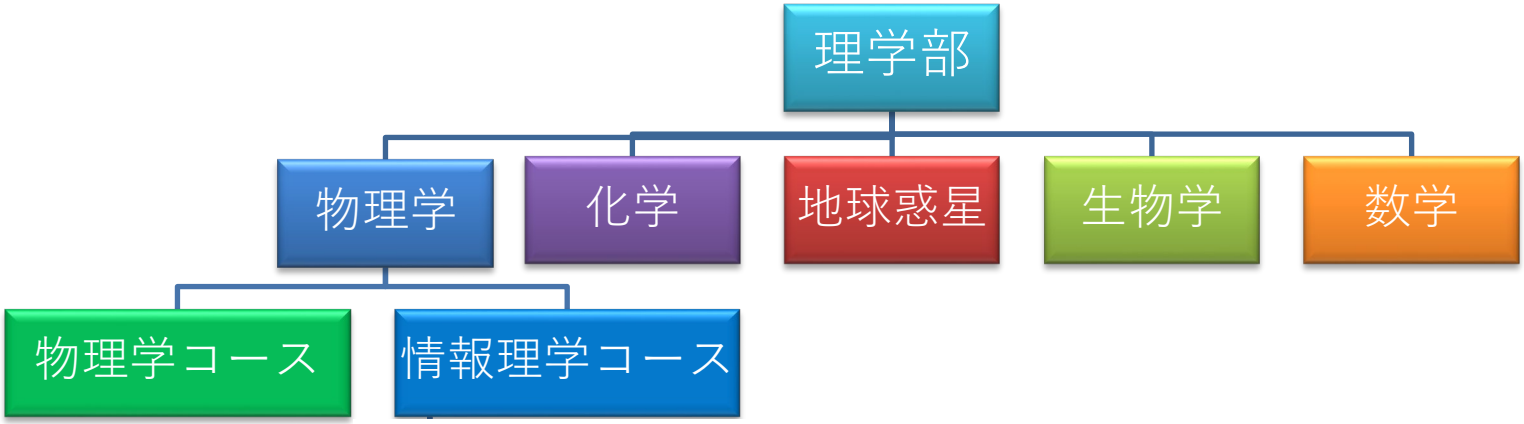
情報科学入門
「コンピュータにどう考えさせるか？」

九州大学 システム情報科学研究院
教授　稲永 俊介

九州大学
KYUSHU UNIVERSITY

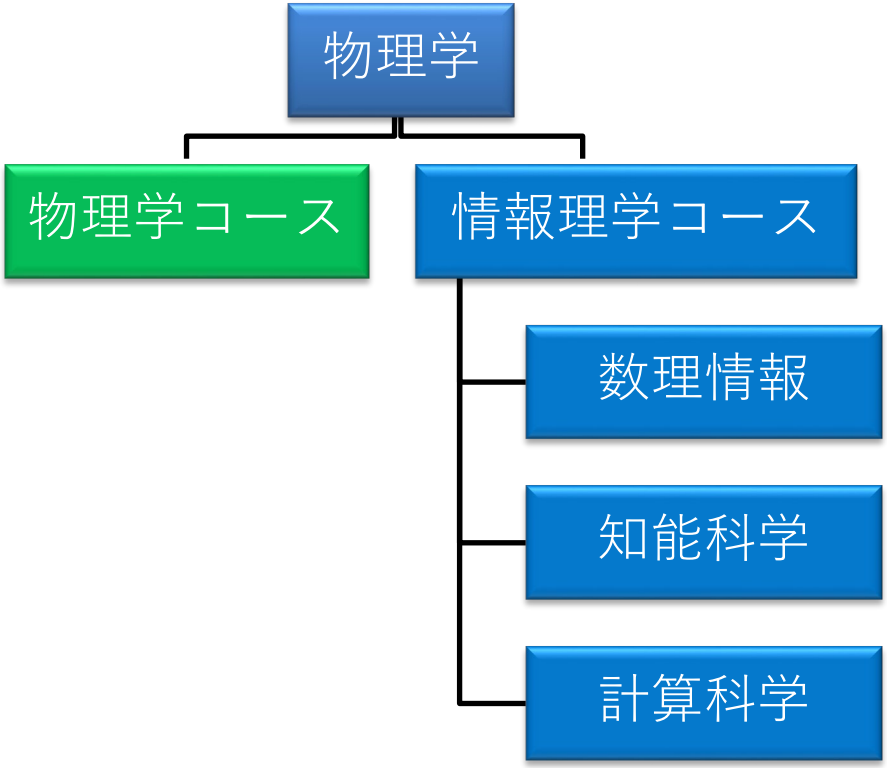
KYUSHU UNIVERSITY 100th 2011
知の新世紀を拓く

九州大学 理学部 物理学科



２年生前期に
コースを選択

九州大学 理学部 物理学科



アルゴリズム，暗号理論など

人工知能，計算学習理論など

計算機シミュレーションなど

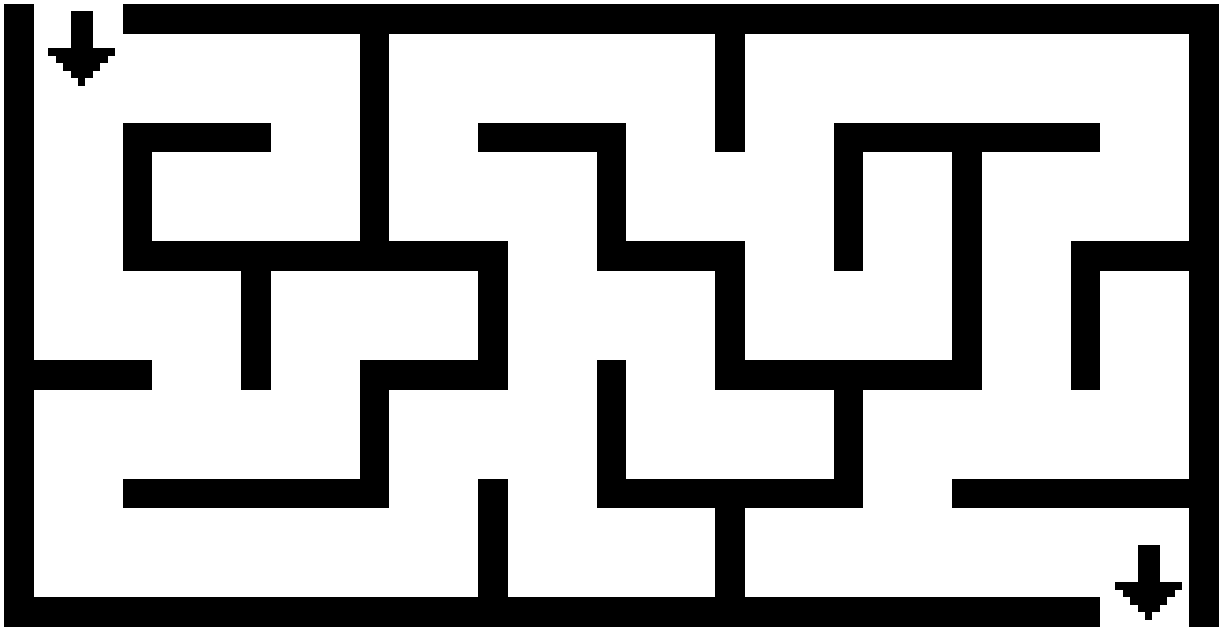
アルゴリズムって？

- コンピュータは、四則演算（＋，－，×，÷）などの単純な計算を高速かつ正確に行える
- しかし、複雑な問題を解くための「手順」をコンピュータ自身が編み出すことはできない
- 単純な計算を組み合わせて、より複雑な**問題を解くための手順**のことを**アルゴリズム**という

アルゴリズムの例（その１）迷路

問題：迷路を脱出せよ！

コンピュータは前後左右に動くことはできるが、
いつどの方向に行けばいいか知らない

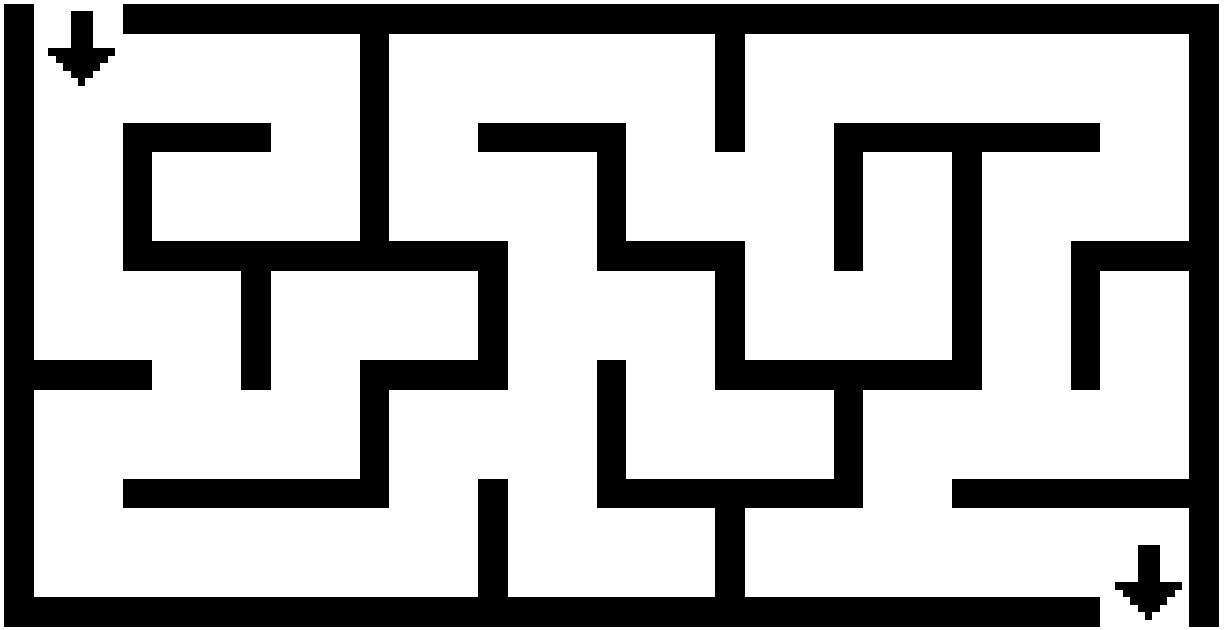


アルゴリズムの例（その１）迷路

問題：迷路を脱出せよ！

行き止まりになったら
どうしたらいいんですか？

スタート



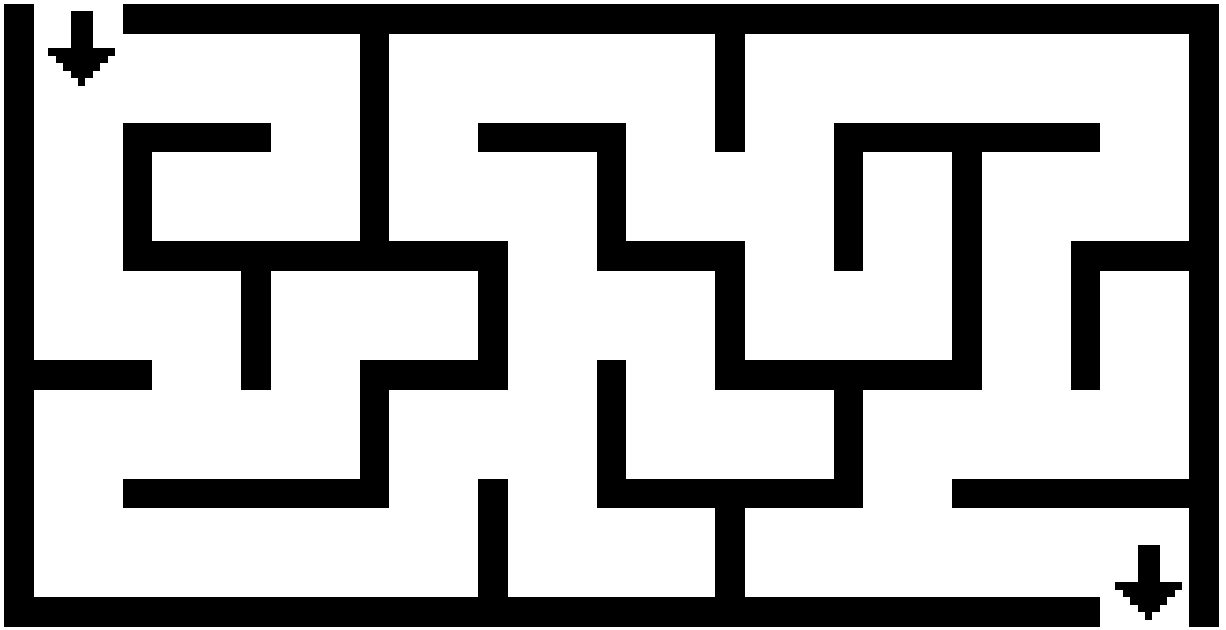
ゴール

アルゴリズムの例（その１）迷路

問題：迷路を脱出せよ！

壁に右手をついた
まま進みなさい（**右手法**）

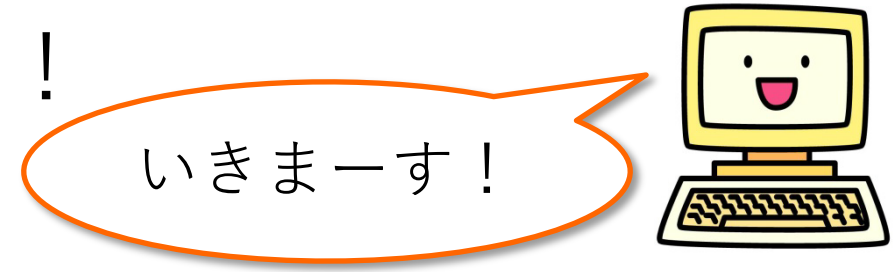
スタート



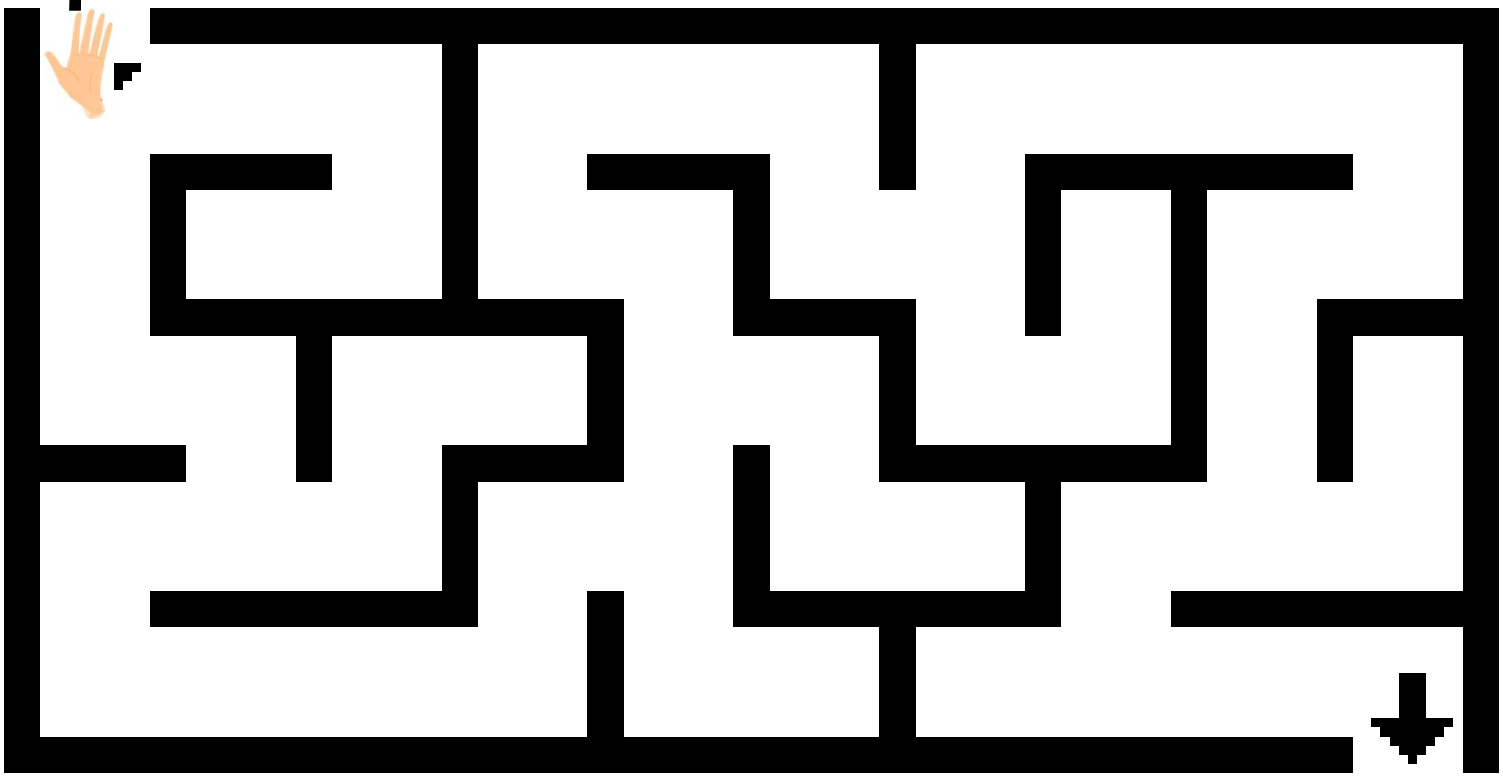
ゴール

アルゴリズムの例（その１）迷路

問題：迷路を脱出せよ！



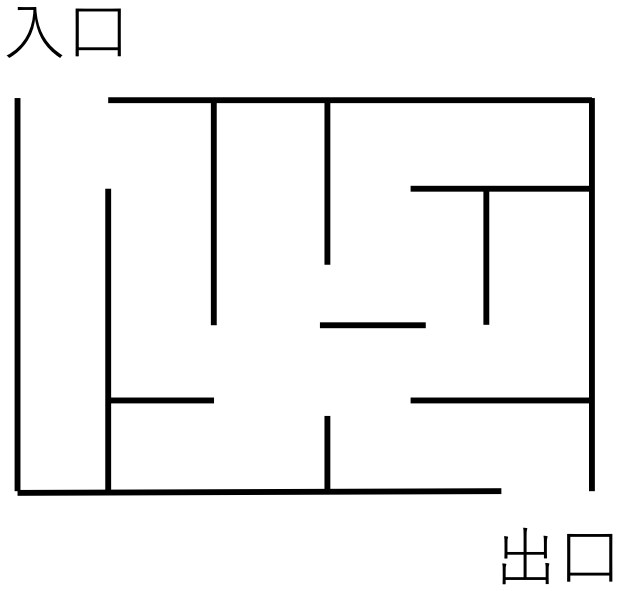
スタート



ゴール

なぜ右手法で迷路を抜けられるのか？

- 迷路を以下の３つのパーツに分割して考える.



