

Podzielność

Definicja

Mówimy, że liczba całkowita a jest **podzielna** przez liczbę całkowitą b , jeżeli istnieje taka liczba całkowita k , że

$$a = kb.$$

Piszemy wtedy

$$\underline{b|a.}$$

Inaczej mówiąc, mamy

$$b|a \iff \bigvee_{k \in \mathbb{Z}} a = kb.$$

$$\boxed{0|0}$$
$$0 = k \cdot 0$$

$$a = k \cdot 0$$

Podzielność

- ⇒ $b|a$,
- ⇒ b dzieli a ,
- ⇒ b jest dzielnikiem a ,
- ⇒ b jest czynnikiem a ,
- ⇒ a jest podzielne przez b ,
- ⇒ a jest wielokrotnością b .

Przypadki szczególne

$$\rightsquigarrow \bigwedge_{n \in \mathbb{Z}} n|0,$$

$$\rightsquigarrow \bigwedge_{n \in \mathbb{Z}} n|n,$$

$$\rightsquigarrow \bigwedge_{n \in \mathbb{Z}} \pm 1|n,$$

$$\rightsquigarrow \bigwedge_{n \in \mathbb{Z}} 0|n \Rightarrow n = 0.$$

$$a \quad k \quad b$$

$$0 = 0 \cdot n$$

$$n = 1 \cdot n$$

$$n = n \cdot 1$$

$$n = (-n) \cdot (-1)$$

Własności relacji podzielności

$(\mathbb{N}, |)$ relacja porządku

Twierdzenie

$|$ jest relacją w \mathbb{Z} oraz

- jeżeli $a|b$ i $b|c$, to $a|c$,
- jeżeli $a|b$ i $a|c$, to dla dowolnych liczb całkowitych s i t zachodzi $a|sb + tc$,
- dla dowolnej liczby całkowitej $c \neq 0$,

$$a|b \iff ca|cb.$$

Ćwiczenie

Udowodnić powyższe twierdzenie.

$$\left. \begin{array}{l} \forall_{k \in \mathbb{Z}} b = k \cdot a \\ \forall_{k' \in \mathbb{Z}} c = k' \cdot b \end{array} \right\} \Rightarrow \begin{array}{l} c = k' \cdot b = \\ = k' \cdot k \cdot a \\ = (k' \cdot k) \cdot a \\ \underbrace{\quad}_{\in \mathbb{Z}} \\ a|c \end{array}$$

Dzielenie z resztą

Twierdzenie o dzieleniu z resztą

Niech $n \in \mathbb{Z}$ oraz $d \in \mathbb{N}$. Istnieje wtedy dokładnie jedna para liczb całkowitych q i r , dla której

$$n = qd + r \quad \text{oraz} \quad \underline{0 \leq r < d.}$$

Dzielenie z resztą

Twierdzenie o dzieleniu z resztą

Niech $n \in \mathbb{Z}$ oraz $d \in \mathbb{N}$. Istnieje wtedy dokładnie jedna para liczb całkowitych q i r , dla której

$$n = qd + r \quad \text{oraz} \quad 0 \leq r < d.$$

Dowód

~> Istnienie.

~> Jedyność.

Przykłady

$$\rightsquigarrow 14 = 2 \cdot 6 + 2,$$

$$\rightsquigarrow 31 = \textcircled{4} \cdot 7 + \textcircled{3},$$

$$\rightsquigarrow 55 = \textcircled{3} \cdot \textcircled{15} + \textcircled{10}.$$

