Pow(x, n)

$$b^x = \underbrace{b \times \cdots \times b}_{x \text{ times}}$$

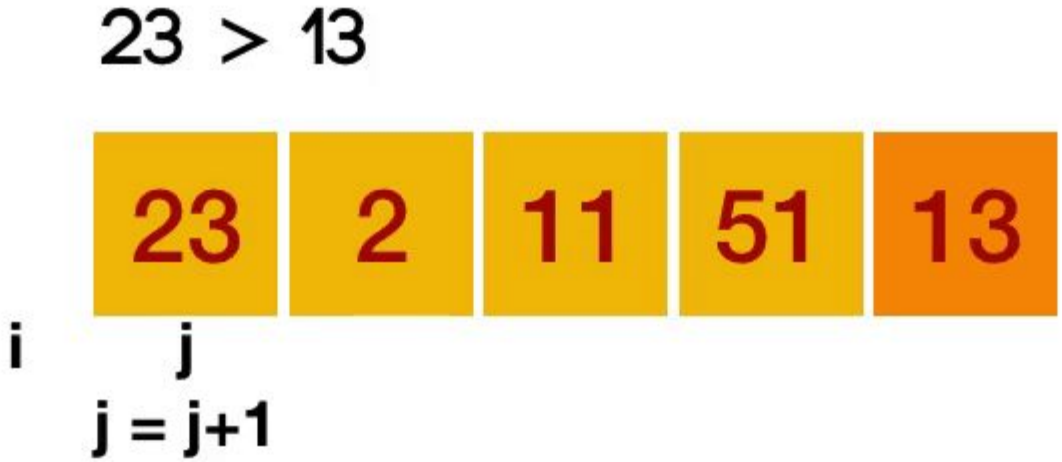
b = base of the logarithm

x = exponent

Wenbo

Last time's follow up

Quick Sort (Still Divide and Conquer)



aaaaaaaaaa|xxxxxxxxxxxxxxxx|bbbbbbbbbb

i j

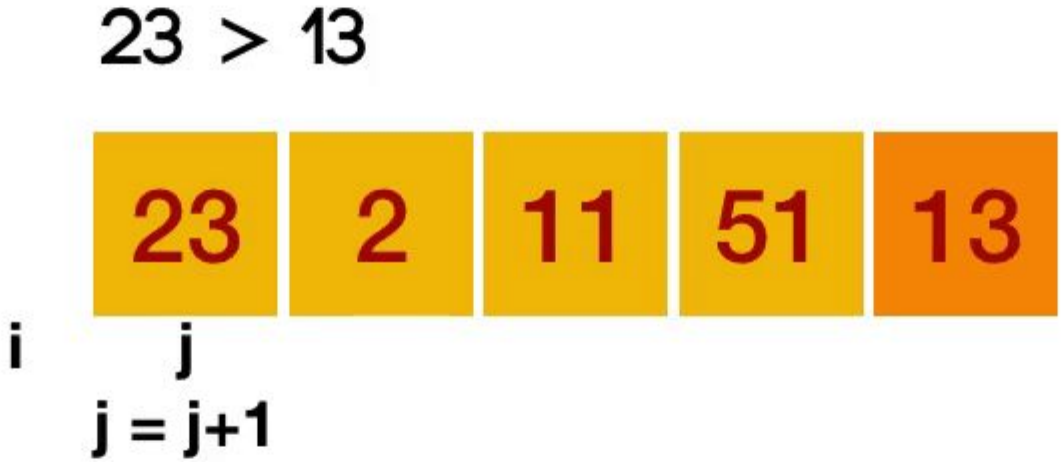
(<=pivot) (undiscovered) (>pivot)

aaaaaaaaaaaaaaaaaa|bbbbbbbbbbbbbbb

j i

(<=pivot) (>pivot)

Follow Up: A better Quick Sort?



aaaaaaaaa|**ppppppppp**|xxxxxxxxxxxxxx|bbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

aaaaaaaaa|**ppppppppp**|bbbbbbbbb

jk i

(<pivot) (==pivot) (>pivot)