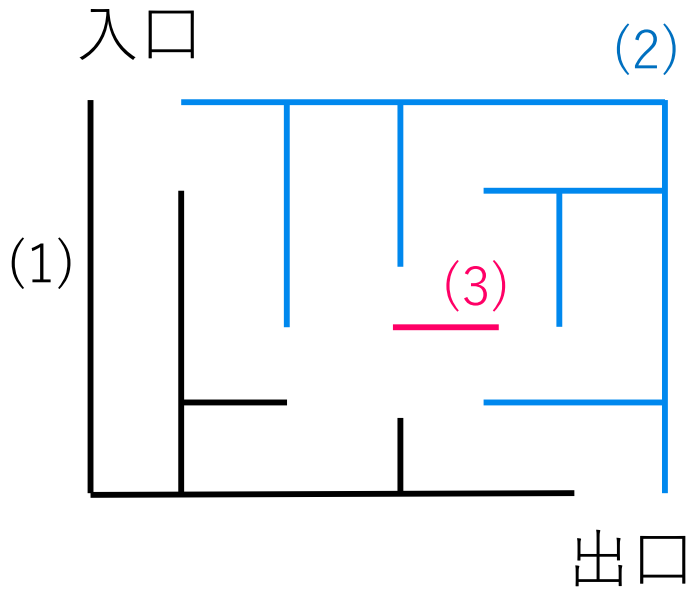
なぜ右手法で迷路を抜けられるのか？

- 迷路を以下の３つのパーツに分割して考える.

  (1) 入口から見て右側の壁に繋がっている部分.

  (2) 入口から見て左側の壁に繋がっている部分.

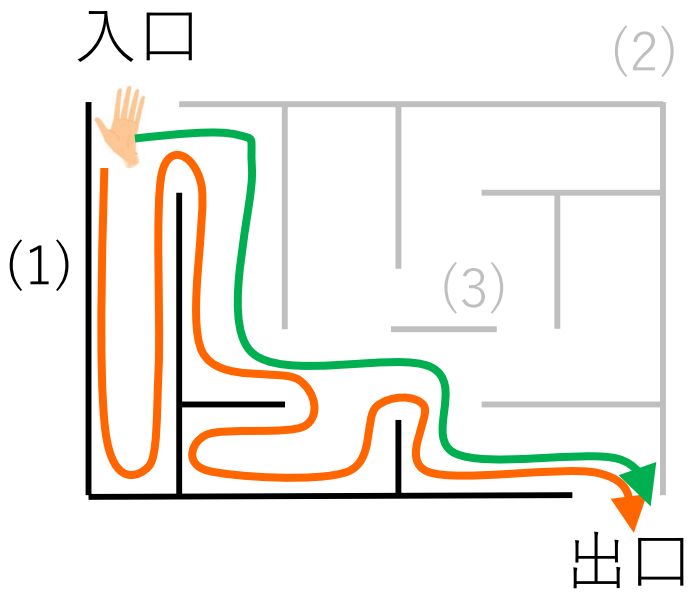
  (3) その他の部分.



なぜ右手法で迷路を抜けられるのか？

- (2)(3) は無視し，(1) のみに着目する．

  (1) の壁沿いに歩けば，出口に通じるのは明らか．



右手法では
「最短経路」
はわからない

迷路をグラフで
表すことで
「最短経路」は
高速に求まる

アルゴリズムの研究

- 問題を解く手順を与えれば、
  コンピュータは人間よりもずっと高速かつ
  正確に問題を解くことができる

- ただし、問題を解く手順（アルゴリズム）を
  考案するのは、人間の役割である

- 備考：アルゴリズムをコンピュータが
  理解できるように書いたものがプログラム

アルゴリズムの「良さ」

- 同じ問題でも、それを解くための
  アルゴリズムは何種類もありえる
  - 解き方は一通りとは限らない

- 「良い」アルゴリズムの条件
  1. 正しく問題を解ける
  2. 高速に問題を解ける

アルゴリズムの例（その２）偽コイン発見

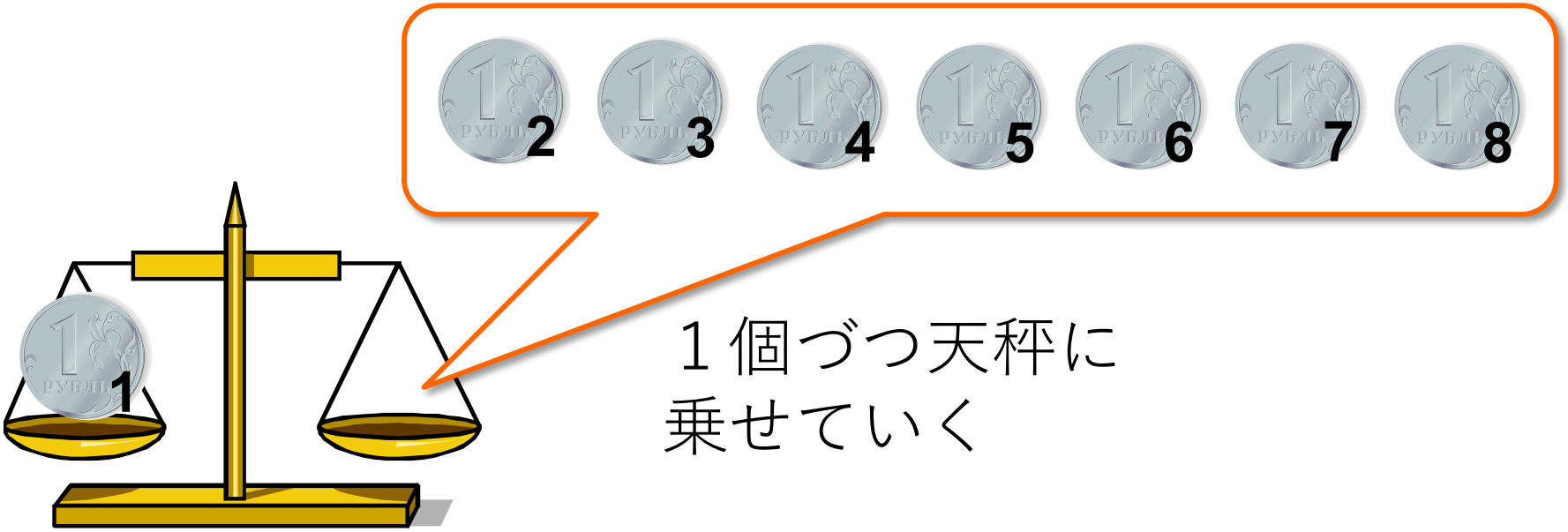
問題：９つのコインがあり、その中に一つだけ
偽物のコインがある。偽物は本物より重い。
天秤を使って、偽物を発見せよ。

