

문제해결기법 - 201921725 안성현

타일 문양 그리기

<2021/10/11>

■ 문제 소개 및 접근법

1-1] 타일 문양 그리기

① 문제

세로 길이가 2이고 가로 길이가 1인 2×1 타일 n 개를 이용하면 $2 \times n$ 도형을 만들 수 있다. 타일은 세로 타일 한 개로 한 열을 채우거나 가로 타일 한 쌍으로 두 열을 채운다. 단, 가로 타일 쌍은 연속적으로 나타나지 않고 세로 타일은 3개 이상 연속적으로 나타나지 않는다. n 이 입력으로 들어 왔을 때 제시된 조건을 만족하는 타일 문양의 개수를 20201212로 나눈 나머지로 출력하라. ($n \leq 1,000,000$ 이고 $n \leq 30$ 인 경우에는 타일 문양을 한 줄에 하나씩 출력한다. 세로 타일은 문자 'I', 가로 타일 쌍은 '='로 나타낸다. '='은 'I'보다 앞선다고 가정한다.)

② 설명

▶ 먼저 계산에 용이하도록 'I'는 1로, '='는 2로 생각했다. 만약 "==I=="라는 문양이라면 2,1,2가 된다. 문양이 만들어질 때 숫자(1,2)의 합은 n 이 된다. 이 때 2,2와 1,1,1이 나타나면 문제의 조건에 위배된다. 또한 $n \geq 3$ 부터는 두 개 이상의 숫자가 합쳐져야 한다. 따라서 두 개의 숫자씩 선택하며 문제를 풀어 보았다. (i) 2,1를 선택한 경우에는 뒤에 올 수 있는 두 개의 숫자가 (2,1),(1,2)가 된다. (ii) 1,2인 경우에는 뒤에 (1,2),(1,1)이 올 수 있다. (iii) 1,1인 경우에는 뒤에 (2,1)이 올 수 있다. 그런데 상황에 따라 뒤에 한 개의 숫자가 올 수도 있다. 따라서 좀 더 체계적으로 분류를 해보았다.

* **ThisSum**: 지금까지의 숫자 합, **dist**: $n - \text{ThisSum}$, **n**: 숫자 합 목표치, **count**: 문양의 현재 개수

* $n=4 \rightarrow 1,1$ 을 선택 $\rightarrow \text{ThisSum}=1+1=2$, $\text{dist}=4-2=2$, 뒤에 2가 붙어야 $\text{count}+1$

(i) 2,1을 선택한 경우

- $\text{dist} \geq 3$: **ThisSum**이 n 이 되기까지 3이상의 차이가 존재

(2,1)을 선택함 or (1,2)를 선택함 ex) $n=6$, 21-21 or 21-12

- $1 \leq \text{dist} \leq 2$

dist에 해당하는 숫자를 뒤에 덧붙이고 count+ 1 ex) n=5, 21-2

- dist=0

목표치를 도달했으므로 count+ 1

(ii) 1,2를 선택한 경우

- dist ≥ 3 : ThisSum이 n이 되기까지 3이상의 차이가 존재

(1,2)를 선택함 or (1,1)을 선택함 ex) n=6, 12-12 or 12-11

- dist=2

(1,1)을 선택함 ex) n=5, 12-11

- dist=1

1을 덧붙이고 count+ 1 ex) n=4, 12-1

- dist=0

목표치를 도달했으므로 count+ 1

(iii) 1,1을 선택한 경우

- dist ≥ 3 : ThisSum이 n이 되기까지 3이상의 차이가 존재

(2,1)를 선택함 ex) n=5, 11-21

- dist=2

2를 덧붙이고 count+ 1 ex) n=4, 11-2

- dist=1

1을 붙이면 조건 불만족하니 pass ex) n=3, 111 불가

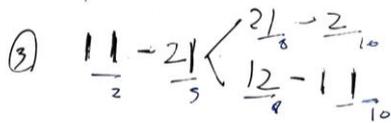
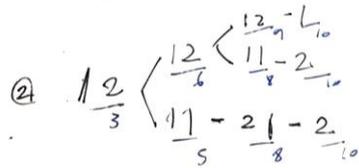
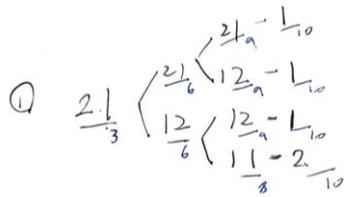
- dist=0

