1 Algorithms

General idea: a set of computational instructions written with if, for, memory access.

Lost Cow problem

```
for i = 1, 2, ...
    move left i
    move right 2i
    move left i
    ...
```

If gate is at x, we will reach it at step $|x|$.

Cost: $\sum_{i=1}^{|x|} 4i = \frac{4|x|(|x|+1)}{2} = 2|x|(|x|+1) \in O(|x|^2)$

$f(n) \in O(g(n))$ if there exist c, n_0 s.t. $\forall n \geq n_0$, $f(n) \leq c \cdot g(n)$. (i.e., if $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} \leq c$)

$n \in O(n^2) \iff \lim\limits_{n\to\infty} \frac{n}{n^2} = \lim\limits_{n\to\infty} \frac{1}{n} = 0$

$f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n))$

$f(n) \in \theta(g(n)) \iff f(n) \in O(g(n)) \wedge g(n) \in O(f(n))$

More efficient version:

```
for i = 1, 2, ...
    move left 2**i
    move right 2*(2**i)
    move left 2**i
    ...
```

Correctness: once $2^i > |x|$, we are done.

Fibonacci

```
def recursiveFib(n):
    if n <= 1
        return 1
    else
       return recursiveFib(n-1) + recursiveFib(n-2)
```

Runtime to compute $f(n)$: $\theta(f(n)) \to \theta(\frac{1+\sqrt{5}}{2})^n \in \Omega(1.618^n)$

Iterative Fibonacci

```
def iterFib(n):
    a[0] = 1
    a[1] = 1
    for i = 2 to n
        a[i] = a[i-1] + a[i-2]
    return a[n]
```

Number of digits is $\theta(\log_2(1.618^n))$