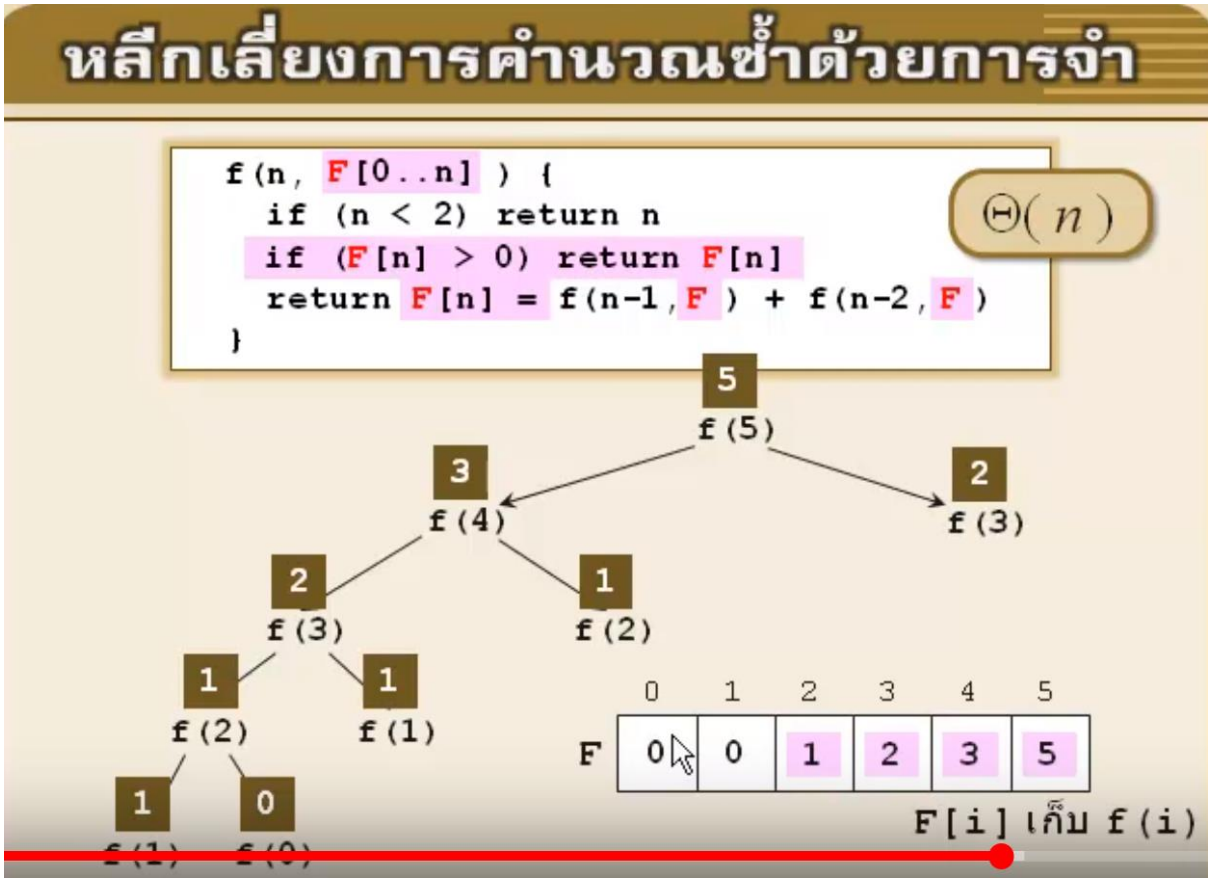
Dynamic programming
กำหนดการพลวัต

Dynamic programming หรือ การกำหนดการเชิงพลวัต (=-=) เป็นรูปแบบของการเขียนโปรแกรมแบบหนึ่งที่จุดสำคัญจะอยู่ที่การตัดส่วนที่ซ้ำซ้อน (overlap) ออกไป (ว้าววว) ทำให้ปะสิทธิภาพของโปรแกรมดีขึ้นแบบทันตาเห็น (ว่ะ ฮ่าๆๆๆ)