~~$$14 = 1 \cdot 6 + \textcircled{8}$$~~

$$n \in \mathbb{Z}, \quad d \in \mathbb{N}$$

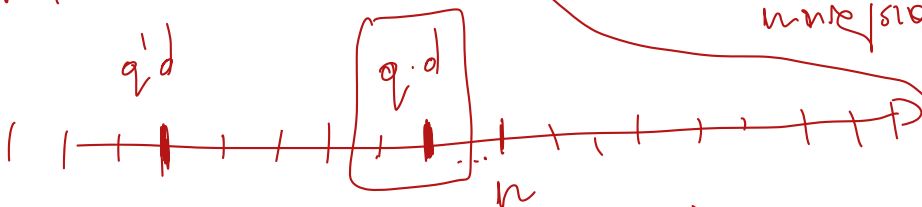
\forall

$$q, r \in \mathbb{Z}$$

$$n = q \cdot d + r \wedge$$

$$0 \leq r < d$$

1) Istnienie



$q \cdot d$ - "multiples" luba
mnożenia n

lub $r \geq 0$

$$q \cdot d \leq n$$

$$n - q \cdot d \geq 0$$

$$n = q \cdot d + \underbrace{(n - q \cdot d)}_r$$

$$r \geq 0? \quad \checkmark$$

$$q \cdot d \leq n < (q+1)d \quad | - q \cdot d$$

$$r < d? \quad \checkmark$$

$$0 \leq r < d$$

2) Jedyność

$$0 \leq r < d$$

$$0 \leq r' < d \quad ?$$

$$n = q \cdot d + r = q' \cdot d + r' \Rightarrow q' = q \wedge r' = r$$

$$\left. \begin{array}{l} n = q \cdot d + r \\ n = q' \cdot d + r' \end{array} \right\} \Rightarrow 0 = q \cdot d + r - (q' \cdot d + r') =$$

$$= (q - q') \cdot d + r - r'$$

$$\Rightarrow (q' - q) \cdot d = r - r' \rightarrow (q' - q) \cdot d = 0$$

jest podzielne
przez d

$$\Rightarrow d \mid r - r'$$

$$0 \leq r, r' < d$$

$$\Rightarrow -d + 0 < r - r' < d - 0$$

$$-d < r - r' < d$$

$$\left. \begin{array}{l} r - r' = 0 \\ r = r' \end{array} \right\}$$

$$r - r' \in \{-d+1, -d+2, \dots, -1, 0, 1, \dots, d-2, d-1\} \in \mathbb{Z}$$

Podłoga i sufit

⇒ Funkcja **podłoga**

$\lfloor x \rfloor :=$ największa liczba całkowita mniejsza lub równa x .

⇒ Funkcja **sufit**

$\lceil x \rceil :=$ najmniejsza liczba całkowita większa lub równa x .

$$\lfloor \frac{1}{2} \rfloor = 0$$

$$\lceil \frac{1}{2} \rceil = 1$$

$$\lfloor -\frac{4}{3} \rfloor = -2$$

$$\lceil -\frac{4}{3} \rceil = -1$$

$$\lfloor \sqrt{2} \rfloor = 1$$

Podłoga i sufit

⇒ Funkcja **podłoga**

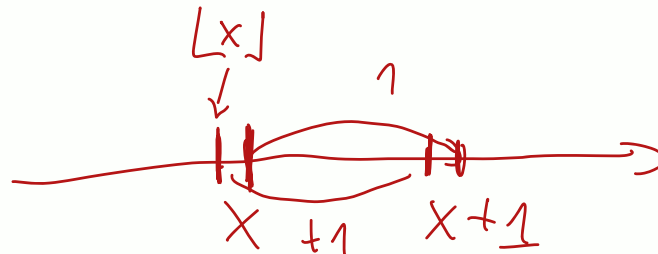
$\lfloor x \rfloor :=$ największa liczba całkowita mniejsza lub równa x .

⇒ Funkcja **sufit**

$\lceil x \rceil :=$ najmniejsza liczba całkowita większa lub równa x .

W szczególności dla dowolnej liczby $x \in \mathbb{R}$ mamy

$$\lfloor x \rfloor \leq x < \lfloor x + 1 \rfloor, \quad x \leq \lceil x \rceil < x + 1.$$



Ćw. 4. Wykresy.