天秤を何回使えば
偽物を発見できる
だろうか？

アルゴリズムの例（その２）偽コイン発見

適当に選んだ１個と残りの８個を
順番に比べれば、最悪の場合でも
８回天秤を使えば偽物が見つかる

より少ない回数で
できるだろうか…？



１個づつ天秤に
乗せていく

アルゴリズムの例（その２）偽コイン発見

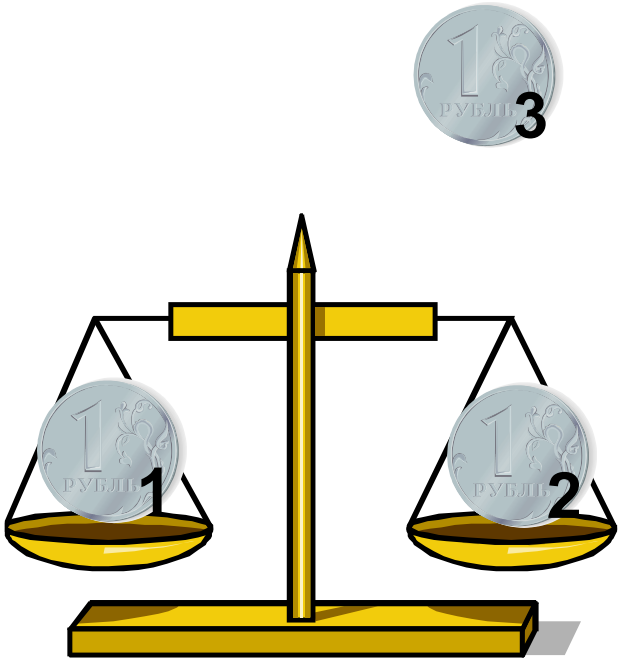
実は、たったの **２回** で発見できる！

考え方

コインが３個の場合を考える

考え方

コインを２個選び、天秤にかける



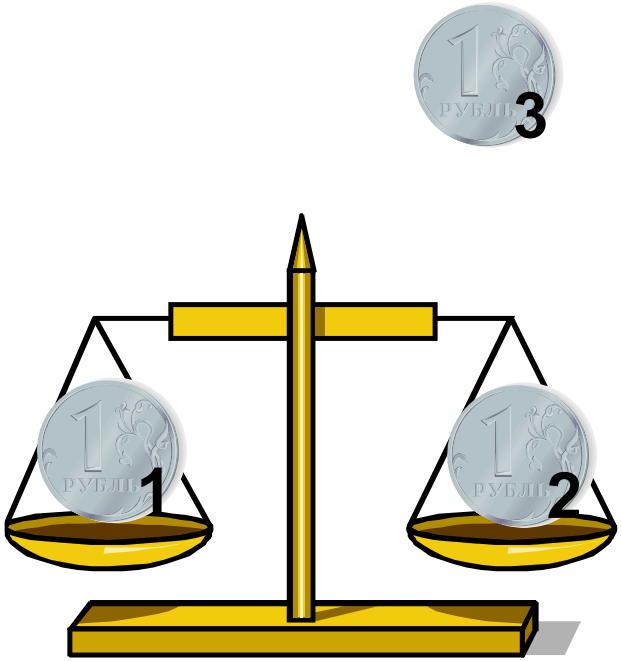
a. １のほうに傾けば、
　１が偽物

b. ２のほうに傾けば、
　２が偽物

c. 釣り合えば、
　３が偽物

考え方

コインが３個のときは、
天秤を **１回** 使えば偽物を発見できる



a. １のほうに傾けば、
　１が偽物

b. ２のほうに傾けば、
　２が偽物

c. 釣り合えば、
　３が偽物

考え方

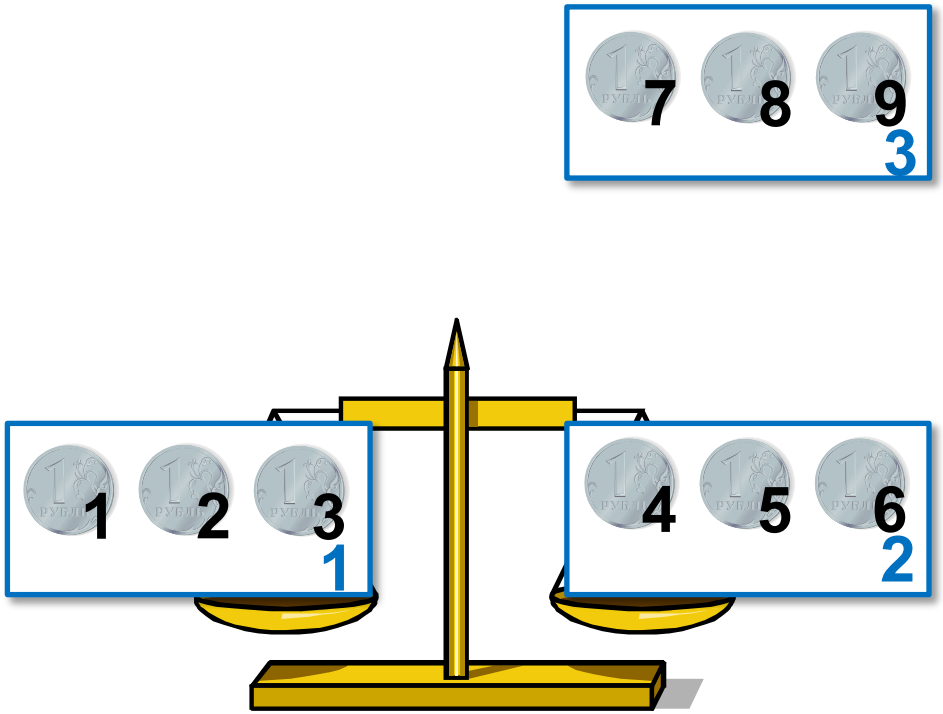
コインが９個のときは、
コインが３個のときのやり方を応用する

考え方

コインを３つのグループに分ける

考え方

グループを２つ選び、天秤にかける



a. 1 のほうに傾いたら、
1 の中に偽物がある

b. 2 のほうに傾いたら、
2 の中に偽物がある

c. 1 と 2 が釣り合えば、
3 の中に偽物がある

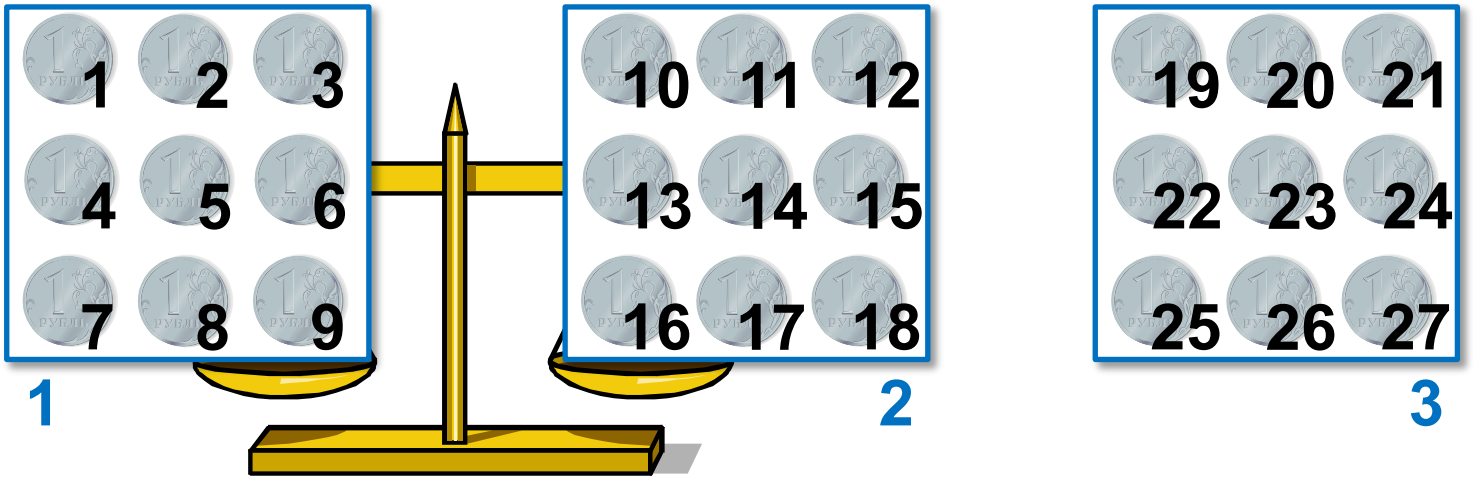
考え方

偽物が入ったグループを見つけるのに１回
そのグループから偽物を見つけるのに１回

よって、コインが９個のときの
天秤の使用回数は **２回**

コインが 27 個のとき

３グループに分け、２つを天秤にかける



天秤を１回使って、偽物が入ったグループを発見
あとは、コインが９個の場合と同様の手順
⇒ 天秤を **３回** 使用すれば偽物を発見できる

天秤の使用回数（コインの個数 $n = 3^k$ のとき）

コインの個数	天秤の使用回数
3	1
9	2
27	3
81	4
⋮	⋮
n	$\log_3 n$

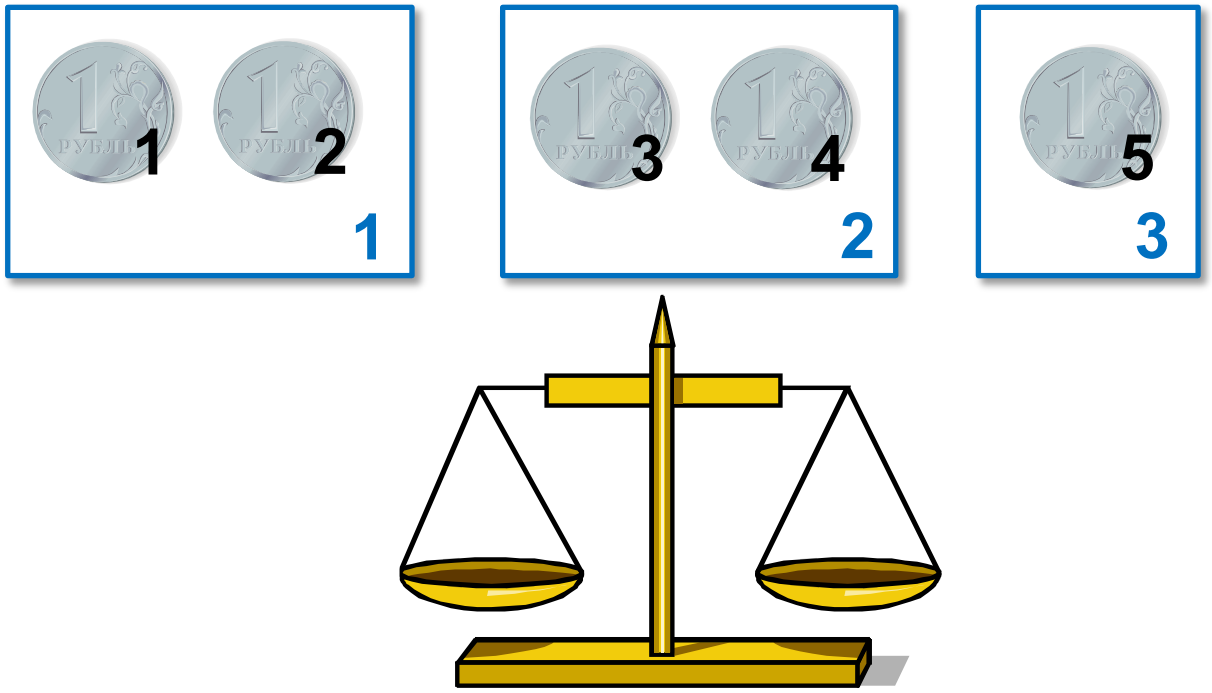
天秤を1回使うと
偽物の候補の数が
3分の1になる.

$n \neq 3^k$ のときは
どーすんの？？

コインが5個のとき

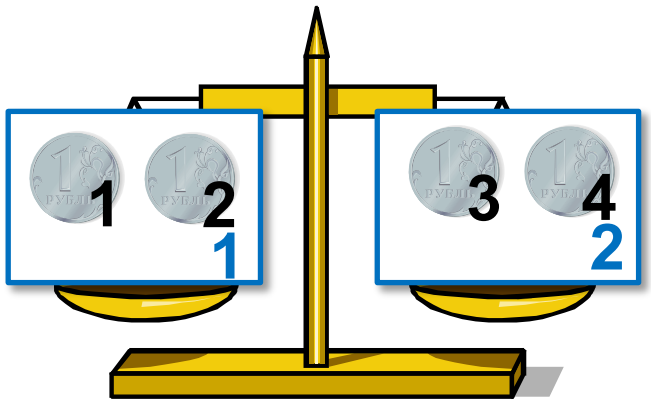
コインが５個のとき

コインを２個、２個、１個の
３つのグループに分ける



コインが5個のとき

コインが２個のグループを天秤にかける



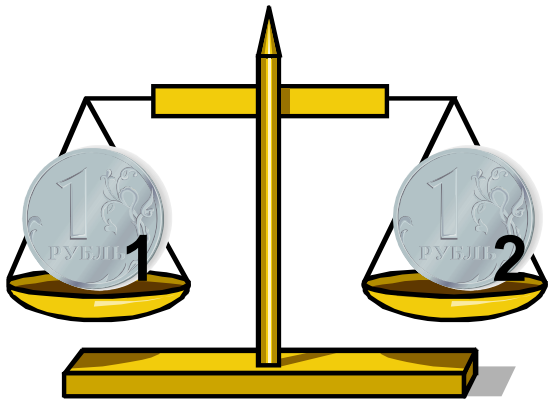
a. 1 のほうに傾いたら、
1 の中に偽物がある

b. 2 のほうに傾いたら、
2 の中に偽物がある

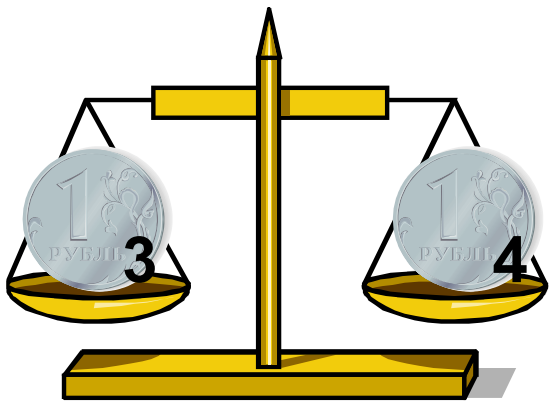
c. 1 と 2 が釣り合えば、
3 の中の 5 が偽物

コインが5個のとき

場合 a のときは，1 と 2 を比べる
場合 b のときは，3 と 4 を比べる



場合 a



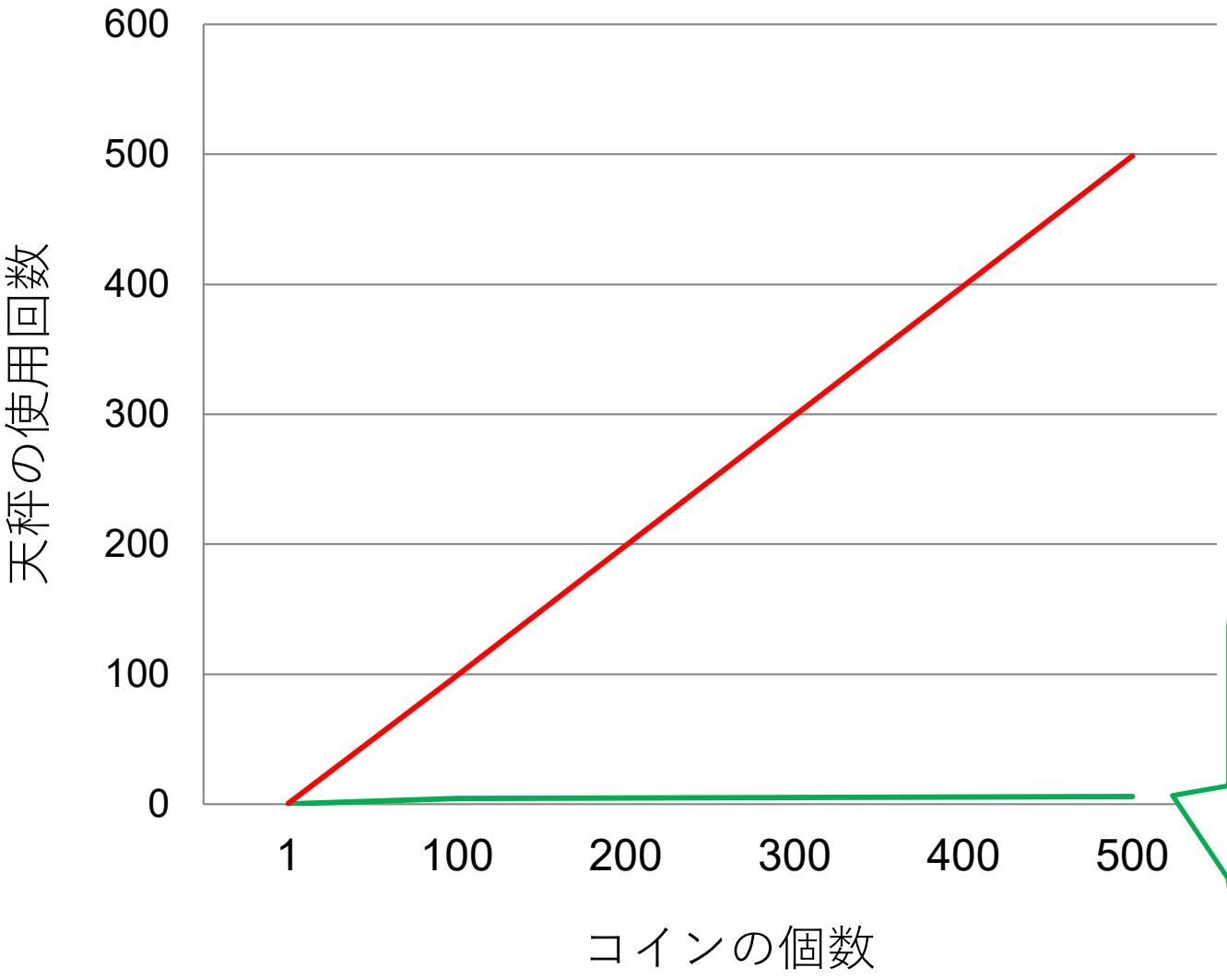
場合 b

コインが５個のとき

場合 a, b のとき，天秤の使用回数は２回
場合 c のとき，天秤の使用回数は１回

よって、コインが５個のときの
天秤の使用回数は高々 **２回**

$\log_3 n$ **vs** $n - 1$



$\log_3 n = n$ を 3 で繰り返し割って 1 以下になる回数

— $\log_3 n$

— n-1

$\log_3 n$ は**最適解**

➔ $\log_3 n$ より少ない天秤の使用回数で偽物を見つけることはできない

アルゴリズムの例（その３）最大公約数問題

問題

入力：正整数 x と y

出力：x と y の最大公約数 $\gcd(x, y)$

- gcd : greatest common divisor

例

- $\gcd(16, 12) = 4$
- $\gcd(321, 99) = 3$
- $\gcd(47, 32) = 1$

最大公約数問題を解くアルゴリズム

1. **しらみつぶし法**
   - 解の候補を大きいほうから順に試していく.
   - 膨大な計算時間がかかる.

2. **ユークリッドの互除法**
   - 明示的に記述された最古のアルゴリズム (BC300 年頃).
   - やり方は高校で学ぶが，計算時間は言及されていない.

数学A「整数」
共通テスト頻出問題

3. **素因数分解に基づく手法**
   - 中学校で学ぶ.
   - 簡単と思われがちだが，実は，現在知られている
     アルゴリズムでは膨大な計算時間がかかる.

