#C언어 반
#6주차

Dp-나머지

T. 김재형
Asst.강민구,정태현

ALOHA
The algorithm club.

지난 시간 요약!

Dp란? 동적계획법이란? Dynamic programming?
★ 기억하며 풀기?
★ 답을 재활용?
★ 앞에서 구했던 답을 이용하고 옆에서도 이용하고..
★ 동적계획법? , 재귀함수?, memorization?
★ 수학? , 점화식?

```
int fibonachi(int n) {
    if (n <= 2) return 1;
    else return fibonachi(n - 1) + fibonachi(n - 2);
}
```

삽질

VS

동적계획법(dynamic programming)

동적계획법(dynamic programming)

1. Top-down 방식

```
#include<stdio.h>

int arr[100];
int fibonachi(int n) {
    if (n <= 1) return 1;
    else if (arr[n] >0) return arr[n];
    else {
        arr[n] = fibonachi(n - 1) + fibonachi(n - 2);
        return arr[n];
    }
}
```

동적계획법(dynamic programming)

2. bottom-up방식

```c
#include<stdio.h>

int fibonachi[105] = { 0,1,1, },N;

int main() {
	for (int i = 2; i < 100; i++) {
		fibonachi[i] = fibonachi[i - 1] + fibonachi[i - 2];
	}
	scanf("%d", &N);
	printf("%d",fibonachi[N]);
}
```

시간복잡도?

- 연산의 횟수!
- NOT 실행시간

- **Example: Sum of Integers less than 10^6**

```
int  sum = 0;
for (i = 0;  i < 1000000; i++)
{
    sum = sum + i ;
}
```

```
float sum(float list[], int n)
{
        float tempsum = 0;                count++; (할당)
        int I;
        for(i=0; i<n; i++) {
                                          count++; (for 문내 연산)
         tempsum += list[i];              count++; (값 계산)
        }
                                          count++; (for문 빠져나오기!)
        return 0;                         count++;  (return 실행)
}
```

```
float sum(float list[], int n)
{
        float tempsum = 0;                  count++; (할당)
        int I;
        for(i=0; i<n; i++) {
                                            count++; (for 문내 연산)
         tempsum += list[i];                count++; (값 계산)
        }
                                            count++; (for문 빠져나오기!)
        return 0;                           count++;  (return 실행)
}
```

2N+3 의 시간복잡도를 가진다!

시간복잡도의 종류

Big-O-Notation(빅오표기법) - O(N)

ω(오메가) 표기법----- ω(N)

Θ(표기법) ----- Θ(N)

등등…..

ALOHA
The algorithm club.

Big –O –Notation 표기법?

- 상수항 무시

  O(2N) -> O(N)
  O($N^2 + 2$) $\rightarrow O(N^2)$

- 영향력 없는 항 무시

  O($N^2 + N$) $\rightarrow O(N^2)$
  O(N^2) 이 가장 지배적이기 때문에 그 외에 영향력이 없는 항들은 무시한다!

For 문 1억번돌면 = 대략 1초!
(백준 기준)

```c
#include <stdio.h>
#include <windows.h>//GetTickCount()를 사용하려면 반드시 써주자

int main() {
	int n, i, j, a, t;
	scanf("%d",&n);
	t = GetTickCount();
	for (i = 0; i<n; i++)for (j = 0; j<n; j++)a = i + j;

	printf("%f", (GetTickCount() - t) / 1000.);//.을 붙혀서 float으 캐스팅한다.
	system("PAUSE");
	return 0;
}
```

이친수 문제 복습!

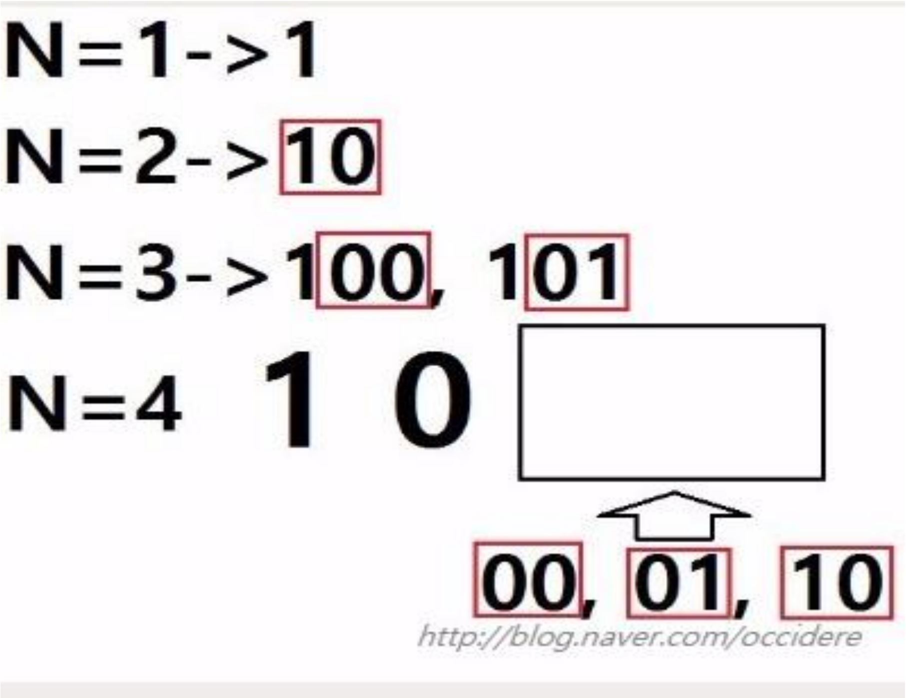
이친수란?　0과1로만 이루어진 수
★ 0으로 시작하지 않는다.
★ 1이 두번 연속으로 나타나지 않는다. (11부분 문자열 X)

