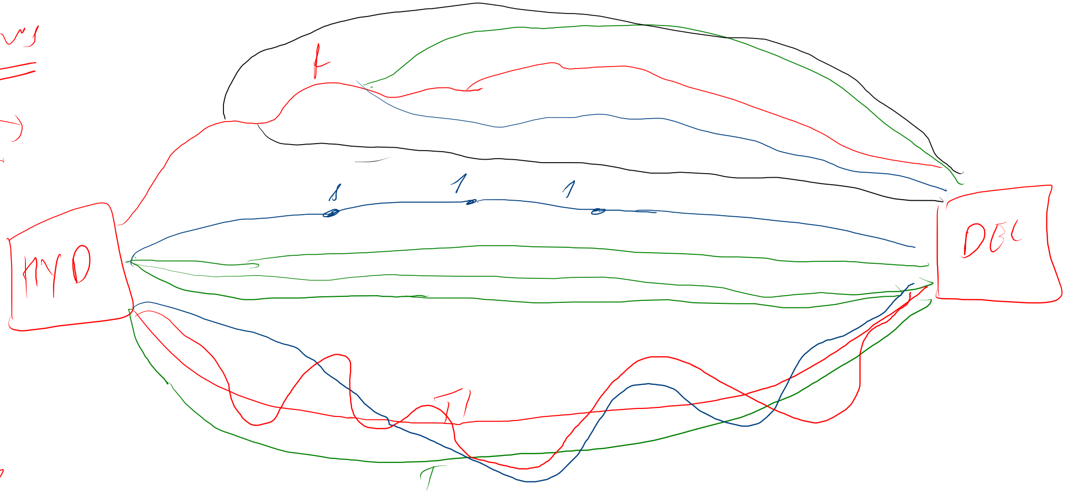
Constraints

Time dist



Recursion

graphs

DP

| | | | | | |
|---|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 | 60 |

$\alpha = a \times b \; (10 \times 20)$

$\beta = b \times c \; (20 \times 30)$

$\gamma = c \times d \; (30 \times 40)$

$\theta = d \times e \; (40 \times 50)$

$\lambda = e \times f \; (50 \times 60)$

$(\alpha = \beta)$

$((\alpha \cdot \beta) \cdot \gamma \cdot (\theta \cdot \lambda))$

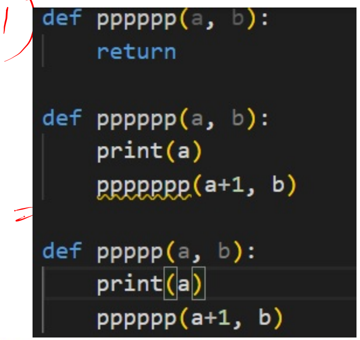
$((((\alpha \cdot \beta) \gamma) \alpha) \gamma)$

$(\alpha (\underline{\beta \cdot \gamma}) \cdot (\underline{\alpha \cdot \lambda}))$

$\alpha \cdot \beta \cdot \gamma \cdot \theta \cdot \lambda$

→ P, (small obvious case)

(0! = 1)

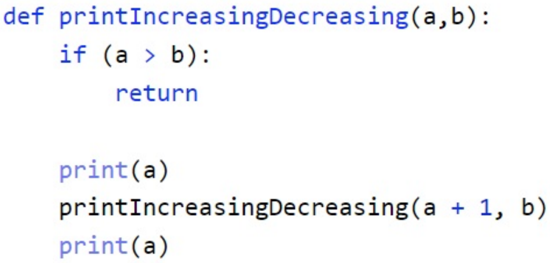


```
def pppppp(a, b):
    return

def pppppp(a, b):
    print(a)
    ppppppp(a+1, b)

def ppppp(a, b):
    print(a)
    pppppp(a+1, b)
```

