gcd (greatest common divisor)

divisor

divisors of 12:	1, 2, 3, 4, 6, 12.
divisors of 30:	1, 2, 3, 5, 6, 10, 15, 30.

common divisors

1, 2, 3, 6.

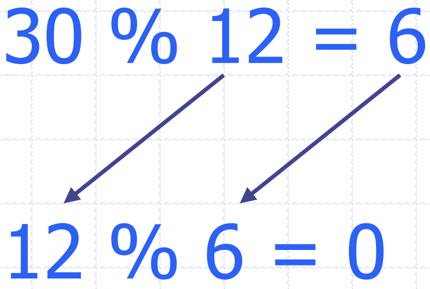
greatest common divisor

6

gcd (greatest common divisor)

```
Algorithm gcdOne(int a, int b)
Step 1. Create a list L1 of all divisors of a.
Step 2. Create a list L2 of all divisors of b.
Step 3. Create a list L3 of common divisors of a and b.
Step 4. Pick the “greatest” from L3.
```

gcd (greatest common divisor)



30 % 12 = 6

12 % 6 = 0

gcd (greatest common divisor)

```
Algorithm gcdTwo(int a, int b)
Step 1. c <- a % b
Step 2. a <- b; b <- c
Step 3. Repeat steps 1 and 2 until b is 0.
Step 4. return a.
```

gcd (greatest common divisor)

```
static int gcd(int a, int b)
{
        if(b == 0)
       {
               return a;
        }
        return gcd(b, a % b);
}
```

Examples of Problems

1. (*GCD Problem*) Given two positive integers m, n, is there a positive integer d that is a factor of both m and n and that is bigger than or equal to every integer d’ that is a factor of m and n?

```
static int gcd(int a, int b)
{
        if(b == 0)
       {
                return a;
        }
        return gcd(b, a % b);
}
```

Given two positive integers, can you show (prove) the algorithm will **halt** (end) ?

What is the time complexity?

Examples of Problems

1. (*GCD Problem*) Given two positive integers m, n, is there a positive integer d that is a factor of both m and n and that is bigger than or equal to every integer d' that is also a factor of m and n?

```
static int gcd(int a, int b)
{
            if(b == 0)
           {
                      return a;
            }
            return gcd(b, a % b);
}
```

```
gcd(13, 8)
gcd(8, 5)            //first recursive call
gcd(5, 3)            //second recursive call
gcd(3, 2)
gcd(2, 1)
gcd(1, 0)
```

There are 5 recursive calls. gcd(Fib(n), Fib(n-1)) will make n-2 recursive calls.

Examples of Problems

1. Given two integers, can you show (prove) the algorithm will halt (end) ?
2. If a = 3, which value of b less than 3 will result in maximum number of recursive calls?
3. If a =5, which value of b less than 5 will result in maximum number of recursive calls?
4. If a =8, which value of b less than 8 will result in maximum number of recursive calls?
5. What is 1, 1, 2, 3, 5, 8, …?
6. Let f(x) = x^2. What is its inverse function?
7. Let g(x) = sqrt(x). Then f(g(x)) = g(f(x)) = x. Hence f and g are inverse of each other.
8. Let f(x) = exp(x). What is its inverse?
9. If Fibonacci has exponential growth, what can you say about the growth of its inverse?

Examples of Problems

3. The subset sum problem

S = {2, 5, 9}

{2}, {5}, {9},

{2, 5}, {2, 9}, {5, 9}

{2, 5, 9}

There are 7 nonempty subsets.

$7 = 8 - 1 = 2^3 - 1$.

Generalize this!

If S has n elements then there $2^n - 1$ nonempty subsets.

All known algorithms take exponential amount of steps. Hence subset problem belongs to EXP.

Lesson 1 Summary

Five problems

- gcd(n, m) log P
- sorting nlogn P
- subset sum exponential EXP
  - maybe someone will come up with a polynomial time algorithm. Then it will belong to class P.
  - It belongs to class NP. That is, given the solution, there is a polynomial time algorithm to verify it.
- n x n chess exponential EXP-complete
  - No polynomial time algorithm is possible.
  - It is not in class NP.
  - It will never belong to P.
- Halting problem No algorithmic solution.

QUIZ ON LESSON 1

What is meant by a problem belongs to class P?

What is meant by a problem belongs to class NP?

What is meant by a problem belongs to class EXP?

What is meant by a problem belongs to class EXP-complete?

What are the two conditions a problem must satisfy to belong to class EXP-complete?

Example of a problem in P, NP, EXP, EXP-complete.

Learn to write an algorithm.

Learn to write a nondeterministic algorithm.

What is the Halting Problem?

Why is Halting Problem important?

QUIZ ON LESSON 1

What is meant by a problem P belongs to class P?

There exists at least one polynomial time algorithm to **solve** the problem P.

What is meant by a problem P belongs to class NP?

There exists at least one polynomial time algorithm to **verify** any solution to the problem P.

What is meant by a problem P belongs to class EXP?

All known algorithms to **solve** the problem P has exponential time complexity.

What is meant by a problem P belongs to class EXP-complete?

(a) All known algorithms to **solve** the problem P has exponential time complexity.

(b) There can never be a polynomial time algorithm to solve the problem P.

What are the two conditions a problem P must satisfy to belong to class EXP-complete?

(a) All known algorithms to **solve** the problem P has exponential time complexity.

(b) There can never be a polynomial time algorithm to solve the problem P.

QUIZ ON LESSON 1

Example of a problem in P, NP, EXP, EXP-complete.

P : Sorting

NP : Sunset Sum

EXP : n x n chess

EXP-complete : n x n chess

Learn to write an algorithm.

Learn to write a nondeterministic algorithm.

What is the Halting Problem?

Write a computer program ℍ with following specifications:

Input : Any computer program ℂ.

Output : **Yes** if ℂ will always halt; **No** otherwise.

Why is Halting Problem important?

There are problems for which there will never be an algorithmic solution. Halting Problem is one such problem.