목표치를 도달했으므로 count+ 1

이 세가지 경우를 그대로 재귀 호출로 구현하고 최종적으로 합산된 count를 반환하면 우리가 원하는 타일 문양의 개수를 얻을 수 있다. 타일 문양 출력을 원한다면 각 경우마다 “==|”, “|==”, “||”를 누적해서 출력하면 된다. 재귀 호출을 할 때마다 리스트에 타일을 추가하고 한 문양이 만들어지면 리스트에 저장된 타일 (ex) ==|==|==|)을 출력하는 방법이다. 한 문양이 만들어 지면 리턴 후 다른 문양을 살펴보는데, 이 때 가장 마지막에 추가한 타일은 리스트에서 제거한다. (ex) ==|==|==| -> ==|==|) 이후 다시 리스트에 다른 타일을 추가하며 다른 문양을 만드는 것이다. (ex) ==|==| -> ==|==| |==)

재귀 호출을 하면 어떻게 동작하는지 n=10인 경우의 예시를 아래에 추가했다.

ex) $n = 10$ \therefore ThisSum

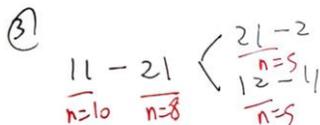
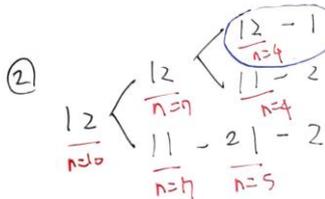
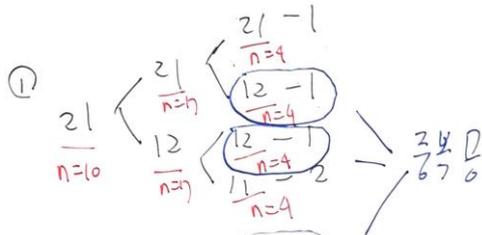


맨 첫 번째 문양만 살펴 보겠다.

2,1을 선택해서 ThisSum은 3, dist=7이 된다. $\text{dist} \geq 3$ 이므로 (2,1)을 덧붙였다. 이후 dist=4가 되므로 다시 (2,1)을 덧붙였다. 이번에는 dist=1이 되므로 1을 덧붙였다. 1을 덧붙이면서 dist=0이 되니 count+1을 한다. 따라서 (2121211)이 완성되고 문양은 “==|==|==|”이 된다.

하지만 지금까지 설명한 방법은 너무 많은 중복 계산으로 인해 효율적이지가 않다.

ex) $n = 10$ $\therefore n$



첫 번째는 $n=10$ 일 때 (2,1)을 선택해서 다음 경우를 본다. 재귀 호출을 하면 $n=7$ 일 때 (2,1)을 선택해서 다음 경우를 본다. 이런 방식으로 생각하면 중복인 케이스를 쉽게 확인할 수 있다.

이후 필자는 효율적인 방법을 생각해보다가 (세 가지 경우)를 수식으로 변환한 뒤 식 정리가 가

능한지 확인해보았다.