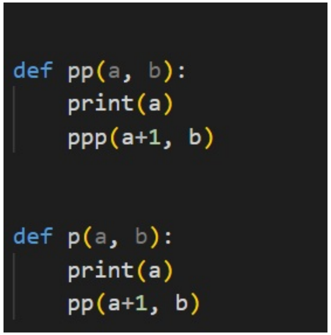
→ (P): (n-1)! =

task: (9, 6)

: (2, 8) → (3, 8)

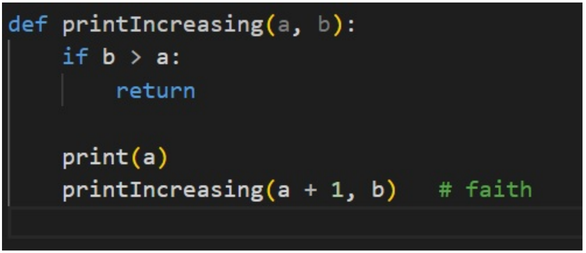


```
def printIncreasingDecreasing(a,b):
    if (a > b):
        return

    print(a)
    printIncreasingDecreasing(a + 1, b)
    print(a)
```

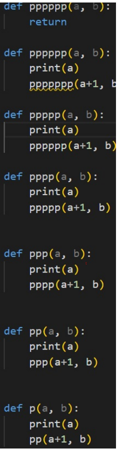


```
def pp(a, b):
    print(a)
    ppp(a+1, b)


def p(a, b):
    print(a)
    pp(a+1, b)
```

(



```
def printIncreasing(a, b):
    if b > a:
        return

    print(a)
    printIncreasing(a + 1, b)    # faith
```

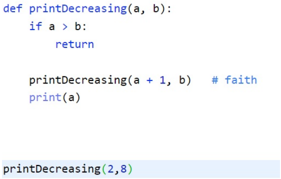


```
def pppppp(a, b):
    return

def pppppp(a, b):
    print(a)
    ppppppp(a+1, b

def ppppp(a, b):
    print(a)
    pppppp(a+1, b)

def pppp(a, b):
    print(a)
    ppppp(a+1, b)


def ppp(a, b):
    print(a)
    pppp(a+1, b)


def pp(a, b):
    print(a)
    ppp(a+1, b)


def p(a, b):
    print(a)
    pp(a+1, b)
```

(2,7)

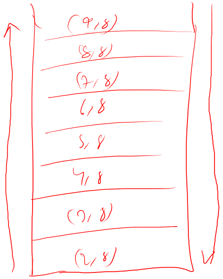
2
3
4
5
6
7

(2, 10)

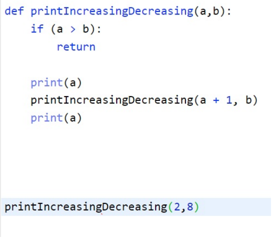


```
def printDecreasing(a, b):
    if a > b:
        return

    printDecreasing(a + 1, b)    # faith
    print(a)


printDecreasing(2,8)
```

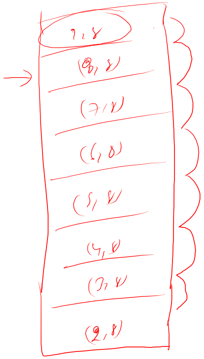
8
7
6
5
4
3
2



(2,8)



```
def printIncreasingDecreasing(a,b):
    if (a > b):
        return

    print(a)
    printIncreasingDecreasing(a + 1, b)
    print(a)



printIncreasingDecreasing(2,8)
```

