Data Structures

3. Recursion

3. Recursion

1. Recursion
2. Factorial
3. Binary Search
4. Permutation

Recursion

- A function is called by itself while running
- possible when the same task is repeated with exit ____ condition
- "Divide & Conquer" strategy
- Iterative function with if-else or while can be rewritten in recursion
- In compiler's view, recursive call is the same as another new function is called

- Advantages
  - Clear representation of algorithm
- Disadvantage
  - Frequent function call overhead

Factorial

$n! = 1 * 2 * \cdots * \underbrace{(n-1)}_{} * n$

$(n-1)!$

$f(4) = 4 * \underline{f(3)}$

$= 3 * f(2)$

- $n! = 1$ where $n = 0$
- $n! = n * (n - 1)!$ where $n \geq 1$

- Program 3.1

```
long factorial (int n)
{
    if (n == 0)
        return 1;
    else
        return factorial (n-1);
}
```

GCD

최대공약수

- Greatest Common Divisor

- gcd (x, y) = gcd (y, x % y)   if y > 0
- gcd (x, 0) = x

- ex) gcd (48, 8) = gcd (8, 48 % 8) = gcd (8, 0) = 8

- ex) gcd (128, 12) = 4

- Program 3.2

Binary Search

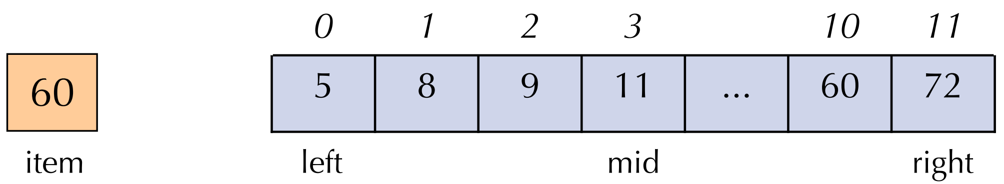
이진탐색

- Problem) search a specified key in a sorted list
  - S = [3, 7, 8, 11, 13, 19, 25]
  - No overlapping number
- Algorithm

```
while (there are any item to compare)
{
    find median position;
    if (item < median)
        change the range with left sub-list
    else if (item > median)
        change the range with right sub-list
    else
        return median position;
}
```

Binary Search

- Program 3.3 P54
  - Iterative Binary Search

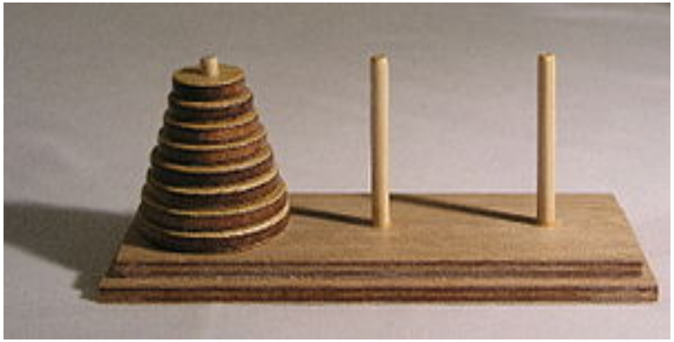


Recursion - Binary Search

- Program 3.4 "재귀적 이진탐색" P55
  - Recursive Binary Search

Hanoi Tower

- Goal of Problem
  - move some discs in a pole to another pole without changing the order of discs
- Condition
  1. Only single disc can be moved at a time
  2. A disc cannot be placed over smaller one



Hanoi Tower

- Algorithm
  - Assuming that there are three poles A, B, C, and all discs of pole A have to be moved to pole B

1. Move n - 1 discs from pole A to pole __C__ → B에 놓으려면 없애야 함
2. Move single disc from pole A to pole B
3. Move n - 1 discs from C to pole __B__

  - Step 1 and 3 should be solved recursively

- Program 3.5

Hanoi Tower P57

- # of movements
- time to move single disk : t(1)

  $t(1) = 1$
- time to move n disks : t(n)

  $$\begin{aligned} t(n) &= 2\ t(n-1) + 1 \\ &= 2\ (2\ t(n-2) + 1) + 1 \\ &= 2\ (2\ (2\ t(n-3) + 1) + 1\ ) + 1 = 2^3 * t(n-3) + 2^2 + 2^1 + 2^0 \\ &= \ldots. \\ &= 2^{n-1} + 2^{n-2} + \ldots + 2^1 + 2^0 \\ &= \underline{2^n-1} \end{aligned}$$

$t(n) = O(2^n)$ // Time complexity 시간복잡도

Big-O 표기법

기하급수적 증가

개당 1초로 가정

$\boxed{\text{17m , 12days , 34yr // n=10, 20, 30}}$

Permutation (순열)

- Find all possible permutations with n elements (n > 1)

- ex) For a set {a, b, c}, find all permutations
  - $P = \{(a, b, c), (a, c, b), (b, a, c), (b, c, a), (c, a, b), (c, b, a)\}$
  - ${}_3P_3 = 3 \times 2 \times 1 = 3! = 6$

Permutation

- Permutations of {a, b, c, d}
  - 24 permutations are divided into 4 groups
    a) (a, b, c, d), (a, b, d, c), .. => (a , Perm(b, c, d)) → 6
    b) (b, Perm(a, c, d)) → 6
    c) (c, Perm(a, b, d)) → 6
    d) (d, Perm(a, b, c)) → 6

    24

    재귀

  - *Perm(b, c, d)* can be solved in the same way
    - (b , Perm(c, d)), (c , Perm(b, d)), (d , Perm (b, c))
  - => a clue for recursive solution

Permutation

- char list[] = { 'a', 'b', 'c', 'd' };

- i : start index of range
- n : end index of range
- ex)
  - perm (list, 0, 3)
  - perm (list, 0, 2)

i	j		
0	*1*	*2*	*3*
'a'	'b'	'c'	'd'
b	a	c	d

P(1,3)

원소자리에게

독특하게 구현한코드

```c
void perm(char list[], int i, int n)
{
    int j;
    if (i == n)
    {
        for (j = 0; j <= n; j++)
            printf ("%c", list[j]);
        printf ("        ");
    }
    else
        for (j = i; j <= n; j++)
        {
            Swap (list[i], list[j]);
            perm (list, i+1, n);
            Swap (list[i], list[j]);
        }
}
```

→ a를제외한

↳ a,b 위치바꾸기.

Permutation

Perm(0, 3)	Perm(1, 3)	Perm(2, 3)	Perm(3, 3)
a (b c d)	a (b (c d)) a (c (b d)) a (d (c b))	a (b (c (d))) a (b (d (c))) a (c (b (d))) a (c (d (b))) a (d (c (b))) a (d (b (c)))	a, b, c, d a, b, d, c a, c, b, d a, c, d, b a, d, c, b a, d, b, c
b (a c d)	b (a (c d)) b (c (a d)) b (d (c a))	b (a (c (d))) b (a (d (c)))	b, a, c, d b, a, d, c
c (b a d)	c (b (a d))	c (b (a (d)))	c, b, a, d
d (b c a)	d (b (c a))	d (b (c (a)))	d, b, c, a

- Exercise 3.3, 3.4

연습문제 4번 (p60)

5개 문자 → <u>120개</u> (5!) 출력

확장된걸로

후보문제 中 하나