Dzielenie z resztą: dowód

⇒ **Istnienie.** Niech

$$q := \left\lfloor \frac{n}{d} \right\rfloor, \quad r := n - qd.$$

Dzielenie z resztą: dowód

⇒ **Istnienie.** Niech

$$q := \left\lfloor \frac{n}{d} \right\rfloor, \quad r := n - qd.$$

Wtedy oczywiście

$$n = n - qd + qd = qd + r$$

oraz

$$\underbrace{q \leq \frac{n}{d} < q + 1}_{\text{dla } q = \left\lfloor \frac{n}{d} \right\rfloor} \Rightarrow qd \leq n < qd + d \Rightarrow 0 \leq r < d.$$

Dzielenie z resztą: dowód

⇒ **Jedyność.** Załóżmy, że dla q, q', r i r' zachodzi

$$n = qd + r = q'd + r', \quad 0 \leq r, r' < d.$$

Dzielenie z resztą: dowód

⇒ **Jedyność.** Załóżmy, że dla q, q', r i r' zachodzi

$$n = qd + r = q'd + r', \quad 0 \leq r, r' < d.$$

Wtedy

$$(q - q')d + (r - r') = 0.$$

Dzielenie z resztą: dowód

⇒ **Jedyność.** Załóżmy, że dla q, q', r i r' zachodzi

$$n = qd + r = q'd + r', \quad 0 \leq r, r' < d.$$

Wtedy

$$(q - q')d + (r - r') = 0.$$

Ponieważ $-d < r - r' < d$, to (dlaczego?) $r - r' = 0$.

Dzielenie z resztą: dowód

⇒ **Jedyność.** Załóżmy, że dla q, q', r i r' zachodzi

$$n = qd + r = q'd + r', \quad 0 \leq r, r' < d.$$

Wtedy

$$(q - q')d + (r - r') = 0.$$

Ponieważ $-d < r - r' < d$, to (dlaczego?) $r - r' = 0$. Wtedy również $(q - q')d = 0$, skąd $q - q' = 0$. Zatem $q = q'$ oraz $r = r'$.

Oznaczenia

Jeżeli

$$n = qd + r, \quad 0 \leq r < d,$$

to q nazywamy **ilorazem**, a r **resztą** z dzielenia n przez d . Piszemy również

$$q = n \operatorname{div} d, \quad r = n \operatorname{mod} d.$$

$$n / d$$

$$n \% d$$

Wzory

Niech $n \in \mathbb{Z}$, $d \in \mathbb{N}$. Wtedy

$$n \operatorname{div} d = \left\lfloor \frac{n}{d} \right\rfloor,$$

$$n \bmod d = \left(\frac{n}{d} - n \operatorname{div} d \right) d.$$

q

q

↑
„złoty dzielenie”

Wzory

Niech $n \in \mathbb{Z}$, $d \in \mathbb{N}$. Wtedy

$$n \operatorname{div} d = \left\lfloor \frac{n}{d} \right\rfloor, \quad n \operatorname{mod} d = \left(\frac{n}{d} - n \operatorname{div} d \right) d.$$

Jako wniosek otrzymujemy

$$n = (n \operatorname{div} d)d + n \operatorname{mod} d, \quad 0 \leq n \operatorname{mod} d < d.$$

$$n = q \cdot d + r$$

Algorytm dzielenia

1 **input:** $n \geq 0, d > 0$ $d \geq 1$

2 **output:** $q, r \in \mathbb{Z}, n = qd + r, 0 \leq r < d$

3 $q := 0$ $n = 0 \cdot d + n$

4 $r := n$

5 **while** $r \geq d$ **do**

6 $q := q + 1$

7 $r := r - d$

8 **end**

Algorytm dzielenia

```
1  input:  $n \geq 0, d > 0$   
2  output:  $q, r \in \mathbb{Z}, n = qd + r, 0 \leq r < d$   
3   $q := 0$   
4   $r := n$   
5  while  $r \geq d$  do  
6       $q := q + 1$   
7       $r := r - d$   
8  end
```

Dlaczego ten algorytm działa?

