자료구조및알고리즘 과제1 202012468 김현서

1. 피보나치 수열을 재귀함수를 통해 구현을 하면 시간복잡도가 $O(2^n)$을 가지는 이유와 문제점

hs11015@DESKTOP-QM04TOL: ~/ict1

```
#include <stdio.h>

int Fibo(int n)
{
        if (n==1)
                return 0;
        else if (n==2)
                return 1;
        else
                return Fibo(n-1) + Fibo(n-2);
}

int main(void)
{
        int num;
        printf("정수를 입력하세요 : ");
        scanf("%d",&num);   //사용자에게 임의의 값을 입력받기
        for (int i=1; i<=num; i++)
                printf("%5d", Fibo(i));   //사용자에게 입력받은 값을 Fibo로 출력
        return 0;
}
```

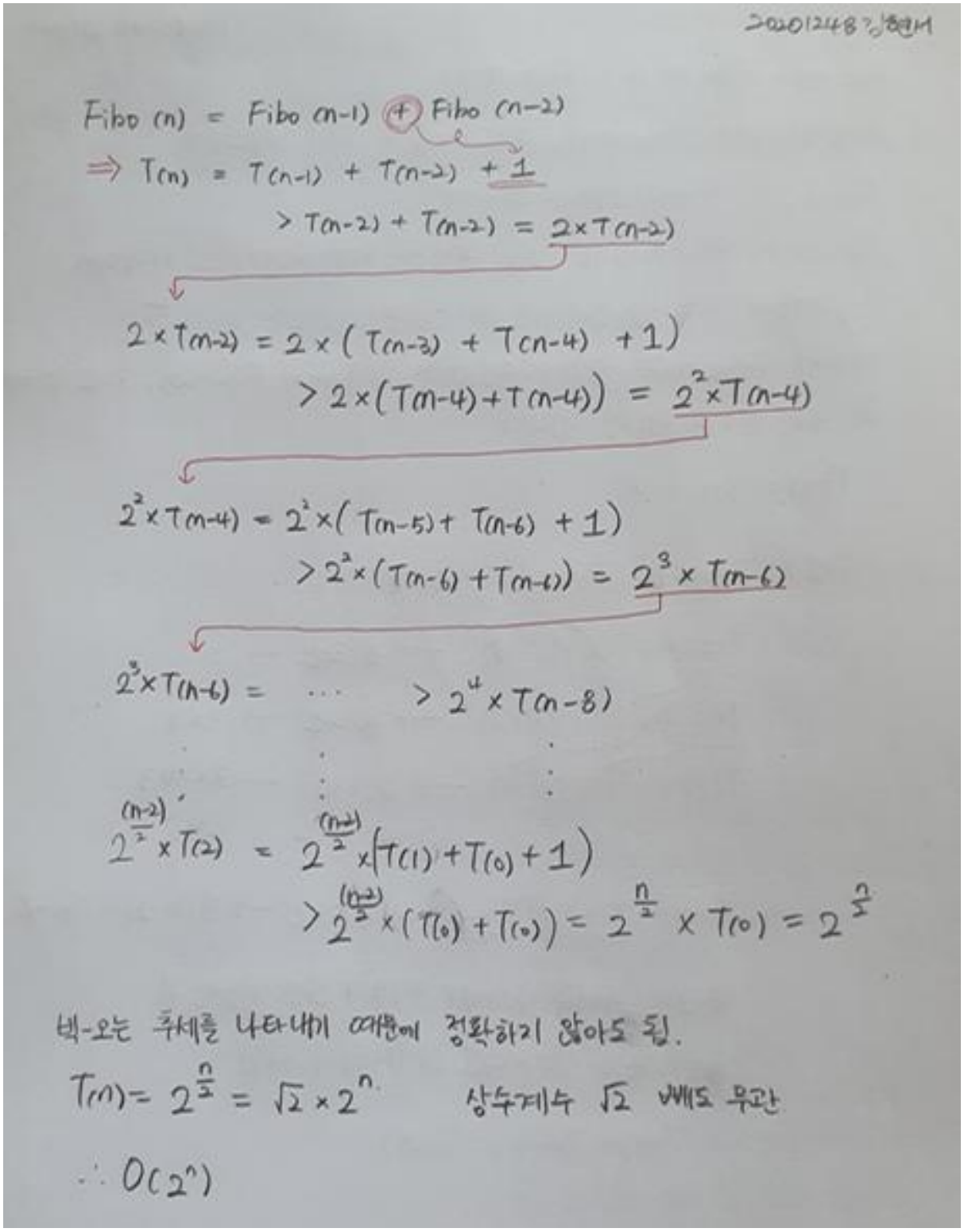
교재에 나온 코드에 main함수만 사용자에게 임의의 값을 입력 받는 형식으로 바꾼 코드입니다.