If ‘x’ < pivot

aaaaaaaaa|**ppppppppp**|xxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ < pivot, ‘x’ becomes ‘a’

aaaaaaaaa|**ppppppppp**|axxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ < pivot, ‘x’ becomes ‘a’

aaaaaaaaa|**ppppppppp**|axxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

Swap ‘p’ and ‘a’

aaaaaaaaa|**ppppppppp**|axxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ < pivot, ‘x’ becomes ‘a’

aaaaaaaaa|**ppppppppp**|axxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

aaaaaaaaa|**appppppppp**|pxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ < pivot, ‘x’ becomes ‘a’

aaaaaaaaa|**ppppppppp**|axxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

aaaaaaaaa|**apppppppp**|pxxxxxxxxxxxxx|bbbbbbbbbb

i = i + 1

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

j = j + 1

If ‘x’ < pivot, ‘x’ becomes ‘a’

aaaaaaaaa|**ppppppppp**|axxxxxxxxxxxxx|bbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

aaaaaaaaa**a|pppppppp**p|xxxxxxxxxxxxx|bbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ > pivot

aaaaaaaaa|**pppppppppp**|xxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ > pivot, ‘x’ becomes ‘b’

aaaaaaaaa|**ppppppppp**|bxxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ > pivot, ‘x’ becomes ‘b’

aaaaaaaaa|**ppppppppp**|bxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

Swap ‘b’ and ‘x’

aaaaaaaaa|**ppppppppp**|bxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ > pivot, ‘x’ becomes ‘b’

aaaaaaaaa|**ppppppppp**|bxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

aaaaaaaaa|**ppppppppp**|xxxxxxxxxxxxxb|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ > pivot, ‘x’ becomes ‘b’

aaaaaaaaa|**ppppppppp**|bxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

aaaaaaaaa|**ppppppppp**|xxxxxxxxxxxxxb|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

k = k - 1

If ‘x’ > pivot, ‘x’ becomes ‘b’

aaaaaaaaa|**ppppppppp**|bxxxxxxxxxxxxx|bbbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

aaaaaaaaa|**ppppppppp**|xxxxxxxxxxxxx|bbbbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ == pivot

aaaaaaaaa|**pppppppppp**|xxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ == pivot

aaaaaaaaa|**ppppppppp**|xxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ == pivot, ‘x’ becomes ‘p’

aaaaaaaaa|**ppppppppp**|pxxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

If ‘x’ == pivot, ‘x’ becomes ‘p’

aaaaaaaaa|**pppppppppp**|pxxxxxxxxxxxxxx|bbbbbbbbbb

j = j + 1

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

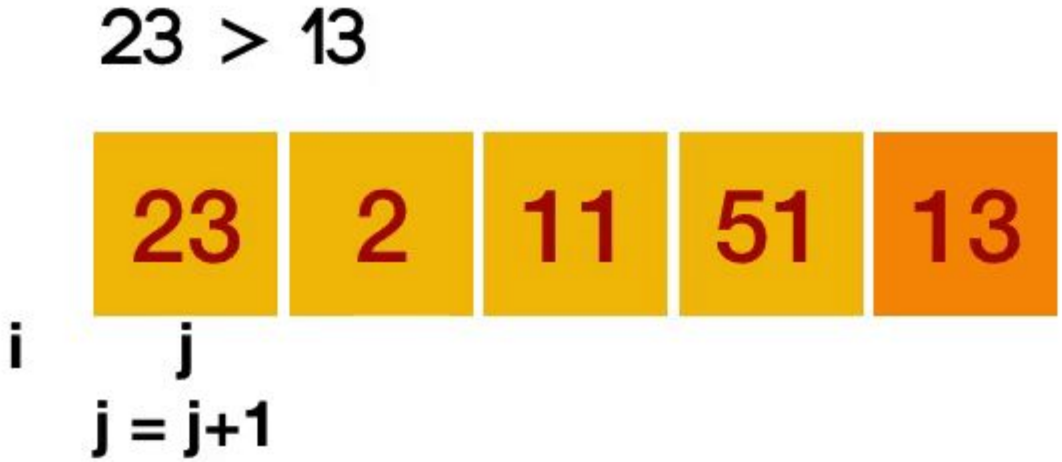
If ‘x’ == pivot, ‘x’ becomes ‘p’

aaaaaaaaa|**ppppppppp**p|xxxxxxxxxxxxxx|bbbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

Keep inspect next ‘x’s, until j and k meet



aaaaaaaaa|**ppppppppp**|xxxxxxxxxxxxxx|bbbbbbbbb

i j k

(<pivot) (==pivot) (undiscovered) (>pivot)

aaaaaaaaa|**ppppppppp**|bbbbbbbbbb

jk i

(<pivot) (==pivot) (>pivot)

Pow(x, n)

Implement pow(x, n), which calculates `x` raised to the power `n` (i.e., x^n).