Algorytm dzielenia: przykład

Niech $n = 31$ i $d = 7$.

Algorytm dzielenia: przykład

Niech $n = 31$ i $d = 7$.

obrót pętli	q	r	$r \geq \overset{d}{\cancel{n}}$
0	0	31	1
1	1	24	1
2	2	17	1
3	3	10	1
4	4	3	0

$$r \geq d$$

$$31 = 4 \cdot 7 + 3$$

Niezmienniki pętli

Zdanie p jest **niezmiennikiem** pętli

```
1   while q do
2       S
3   end
```

jeżeli spełniony jest następujący warunek:

Niezmienniki pętli

Zdanie p jest **niezmiennikiem** pętli

```
1   while q do
2       S
3   end
```

jeżeli spełniony jest następujący warunek:

Jeśli zdania p i q są prawdziwe zanim wykonamy kroki S , to zdanie p będzie prawdziwe po wykonaniu S .

$$p \wedge q \xrightarrow{S} p$$

Niezmienniki pętli

Twierdzenie o niezmiennikach (R. Floyd, 1967 r.)

Założmy, że p jest niezmiennikiem pętli

```
1   while q do
2       S
3   end
```

oraz, że zdanie p jest prawdziwe przed wejściem w pętlę. Wtedy zdanie p jest prawdziwe po każdej iteracji pętli. Jednocześnie jeśli pętla się kończy, to po jej zakończeniu zdanie p jest prawdziwe, a zdanie q fałszywe.

Algorytm dzielenia

$p \wedge \neg q :$

$$n = q \cdot d + r \wedge r \geq 0 \wedge r < d$$

3) $r' = r - d < r$

po 11 sp bnie

4) \Rightarrow po zlozonym
p11 mery

1 **input:** $n \geq 0, d > 0$

2 **output:** $q, r \in \mathbb{Z}, n = qd + r, 0 \leq r < d$

3 $q := 0$

4 $r := n$

5 $\# n = q \cdot d + r \wedge r \geq 0$

6 **while** $r \geq d$ **do**

7 $q := q + 1$

8 $r := r - d$

9 **end**

1) p jest prawdziwe przed while { p: $n = 0 \cdot d + n$

2) $p \wedge q \xrightarrow{S} p, n = q \cdot d + r \wedge r \geq 0 \wedge r \geq d$

$$\begin{aligned} q' &= q + 1 \\ r' &= r - d \\ n &= (q+1)d + r - d \\ r' &= r - d \Rightarrow r' \geq 0 \end{aligned}$$

Algorytm dzielenia

```
1  input:  $n \geq 0, d > 0$ 
2  output:  $q, r \in \mathbb{Z}, n = qd + r$ 
3   $q := 0$ 
4   $r := n$ 
5  # niezmiennik:  $qd + r = n, r \geq 0$ 
6  while  $r \geq d$  do
7       $q := q + 1$ 
8       $r := r - d$ 
9  end
```

Algorytm dzielenia: dowód poprawności

- ⇒ Zdanie $qd + r = n \wedge r \geq 0$ jest niezmiennikiem pętli, ponieważ $r - d \geq 0$ dla $r \geq d$ oraz

$$(q + 1)d + (r - d) = qd + r + d - d = n.$$

- ⇒ Przed wykonaniem pętli zdanie $qd + r = n \wedge r \geq 0$ jest prawdziwe, gdyż