이 코드로 실행하게 되면 이렇게 결과는 옳게 나오지만,

hs11015@DESKTOP-QM04TOL: ~/ict1

```
hs11015@DESKTOP-QM04TOL:~/ict1$ vi Fibo.c
hs11015@DESKTOP-QM04TOL:~/ict1$ gcc Fibo.c
hs11015@DESKTOP-QM04TOL:~/ict1$ ./a.out
정수를 입력하세요 : 7
    0    1    1    2    3    5    8hs11015@l
```

피보나치 수열을 재귀함수를 통해 구현을 하게 되면 "Fibo(n)=Fibo(n-1)+Fibo(n-2)" 형태로 구현됩니다. 즉, 두 개의 Fibo 함수가 다시 호출되는 것인데 C언어는 절차 지향 언어이기 때문에 '+'연산자 왼편에 있는 Fibo함수의 호출이 완료되어야 비로소 '+'연산자 오른편에 있는 함수 호출이 진행됩니다. 그렇기 때문에 Fibo(n)을 계산하기 위해 Fibo 함수를 호출하는 횟수인 T(n)은 중복 계산되는 Fibo의 값들을 모두 포함한 값입니다. 그렇기 때문에 굉장히 느리게 시행됩니다.



이러한 계산식을 통해 위 코드는 $O(2^n)$이라는 값을 갖게 됨을 보일 수 있습니다. $O(2^n)$은 시간복잡도 측면에서 보면 효율이 많이 떨어지기 때문에 치명적이라고 말 할 수 있습니다

2. O(n)의 시간 복잡도를 가지도록 하기 위해 1번에서 말한 문제점을 어떤 방법으로 수정했는지에 대한 설명

교수님께서 강의에서 중복되어 실행되는 값을 제거하여 한 번만 실행되도록 만들면 된다는 힌트를 주셨는데 처음에는 동적 할당을 이용해서 배열 만들어서 해보는 방식으로 접근했는데, 이렇게 하면 배열을 사용해 다음 배열을 도출하는 방법만 생각이 나고, 재귀함수를 사용해 다음 배열을 도출하는 방법을 알아내지 못했습니다.

```c
#include <stdio.h>
#include <stdlib.h>

int Fibo(int n)
{
    printf("func call param %d₩n", n);
    int* arr, count = 0, size = 3, a = n - 1;
    arr = (int*)calloc(size, sizeof(int));  // 3개의 저장공간 할당


    if (n == 1)
    {
        arr[a] = 0;
        return arr[a];
    }

    else if ((n == 2) || (n == 3))
    {
        arr[a] = 1;
        return arr[a];
    }

    //n==3까지 했을 때 arr[2]까지 할당(3개)

    else if (n > 3)
    {
        for (int i = 0; i <= n; i++)
        {
            count += 1;
            if (count == size)   // 할당했던 3개의 저장공간을 모두 사용하면
            {
                size += 3;      //크기를 늘려 재할당시킴
                arr = (int*)realloc(arr, size * sizeof(int));



            }

        }

        if (arr[a] > 0)
            return arr[a];

        else if (arr[a] <= 0)
            return arr[a] = Fibo(n - 1) + Fibo(n - 2);

    }

    else
    {
        printf("양의 정수를 입력해 주세요.");
        return -1;
    }

    free(arr);
}

int main(void)
{
    int num, cnt = 0;

    printf("정수를 입력하시면,₩n입력하신 정수번째까지의 피보나치 수열을 출력해드립니다.₩n(단, 피보나치 수열을 0부터 시작) : ");
    scanf_s("%d", &num);
    printf("₩n가독성을 높이기 위해 한 줄에 숫자 5개씩 출력하겠습니다.₩n₩n");

    for (int i = 1; i <= num; i++)
    {
        printf("%d₩t₩t", Fibo(i));
        cnt++;
        if ((cnt % 5) == 0)
            printf("₩n");
    }

    return 0;
}
```

위처럼 동적 할당을 메모이제이션으로 사용해보려고 했는데, 이런 방법으로는 쓸 수 없는 것인지 printf("func call param %d₩n", n)을 이용해 코드의 반복 횟수를 살펴보니 O(n)이 아닌 $O(2^n)$으로 나옵니다.