$n=3$,

- ① 2,1 이용 \Rightarrow 1가지
 - ② 1,2 이용 \Rightarrow 1가지
 - ③ 1,1 이용 \Rightarrow 0가지
- $H+1=2 = \text{Count}$
 $=1$
 $1=1$

$4 \leq n \leq 5$,

- ① 2,1 이용 \Rightarrow 1가지
 - ② 1,2 이용 \Rightarrow 1가지
 - ③ 1,1 이용 \Rightarrow 1가지
- $H+1=3 = \text{Count}$
 $n=4$ $n=5$
 $=11$ $=1=1$
 $1=1$ $1=11$
 $11=$ $11=1$

$n \geq 6$,

- ① 2,1 이용 $\Rightarrow (2,1 \text{ 이용}; n-3) + (1,2 \text{ 이용}; n-3)$
- ② 1,2 이용 $\Rightarrow (1,2 \text{ 이용}; n-3) + (1,1 \text{ 이용}; n-3)$
- ③ 1,1 이용 $\Rightarrow (2,1 \text{ 이용}; n-2)$

$n < 3$ 일 때는 이 방식으로 풀 수가 없다. (두 개의 숫자를 선택하는 구조이기에) $3 \leq n \leq 5$ 일 때는 각 경우에 대해 예측이 쉽기 때문에 따로 식으로 정리하지는 않았다.

$n \geq 6$ 일 때는 초기 선택 후 $\text{dist} \geq 3$ 을 반드시 만족한다. 따라서 위 그림과 같은 식을 만족한다.

- ① 2,1 이용 = a $n \geq 6$ 인 n에 대해서
- ② 1,2 이용 = b a, b, c 이용
- ③ 1,1 이용 = c $a_n \quad b_n \quad c_n$

ex) a_6 : $n=6$ 일 때 (2,1)을 선택해서
문양 개수를 따져봄

$$\underbrace{2,1}_{n=6} < \underbrace{2,1}_{n=3} \Rightarrow 2721$$

$$\left\{ \begin{array}{l} a_n = a_{n-3} + b_{n-3} \\ b_n = b_{n-3} + c_{n-3} \\ c_n = a_{n-2} \end{array} \right.$$

원하는 것: $a_n + b_n + c_n = \text{Count}$

이 2721

$$\left\{ \begin{array}{l} a_n = 2a_{n-3} - a_{n-6} + a_{n-9} \\ \text{Count} \\ = a_{n-2} + 3a_{n-3} + a_{n-5} - 2a_{n-6} + 2a_{n-8} \end{array} \right.$$

이후 (i,ii,iii)각 경우를 a,b,c로 단순하게 표현하고 6이상인 n에 대해서 a,b,c를 이용하는 것을 각각 a_n, b_n, c_n 으로 표현했다. 그러면 위 그림과 같이 a_n, b_n, c_n 에 대한 각각의 수식이 나온다. 이 수식을 정리하면 $a_n, count$ 를 새롭게 나타낼 수 있다. 여기서 식을 보면 a_{n-8} 이 포함되어 있는데 처음에 말했다 싶이 a_0, a_1, a_2 와 같이 n이 3보다 작은 경우는 존재하지 않는다. 따라서 a_n 과 count수식은 n이 11이상일 때 성립한다. 이후 필자는 $a_3 \sim a_{10}$ 인 값을 찾은 뒤 a_n 값을 담은 array배열에 저장했다. 그러면 반복문을 통해 i가 11부터 n이 될 때까지 $a_i(10 < i < n)$ 를 쉽게 저장할 수 있다. 마지막으로 $count(a_{n-2} + 3a_{n-3} + a_{n-5} - 2a_{n-6} + 2a_{n-8})$ 을 반환하면 된다. 앞에서 굳이 설명은 안 했지만 문제에서 출력은 20201212로 나눈 나머지를 요구했기 때문에 a_i 를 저장할 때 20201212로 나눈 나머지를 저장하고 count도 20201212로 나눈 나머지로 리턴하면 된다.

$(a + b) \% d = ((a \% d) + (b \% d)) \% d$ 인 모듈러 연산 성질을 이용하였다.

