A. B. & Mayfield, C. (2020). *Think Java: How to think like a computer scientist*. O'Reilly.

Filho, W. F. (2018). *Computer science distilled: Learn the art of solving computational problems*. Code Energy LLC.

Petzold, C. (2000). *Code: The hidden language of computer hardware and software*. Microsoft Press.

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9 (1), 36-45.

<https://doi.org/10.1145/365153.365168>

Whittington, J. (2016). *A machine made this book: Ten sketches of computer science*. Coherent Press.

## **ADDITIONAL MATERIALS**

<https://github.com/tailequy/IU-IntroductionComputerScience>

Tai.le-quy@iu.org