그래서 결국 동적 할당 대신 정적 할당을 이용하기로 했습니다.

```
hs11015@DESKTOP-QM04TOL: ~/ict1
#include <stdio.h>

int Fibo(int n)
{
        //printf("func call param %d ₩n", n);  // 코드가 몇 번 반복됐는지 알아보기 위함
        static int arr[100];                    // 배열의 정적 할당을 이용(단, 이렇게 하면 100까지만 할당 가능)
        if (n ==1)              // 1을 입력하면 0, 2를 입력하면 1 출력.
        {
                arr[n] = 0;
                return arr[n];
        }

        else if ((n == 2) || (n == 3))
        {
                arr[n] = 1;
                return arr[n];
        }

        else
        {
                if (arr[n] > 0)                 // arr[100]을 함으로써 100개의 공간이 0으로 모두 할당됨.
                        return arr[n];          //따라서 0이 아닌 n번째 값이 배열 arr[n]에 저저장되어 있다면 그걸 return함.

                else
                {
                        arr[n] = Fibo(n-1) + Fibo(n-2);
                        return arr[n];  // n번째 값이 배열 arr[n]에 저장되어있지 않다면 계산해서 저장 후 return
                }
        }
}

int main(void)
{
        int num, cnt;

        printf("1과 100 사이의 정수를 입력하시면,₩n입력하신 정수번째까지의 피보나치 수열을 출력해드립니다.₩n(단, 피보나치 수열은 0부터 시작) : ");
        scanf("%d",&num);   //사용자에게 임의의 값을 입력받기
        printf("₩n가독성을 높이기 위해 한 줄에 숫자 5개씩 출력하겠습니다.₩n₩n");
-- INSERT --                                                                    17,2-9        Top
        for (int i=1; i<=num; i++)
        {
                printf("%d₩t", Fibo(i));
                cnt++;
                if ((cnt % 5) == 0)       // 5번째 수부터 5씩 끊어서 출력
                        printf("₩n");
        }

        return 0;
}

/*0 1 1 2 3 5 8 13 21 34
  55 89 ..... 이런 형식*/
```

