Decrease and Conquer

Topics

- Exponentiation by squaring
  - modPow
  - Solving linear recursion
- Greatest common divisor
- Extended Euclidean algorithm
- Multiplicative inverse
  - mod prime
  - mod composite

Exponentiation by squaring

- $a^1=a$
- $a^2=a \cdot a$
- $a^k=a^{k-1} \cdot a$
- Operator $\cdot$ is associative: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - Addition + is associative
  - Multiplication × is associative
  - Subtraction - is not associative
  - Division ÷ is not associative

Exponentiation by squaring

- Goal: Computing a^x efficiently.
- If the operator $\cdot$ is associative, then
  - $a^{2k}=a^k \cdot a^k$
  - $a^{2k+1}=a \cdot a^k \cdot a^k$
- Idea:
  - Compute $b=a^{x/2}$ first.
  - If x is odd, return $a \cdot b \cdot b$
  - If x is even, return $b \cdot b$
- Time complexity: O(logx)

modPow

- $a^2 \bmod m = (a \cdot a) \bmod m$
  $= (a \bmod m) \cdot (a \bmod m) \bmod m$
- We can perform modulo operation after each multiplication.
- So modPow is essentially exponentiation.
  - Can be done efficiently

Matrix exponentiation

- Matrix multiplication C=AB
  - Not communitive: AB=BA does not hold in general!
  - Associative: (AB)C = A(BC)
  - If A is p-by-q and B is q-by-r, then AB can be computed in O(pqr) by definition.
    - Can be faster by some divide-and-conquer approaches
- Matrix exponentiation can be computed efficiently

Linear recursion

- $f(i)=x_i$ for $i < k$
- $f(n) = a_0+a_1f(n-1)+a_2f(n-2)+\ldots+a_kf(n-k)$

Linear recursion

- We can use matrix exponentiation to solve linear recursion efficiently.
- Ex: Fibonacci numbers
  - $F(1)=F(2)=1$
  - $F(n)=F(n-1)+F(n-2)$
- We can compute $F(n)$ in $O(\log n)$ time.

Great common divisor

- GCD: $gcd(x,y)=max\{ d: x=pd, y=qd \text{ and } x,y \in Z\}$
  - $gcd(x,y)=gcd(y,x)=gcd(-x,y)=gcd(y-x,x)$
  - $gcd(0,x)=|x|$
- How to compute?
  - Enumeration: $O(min(x,y))$
  - Euclidean algorithm
    - Assume x and y are non-negative and $x+y>0$.
    - If $y=0$ return x
    - return gcd(y, x%y)

Extended Euclidean algorithm

- Problem: Given x and y, to find a and b such that $ax+by=\gcd(x,y)$
- Idea: Think the process of Euclidean algorithm
  - Ex: $x=64$, $y=10$
    $64 - 6\cdot 10 = 4$
    $10 - 2\cdot 4 = 2$
    $0 = 2 - 1\cdot 2$
  - Reverse the equations:
    $4 = 64 - 6\cdot 10$
    $2 = 10 - 2\cdot 4 = 10 - 2\cdot(64 - 6\cdot 10) = -2\cdot 64 + 13\cdot 10$
    $0 = 2 - 1\cdot 2$ (note: $\gcd(64,10)=2$)

Multiplicative inverse

- $x \cdot (1/x) = 1$
- $a \cdot x = 1 \pmod{p}$
- $ax + bp = 1$
- Can be computed by extended Euclidean Algorithm

Fermat little theorem

- For any prime p and any integer $b > 0$, we have that $b^{p-1}=1 \pmod p$
- Not going to show the correctness here
  - Number theory
  - Group theory
- Can be computed by modPow
- Compute multiplicative inverse for prime modulo
  - $b^{p-2}=b^{-1} \pmod p$

Euler's totient

- $\phi(n)$ is the number of positive integers in [1,n] such that are relative prime to n.
- $\phi(6)=|\{1,5\}|=2$
- $\phi(30)=|\{1,7,11,13,17,19,23,29\}|=8$
- $\phi(p)=p-1$ if p is a prime
- $\phi(p^k)=p^{k-1}(p-1)$
- $\phi(xy)=\phi(x)\cdot\phi(y)$ if $\gcd(x,y)=1$
- If $\gcd(b,m)=1$, then $b^{\phi(m)}=1 \pmod m$