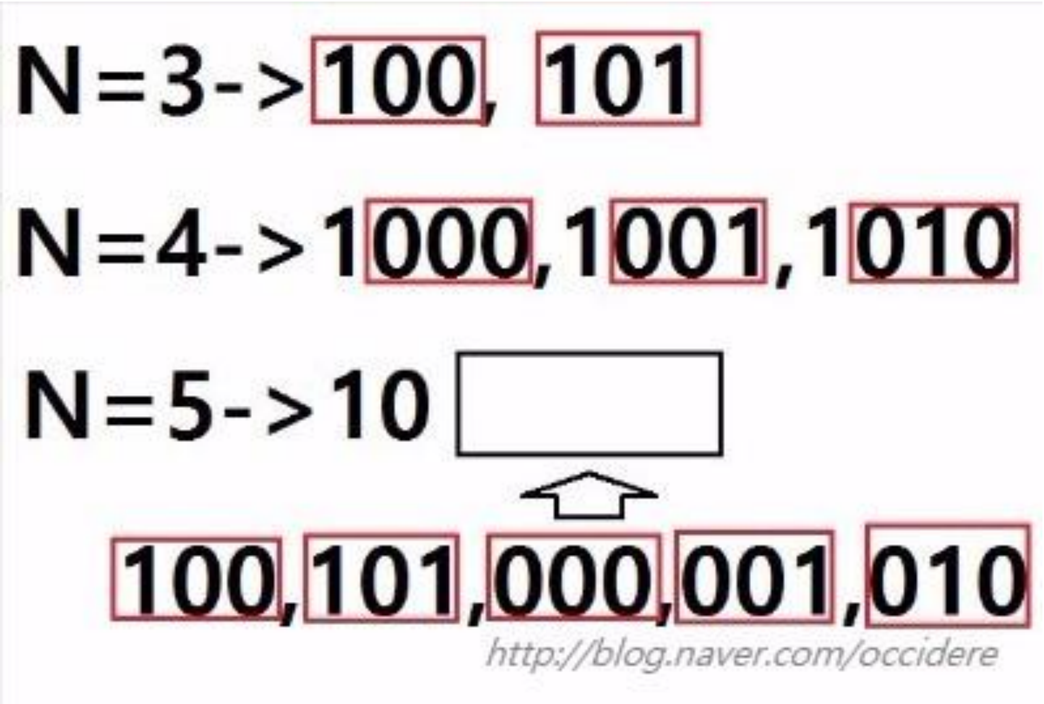
N 자리의 이친수의 개수?

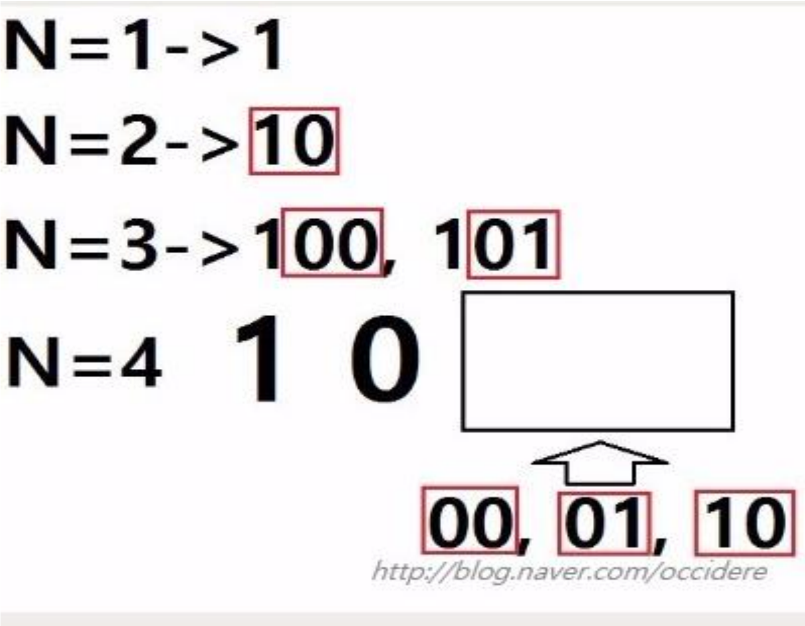
■ 지금부터 두가지 방식으로 풀어봅시다!