소스 코드와 실행 결과

2-1] 소스 코드와 실행 결과

① 코드

```
#include<time.h>
#include <stdio.h>
#include <stdlib.h>
#pragma warning(disable:4996)
#define max 30

typedef char element;
typedef struct {
    element array[max];
    int size;
}ArrayListType;
typedef ArrayListType * ArrayList_ptr;

// 리스트 초기화
void init(ArrayList_ptr list) {
    list->size = 0;
}

// 리스트 생성
ArrayList_ptr create() {
    return (ArrayList_ptr)calloc(1, sizeof(ArrayListType));
}

// 빈 리스트 확인
int is_empty(ArrayList_ptr list) {
    if (list->size == 0)
        return 1;
    else
        return 0;
}

// 풀 리스트 확인
int is_full(ArrayList_ptr list) {
    if (list->size == max)
        return 1;
    else
        return 0;
}

// 아이템 삽입
void insert_last(ArrayList_ptr list, element item) {
    if (!is_full(list)) {
        int pos = list->size;
        list->array[pos] = item;
    }
}
```

```

        list->size++;
    }
    else
        printf("리스트가 꽉 차있습니다\n");
}

// 아이템 삭제
void delete_last(ArrayList_ptr list) {
    if (!is_empty(list)) {
        list->size--;
    }
    else
        printf("리스트가 비어 있습니다\n");
}

int n; // 2xn개의 타일에서 'n'
int count; // 문양의 개수
char tiles[500][31]; // 문양 모음

// recursion-call
int tile30(int i, int j, int ThisSum, ArrayList_ptr list) {
    // n=1
    if (n == 1) {
        tiles[count++][0] = '|'; // 1tile
        return count;
    }

    // n=2
    if (n == 2) {
        tiles[count][0] = '=';
        tiles[count++][1] = '='; // 1tile
        tiles[count][0] = '|';
        tiles[count++][1] = '|'; // 2tiles
        return count;
    }

    // first call
    if (i == 0 && j == 0) {
        tile30(2, 1, 0, list);
        tile30(1, 2, 0, list);
        tile30(1, 1, 0, list);
    }
    ThisSum += i + j;
    int dist = n - ThisSum;

    // case 1
    if (i == 2 && j == 1) {
        insert_last(list, '=');
        insert_last(list, '=');
        insert_last(list, '|');

        if (dist >= 3) {
            // string incomplete -> call

```

```

        tile30(2, 1, ThisSum, list);
        tile30(1, 2, ThisSum, list);
        delete_last(list);
        delete_last(list);
        delete_last(list);
    }
    else if (dist == 2) {
        // string complete -> return
        insert_last(list, '=');
        insert_last(list, '=');
        for (int j = 0; j < list->size; j++) {
            tiles[count][j] = list->array[j];
        }
        delete_last(list);
        delete_last(list);
        delete_last(list);
        delete_last(list);
        delete_last(list);

        count++;
        return 0;
    }
    else if (dist == 1) {
        // string complete -> return
        insert_last(list, '|');
        for (int j = 0; j < list->size; j++) {
            tiles[count][j] = list->array[j];
        }
        delete_last(list);
        delete_last(list);
        delete_last(list);
        delete_last(list);

        count++;
        return 0;
    }
    else {
        // string complete -> return
        for (int j = 0; j < list->size; j++) {
            tiles[count][j] = list->array[j];
        }
        delete_last(list);
        delete_last(list);
        delete_last(list);

        count++;
        return 0;
    }
}
// case 2
else if (i == 1 && j == 2) {
    insert_last(list, '|');
    insert_last(list, '=');