$$0 \cdot d + n = n \quad \wedge \quad n \geq 0.$$

- ⇒ Algorytm się zatrzymuje, ponieważ

$$r - d < r.$$

- ⇒ Na mocy twierdzenia o niezmiennikach po zakończeniu mamy

$$qd + r = n \quad \wedge \quad r \geq 0 \quad \wedge \quad \neg(r \geq d)$$

czyli

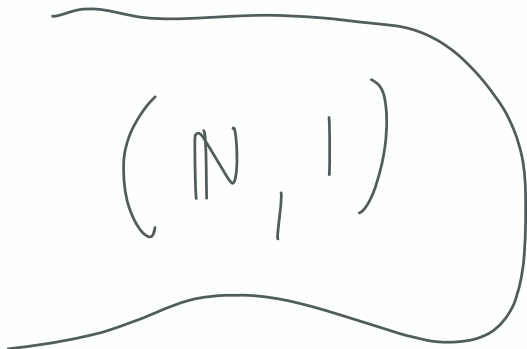
$$qd + r = n \quad \wedge \quad 0 \leq r < d.$$

Największy wspólny dzielnik

Niech m i n będą liczbami całkowitymi.

- ~> Zbiór wspólnych dzielników dodatnich liczb m i n jest niepusty ponieważ $1|m$ i $1|n$.
- ~> Jeżeli $m \neq 0$ lub $n \neq 0$, to liczby m i n mają tylko skończenie wiele wspólnych dzielników dodatnich.
- ~> Jeżeli $m \neq 0$ lub $n \neq 0$, to największy wspólny dzielnik dodatni liczb m i n oznaczamy przez

$$\text{NWD}(m, n) = \gcd(m, n)$$



A hand-drawn diagram consisting of a large, irregular oval shape. Inside the oval, the expression $(\mathbb{N}, 1)$ is written in a simple, handwritten style.

Największy wspólny dzielnik

Niech m i n będą liczbami całkowitymi.

- ⇒ Zbiór wspólnych dzielników dodatnich liczb m i n jest niepusty ponieważ $1|m$ i $1|n$.
- ⇒ Jeżeli $m \neq 0$ lub $n \neq 0$, to liczby m i n mają tylko skończenie wiele wspólnych dzielników dodatnich.
- ⇒ Jeżeli $m \neq 0$ lub $n \neq 0$, to największy wspólny dzielnik dodatni liczb m i n oznaczamy przez

$$\text{NWD}(m, n).$$

Jak znaleźć $\text{NWD}(m, n)$?

NWD

Zadanie

$\text{NWD}(135, 120)?$

NWD

Zadanie

NWD(135, 120)?

135	3
45	3
15	3
5	5
1	

$$135 = \underline{3^3} \cdot \underline{5}, \quad 120 = 2^3 \cdot \underline{3} \cdot \underline{5}.$$
$$3^1 \cdot 5^1 = 15$$

NWD

Zadanie

$$\text{NWD}(135, 120)?$$

$$135 = 3^3 \cdot 5, \quad 120 = 2^3 \cdot 3 \cdot 5.$$

Stąd

$$\text{NWD}(135, 120) = 15.$$

NWD – wersja naiwna

$\{0, 1, 2, \dots\}$

$m > 0 \vee n > 0$

```
1  input:  $m, n \in \mathbb{N}_0$ ,  $m + n > 0$ 
2  output:  $d = \text{NWD}(m, n)$ 
3   $d := 1$ ,  $k := 2$ 
4  while  $k \leq m \wedge k \leq n$  do
5      if  $k | m \wedge k | n$  do
6           $d := d * k$ 
7           $m := m / k$ 
8           $n := n / k$ 
9      else
10          $k := k + 1$ 
11     end
12 end
```

NWD – wersja naiwna

```
1  input:  $m, n \in \mathbb{N}_0, m + n > 0$ 
2  output:  $d = \text{NWD}(m, n)$ 
3   $d := 1, k := 2$ 
4  while  $k \leq m \wedge k \leq n$  do
5      if  $k|m \wedge k|n$  do
6           $d := d * k$ 
7           $m := m/k$ 
8           $n := n/k$ 
9      else
10          $k := k+1$ 
11     end
12 end
```

Dla $m, n \sim 2^{100}$ pętla może obrócić się około 2^{100} razy.