アルゴリズムの例（その３）最大公約数問題

問題

入力：正整数 x と y

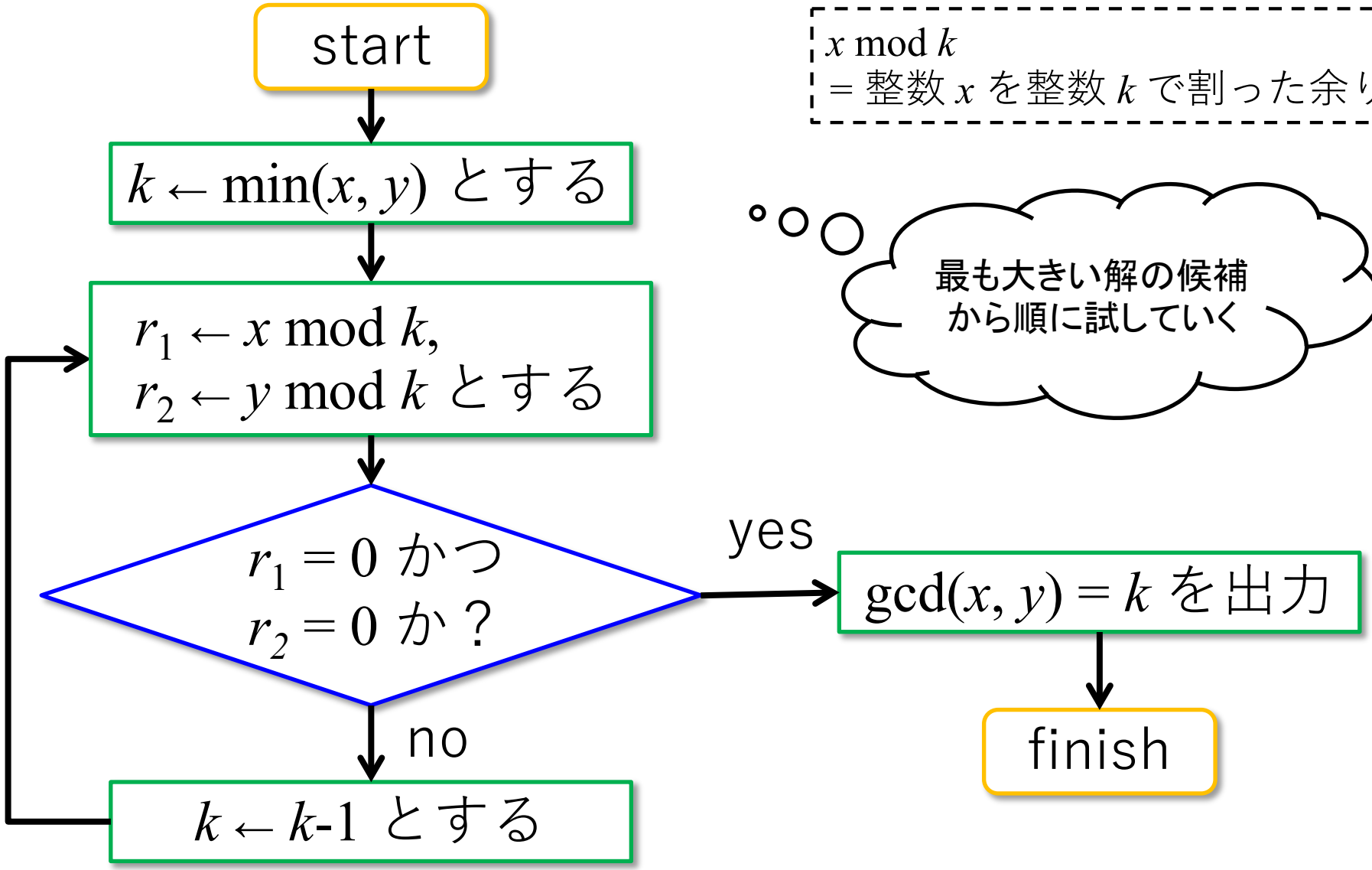
出力：x と y の最大公約数 $\gcd(x, y)$

- gcd : greatest common divisor

例

- $\gcd(16, 12) = 4$
- $\gcd(321, 99) = 3$
- $\gcd(47, 32) = 1$

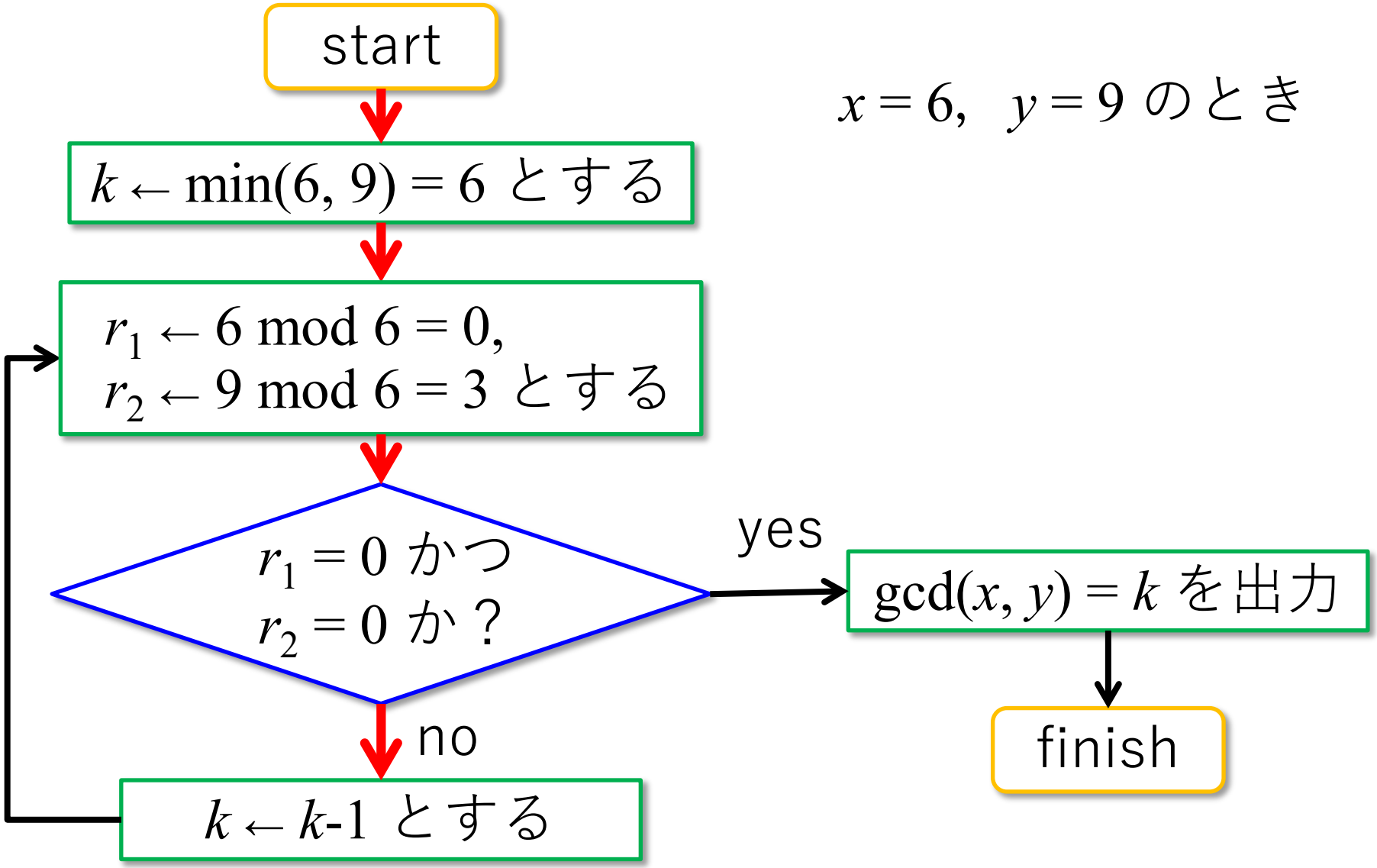
しらみつぶし法 naive_gcd



naive_gcd の実行例

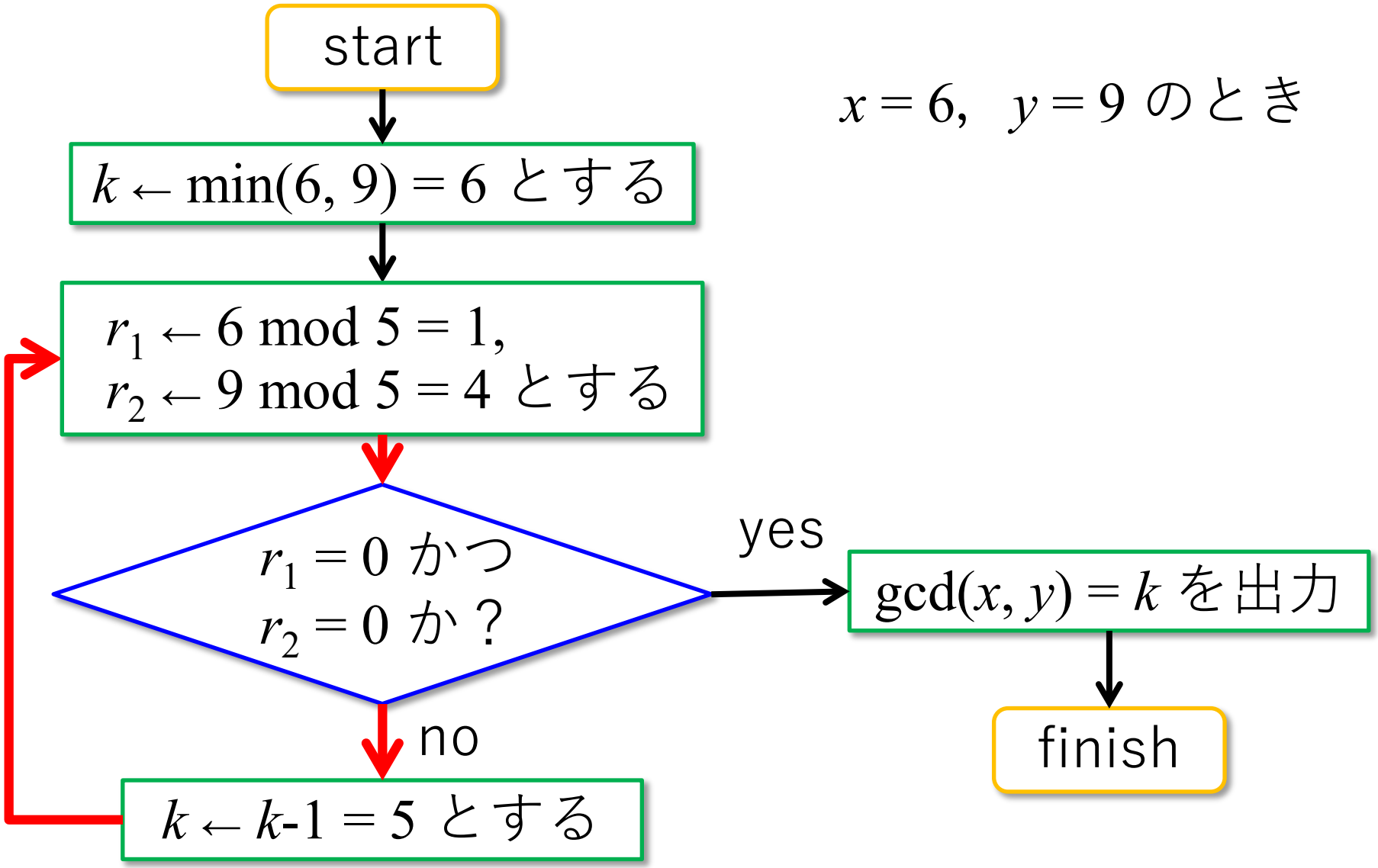
$x \bmod k$
= 整数 x を整数 k で割った余り

$x = 6,\quad y = 9$ のとき



naive_gcd の実行例

$x \bmod k$
= 整数 x を整数 k で割った余り

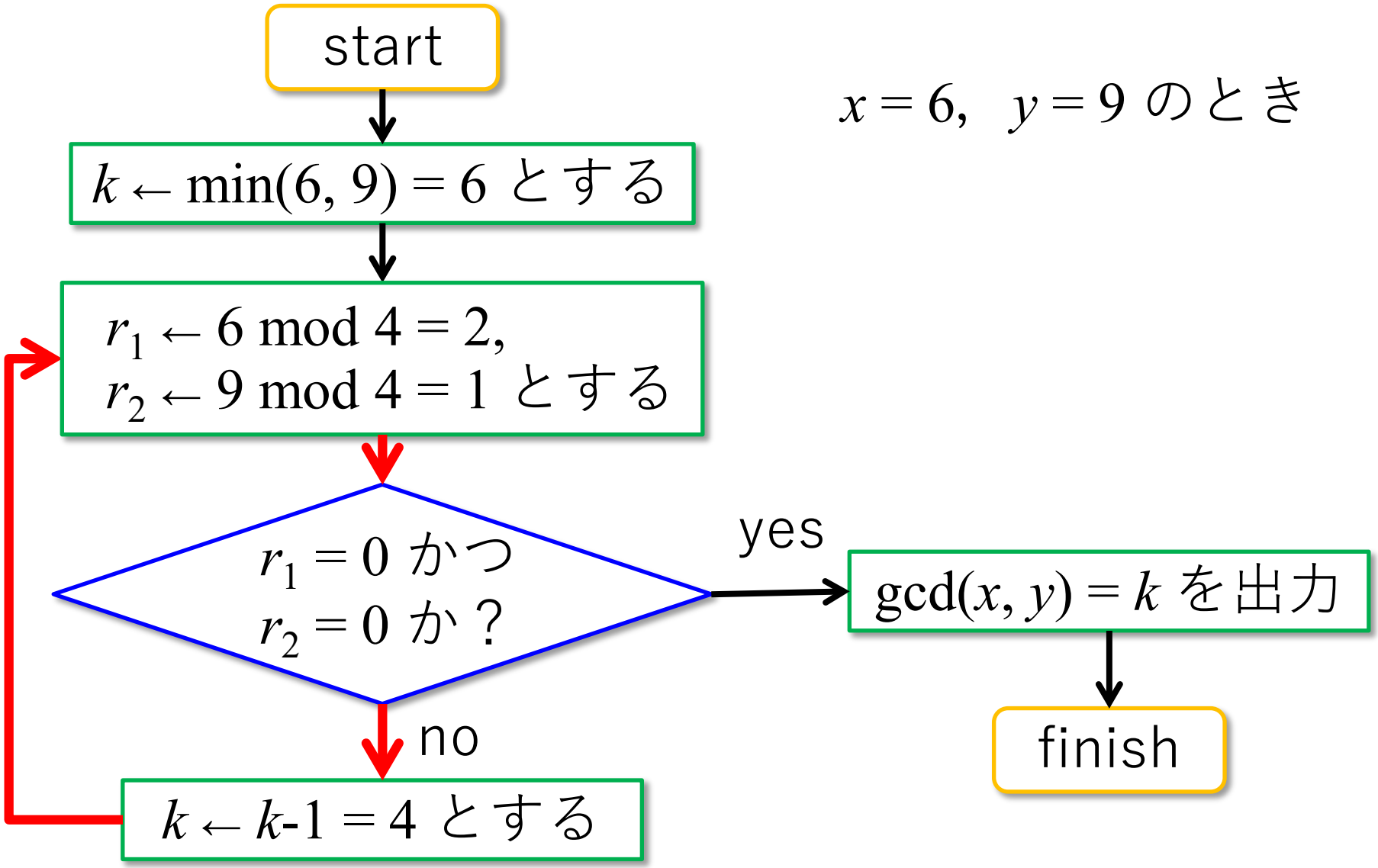


$x = 6,\quad y = 9$ のとき

naive_gcd の実行例

$x \bmod k$
= 整数 x を整数 k で割った余り

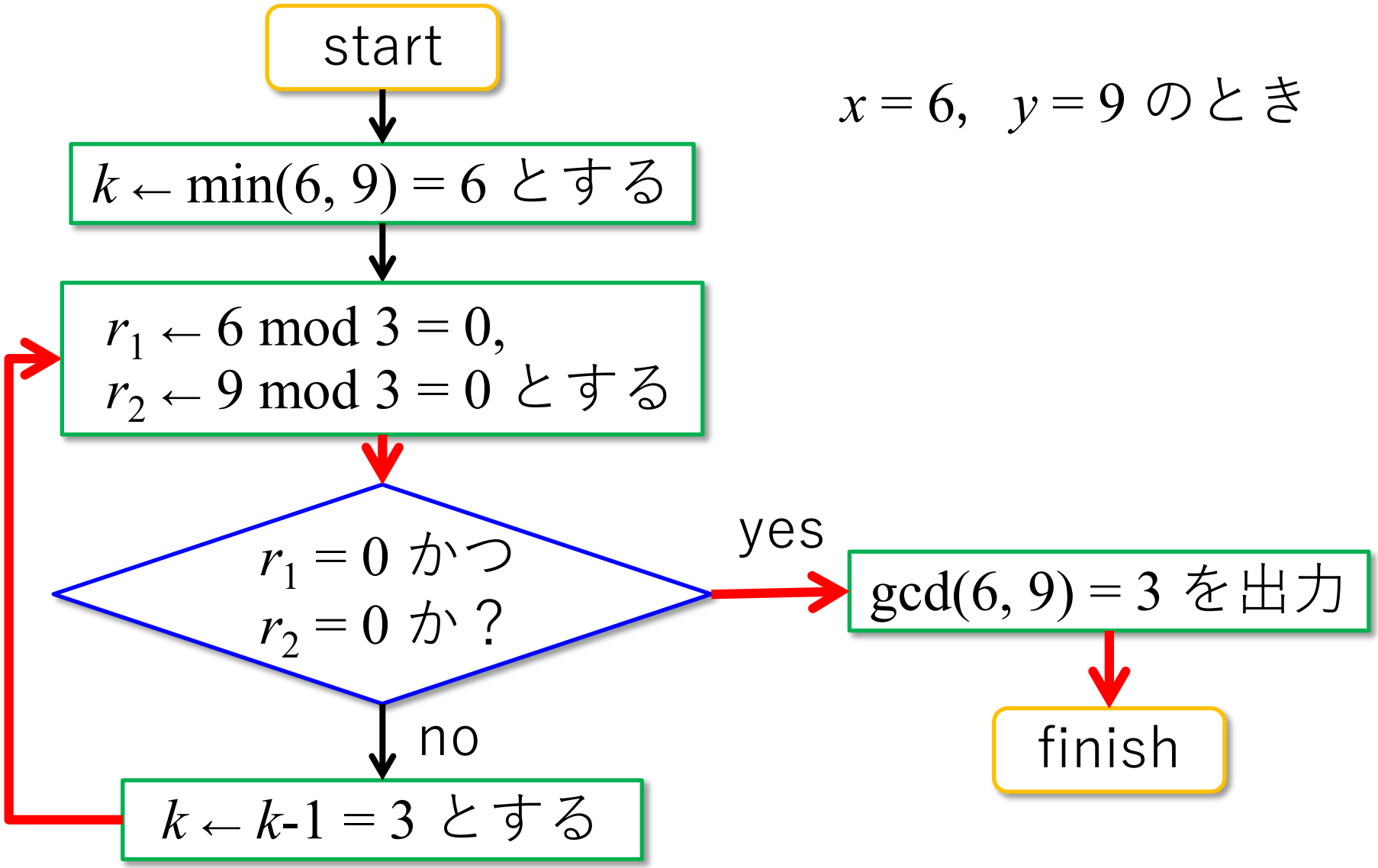
$x = 6,\quad y = 9$ のとき



naive_gcd の実行例

$x \bmod k$
= 整数 x を整数 k で割った余り

$x = 6,\quad y = 9$ のとき



計算ステップ数

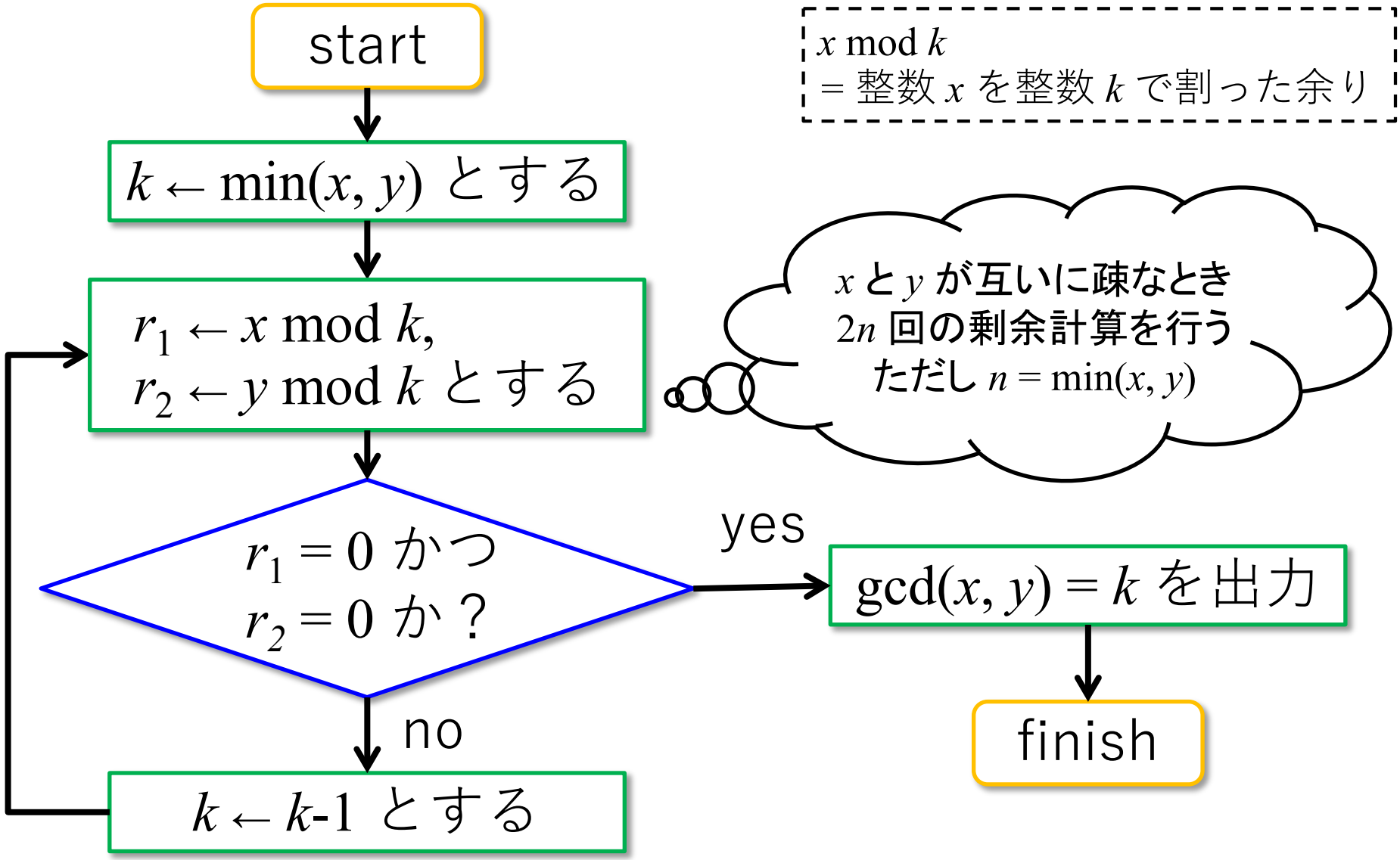
- 計算ステップ数とは，アルゴリズムが終了するまでに行われる基本演算の回数である．
- 計算ステップ数が少ないほど，アルゴリズムの実行時間は短くなる．
- 最大公約数問題では 剰余 (mod) を基本演算と考える．