이런 식으로 작성하면

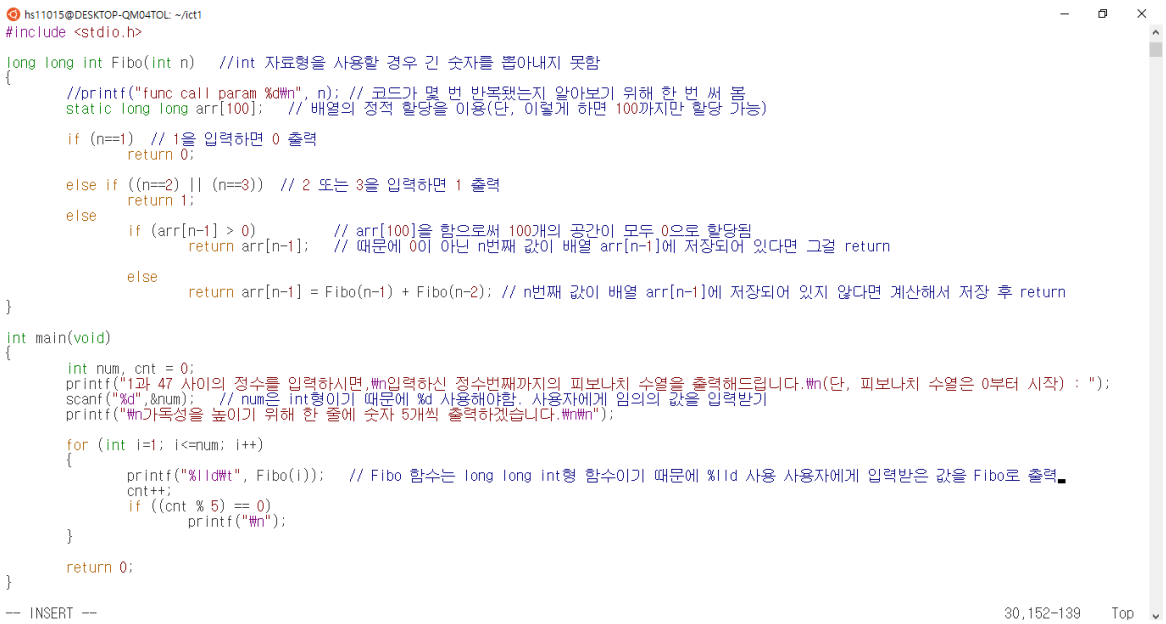
```
       |    ~~~
hs11015@DESKTOP-QM04TOL:~/ict1$ vi Fibo.c
hs11015@DESKTOP-QM04TOL:~/ict1$ gcc Fibo.c
hs11015@DESKTOP-QM04TOL:~/ict1$ ./a.out
1과 100 사이의 정수를 입력하시면,
입력하신 정수번째까지의 피보나치 수열을 출력해드립니다.
(단, 피보나치 수열은 0부터 시작) : 100

가독성을 높이기 위해 한 줄에 숫자 5개씩 출력하겠습니다.

0       1       1       2       3
5       8       13      21      34
55      89      144     233     377
610     987     1597    2584    4181
6765    10946   17711   28657   46368
75025   121393  196418  317811  514229
832040  1346269 2178309 3524578 5702887
9227465 14930352        24157817        39088169        63245986
102334155       165580141       267914296       433494437       701408733
1134903170      1836311903      -1323752223     512559680       -811192543
-298632863      -1109825406     -1408458269     1776683621      368225352
2144908973      -1781832971     363076002       -1418756969     -1055680967
1820529360      764848393       -1709589543     -944741150      1640636603
695895453       -1958435240     -1262539787     1073992269      -188547518
885444751       696897233       1582341984      -2015728079     -433386095
1845853122      1412467027      -1036647147     375819880       -660827267
-285007387      -945834654      -1230842041     2118290601      887448560
-1289228135     -401779575      -1691007710     -2092787285     511172301
-1581614984     -1070442683     1642909629      572466946       -2079590721
-1507123775     708252800       -798870975      -90618175       -889489150
hs11015@DESKTOP-QM04TOL:~/ict1$
```

