

알고리즘 - 4주차 - 재귀+

201902694 - 박재우

14	201902694	7	1784	78 3 tries	332 4 tries	699 5 tries	495 1 try
----	-----------	---	------	---------------	----------------	----------------	--------------

문제 1

a. 문제 / 목표

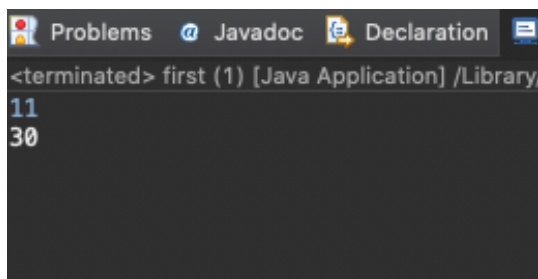
첫달에 토끼 한쌍 두달후 새끼 한마리->n 달 토끼 수는?

b. 해결 방법 (주요 소스코드 첨부. 스크린샷 or 코드 CV)

```
public class first {  
    public static int fibo(int n) {  
        if (n<=2)  
            return 2;  
        return fibo(n-1) + fibo((n-2))/2;  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        System.out.print(fibo(n));  
    }  
}
```

2 달까지의 토끼수는 2로 동결 이후 n 번째 달 2 달전에 두쌍이 한마리의 새끼를 낳고 또한 한달 전 토끼수는 유지. 따라서 이를 재귀로 구현하면 된다.

c. 결과 (입력, 출력 결과)



```
Problems Javadoc Declaration  
<terminated> first (1) [Java Application] /Library  
11  
30
```

1. 시간복잡도 : $T(n) = T(n-1) + T(n-2) + 1$ for $(n \geq 2)$

자신의 생각:

피보나치 수열을 구할때와 비슷한 구조라고 생각한다. 다만 수가 증가할때까지 2달이라는 시간이 존재하고 또한 토끼 2마리가 존재해야 수가 1 증가하기에 여러모로 조건을 충족해야하는 까다로운 문제라고 생각한다.

2. 질문 : 문제를 해결했기에 딱히 존재하지 않는다.

3. 느낀점 :

1번문제부터 해결하지 못하는 것을 보면 아직 공부가 더 필요하다는 것을 깨달았다.

또한 재귀에 대해서도 사고가 제대로 이루어지지 않는것을 보아 재귀에 대해서 많은 공부가 필요한 것 같다.

문제 2

a. 문제 / 목표

가위바위보에 따라 x,y,z 칸 이동. 거리가 n 일때 가장 빨리만나는 게임횟수

b. 해결 방법 (주요 소스코드 첨부. 스크린샷 or 코드 CV)

```
public class second {
    static int flag = 0;
    public static int check(int n, int[] xyz, int sum, int num) {
        if (n == sum)
            return num;
        else if (n < sum)
            return 99999;
        int m = 99999;
        for (int i = 0; i < 3; i++) {
            m = Math.min(m, check(n, xyz, sum + xyz[i], num + 1));
        }
        return m;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] xyz = new int[3];
        Arrays.sort(xyz);
        for (int i = 0; i < 3; i++)
            xyz[i] = sc.nextInt();

        int n = sc.nextInt();
        int c;
        if ((c = check(n, xyz, 0, 0)) == 99999)
            System.out.print(-1);
        else
            System.out.print(c);
    }
}
```

X, Y, Z 를 더해가면서 그 거리가 나올때까지 탐색.

나온 결과중에서 가장 횟수가 적은 경우를 min 함수를 통해서 출력

c. 결과 (입력, 출력 결과)

Problems Javadoc Declaration Console X

```
<terminated> second (1) [Java Application] /Library/Java/JavaVirtual  
3 5 22  
80  
7
```

시간복잡도 : 각 자리수 마다 나머지 연산해서 찾아서 $n/2$ 이다.

자신의 생각:

문제를 해결하는데 있어서 상당히 오래 걸렸다. 문제 해결을 위한 알고리즘방법이 제대로 떠오르는데까지 상당히 오래걸렸으며 이방법또한 제대로 출제의도에 맞게 재귀를 제대로 사용하여 푼것 같지는 않다.

질문 :

분명 제대로 재귀를 사용해서 문제를 해결하는 방법이 있다고 들었다.
내가 푼 방식 말고 다른 알고리즘을 사용해서 푸는 방법이 궁금하다.

느낀점 :

재귀에 대한 사고가 제대로 이루어지지 않다 보니 제대로 푼 것 같지가 않다.
이러한 방법 이외에 조금 더 간단한 방법이 있다는 것을 알지만 그것을 스스로 알아낼 수 없다는 것이 아쉬웠다.