```

```

insert_last(list, '|');

if (dist >= 3) {
    // string incomplete -> call
    tile30(1, 2, ThisSum, list);
    tile30(1, 1, ThisSum, list);
    delete_last(list);
    delete_last(list);
    delete_last(list);
}
else if (dist == 2) {
    // string incomplete -> call
    tile30(1, 1, ThisSum, list);
    delete_last(list);
    delete_last(list);
    delete_last(list);
}
else if (dist == 1) {
    // string complete -> return
    insert_last(list, '|');
    for (int j = 0; j < list->size; j++) {
        tiles[count][j] = list->array[j];
    }
    delete_last(list);
    delete_last(list);
    delete_last(list);
    delete_last(list);

    count++;
    return 0;
}
else {
    // string complete -> return
    for (int j = 0; j < list->size; j++) {
        tiles[count][j] = list->array[j];
    }
    delete_last(list);
    delete_last(list);
    delete_last(list);
    count++;
    return 0;
}
}
// case 3
else if (i == 1 && j == 1) {
    insert_last(list, '|');
    insert_last(list, '|');

    if (dist >= 3) {
        // string incomplete -> call
        tile30(2, 1, ThisSum, list);
        delete_last(list);
        delete_last(list);
    }
}

```

```

    }
    else if (dist == 2) {
        // string complete -> return
        insert_last(list, '=');
        insert_last(list, '=');
        for (int j = 0; j < list->size; j++) {
            tiles[count][j] = list->array[j];
        }
        delete_last(list);
        delete_last(list);
        delete_last(list);
        delete_last(list);

        count++;
        return 0;
    }
    else if (dist == 1) {
        // string incomplete -> pass & return
        delete_last(list);
        delete_last(list);
        return 0;
    }
    else {
        // string complete -> return
        for (int j = 0; j < list->size; j++) {
            tiles[count][j] = list->array[j];
        }
        delete_last(list);
        delete_last(list);

        count++;
        return 0;
    }
}
return count;
}

int one_array[1000001];
int two_array[1000001];
int thr_array[1000001];

// dynamic-programming (사용 x - stack overflow)
int tile(int i, int j, int ThisN) {
    // first call
    if (i == 0 && j == 0) {
        int r;
        for (r = 0; r <= n; r++) {
            one_array[r] = two_array[r] = thr_array[r] = -1;
        }
        one_array[ThisN] = tile(2, 1, n - 3) % 20201212;
        two_array[ThisN] = tile(1, 2, n - 3) % 20201212;
        thr_array[ThisN] = tile(1, 1, n - 2) % 20201212;
    }
}

```

```

count = ((one_array[ThisN] + two_array[ThisN] + thr_array[ThisN])) %
20201212;
    return count;
}

// base_case
if (ThisN <= 2) {
    if (ThisN == 1 && (i == 1 && j == 1))
        return 0;
    else
        return 1;
}

// return cache
if (i == 2 && j == 1) {
    if ((one_array[ThisN] != -1) && (two_array[ThisN] != -1)) {
        return one_array[ThisN] + two_array[ThisN];
    }
}
else if (i == 1 && j == 2) {
    if ((two_array[ThisN] != -1) && (thr_array[ThisN] != -1)) {
        return two_array[ThisN] + thr_array[ThisN];
    }
}
else if (i == 1 && j == 1) {
    if (one_array[ThisN] != -1) {
        return one_array[ThisN];
    }
}

// case 1
if (i == 2 && j == 1) {
    one_array[ThisN] = tile(2, 1, ThisN - 3) % 20201212;
    two_array[ThisN] = tile(1, 2, ThisN - 3) % 20201212;
    return one_array[ThisN] + two_array[ThisN];
}
// case 2
else if (i == 1 && j == 2) {
    two_array[ThisN] = tile(1, 2, ThisN - 3) % 20201212;
    thr_array[ThisN] = tile(1, 1, ThisN - 2) % 20201212;
    return two_array[ThisN] + thr_array[ThisN];
}
// case 3
else if (i == 1 && j == 1) {
    one_array[ThisN] = tile(2, 1, ThisN - 3) % 20201212;
    return one_array[ThisN];
}
}

// use two expressions
// an = 2a(n-3)-a(n-6)+a(n-8)
// an+bn+cn=a(n-2)+3a(n-3)+a(n-5)-2a(n-6)+2a(n-8)
int tile_iter() {