이런 결과를 얻게 되는데 피보나치 수열 48번째부터는 갑자기 정상적인 값이 도출되지 않았습니다

왜 이럴까 고민을 해 본 결과 int형 상수라서 일정 범위 밖으로 벗어나면 출력이 되지 않는 것 같았습니다. 그래서 int 대신 long long 자료형을 써서 코드를 작성했습니다.



```
hs11015@DESKTOP-QM04TOL: ~/ict1
#include <stdio.h>

long long int Fibo(int n)   //int 자료형을 사용할 경우 긴 숫자를 못아내지 못함
{
        //printf("func call param %d₩n", n); // 코드가 몇 번 반복됐는지 알아보기 위해 한 번 써 봄
        static long long arr[100];   // 배열의 정적 할당을 이용(단, 이렇게 하면 100까지만 할당 가능)

        if (n==1)  // 1을 입력하면 0 출력
                return 0;

        else if ((n==2) || (n==3))  // 2 또는 3을 입력하면 1 출력
                return 1;
        else
                if (arr[n-1] > 0)          // arr[100]을 함으로써 100개의 공간이 모두 0으로 할당됨
                        return arr[n-1];   // 때문에 0이 아닌 n번째 값이 배열 arr[n-1]에 저장되어 있다면 그걸 return

                else
                        return arr[n-1] = Fibo(n-1) + Fibo(n-2); // n번째 값이 배열 arr[n-1]에 저장되어 있지 않다면 계산해서 저장 후 return
}

int main(void)
{
        int num, cnt = 0;
        printf("1과 47 사이의 정수를 입력하시면,₩n입력하신 정수번째까지의 피보나치 수열을 출력해드립니다.₩n(단, 피보나치 수열은 0부터 시작) : ");
        scanf("%d",&num);   // num은 int형이기 때문에 %d 사용해야함. 사용자에게 임의의 값을 입력받기
        printf("₩n가독성을 높이기 위해 한 줄에 숫자 5개씩 출력하겠습니다.₩n₩n");

        for (int i=1; i<=num; i++)
        {
                printf("%lld₩t", Fibo(i));   // Fibo 함수는 long long int형 함수이기 때문에 %lld 사용 사용자에게 입력받은 값을 Fibo로 출력
                cnt++;
                if ((cnt % 5) == 0)
                        printf("₩n");
        }

        return 0;
}
-- INSERT --                                                                    30,152-139    Top
```

```
hs11015@DESKTOP-QM04TOL:~/ict1$ vi Fibo.c
hs11015@DESKTOP-QM04TOL:~/ict1$ gcc Fibo.c
hs11015@DESKTOP-QM04TOL:~/ict1$ ./a.out
1과 47 사이의 정수를 입력하시면,
입력하신 정수번째까지의 피보나치 수열을 출력해드립니다.
(단, 피보나치 수열은 0부터 시작) : 100

가독성을 높이기 위해 한 줄에 숫자 5개씩 출력하겠습니다.

0       1       1       2       3
5       8       13      21      34
55      89      144     233     377
610     987     1597    2584    4181
6765    10946   17711   28657   46368
75025   121393  196418  317811  514229
832040  1346269 2178309 3524578 5702887
9227465 14930352        24157817        39088169        63245986
102334155       165580141       267914296       433494437       701408733
1134903170      1836311903      2971215073      4807526976      7778742049
12586269025     20365011074     32951280099     53316291173     86267571272
139583862445    225851433717    365435296162    591286729879    956722026041
1548008755920   2504730781961   4052739537881   6557470319842   10610209857723
17167680177565  27777890035288  44945570212853  72723460248141  117669030460994
190392490709135 308061521170129 498454011879264 806515533049393 1304969544928657
2111485077978050        3416454622906707        5527939700884757        8944394323791464        14472334024676221
23416728348467685       37889062373143906       61305790721611591       99194853094755497       160500643816367088
259695496911122585      420196140727489673      679891637638612258      1100087778366101931     1779979416004714189
```

그 결과 이렇게 값을 출력할 수 있게 되었습니다. 그럼에도 93번째 피보나치 수열 이후의 값은 나오지 않았습니다. 제가 알고 있는 숫자 자료형 중 숫자를 최대한으로 입력할 수 있는 자료형은 long long이기 때문에 이 이상 출력할 수 없다고 판단했습니다.

따라서 이게 완성된 코드 입니다.

hs11015@DESKTOP-QM04TOL: ~/ict1

```c
#include <stdio.h>

long long int Fibo(int n)   //int 자료형을 사용할 경우 긴 숫자를 뽑아내지 못함
{
        //printf("func call param %d₩n", n); // 코드가 몇 번 반복됐는지 알아보기 위해 한 번 써 봄
        static long long arr[100];   // 배열의 정적 할당을 이용(단, 이렇게 하면 100까지만 할당 가능)

        if (n==1)  // 1을 입력하면 0 출력
                return 0;

        else if ((n==2) || (n==3))  // 2 또는 3을 입력하면 1 출력
                return 1;
        else
                if (arr[n-1] > 0)          // arr[100]을 함으로써 100개의 공간이 모두 0으로 할당됨
                        return arr[n-1];   // 때문에 0이 아닌 n번째 값이 배열 arr[n-1]에 저장되어 있다면 그걸 return

                else
                        return arr[n-1] = Fibo(n-1) + Fibo(n-2); // n번째 값이 배열 arr[n-1]에 저장되어 있지 않다면 계산해서 저장 후 return
}

int main(void)
{
        int num, cnt = 0;
        printf("1과 93_사이의 정수를 입력하시면,₩n입력하신 정수번째까지의 피보나치 수열을 출력해드립니다.₩n(단, 피보나치 수열은 0부터 시작) : ");
        scanf("%d",&num);   // num은 int형이기 때문에 %d 사용해야함. 사용자에게 임의의 값을 입력받기
        printf("₩n가독성을 높이기 위해 한 줄에 숫자 5개씩 출력하겠습니다.₩n₩n");

        for (int i=1; i<=num; i++)
        {
                printf("%lld₩t", Fibo(i));   // Fibo 함수는 long long int형 함수이기 때문에 %lld 사용 사용자에게 입력받은 값을 Fibo로 출력
                cnt++;
                if ((cnt % 5) == 0)
                        printf("₩n");
        }

        return 0;
}
```