문제 3

a. 문제 / 목표

가위바위보에 따라 x, y, z 칸 이동. 거리가 n 일때 가장 빨리만나는 게임횟수

b. 해결 방법 (주요 소스코드 첨부, 스크린샷 or 코드 CV)

```
package algorithm.week4;

import java.util.Arrays;

public class third {
    static int check = 0;
    static char[] save = new char[9999];

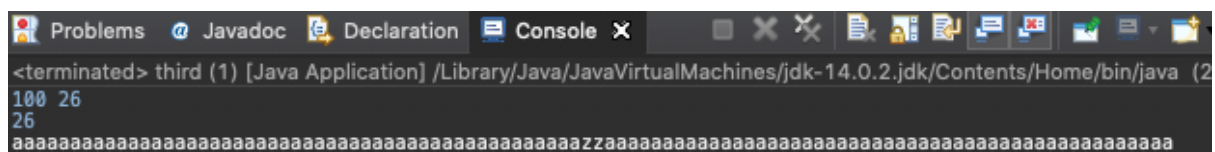
    public static void test(int n, int m, char[] a, int num) {
        if(n == 0 && check < num) {
            for(int i=0; i<m; i++) {
                a[0] +=1;
                check+=1;
                if(check == num) {
                    for(int j=0; j<a.length; j++)
                        save[j] = a[j];
                }
            }
        }
        else {
            for(int i=0; i<m && check< num; i++) {
                if(n>0 && a[n-1] == 'a'+(m-1))
                    a[n-1] = 'a'-1;
                a[n] +=1;
                test(n-1, m, a, num);
            }
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] nm = new int[2];
        for(int i = 0; i < 2; i++)
            nm[i] = sc.nextInt();
        int num = sc.nextInt();
        char[] set = new char[nm[0]+1];
        int length;
        if(nm[0]%2 == 0)
            length = nm[0]/2;
        else
            length = nm[0]/2+1;
        char[] a = new char[length];
        for(int i=0; i<a.length; i++)
            a[i] = 'a'-1;
        test(length-1, nm[1], a, num);
        if(nm[0]%2 == 0)
            length -=1;
        for(int k=0; k<a.length; k++)
            set[k] = save[length-k];
        for(int j=0; j<a.length; j++)
            set[a.length+j] = save[j];
        String str = new String(set);
        System.out.print(str.trim());
    }
}
```

입력된 길이의 반중 오른쪽의 결과만 우선 계산한다. 그 길이만큼 배열을 생성한 뒤에 test 재귀 함수를 통해 a[0]부터 m 종류의 알파벳을 순서대로 증가시킨뒤 M 까지 증가한뒤에 a[1]으로 이동후 증가 다시 a[0]를 a 부터 증가 시켜가면서 이를 배열의 끝까지 x 번째 문자열이 나올때까지 재귀를 통해 작동시킨다.

그후 규칙에 맞게 a 배열을 거꾸로 출력후 정방향으로 출력하면 끝이 난다.

c. 결과 (입력, 출력 결과)



시간복잡도 : $T(n)=T(n-1)$ for $(n \geq x)$
 자신의 생각 및 느낀점 :

실습문제 4문제중 가장 까다로운 문제였다고 생각한다.
또한 재귀의 사용뿐만이 아니라 홀수

문제 4

a. 문제 / 목표

세 가지 괄호를 각각 l,m,n개 배치할 때, 배치 규칙에 맞는 경우는 몇 가지인지 구하는 문제

b. 해결 방법 (주요 소스코드 첨부. 스크린샷 or 코드 CV)

```
public class fourth {  
  
    public static int fac(int n) {  
        if(n == 1)  
            return 1;  
        return n*fac(n-1);  
    }  
  
    public static int cal2(int n, int n2) {  
        if(n == n2)  
            return n2;  
        return n*cal2(n-1,n2);  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int[] abc = new int[3];  
        for(int i = 0; i < 3; i++)  
            abc[i] = sc.nextInt();  
        int n = abc[0]+abc[1]+abc[2];  
        System.out.print(cal2(2*n,n2)/(fac(abc[0])*fac(abc[1])*fac(abc[2])));  
    }  
}
```

카탈란 공식을 응용하여 푸는 문제.

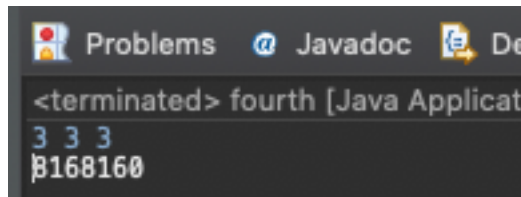
$$c_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$

에서 n!은 같은 괄호의 카운팅을 제거하는 방법이므로 이를 제외하고 식을 구현한다.

단, 2n!을 재귀를 사용해서 구하면 오버플로우가 발생함으로 (2n)!/(n+1)!은

약분하여 계산한뒤에 이를 각각의 괄호갯수의 팩토리얼로 나누면 원하는 결과값이 나온다.

c. 결과 (입력, 출력 결과)



시간복잡도 : $T(n) = T(n-1)$ 이다

자신의 생각:

생각보다 이 문제는 앞선 문제보다 쉬웠던것 같다.

대칭쌍의 개수를 구하는 공식이 카탈란 공식인데 이를 활용하여 변형하면 원하는 값을 쉽게 얻을 수 있었다.

질문 : 딱히 존재하지 않는다.

느낀점 :

알고리즘을 푸는데 있어서 기존 수학실력이 많이 필요하다는 것을 다시한번 느꼈다.

4번문제 또한 카탈란 공식을 알고 있으면 금방 쉽게 해결 할 수 있는 문제로 중요 알고리즘 방식 이외에도 수학공부 역시 중요한것 같다.