```

```

int infor[11] = { -1,-1,-1,1,1,1,2,2,2,3,4 };
int* array = (int*)malloc(sizeof(int) * 1000001);
for (int i = 0; i <= 10; i++) {
    array[i] = infor[i];
}
for (int i = 11; i <= n; i++) {
    array[i] = ((2 * array[i - 3]) - array[i - 6] + array[i - 8]) % 20201212;
}
count = (array[n - 2] + (3 * array[n - 3]) + array[n - 5] - (2 * array[n - 6]) +
(2 * array[n - 8])) % 20201212;
return count;
}

int main() {
    count = 0;
    scanf("%d", &n);

    // n이 30이하일 때
    if (n <= 30) {
        ArrayList_ptr list = create();
        init(list);
        printf("%d\n", tile30(0, 0, 0, list));
        for (int i = 0; i < count; i++) {
            printf("%s\n", tiles[i]);
        }
    }

    // n이 30초과일 때
    else {
        printf("%d\n", tile_iter());
    }

    return 0;
}

```

<오류 발견 / 2021-10-12 pm 21:40>

a_n 으로 정리한 수식 자체는 올바르나 나머지 연산을 하면서 a_{n-3} 이 a_{n-6} 보다 작아지면서 음수가 나오는 문제가 발생하였다. 이것은 정리한 수식에 뺄셈 연산이 존재해서 문제가 된다. 따라서 수식을 정리할 때 덧셈 연산으로만 되게 하면 문제가 해결된다. 일단 가장 쉽게 하는 방법은 b_n 수식만 $b_{n-3} + a_{n-5}$ 로 바꾸고 문제를 다시 푸는 것이다. $n=8$ 이상부터 이 식은 성립한다. ($n=7$ 일 때 b_7 을 구하려면 a_2 를 알아야 되는데 $n \geq 3$ 을 항상 만족해야 됨)

<변경 코드>

```

// use three expressions
// an = a(n-3)+b(n-3)
// bn = b(n-3)+a(n-5)

```

```

// cn = a(n-2)
// count = an+bn+cn = a(n-3)+2*b(n-3)+a(n-5)+a(n-2)
// n=8 이상부터 식이 성립함 (n=7일 때 b(7)을 구하려면 a(2)를 알아야 되는데 n>=3을 항상 만족해야 됨)
int tile_iter() {
    int a_infor[8] = { -1,-1,-1,1,1,1,2,2 }; // a0~a7
    int b_infor[8] = { -1,-1,-1,1,1,1,1,2 }; // b0~b7
    int* a_array = (int*)malloc(sizeof(int) * 1000001);
    int* b_array = (int*)malloc(sizeof(int) * 1000001);

    for (int i = 0; i <= 7; i++) {
        a_array[i] = a_infor[i];
        b_array[i] = b_infor[i];
    }

    for (int i = 8; i <= n; i++) {
        a_array[i] = (a_array[i - 3] + b_array[i - 3])%20201212;
        b_array[i] = (b_array[i - 3] + a_array[i - 5])%20201212;
    }

    count = (a_array[n-3]+(2*b_array[n-3])+a_array[n-5]+a_array[n-2])%20201212;
    return count;
}

```

② DEV-C++ 컴파일러 이용 실행 결과

```
E:\대학과 과제\3-2] CUK#문제해결기법#과제5 - 타일 문양#과제 제출#tile.exe
15
24
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
==|==|==|==|
-----
Process exited after 5.022 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

입력: 도형의 가로 길이 (15 -> 2x15)

출력: 문양의 개수 % 20201212 = 24, 가능한 문양 출력

```
E:\대학과 과제\3-2] CUK#문제해결기법#과제5 - 타일 문양#과제 제출#tile.exe
1000000
9697641
-----
Process exited after 2.217 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

입력: 도형의 가로 길이 (1,000,000 -> 2x1,000,000)

출력: 문양의 개수 % 20201212 = 9697641