naive_gcd の計算時間見積もり

- $n = \min(x, y)$ とする。
  naïve_gcd は最悪時には $2n$ 回の剰余計算を行う。

➔ 計算ステップ数は高々 $2n$ である。

しらみつぶし法 naive_gcd（おさらい）



naive_gcd の計算時間見積もり

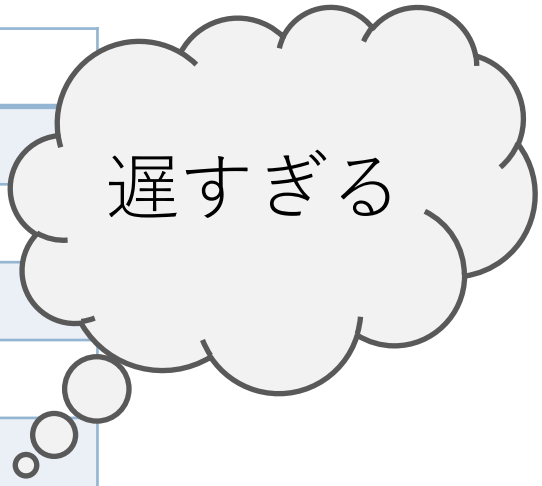
- $n = \min(x, y)$ とすると，計算ステップ数は高々 $2n$
- 剰余の1回の計算時間
  - パソコン 1ステップ 10^{-8} 秒
  - スパコン 1ステップ 10^{-11} 秒

n	パソコン	スパコン
10000	0.2 × 10^{-3} 秒	0.2 × 10^{-6} 秒
100万	0.02 秒	0.2 × 10^{-4} 秒
1億	2 秒	0.002 秒
1兆	5.55 時間	20 秒
1京	6.3 年	55.5 時間

naive_gcd の計算時間見積もり

- $n = \min(x, y)$ とすると，計算ステップ数は高々 $2n$
- 剰余の1回の計算時間
  - パソコン 1ステップ 10^{-8} 秒
  - スパコン 1ステップ 10^{-11} 秒

n	パソコン	スパコン
10000	0.2 × 10^{-3} 秒	0.2 × 10^{-6} 秒
100万	0.02 秒	0.2 × 10^{-4} 秒
1億	2 秒	0.002 秒
1兆	5.55 時間	20 秒
1京	6.3 年	55.5 時間
10^{50}	莫大	6.3 × 10^{31} 年
10^{100} ($\approx 2^{332}$)	莫大	6.3 × 10^{81} 年



高速化への道 ～ユークリッドの互除法

- 最大公約数問題を，
  「なるべく大きな正方形で長方形を埋める問題」
  に置き換えて考える．

問題

縦の長さ x，横の長さ y の長方形が与えられたとき，長方形を敷き詰める最大の正方形 S を求めよ．

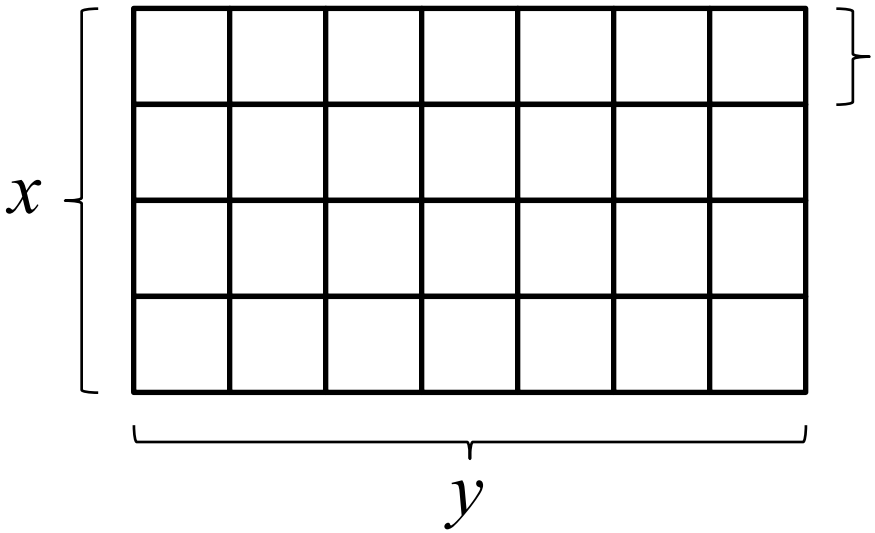
x，y は正整数

長方形敷き詰め問題

問題

縦の長さ x，横の長さ y の長方形が与えられたとき，長方形を敷き詰める最大の正方形 S を求めよ．

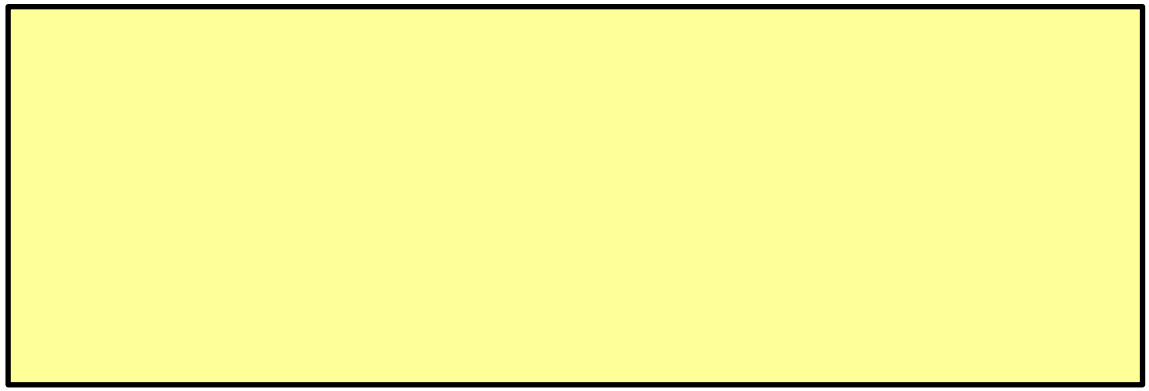
x，y は正整数



- 長方形を埋める正方形の一辺の長さ＝ x と y の公約数
- そのような最大の正方形 S の一辺の長さ＝ $\gcd(x, y)$

例 1

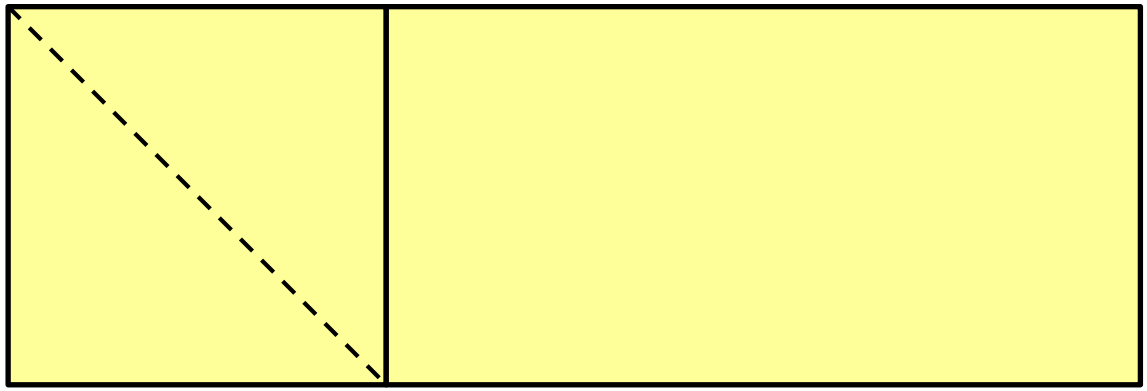
- 以下の長方形をなるべく大きな正方形に分割せよ.



- **ただし定規を使ってはいけない!!**

例１（解答）

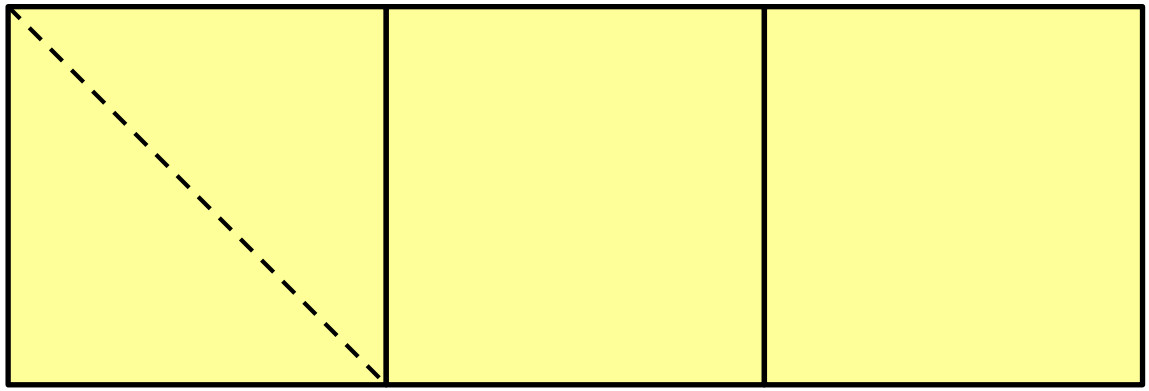
- 以下の長方形をなるべく大きな正方形に分割せよ．



  - なるべく大きな正方形を作る。

例１（解答）

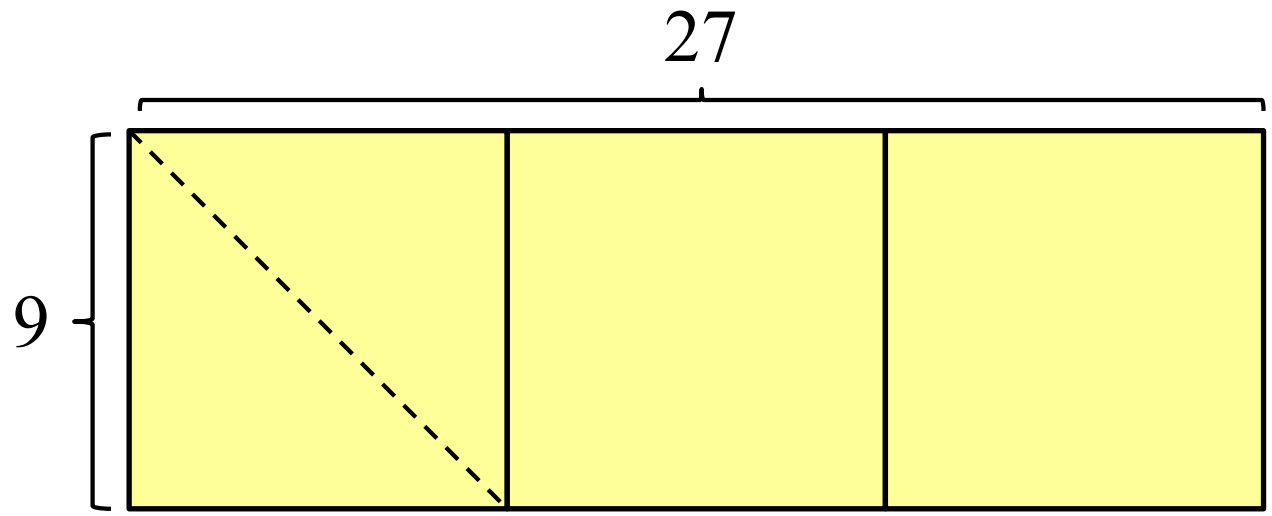
- 以下の長方形をなるべく大きな正方形に分割せよ.



- 正方形を入るだけ入れる＝長い辺を短い辺で割る.
- ぴったり割れたら，短い辺の長さが最大公約数.

例１（解答）

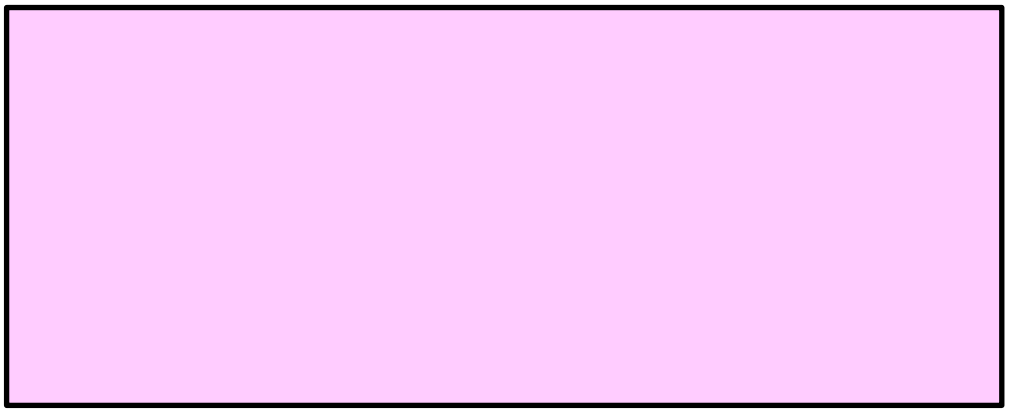
- 以下の長方形をなるべく大きな正方形に分割せよ．



- 正方形を入るだけ入れる＝長い辺を短い辺で割る．
- ぴったり割れたら，短い辺の長さが最大公約数．

例 2

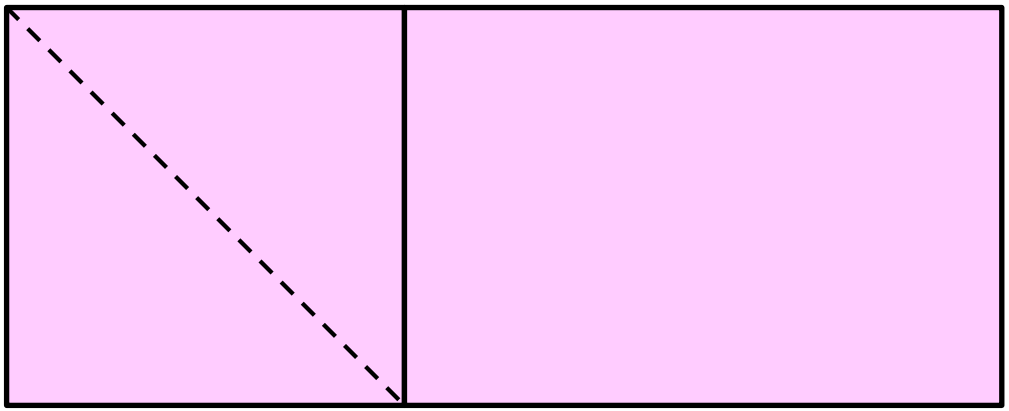
- 以下の長方形をなるべく大きな正方形に分割せよ.



