Recursion

***ผู้เขียน**: ทศพร แสงจ้า*

แนวคิด

- Recursion คือกระบวนการเรียกตัวเอง
- ตามปกติแล้วแบ่งกรณีเป็น
  - base case
  - recursion case

  โดยที่จะย่อยปัญหาให้เล็กลงจนกระทั่งแก้ได้ด้วย base case

การเขียนโปรแกรม

จะเขียนกระบวนการนั้นด้วยฟังก์ชัน และการเรียกตัวเองจะเป็นการเรียกฟังก์ชันที่เขียนขึ้นมาภายในฟังก์ชันนั้น

ตัวอย่าง การ เขียน recursive function ใน ภาษา C++ โดยยังไม่คำนึงถึงการนำไปใช้แก้โจทย์ปัญหา

```cpp
void recursive_function() {
    if (base condition)
        return; // base case
    else
        recursive_function(); // recursion case
}
```

วิเคราะห์ประสิทธิภาพ

จะใช้วิธีการนับจำนวนครั้งที่จะมีการเรียกฟังก์ชันเกิดขึ้นตามพารามิเตอร์ของฟังก์ชันนั้น

โจทย์ตัวอย่าง - Factorial

เนื้อหาโจทย์

จงหาค่าของ $n!$ โดย $n! = n \times (n-1) \times (n-2) \times \cdots \times 1$

แนวคิด

โจทย์นี้สามารถใช้โครงสร้าง loop ในการแก้ปัญหาได้ แต่นำมาเป็นตัวอย่างเริ่มต้นในการใช้โครงสร้างของ recursion ในการแก้ปัญหา

โดยมองค่าของ $n!$ เป็น recursive function ดังนี้

$$f(n) = \begin{cases} 1 & \text{if } n \leq 1, \\ n \times f(n-1) & \text{else} \end{cases} \tag{1}$$

รหัสเทียม

```cpp
#include <bits/stdc++.h>

using namespace std;

int factorial(int n) {
  if (n <= 1) { // base case
    return n;
  } else {
    return n * factorial(n - 1);
  }
}

int main() {
```

```
  cout << factorial(4);
  return 0;
}
```

จะได้ output

```
24
```

โจทย์ตัวอย่าง - Fibonacci

เนื้อหาโจทย์

จงหาค่าของเลขฟิโบนัชชีลำดับที่ n โดยที่ เลขฟิโบนัชชีลำดับที่ n จะมีค่าเท่ากับผลบวกของเลขฟิโบนัชชีลำดับที่ $n-1$ และ $n-2$

แนวคิด

โจทย์นี้สามารถใช้โครงสร้าง loop ในการแก้ปัญหาได้เช่นกัน แต่นำมาเป็นอีกตัวอย่างหนึ่งในการใช้โครงสร้างของ recursion ในการแก้ปัญหา

โดยมองค่าของเลขฟิโบนัชชีลำดับที่ n เป็น recursive function ดังนี้

$$f(n) = \begin{cases} n & \text{if } n \leq 1, \\ f(n-1) + f(n-2) & \text{else} \end{cases} \tag{2}$$

รหัสเทียม

```
#include <bits/stdc++.h>

using namespace std;

int fibo(int n) {
  if (n <= 1) { // base case
    return 1;
  } else {
    return fibo(n - 1) + fibo(n - 2);
  }
}
```

```

int main() {
  cout << fibo(4);
  return 0;
}
```

จะได้ output

```
3
```

โจทย์ตัวอย่าง - Permutation

เนื้อหาโจทย์

จงหาลำดับที่มีสมาชิก n ตัว โดยที่แต่ละสมาชิกมีค่าน้อยกว่า r ทั้งหมดที่เป็นไปได้ (จำเป็นต้องใช้ array)

แนวคิด

เป็นการค้นทุกวิธีที่เป็นไปได้ (complete search) ซึ่งจะนำ recursive function เข้ามาใช้ โดย

เริ่มต้นด้วยการสร้างลำดับที่ยังไม่มีสมาชิกใด ๆ และค่อย ๆ เติมสมาชิกเข้าไปในลำดับนั้น

base case จะเป็นเงื่อนไขเมื่อลำดับมีสมาชิกครบ n

recursive case จะเป็นการเติมสมาชิกลำดับถัดไปด้วยทุกค่าที่เป็นไปได้

รหัสเทียม

```
#include <bits/stdc++.h>

using namespace std;

int a[10];

void perm(int d, int r, int n) {
```

```
  if (d == n) { // base case
    for (int i = 0; i < n; i++) {
      cout << a[i] << " ";
    }
    cout << "\n";
    return;
  }
  for (int i = 0; i < r; i++) {
    a[d] = i;
    perm(d + 1, r, n); // recursive case
  }
}

int main() { perm(0, 3, 3); }
```

จะได้ output

```
0 0 0
0 0 1
0 0 2
0 1 0
0 1 1
0 1 2
0 2 0
0 2 1
0 2 2
1 0 0
1 0 1
1 0 2
1 1 0
1 1 1
1 1 2
1 2 0
1 2 1
1 2 2
2 0 0
2 0 1
2 0 2
2 1 0
2 1 1
2 1 2
2 2 0
2 2 1
2 2 2
```

ปัญหาที่พบบ่อย

Overflow

กรณีไม่มี base case หรือ base case ไม่ครอบคลุม จะทำให้เกิดการ overflow ได้

```cpp
#include <bits/stdc++.h>

using namespace std;

int factorial(int n) { return n * factorial(n - 1); }

int main() {
  cout << factorial(4);
  return 0;
}
```

จะเกิด error (ซึ่งอาจจะแตกต่างกันไปขึ้นอยู่กับระบบปฏิบัติการและคอมไพเลอร์) เช่น

```
"./a.out" terminated by signal SIGSEGV (Address boundary error)
```

โจทย์แนะนำ

1. จงหา permutation โดยแต่ละข้อมีเงื่อนไขแตกต่างกันดังต่อไปนี้
   (a) จำนวน 5 หลัก และแต่ละหลักมีค่าไม่เกิน 6
   (b) ตัวเลขหลักก่อนหน้าต้องมีค่าน้อยกว่าหรือเท่ากับตัวเลขหลักถนัดไปเสมอ
   (c) ตัวเลขห้ามซ้ำ
2. จงกลับข้อความ (reverse string) โดยห้ามใช้ loop, array หรือ string และมีเงื่อนไขเพิ่มเติมแตกต่างกันตามแต่ละข้อต่อไปนี้
   (a) กลับทั้งข้อความ
   ตัวอย่างข้อมูลนำเข้า

   ```
   iamastring
   ```

ตัวอย่างข้อมูลส่งออก

```
gnirtsamai
```

(b) กลับเฉพาะภายในคำที่คั่นด้วยเว้นวรรค

ตัวอย่างข้อมูลนำเข้า

```
i am a string
```

ตัวอย่างข้อมูลส่งออก

```
i ma a gnirts
```

3. Programming.in.th

   (a) 0019 Perket

   (b) 0039 Food

4. cses.fi

   (a) 2205 Gray Code

   (b) 2165 Tower of Hanoi

   (c) 1622 Creating Strings

   (d) 1623 Apple Division