pow(x, n)

- binary exponentiation

x^n

$n \rightarrow \log_2 n + 1$

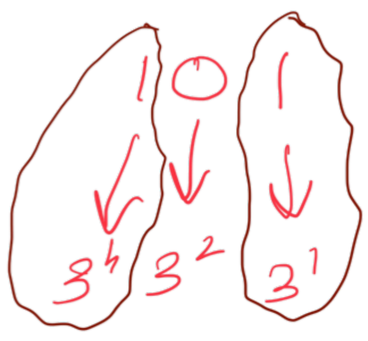
dec → binform

$n = 8 \Rightarrow 1000$

$\log_2 8 + 1 = 3 + 1 = 4$ digits

if we had to find 2^8 we will have to run the loop 8 times but if we used the binary form of 8 and somehow made it work then the loop will run for 4 times which is half the amount of loops needed

$x^n = 3^5$



ignore zeros and multiply the 1s

$\therefore 3^4 \times 3^1 = 3^5$

$x = 3^1 \cdot 3^1$

$x^2 = 3^2 \cdot 32$

$x^4 = 3^4 \cdot 3^4$ keep on taking squares in the loop so we don't waste time

$x^8 = 3^8$

$\log n$ iterations

already calculate lots range

Dry run

binform = n (101), ans = 1, x = 3^1

1 ① ans = ans * [x] = 3^1

x = x * x = 9

0 ② x = x^2 = 81

3 steps because $\log_2 n + 1 = \log_2 8 + 1 = 3$

1 ③ ans = ans * x = $3 \times 81 = 243 = 3^5$

x = x^2 = 81^2

ignore when zero cases don't update ans only x

Code:

```
binform = n, ans = 1, x
if (n < 0) {
    x = 1/x;
    binform = -binform;
}
while (binform > 0) {
    if (binform % 2 == 1) {        // finds last digit
        ans *= x                   // if last digit is 1 then go in loop
    }                              // update ans
    x *= x                         // square the existing x
    binform /= 2;                  // go to the next digit (1 or 0)
}
```

time complexity : $O(\log n)$

Corner cases

$n = 0 \Rightarrow 1$

$x = 0 \Rightarrow 0$

$x = 1 \Rightarrow 1$

$x = -1$ & pow even $\Rightarrow +1$

$x = -1$ & pow odd $\Rightarrow -1$

```
if (n == 0)  return 1.0;
if (x == 0)  return 0.0;
if (x == 1)  return 1.0;
if (x == -1 && n%==0) return 1.0;
if (x == -1 && n%!=0) return -1.0;
```