- **ただし定規を使ってはいけない!!**

例２（解答）

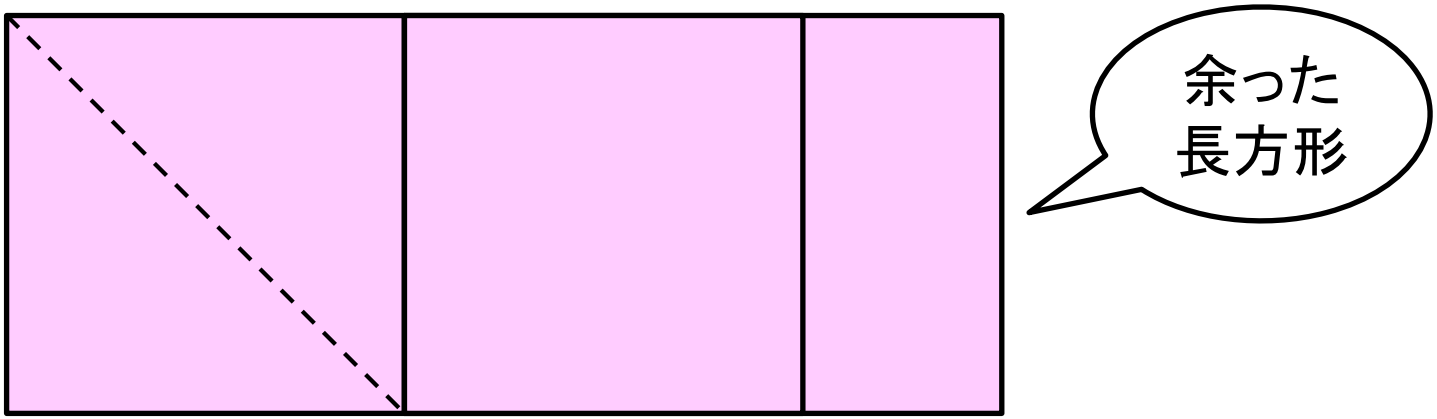
- 以下の長方形をなるべく大きな正方形に分割せよ．



  - なるべく大きな正方形を作る。

例２（解答）

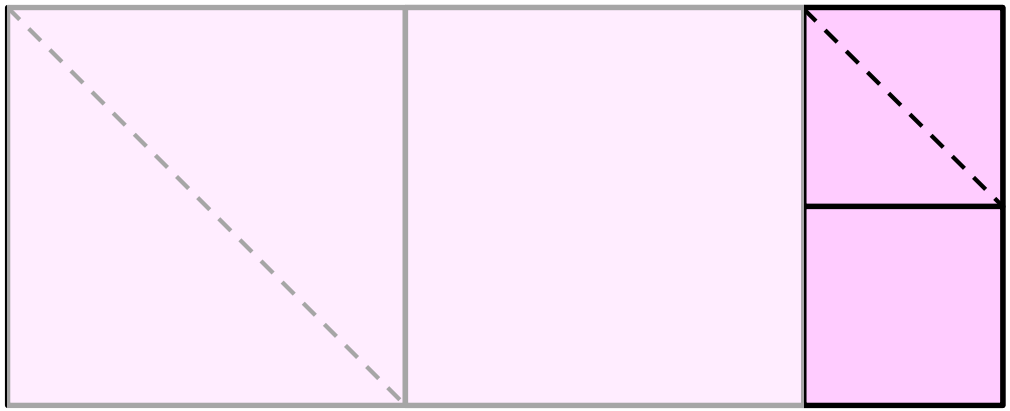
- 以下の長方形をなるべく大きな正方形に分割せよ．



  - 正方形を入るだけ入れる＝長い辺を短い辺で割る．
  - ぴったり割れないときは，余った長方形の長い辺を短い辺で割る．

例２（解答）

- 以下の長方形をなるべく大きな正方形に分割せよ．

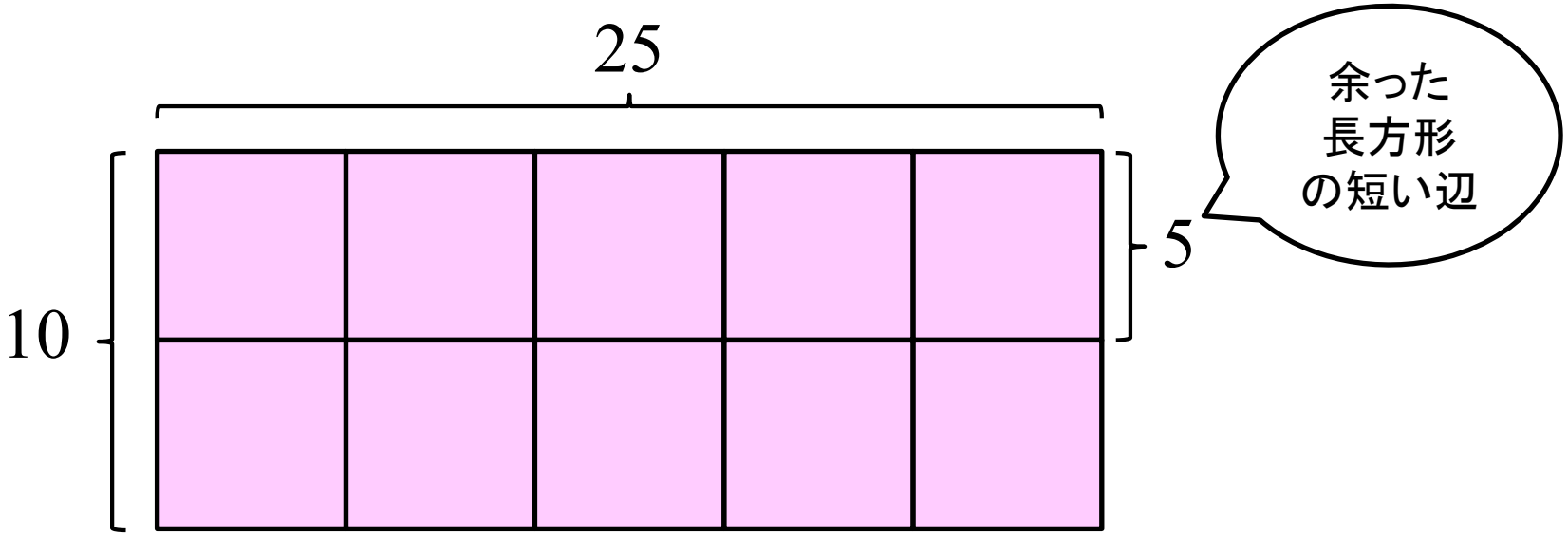


余った
長方形
の短い辺

- 正方形を入るだけ入れる＝長い辺を短い辺で割る．
- ぴったり割れたら，短い辺の長さが最大公約数．

例２（解答）

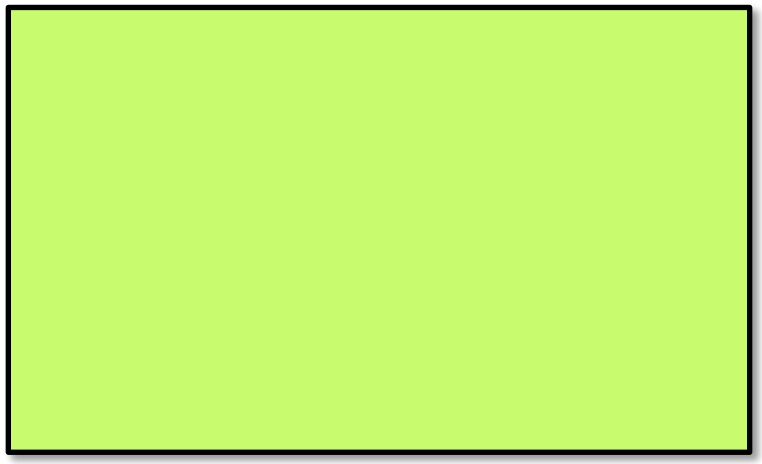
- 以下の長方形をなるべく大きな正方形に分割せよ．



- 正方形を入るだけ入れる＝長い辺を短い辺で割る．
- ぴったり割れたら，短い辺の長さが最大公約数．

例 3

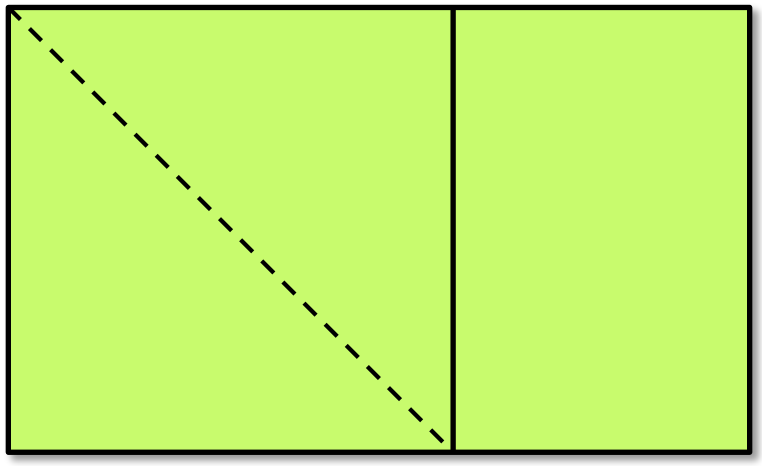
- 長方形の紙をなるべく大きな正方形に分割せよ.



- **ただし定規を使ってはいけない!!**

例３（解答）

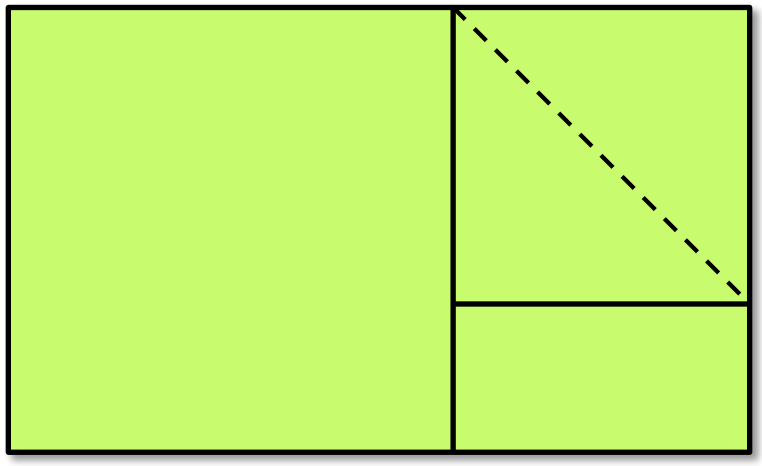
- 長方形の紙をなるべく大きな正方形に分割せよ．
- ただし，最初に定規で辺の長さを測ってはいけない．



- 長い辺を短い辺で割る．
- ぴったり割れないときは，余った長方形の長い辺を短い辺で割る．

例３（解答）

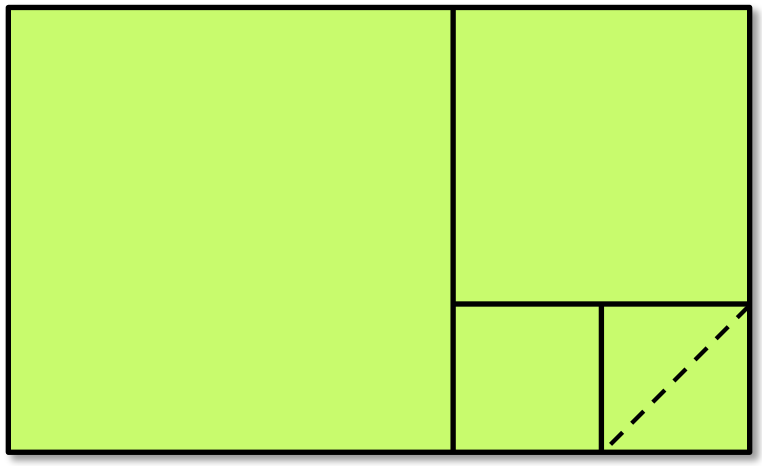
- 長方形の紙をなるべく大きな正方形に分割せよ．
- ただし，最初に定規で辺の長さを測ってはいけない．



- 長い辺を短い辺で割る．
- ぴったり割れないときは，余った長方形の長い辺を短い辺で割る．

例３（解答）

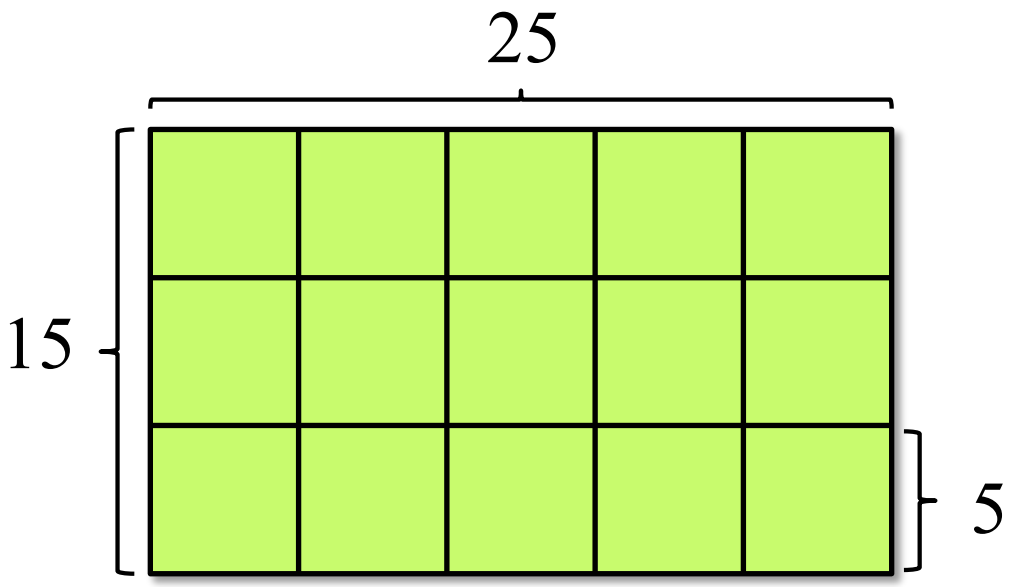
- 長方形の紙をなるべく大きな正方形に分割せよ．
- ただし，最初に定規で辺の長さを測ってはいけない．



- ぴったり割れたら，短い辺の長さが最大公約数．

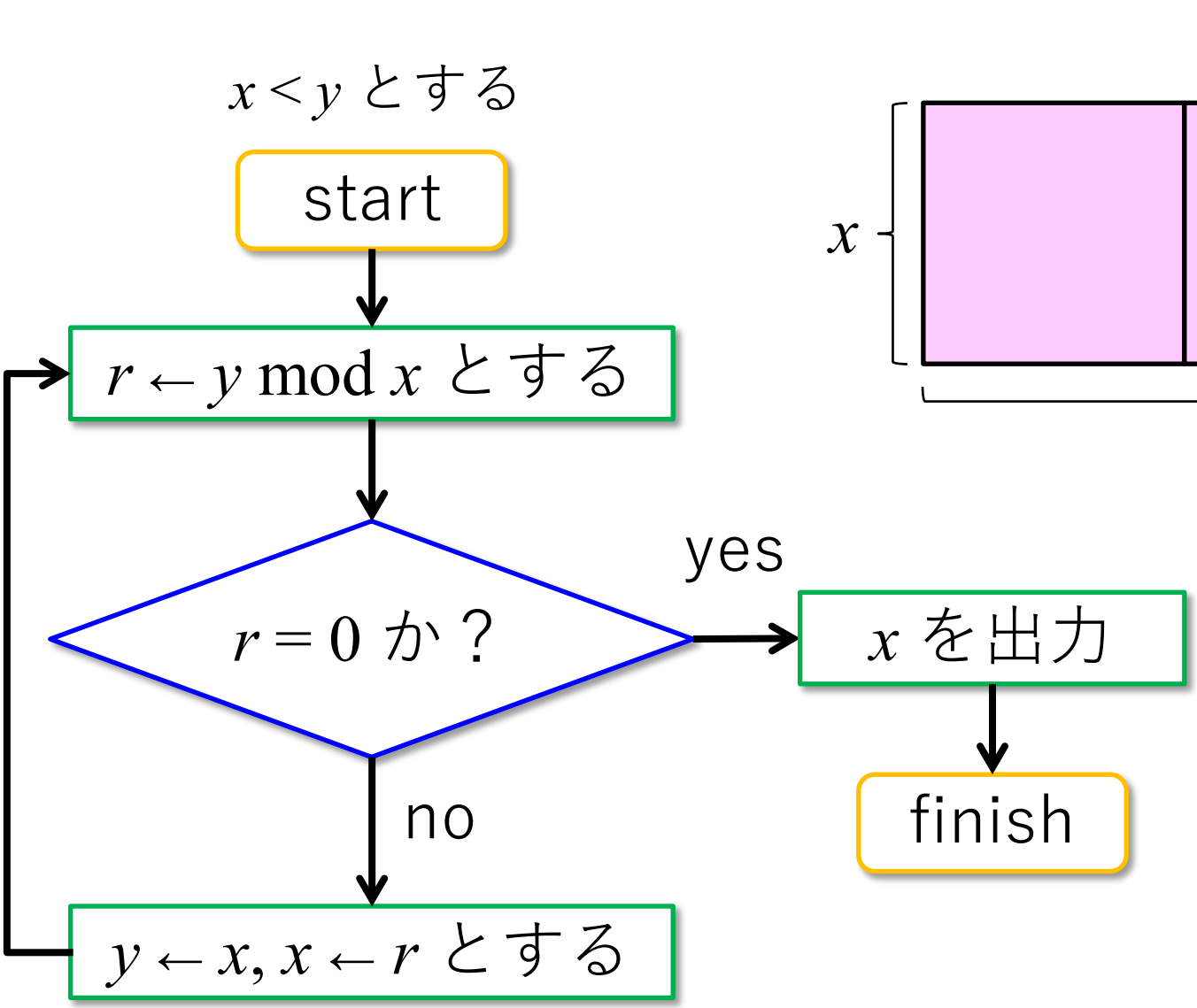
例３（解答）

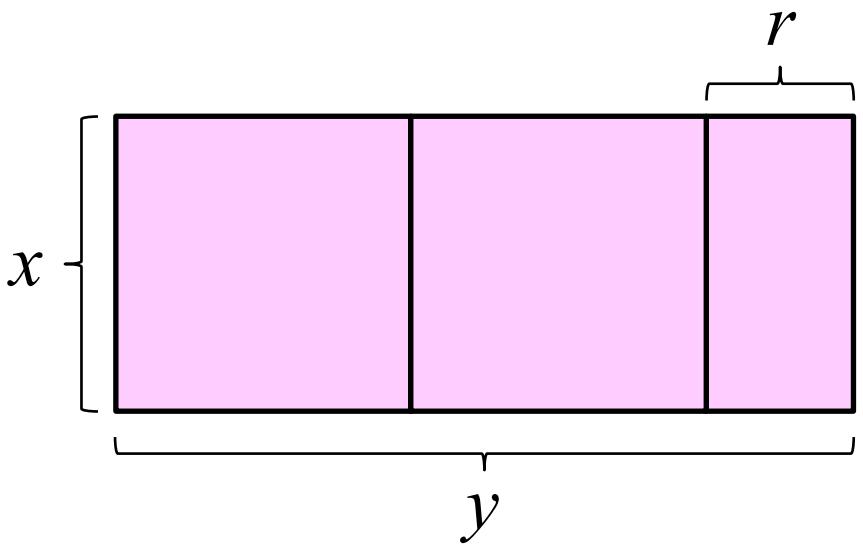
- 長方形の紙をなるべく大きな正方形に分割せよ．
- ただし，最初に定規で辺の長さを測ってはいけない．



