



## 3 / 29

- 과제 내용
  - 퀵 소트, 머지 소트, 하노이
- 하노이 타워의 점근적 분석
  - 반복 대치
    - $T(n) = 2 * T(n - 1) + 1$
    - $= 2 * (2 * T(n - 2) + 1) + 1 = 2^2 * T(n - 2) + 1 + 2$
    - ...
    - $= 2^{(n-1)} * T(1) + (1 + 2 + 4 + ... + 2^{n-2})$
    - $= (2^{(n-1)}) / (2 - 1) = 2^n - 1$
    - $2^n - 1 < 2 * 2^n \rightarrow O(2^n)$
    - $2^n - 1 > 1 / 2 * 2^n \rightarrow \Omega(2^n)$
    - 세타( $2^n$ )
  - 추정후 증명

# 추정후 증명

$$T(n) = 2T(n-1) + 1$$

$$T(1) = 1$$

→  $T(n) = \Theta(2^n)$ 을 증명

경계조건 :

$$T(1) = 2T(0)+1 = 1 \leq 2 * 2^1, \geq 1/3 * 2^1$$

$$T(2) = 2T(1)+1 = 3 \leq 2 * 2^2, \geq 1 * 2^2$$

가정 :  $n-1$ 일때,  $T(n-1) \leq c * 2^{n-1}$  또는  
 $T(n-1) \geq c * 2^{n-1}$  인  $c$ 가 존재한다고 가정

전개1 :  $n$  일때,  $O(2^n)$  증명

$$\begin{aligned} T(n) &= 2T(n-1)+1 \\ &\leq 2 * c * 2^{n-1} + 1 \\ &= c * 2^n + 1 \\ &\leq k * 2^n, \text{ if } c < k \Rightarrow O(2^n) \end{aligned}$$

전개2 :  $n$  일때,  $\Omega(2^n)$  증명 : 마찬가지로..  
 $\geq k * n, \text{ if } c > k \Rightarrow \Omega(2^n)$

→  $\Theta(2^n)$

◦ 하노이 타워를 iteration

■ Recursion 처리 과정의 완벽한 이해

```
public static void hanoi(n, i, j){
    if(n == 1) move(1, i, j);
    else{
        hanoi(n - 1, i, k);
        move(n, i, j);
        hanoi(n - 1, k, j);
    }
}
```



- Array vs ArrayList
  - 정적 배열과 동적 배열의 차이
- ArrayList vs LinkedList
  - 비교 관점
  - Add, Remove, Search
- ArrayList 구현하기
  - 데이터 : int (non-generic)
  - Methods
    - String get(int index) : return i-th data;
    - int search(int data) : return index of given data
    - void add(int data) : save data at the end
    - void add(int index, int data) : save data at i-th position
    - String remove(int index) : remove & return i-th data
    - int remove(int data) : remove given data & return the position

- int sizeOf() : return number of data added
- int arrSize() : return current array max size
- String toString() : return output string
- 상황별 성능 추정
  - String get(int index) : return i-th data;
    - Index로 찾기 때문에  $O(1)$ 만큼의 시간이 들음
  - int search(int data) : return index of given data
    - for문을 이용해 찾으므로  $O(N)$ 만큼의 시간이 들음
  - void add(int data) : save data at the end
    - 값을 추가할 때,  $O(1)$
    - 배열을 추가해야 될 경우 복사, 생성, 붙여넣기  $O(2N) \rightarrow O(N)$
  - void add(int index, int data) : save data at i-th position
    - 중간에 값을 추가할 경우, 배열을 옮겨야 하기 때문에,  $O(N)$
  - String remove(int index) : remove & return i-th data
    - 삭제하고, 배열을 다시 이어줘야 하기 때문에  $O(N)$
  - int remove(int data) : remove given data & return the position
    - 삭제하고, 배열을 다시 이어줘야 하기 때문에  $O(N)$
  - int sizeOf() : return number of data added
    - 아이템의 갯수를 별도로 세기 때문에  $O(1)$
  - int arrSize() : return current array max size
    - 길이를 별도로 세기 때문에  $O(1)$ ;
  - String toString() : return output string
    - 아이템을 출력하기위해 For문을 사용하기 때문에  $O(N)$