-- INSERT -- 24,17-23 Top

3. 자신의 코드가 O(n)임을 증명하기 위해 시간복잡도 계산 (중간 과정 포함)

계산 과정 이전에 printf("func call param %d₩n", n);을 사용해 O(n)임을 보이겠습니다.

```
hs11015@DESKTOP-QMO4TOL:~/ict1$ vi Fibo.c
hs11015@DESKTOP-QMO4TOL:~/ict1$ gcc Fibo.c
hs11015@DESKTOP-QMO4TOL:~/ict1$ ./a.out
1과 93 사이의 정수를 입력하시면,
입력하신 정수번째까지의 피보나치 수열을 출력해드립니다.
(단, 피보나치 수열은 0부터 시작) : 50

가독성을 높이기 위해 한 줄에 숫자 5개씩 출력하겠습니다.

func call param 1
0       func call param 2
1       func call param 3
1       func call param 4
func call param 3
func call param 2
2       func call param 5
func call param 4
func call param 3
3
func call param 6
func call param 5
func call param 4
5       func call param 7
func call param 6
func call param 5
8       func call param 8
func call param 7
func call param 6
13      func call param 9
func call param 8
func call param 7
21      func call param 10
func call param 9
func call param 8
34
func call param 11
func call param 10
func call param 9
55      func call param 12
func call param 11
func call param 10
89      func call param 13
func call param 12
func call param 11
144     func call param 14
func call param 13
func call param 12
233     func call param 15
func call param 14
func call param 13
377
func call param 16
func call param 15
func call param 14
610     func call param 17
func call param 16
func call param 15
987     func call param 18
func call param 17
func call param 16
1597    func call param 19
func call param 18
func call param 17
2584    func call param 20
func call param 19
func call param 18
4181
func call param 21
func call param 20
func call param 19
6765    func call param 22
func call param 21
func call param 20
10946   func call param 23
func call param 22
func call param 21
17711   func call param 24
func call param 23
func call param 22
28657   func call param 25
func call param 24
func call param 23
46368
func call param 26
func call param 25
func call param 24
75025   func call param 27
func call param 26
func call param 25
121393  func call param 28
func call param 27
func call param 26
196418  func call param 29
func call param 28
func call param 27
317811  func call param 30
func call param 29
func call param 28
514229
func call param 31
func call param 30
func call param 29
832040  func call param 32
func call param 31
func call param 30
1346269 func call param 33
func call param 32
func call param 31
2178309 func call param 34
func call param 33
func call param 32
3524578 func call param 35
func call param 34
func call param 33
5702887
func call param 36
func call param 35
func call param 34
9227465 func call param 37
func call param 36
func call param 35
14930352        func call param 38
func call param 37
func call param 36
24157817        func call param 39
func call param 38
func call param 37
39088169        func call param 40
func call param 39
func call param 38
63245986
func call param 41
func call param 40
func call param 39
102334155       func call param 42
func call param 41
func call param 40
165580141       func call param 43
func call param 42
func call param 41
267914296       func call param 44
267914296       func call param 44
func call param 43
func call param 42
433494437       func call param 45
func call param 44
func call param 43
701408733
func call param 46
func call param 45
func call param 44
1134903170      func call param 47
func call param 46
func call param 45
1836311903      func call param 48
func call param 47
func call param 46
2971215073      func call param 49
func call param 48
func call param 47
4807526976      func call param 50
func call param 49
func call param 48
7778742049
```

이런 식으로 3번씩 반복되어 실행되기 때문에 T(n) = 3n+@ 꼴로 나옵니다. 3n을 빅-오로 바꿀 때는 최고차항의 상수계수는 날려버림으로 O(n)이 됩니다.