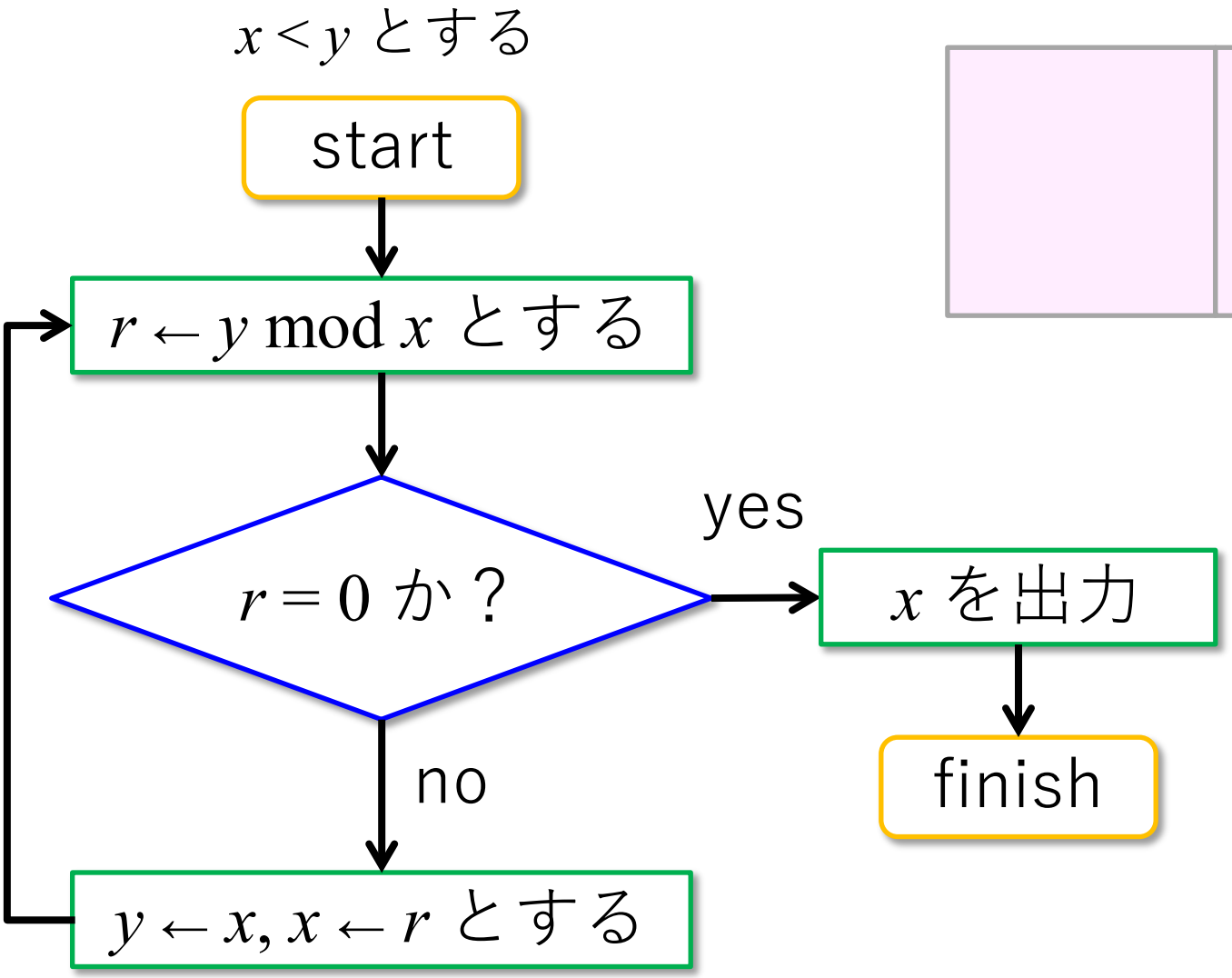
- ぴったり割れたら，短い辺の長さが最大公約数．

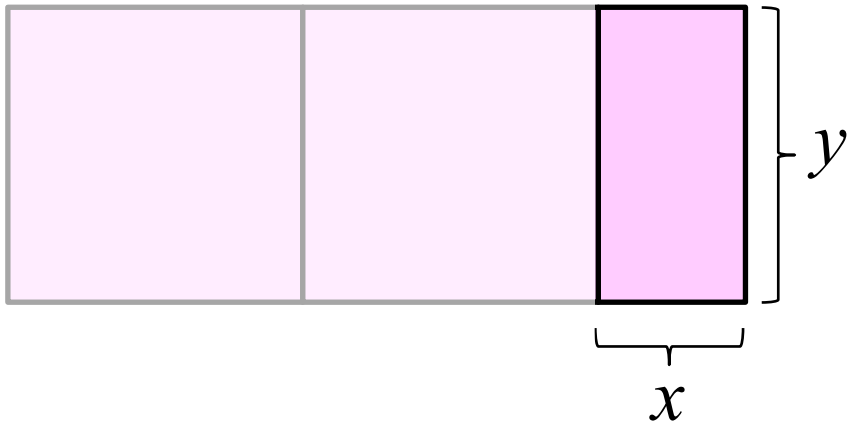
賢いアルゴリズム euclid_gcd



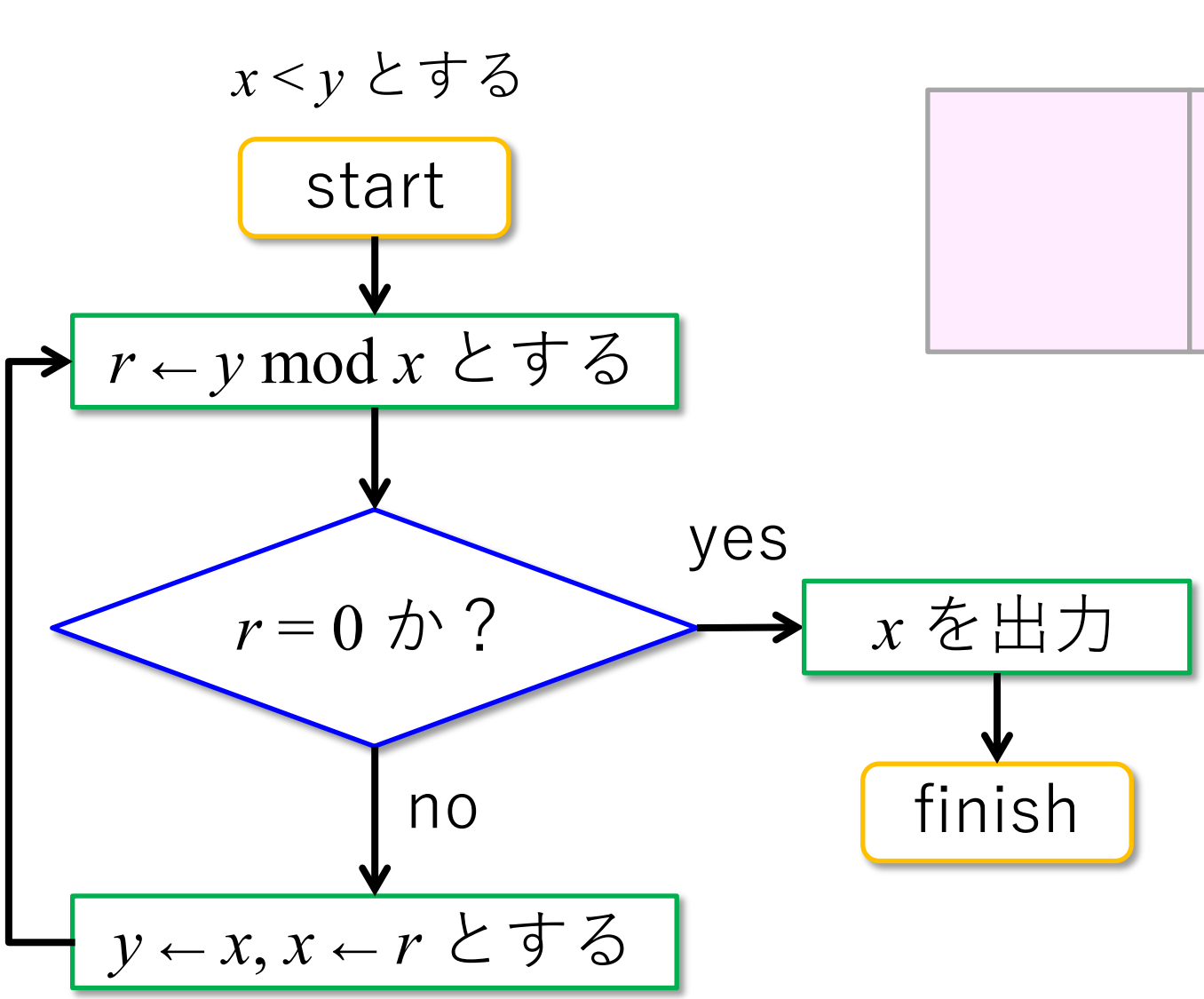


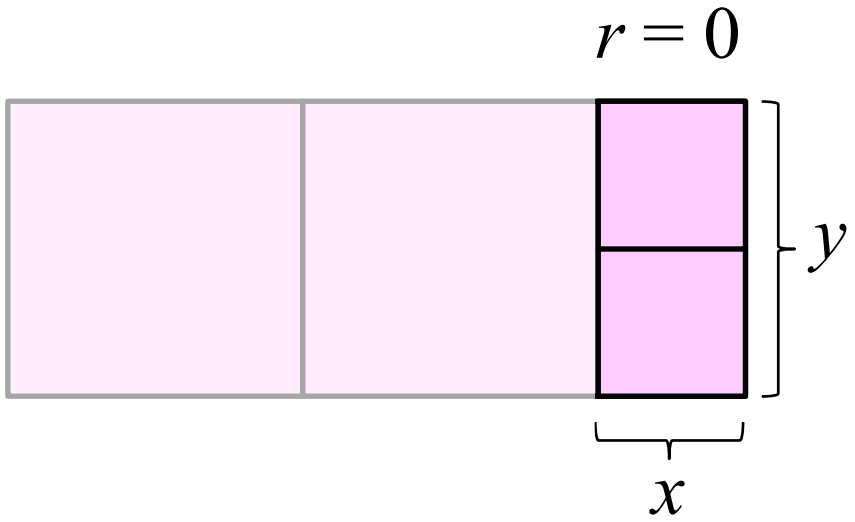
賢いアルゴリズム euclid_gcd



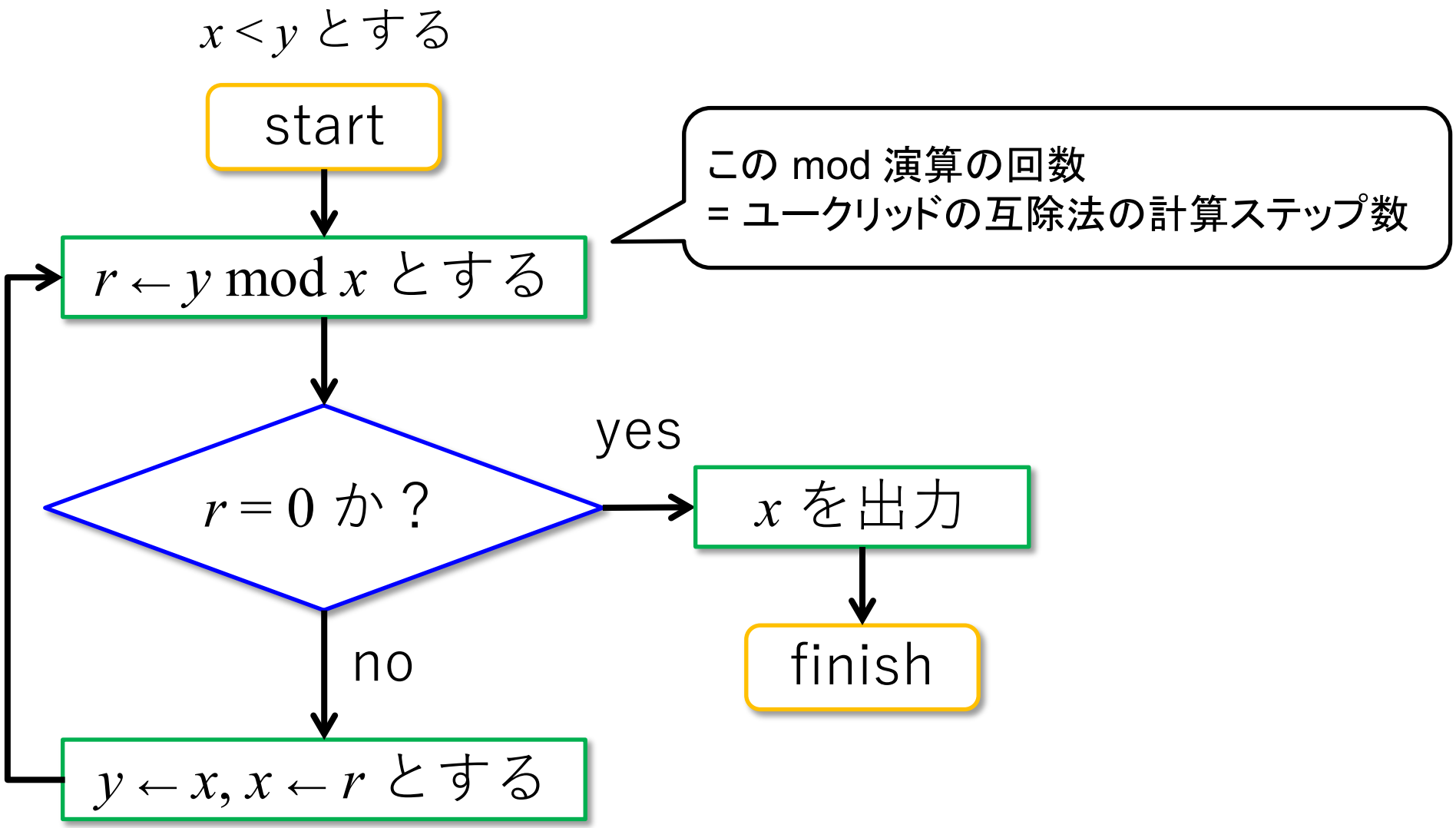


賢いアルゴリズム euclid_gcd





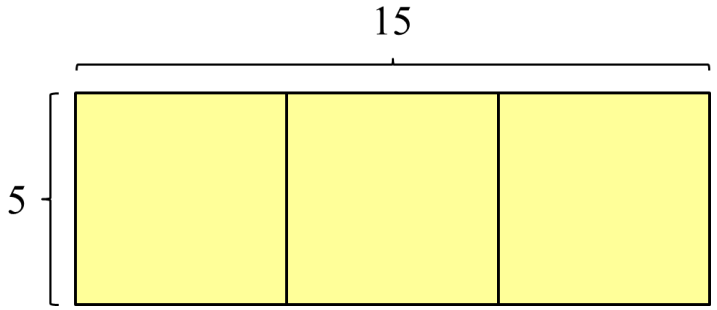
ユークリッドの互除法の計算ステップ



ユークリッドの互除法の計算ステップ

$x = 5, y = 15$ のとき
ステップ1： $15 \bmod 5 = 0$

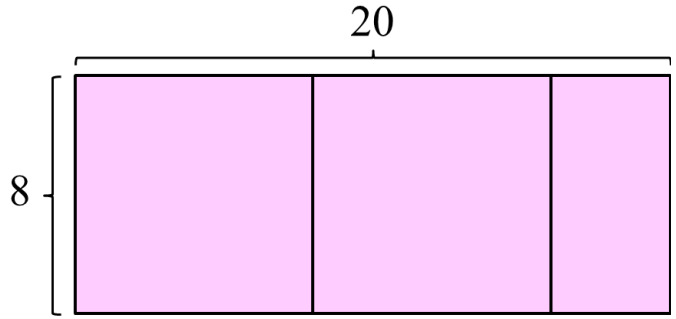
➔ 計算ステップ数 = 1



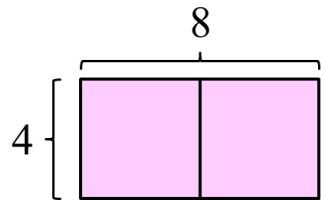
ステップ1

$x = 8, y = 20$ のとき
ステップ1： $20 \bmod 8 = 4$
ステップ2： $8 \bmod 4 = 0$

➔ 計算ステップ数 = 2

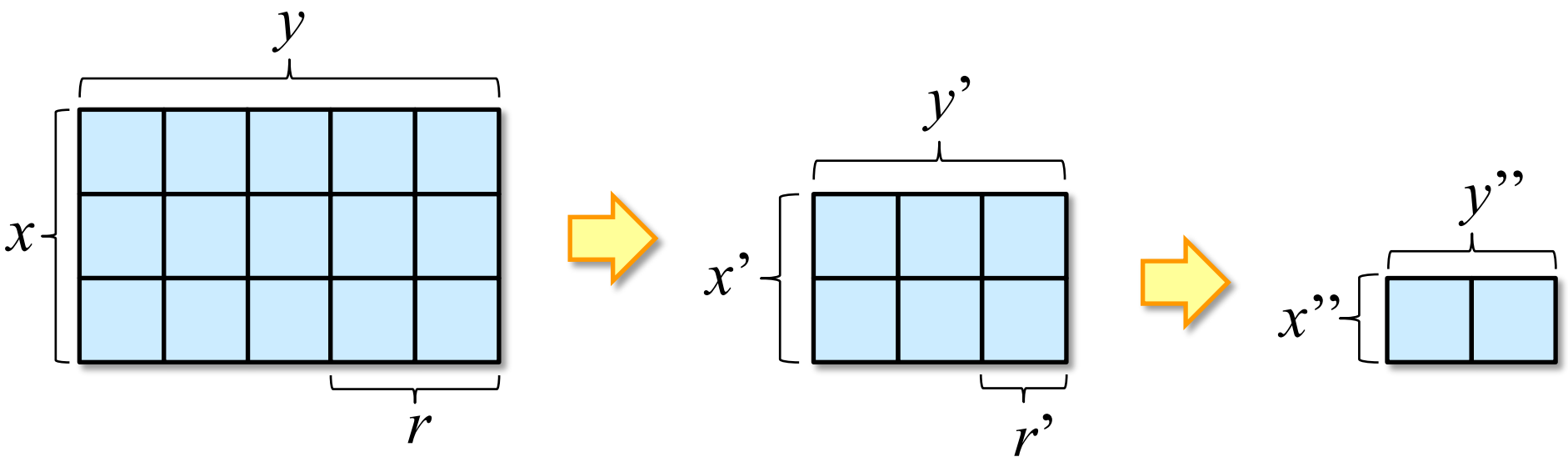


ステップ1



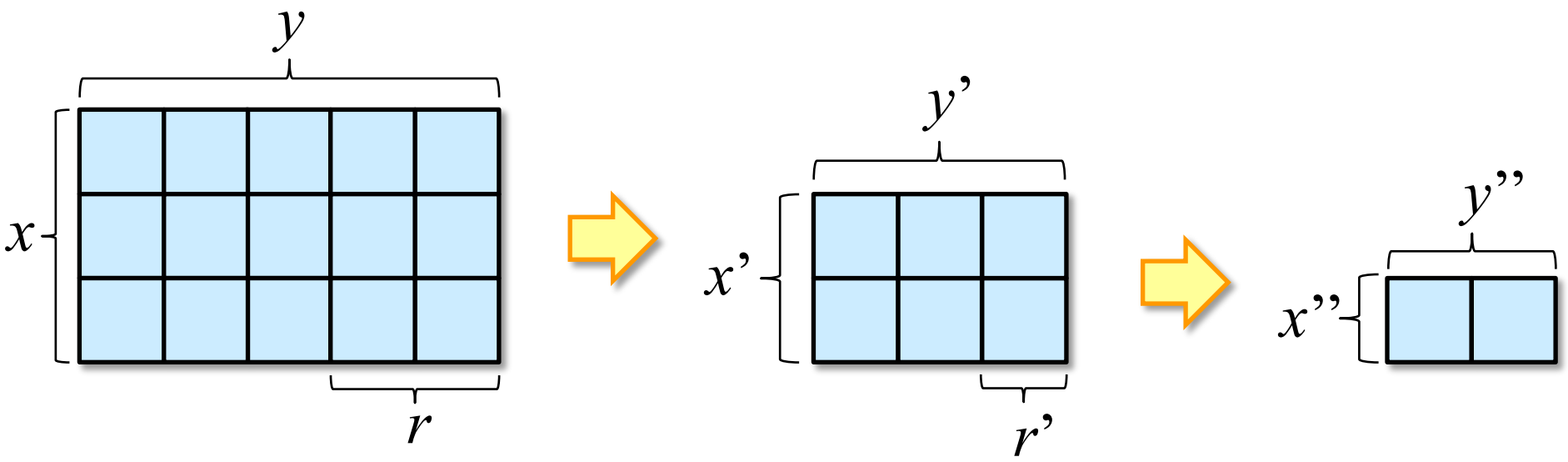
ステップ2

ユークリッドの互除法の計算ステップ数



- y と x の更新を 2 回繰り返すと，元の値の半分未満になる.
  - $y = zx + r$ とおく.
  - $y > x$ より $z \geq 1$ ➔ $y = zx + r \geq x + r$.
  - y は x で割り切れていないので $x > r$ ➔ $y > 2r = 2y''$ ➔ $\frac{y}{2} > y''$
  - x についても同様.

ユークリッドの互除法の計算ステップ数



- y と x の更新を 2 回繰り返すと，元の値の半分未満になる．
- x が 1 になったら，x は必ず y を割り切れる．
  ➔ x が 1 になるか，その前までに，必ず計算が終わる．
- よって，$n = \min(x, y)$ とすると，ユークリッドの互除法の計算ステップ数は高々 $2\log_2 n$ である．

ユークリッドの互除法の計算時間見積もり

- ユークリッドの互除法の計算ステップ数 $\approx 2\log_2 n$
- 剰余の1回の計算時間
  - パソコン 1ステップ 10^{-8}秒
  - スパコン 1ステップ 10^{-11}秒

n	naive_gcd スパコン	euclid_gcd パソコン	euclid_gcd スパコン
1億	0.002 秒	0.56×10^{-3} 秒	0.56×10^{-6} 秒
1兆	20 秒	0.82×10^{-3} 秒	0.82×10^{-6} 秒
1京	55.5 時間	1.08×10^{-3} 秒	1.08×10^{-6} 秒
10^{50}	6.3×10^{31} 年	3.34×10^{-3} 秒	3.34×10^{-6} 秒