ลักษณะของการเขียนมันจะดูๆคล้ายกับการตัดสิ่งที่ไม่จำเป็นต้องคิดออก หรือไม่ก้รวมสิ่งที่คิดเหมือนๆกันเข้าไว้ด้วยกันอ่ะคับ พูดแบบนี้อาจจะไม่เข้าใจ =w= ลองดูตัวอย่างเลยดีกว่า



วางโดมิโน

กำหนดตารางขนาด 2 x n มาให้ เราจะสามารถวาง โดมิโน ขนาด 1x 2 หรือ 2 x 1 ได้กี่วิธี (n < 10^6)

สมมติว่า n = 3 เราก็เลือกวางได้ 3 วิธีคือ =| , |= , |||
โดยที่ | คือโดมิโน 2 x 1
และ = คือโดมิโน 1 x 2 จำนวน 2 ตัว 55+

มาที่วิธีปกติกันก่อนเราก้ต้องเลือกว่าที่ตำแหน่งปัจจุบันเราจะวาง 2×1 หรือว่า 1×2 2 ตัวดีไปเรื่อยๆ ถ้าขนาดครบ n แล้วก้เพิ่มค่าคำตอบ แบบนี้

```
int search( int now , int n ){
	if( now == n ) return 1;
	if( now > n ) return 0;
	return search( now + 1, n ) + search( now + 2 , n );
}
main(){
	ans = search( 0 , n );
}
```

จบแล้วคับ ทำงานได้ใน O 2^n ได้มั้ง อนาถมาก

รายละเอียดของฟังก์ชั่นนี้

แต่ !! จากโค้ดข้างบนนี้เราจะสังเกตเห็นได้ว่ามันจะเรียก search(i , n) โคตรจะหลายครั้งๆ ทั้งๆที่การเรียก search(i , n) แต่ละครั้งมันให้คำตอบเท่าเดิม =.= ลองดูโค้ดแบบที่สองนะครับ

```
int keep[ MAX_N ];
int search( int now , int n ){
	if( keep[now] != –1 ) return keep[now];
	if( now == n ) keep[now] =1;
	if( now > n ) keep[now] = 0;
	keep[now] = search( now + 1, n ) + search( now + 2 , n );
	return keep[now];
}
main(){
	for( int c = 0 ; c < MAX_N ; c ++ ) keep[c] = –1;
	ans = search( 0 , n );
}
```

โค้ดข้างบนนี้ก้จะเก็บค่า search(i , n) สำหรับแต่ละค่า i ไว้ทำให้ถ้าเข้าไปซ้ำก็เรียกตอบได้ทันที ซึ่งหากพิจารณาดีๆแล้วโค้ดนี้จะทำงานได้ใน O(n) เลยทีเดียว > < เริ่มเห็นประโยชน์ของ dynamic programming กันแล้วไช่มั๊ยคับบบ จาก 2^n –> n เลยย

ถ้าลองสังเกตโค้ดข้างบนดีดี (อีกครั้ง) ก้จะพบนะคับว่าโจทย์ข้อนี้ คือ การหาค่าของ search(n) โดย search(0) = 1, search(1) = 1, search(i) = search(i – 1) + search(i – 2);
เพราะตารางขนาด 2xn จะมาจากการเอาขนาด 2x(n-1) มาเติมแนวตั้ง หรือเอาขนาด 2x(n-1) เติมแนวนอน 2 อัน ดังนั้นจะสามารถเขียนโปรแกรมแบบง่ายๆกว่าเดิมได้เป็น

```
int ans[MAXN_N];
main(){
        ans[0] = ans[1] = 1;
        for( int c = 2 ; c < MAX_N ; c ++ ) ans[c] = ans[c-1] + ans[c-2];
}
```

หลังจากรันข้างบนเสร็จเราสามารถตอบคำถามของทุกกระเบื้องขนาด 2xn ได้จากค่า ans[n] เลยคับบ