Miscellaneous math

Fast pow, Fibonacci, tortoise and hare

beCP Training

April 3, 2016

Table of Contents

Powers

Definition:

- Chain multiplication
- "n-th power of b"
- b is the base, n is the exponent
- $b^n = \underbrace{b \times \cdots \times b}_{n \text{ times}}$

Examples:

- $3^0 = 1$ (by definition)
- $3^1 = 3$
- $3^2 = 3 \times 3 = 9$ (square)
- $3^3 = 3 \times 3 \times 3 = 27$ (cube)

Power computation: linear

Problem: compute the power the n-th power of b, for given b and n.

Solution 1: Simple loop

```
int nthPower(int b, int n)
{
    int power = 1;
    for (int i = 0; i < n; i++)
        power *= b;
    return power;
}
```

Complexity: $O(n)$

Power computation: logarithmic (1)

Can we do it faster? Yes, because associativity!

For example, to compute 3^{10}, we can compute 3^5 then square it:

- $3^2 = 3 \times 3 = 9$
- $3^5 = 3^2 \times 3^2 \times 3 = 9 \times 9 \times 3 = 243$
- $3^{10} = 3^5 \times 3^5 = 243 \times 243 = 59049$

Only 4 multiplications instead of 9.

Power computation: logarithmic (2)

Solution 2: Recursive function

```
int nthPower(int b, int n)
{
    // Initial case
    if (n == 0)
        return 1;

    // Recursive case
    int power = powerOfThree(b, n/2);
    power *= power;
    if (n % 2 == 1)
        power *= b;
    return power;
}
```

We divide n by 2 on every call $\Rightarrow O(\log n)$

Fast pow: usage

When to use it:

- When linear time is too slow
- Typically when computing a number of possibilities

Limits:

- Exponent $\leq 10^{18}$ if using `long long` (or more!)
- Many powers with the same base $\Rightarrow$ store in an array
- Be careful with overflows! Often, the statement asks for the result *modulo* some number.

Table of Contents

Table of Contents

Table of Contents