10^{100} ($\approx 2^{332}$)	6.3×10^{81} 年	6.66×10^{-3} 秒	6.66×10^{-6} 秒

ユークリッドの互除法の正当性

- $\gcd(r, x)$ を求めることで,
  なぜ $\gcd(x, y)$ が求まるのだろうか？

数学Aの教科書にも載っています

定理

正整数 x と y について,
$y = zx + r$ かつ $0 < r < x$ であるとき
(つまり $y \bmod x = r$ かつ $r \neq 0$ のとき),
$\gcd(x, y) = \gcd(r, x)$ が成り立つ.

ユークリッドの互除法の正当性

【定理の証明の方針】

(i) $\gcd(x, y) \geqq \gcd(r, x)$ と

(ii) $\gcd(x, y) \leqq \gcd(r, x)$ が

両方とも成り立つことを示して,

$\gcd(x, y) = \gcd(r, x)$ を導く.

ユークリッドの互除法の正当性

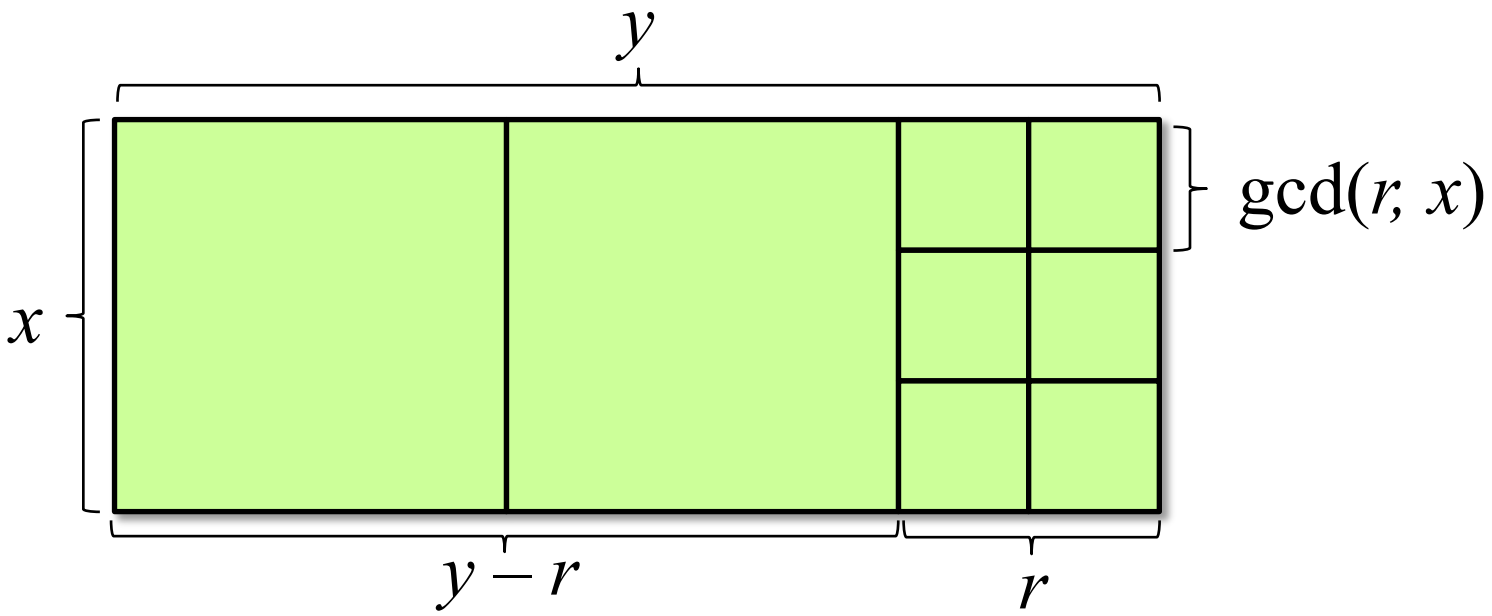
証明(i): r は $\gcd(r, x)$ で割り切れ，$y - r$ は x で割り切れる．

x も $\gcd(r, x)$ で割り切れるので，

$r + (y - r) = y$ は $\gcd(r, x)$ で割り切れる．

よって，$\gcd(r, x)$ は x と y の公約数である．

最大公約数の定義より，$\gcd(x, y) \geqq \gcd(r, x)$ である．



ユークリッドの互除法の正当性

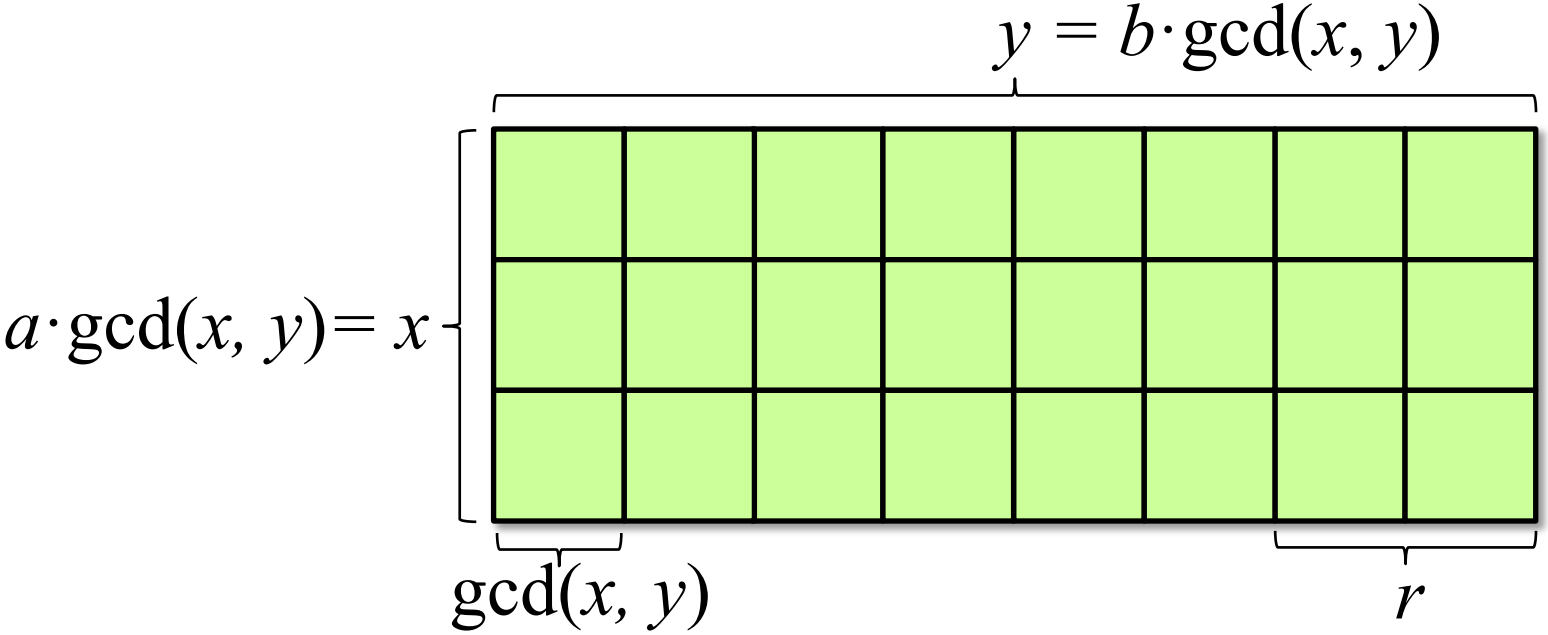
証明(ii): $x = a \cdot \gcd(x, y), y = b \cdot \gcd(x, y)$ とおくと,

$y - zx = \gcd(x, y) \cdot (b - za) = r > 0$ が成り立つ.

ここで b, z, a は整数なので, r は $\gcd(x, y)$ で割り切れる.

よって, $\gcd(x, y)$ は r と x の公約数である.

最大公約数の定義より, $\gcd(x, y) \leqq \gcd(r, x)$である.

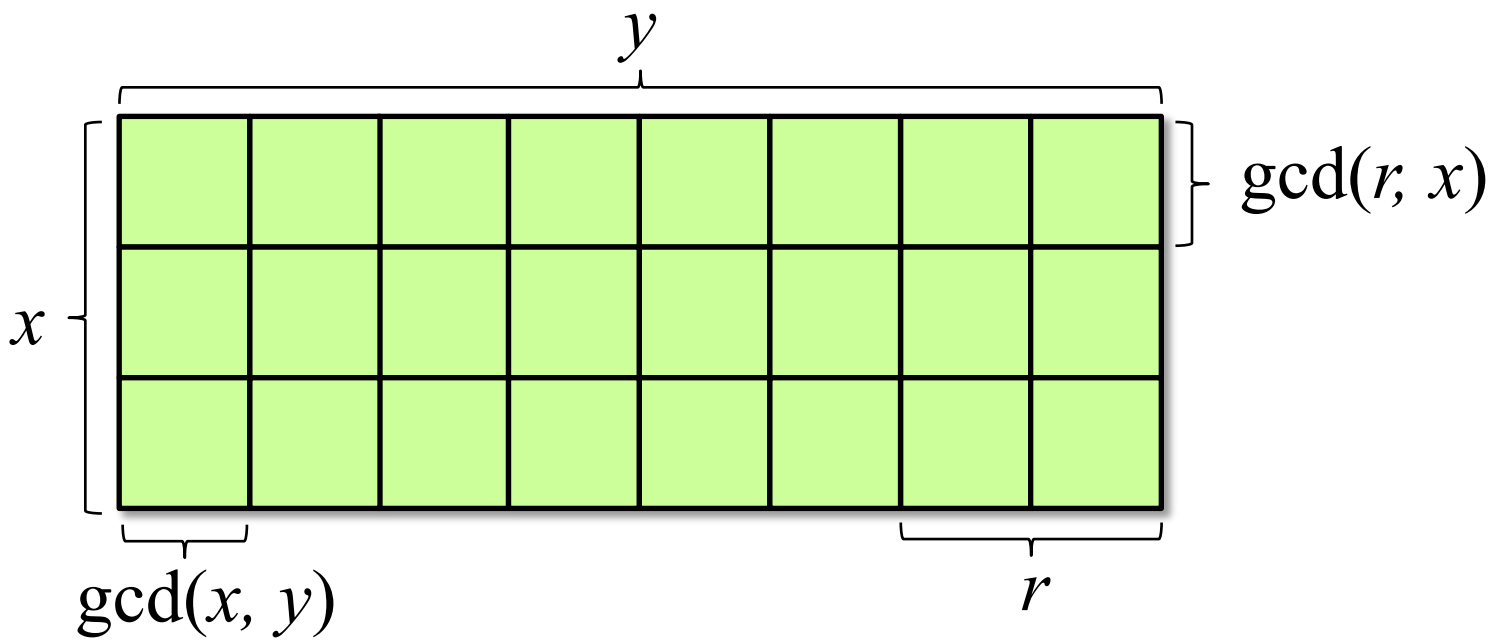


ユークリッドの互除法の正当性

【証明つづき】

$\gcd(x, y) \geqq \gcd(r, x)$ かつ $\gcd(x, y) \leqq \gcd(r, x)$,

すなわち $\gcd(x, y) = \gcd(r, x)$ である. （証明終）



まとめ

大学以降の情報系科目では **数学** を駆使します

- 例）整数，集合，論理，確率，統計など

計算理論：アルゴリズムの正しさや計算時間を
論理的に証明 ➔ **プログラム性能保証**

情報理論：**確率統計**に基づく情報エントロピー，
データ圧縮，暗号理論 ➔ **情報通信技術**

人工知能理論：機械学習のための**微分積分**，
線形代数，**確率統計** ➔ **AI技術**

数学抜きで情報科学を正しく理解することは
できない ➔ 数学はやっぱり大事！