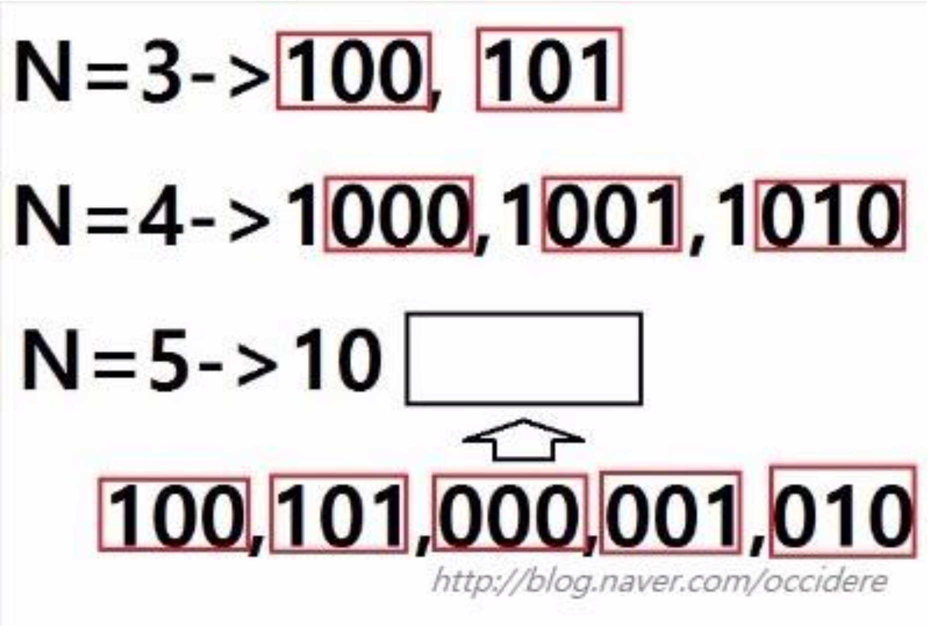
풀이1. 1차원 점화식

Point : 이친수란 10 으로 시작하는 숫자이다!(n≥ 2)









★**10** 아래에는 **1**로 시작할수 있으며 **0**으로도 시작할수 있다**!**★

Point : 이친수란 10 으로 시작하는 숫자이다!(n≥ 2)

N-2 자리의 이친수 = 1로 시작하며 N자리의 이친수중 10아래의 부분수열들!

N-1 자리의 1아래의 이친수 = 0으로 시작하며 N자리의 이친수중 10아래의 부분수열들!

※N자리의 이친수를 표현

1	0	1	.	.	.	.	.	.	.

1	0	0	.	.	.	.	.	.	.

※N-1자리의 이친수를 표현

1	0	.	.	.	.	.	.	.

※N-2자리의 이친수를 표현

1	.	.	.	.	.	.	.

★ 결론 : N 자리의 개수 = N-1 자릿수+N-2 자릿수

코드로 표현을 해봅시다!

1차원 배열임에 주목하자!

```c
#include<stdio.h>

int N;
long long answer, dp[94];
//여유롭게 배열을 설정하자!
int main() {
    dp[1] = 1;
    dp[2] = 1;
    for (int i = 2; i < 93; i++)  dp[i] = dp[i - 1] + dp[i - 2];
    scanf("%d", &N);
    printf("%lld", dp[N]); // long long 은 lld 로 받는다!
    return 0;
}
```

풀이2. 2차원 점화식

★N 자리의 이친수는 0으로 끝나거나 1로 끝나거나 둘중 하나이다!

★N 자리의 이친수는 0으로 끝나거나 1로 끝나거나 둘중 하나이다!

★N자리의 이친수중 0으로 끝나는 수는 N-1 자리의 이친수의 개수와 같다!

★N자리의 이친수중 1로 끝나는 수는 N-1 자리의 이친수 중 0으로 끝나는 수의 개수와 같다 !

★N자리의 이친수중 0으로 끝나는 수는 N-1 자리의 이친수의 개수와 같다!

1	0						1,0

+ 0

★N자리의 이친수중 1로 끝나는 수는 N-1 자리의 이친수 중 0으로 끝나는 수의 개수와 같다 !