Example 1:

```
Input: x = 2.00000, n = 10
Output: 1024.00000
```

Example 2:

```
Input: x = 2.10000, n = 3
Output: 9.26100
```

Example 3:

```
Input: x = 2.00000, n = -2
Output: 0.25000
Explanation: 2^-2 = 1/2^2 = 1/4 = 0.25
```

Constraints:

- $-100.0 < x < 100.0$
- $-2^{31} <= n <= 2^{31}-1$
- `n` is an integer.
- $-10^4 <= x^n <= 10^4$

intuition

```
rslt = b
for i = 2 to x
        rslt = rslt * b
end for
return rslt
```

$$b^x = \underbrace{b \times \cdots \times b}_{x \text{ times}}$$

b = base of the logarithm

x = exponent

Approach 1 - recursive

x ^ 17:

Intermediate result = x ^ 8

return (Intermediate result) * (Intermediate result) * x ^ 1

x ^ 8:

Intermediate result = x ^ 4

return (Intermediate result) * (Intermediate result)

x ^ 2:

return x * x

x ^ 4:

Intermediate result = x ^ 2

return (Intermediate result) * (Intermediate result)

Approach 1 - recursive

x ^ -17:

Intermediate result = x ^ 8

return 1.0/((Intermediate result) * (Intermediate result) * x ^ 1)

x ^ 8:

Intermediate result = x ^ 4

return (Intermediate result) * (Intermediate result)

x ^ 2:

return x * x

x ^ 4:

Intermediate result = x ^ 2

return (Intermediate result) * (Intermediate result)

Edge case

Constraints:

- $-100.0 < x < 100.0$
- $\boxed{-2^{31}} <= n <= 2^{31}-1$
- n is an integer.
- $-10^4 <= x^n <= 10^4$

When x = - 2^31 = -2147483648, -x still = -2147483648

Range of INT32:

-2147483648 (-2^31) to 2147483647 (2^31 - 1)

So, -(-2147483648) = 2147483648 > 2147483647

Out of range

Approach 2 - recursive

x ^ -17:

Intermediate result = x ^ -8

return 1.0/((Intermediate result) * (Intermediate result) * x ^ 1)

x ^ -8:

Intermediate result = x ^ -4

return (Intermediate result) * (Intermediate result)

x ^ -2:

return x * x

x ^ -4:

Intermediate result = x ^ -2

return (Intermediate result) * (Intermediate result)

Approach 3 - ADVANCED

6 5 4 3 2 1 0

89(Decimal) = 1 0 1 1 0 0 1(Binary)

= 2^6 + 2^4 + 2^3 + 2^0

= 64+16+8+1

x^89 = x^(64+16+8+1) = (x^64) * (x^16) * (x^8) * (x^1)

Approach 3 - ADVANCED

$$\begin{array}{rccccccc} & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 89(\text{Decimal}) = & 1 & 0 & 1 & 1 & 0 & 0 & 1(\text{Binary}) \end{array}$$

$= 2^6 + 2^4 + 2^3 + 2^0$

$= 64+16+8+1$

$x^{89} = x^{(64+16+8+1)} = (x^{64}) * (x^{16}) * (x^8) * (x^1)$

$$x^{64} \leftarrow x^{32} \leftarrow x^{16} \leftarrow x^8 \leftarrow x^4 \leftarrow x^2 \leftarrow x$$

rslt = $(x^{64}) * (x^{16}) * (x^8) * (x^1)$

Approach 3 - ADVANCED

$$
\begin{array}{ccccccc}
6 & 5 & 4 & 3 & 2 & 1 & 0
\end{array}
$$

89(Decimal) = 1 0 1 1 0 0 1(Binary)

= 2^6 + 2^4 + 2^3 + 2^0

= 64+16+8+1

x^89 = x^(64+16+8+1) = (x^64) * (x^16) * (x^8) * (x^1)

x^64 ⟵ x^32 ⟵ x^16 ⟵ x^8 ⟵ x^4 ⟵ x^2 ⟵ x

rslt = (x^64) * (x^16) * (x^8) * (x^1)

Bitwise operations!

Next Week: Climbing Stairs

You are climbing a staircase. It takes `n` steps to reach the top.

Each time you can either climb `1` or `2` steps. In how many distinct ways can you climb to the top?

Example 1:

```
Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps
```

Example 2:

```
Input: n = 3
Output: 3
Explanation: There are three ways to climb to the top.
1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step
```

Constraints:

- `1 <= n <= 45`