$x^0 = 1$

pow(a, b)

dost (recursive call) = pow(a, b-1)

= $a^{(b-1)} * a$

= a^b

T(N)

```
public static int power_basic(int a, int b) {
                                    → O(1)
    int smallAns = power_basic(a, b - 1);  → T(N-1)
    return smallAns * a;  → O(1)
}
```

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

$T(N) = \lambda + T(N-1)$

$T(N-1) = \lambda + T(N-2)$

$T(N-2) = \lambda + T(N-3)$

$T(N-3) = \lambda + T(N-4)$

$T(1) = \lambda + T(0)$

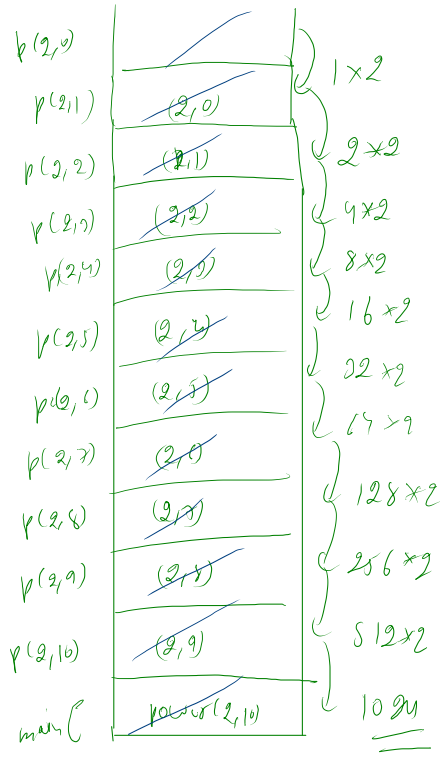
$T(N) = N\lambda + T(0)$

$T(N) = O(N)$

$T(N) \simeq O(N)$

2^{1000}

$2^{10^6} \simeq (10^6$ unit second)



pow(9, 6)

me → dost (Recursion)

small Ans = pow(a, 6/2)

(small Ans *= small Ans)

3^8 → me → dost → 3^4

$(3^4)(3^4) = 3^8$

$9/2 = 4$

3^9 → me → dost → $(3^4)(3^4)$ · 3^8

$3^9 = 3^{4.5} \cdot 3^{4.5}$

$= 3^4 \cdot 3^4 \cdot 3^{.5} \cdot 3^{.5}$

$= (3^4)(3^4) \cdot 3^1$

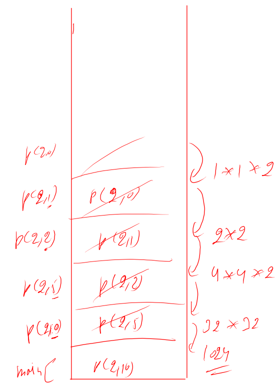
T(N)

```
public static int power_opti(int a, int b) {
    if (b == 0) {
        return 1;
    }

    int smallAns = power_opti(a, b - 1);
    smallAns *= smallAns;

    return b % 2 == 0 ? smallAns : smallAns * a;
}
```

→ 1, /2 → T(N/2), → (1), → (1)



$T(N) = 2 + T(N/2)$

$T(N) = \lambda + T(N/2)$

$T(N/2) = \lambda + T(N/4)$

$T(N/4) = \lambda + T(N/8)$

$T(1) = \lambda + T(0)$

$T(N) = K \cdot \lambda + T(0)$

$T(N) = \lambda(\log_2(N) + 1) + T(0)$

$T(N) = O(\log_2(N))$

$S(N) = O(\log_2(N))$

$N, N/2, N/4, N/8, N/16 \ldots\ldots 4, 2, 1$

$a_0 = N$

$a_K = 1$

$r = (1/2)$

$a_K = a_0 r^{K-1}$

$1 = N (\frac{1}{2})^{K-1}$

$2^{K-1} = N$

$\log_2 2^{K-1} = \log_2(N)$

$(K-1) \log_2(2) = \log_2(N)$

$K-1 = \log_2(N)$

$K = \log_2(N) + 1$