자료구조 hw1

하노이 타워 만들기

20231604 정유연

1. Write a recursive function in C.

1) c code

```c
void hanoi(int n, char first, char second, char third, int* cnt) {
    if (n == 1) {
        printf("no.1 disk moves from %c to %c.₩n", first, third);
        (*cnt)++;
    }
    else {
        hanoi(n - 1, first, third, second, cnt);
        printf("no.%d disk moves from %c to %c.₩n", n, first, third);
        (*cnt)++;
        hanoi(n - 1, second, first, third, cnt);
    }
}

int main() {
    int n, cnt = 0;
    clock_t start, end;
    double cpu_time;

    printf("Put the number of disks:");
    scanf("%d", &n);
    hanoi(n, 'A', 'B', 'C', &cnt);

    printf("The number of moves needed is %d₩n", cnt);

    return 0;
}
```

2) prompt

```
Put the number of disks:3
no.1 disk moves from A to C.
no.2 disk moves from A to B.
no.1 disk moves from C to B.
no.3 disk moves from A to C.
no.1 disk moves from B to A.
no.2 disk moves from B to C.
no.1 disk moves from A to C.
The number of moves needed is 7
```

2. Determine the space complexity.

공간 복잡도는 프로그램을 실행시키는데 필요한 메모리의 양이므로 재귀함수를 몇 번이나 호출하는지가 공간 복잡도에 영향을 미친다. 위 코드에서 우리는 입력 받은 n만큼 재귀함수를 호출하므로 스택에는 hanoi 함수가 n부터 1까지 n개가 쌓인다. 이때 하노이 함수의 매개변수는 총 5개(constant)이고 동적할당을 하지 않으므로 function call만 공간 복잡도에 영향을 준다.

따라서 하노이타워의 공간 복잡도는 S(n)이다.

3. Determine the value of count when the function ends, and show the time complexity using O.

1의 코드에서 봤듯이 the number of execution을 세기 위해 disk를 옮길 때마다 cnt를 1씩 추가했다. 예를 들어 n이 4일 때 function은 3,1,3를 호출하고 3에서 2,1,2를, 2에서 1,1,1을 호출한다. 따라서 hanoi(n)는 2*hanoi(n-1)+1 를 재귀적으로 호출하므로 hanoi(n)의 number of execution은 2^n-1 이다.

즉 하노이타워의 시간 복잡도는 O(2^n)이다.

4. Measure the performance with n=3, 10, 15, 20, and summarize your findings.

1) c code

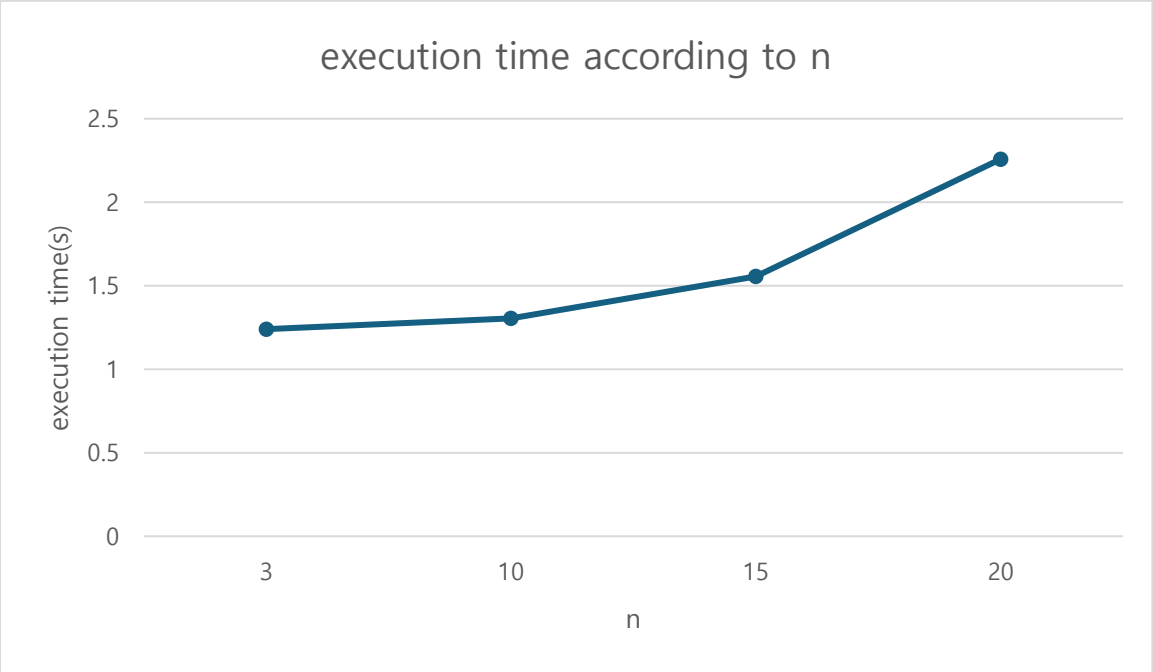
```c
int main() {
    clock_t start=clock(), end;
    int n, cnt = 0;
    double cpu_time;
        cnt = 0;
        printf("Put the number of disks:");
        scanf("%d", &n);

        hanoi(n, 'A', 'B', 'C', &cnt);
        end = clock();
        //printf("The number of moves needed is %d₩n", cnt);
        cpu_time = ((double)(end - start)) / CLOCKS_PER_SEC;

        printf("For n = %d, Execution time: %.10f seconds, Number of executions: %d₩n", n, cpu_time, cnt);
    return 0;
}
```

2) result

n	execution time
3	1.240000
10	1.305000
15	1.556000
20	2.258000



n이 커질수록 execution time이 기하급수적으로 커지는 것을 관찰하였다.

위 코드로 n에 따른 number of execution이 3에서 찾은 2^n-1와 일치함을 확인했다.