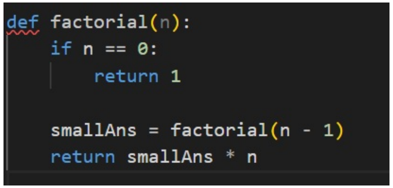


2
3
4
5
6
7
8
8
7
6
5
4
3
2

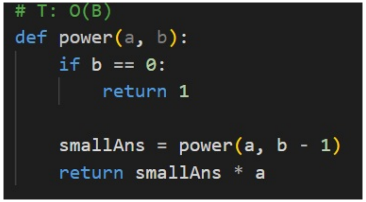
$n! = f(n-1)$

$24 \times 5$

$n = 5$

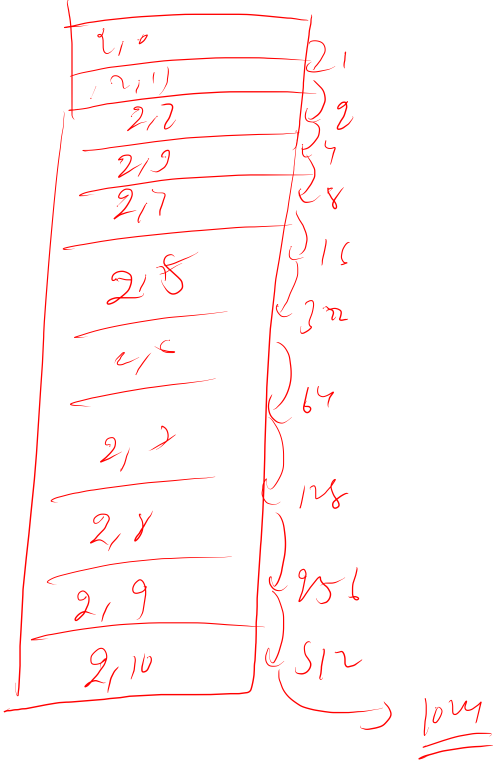


```
def factorial(n):
    if n == 0:
        return 1

    smallAns = factorial(n - 1)
    return smallAns * n
```



```
# T: O(B)
def power(a, b):
    if b == 0:
        return 1

    smallAns = power(a, b - 1)
    return smallAns * a
```

$(8) \times 6$



friend = pow(2,3) = $2^3$

pow(2,7) = (friend)(friend) = $2^3 \cdot 2^3 \cdot = 2^6$

pow(2,6) = (friend)(friend) = $2^3 \cdot 2^3 = 2^6$

$2^7 = (2^{7/2})\, 2^{(7/2)}$

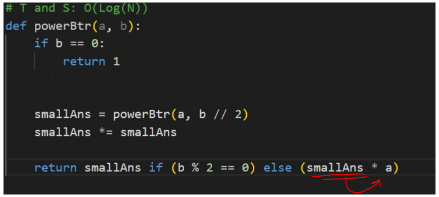
$= 2^{3.5} \cdot 2^{3.5}$

$= 2^3 \cdot 2^3 \cdot 2^{.5} \cdot 2^{.5}$

$= (2^3)(2^3) \cdot 2^{.5+.5}$

$= (7)(7) \cdot 2$

$= (7)(7)(9)$



```
# T and S: O(Log(N))
def powerBtr(a, b):
    if b == 0:
        return 1

    smallAns = powerBtr(a, b // 2)
    smallAns *= smallAns

    return smallAns if (b % 2 == 0) else (smallAns * a)
```

$6 = 10^6$

$2^{10^6}$

$\log_2 10^6$

$6 \log_2 10$

$\frac{6}{0.5} =$

$\frac{50}{5}$ 20

$6, \frac{6}{2}, \frac{6}{4}, \frac{6}{8}, \frac{6}{16}, \frac{6}{32}$ ----- 8, 4, 2, 1

$a_0 = 6$

$a_k = 1$

$r = 1/2$

$a_k = a_0 r^k$

$1 = 6 \frac{1}{2^k}$

$2^k = 6$

$\boxed{K = \log_2(6)}$

2,0

2,1

2,2

2,5

2,10

2,1

1 * 2

2 * 2

4 * 4 * 2

32 * 32

1024

$2^9 = (2^7)\, 2^7 * 2$

$2^8 = 2^5 \cdot 2^5$