1	0						0

+ 1

풀이 요약

N자리의 이친수의 개수= (N자리중) 1로끝나는 수+ (N자리중) 0으로 끝나는수
= (N-1자리의 이친수 중 0으로끝나는 수)+ (N-1자리의 이친수의 개수)

코드로 표현해 봅시당!

2차원 배열임에 주목하자!

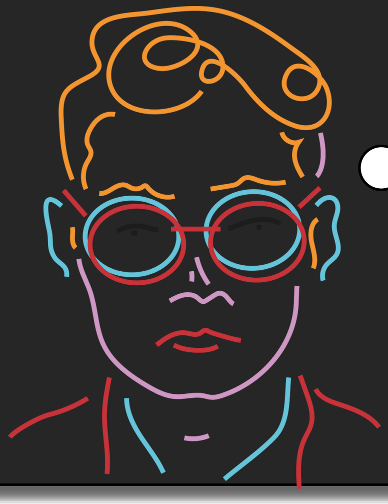
```
#include<stdio.h>

int N;
long long   dp[93][3];
int main() {
    scanf("%d", &N);
    dp[1][0] = 0; dp[1][1] = 1; dp[1][2] = 1;
    for (int i = 2; i < 92; i++) {
        dp[i][0] = dp[i - 1][2];
        //dp[i][0]= i자리수의 이친수중 0으로 끝나는 수의 개수
        dp[i][1] = dp[i - 1][0];
        //dp[i][1]= i자리수의 이친수중 1으로 끝나는 수의 개수
        dp[i][2] = dp[i][0] + dp[i][1];
        //dp[i][2]= i자리수의 이친수의 개수
    }
    printf("%lld", dp[N][2]);
}
```

#1

점화식(1차원 및 2차원)

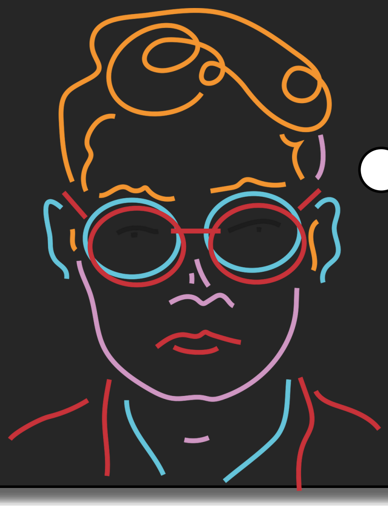
11727	2*N 타일링	11057	오르막수
11659	●구간 합 국하기4	10844	쉬운 계단 수
9095	1,2,3 더하기		

ALOHA
The algorithm club.



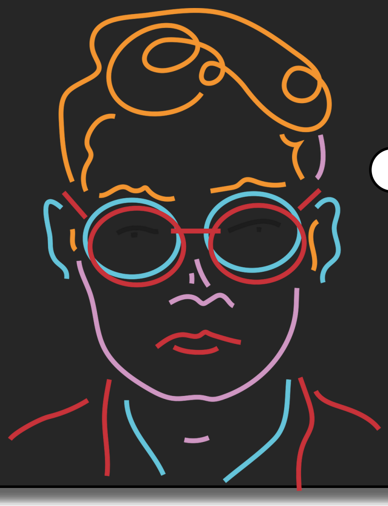
이거 그냥 배열세려박
고 각각의 케이스 마다
i 번째 부터 j 번째 수
까지 합을 구하면 되는
거 아닌가?

Array[1000000] 칸..
i 번째부터 j 까지
For(int a=i ; a<=j ; a++)
Sum +=arr[a] 이런식으로...



최악의 상황은 배열
100000 개 입력받고 ...
100000 번 모두 1번째
부터 100000번째..

그럼 연산 횟수는
100000번 동안 각 한
번동안 100000번 연산
하니 **1000억!**




어카지..

n 번째 칸의 배열 = 첫번째부터 n 번째 칸까지의 합!

i번째부터 j 번째 까지의 합 = j번째 배열 - i번째 배열!

힌트 : 어려운 문제는 사고의 확장

#2

점화식 심화

★1149 RGB거리
2579 계단 오르기
2156 포도주 시식

2670 ★연속부분 최대곱
1912 연속합

▲1149번

모든 경우의수를 따지면 시간 복잡도 3^n 이 나온다!
그렇다면 이것또한 조금더 효율적으로 풀수있는 방법은 없을까?

▲2670번
모든 경우의 수를 다 곱해서 비교하기에는 좀 그렇고...
어떻게 곱해야 조금더 숫자가 커질까?

#3

수학

1932 정수삼각형
2869 ★달팽이는 올라가고싶다.

▲1932번
안될땐 노가다를 해보자!

수고했어요!