Algorytm Euklidesa

$$m \geq 0 \quad n \geq 1$$

Twierdzenie

Załóżmy, że $m \in \mathbb{N}_0$ i $n \in \mathbb{N}$. Wtedy zbiór wspólnych dzielników liczb m i n jest taki sam jak zbiór wspólnych dzielników liczb n i $(m \bmod n)$.

Algorytm Euklidesa

Twierdzenie

Założmy, że $m \in \mathbb{N}_0$ i $n \in \mathbb{N}$. Wtedy zbiór wspólnych dzielników liczb m i n jest taki sam jak zbiór wspólnych dzielników liczb n i $(m \bmod n)$.

Dowód.

~> Wystarczy sprawdzić (dlaczego?), że

$$\bigwedge_{k \in \mathbb{N}} (k|m \wedge k|n) \iff [k|n \wedge k|(m \bmod n)].$$



Algorytm Euklidesa

Twierdzenie

Założmy, że $m \in \mathbb{N}_0$ i $n \in \mathbb{N}$. Wtedy zbiór wspólnych dzielników liczb m i n jest taki sam jak zbiór wspólnych dzielników liczb n i $(m \bmod n)$.

Dowód.

~> Wystarczy sprawdzić (dlaczego?), że

$$\bigwedge_{k \in \mathbb{N}} (k|m \wedge k|n) \overset{\Rightarrow}{\iff} [k|n \wedge k|(m \bmod n)].$$

~> Powyższa równoważność wynika (dlaczego?) z równości

$$m = (m \operatorname{div} n)n + m \bmod n.$$

$m = qn + r$

$k|m \wedge k|n \Rightarrow k|m \bmod n$

Algorytm Euklidesa

$$\text{dzielniki}(m, n) = \text{dzielniki}(n, m \bmod n)$$

Wniosek

Jeżeli $m \in \mathbb{N}_0$ i $n \in \mathbb{N}$, to

$$\text{NWD}(m, n) = \text{NWD}(n, m \bmod n).$$

Algorytm Euklidesa

Wniosek

Jeżeli $m \in \mathbb{N}_0$ i $n \in \mathbb{N}$, to

$$n \geq 1$$

$$\text{NWD}(n, 0) = n$$

$$\text{NWD}(m, n) = \text{NWD}(n, m \bmod n).$$

Przykłady:

$$\text{NWD}(135, 120) = \text{NWD}(120, 15) = \text{NWD}(15, 0) = 15,$$

$$\left(\begin{array}{c} (120, 135 \bmod 120) \\ \text{"} \\ (120, 15) \end{array} \right) \rightarrow \left(\begin{array}{c} (15, 120 \bmod 15) \\ \text{"} \\ 0 \end{array} \right)$$

Algorytm Euklidesa

Wniosek

Jeżeli $m \in \mathbb{N}_0$ i $n \in \mathbb{N}$, to

$$\text{NWD}(m, n) = \text{NWD}(n, m \bmod n).$$

Przykłady:

$$\text{NWD}(135, 120) = \text{NWD}(120, 15) = \text{NWD}(15, 0) = 15,$$

$(40, 135 \bmod 40)$
 $\quad \quad \quad \parallel$
 $\quad \quad \quad 15$

$(15, 40 \bmod 15)$
 $\quad \quad \quad \parallel$
 $\quad \quad \quad 10$

$$\begin{aligned} \text{NWD}(135, 40) &= \text{NWD}(40, 15) = \text{NWD}(15, 10) = \\ &= \text{NWD}(10, 5) = \text{NWD}(5, 0) = 5. \end{aligned}$$

Algorytm Euklidesa

1 **input:** $m, n \in \mathbb{N}_0, m + n > 0$

2 **output:** $d = \text{NWD}(m, n)$

3 $d := m$

4 $k := n$

$\text{NWD}(m, n)?$

5

6 **while** $k \neq 0$ **do**

7 $(d, k) := (k, d \bmod k)$

8 **end**

$d = k$
 $k = d \bmod k$

$\text{NWD}(d, k) = \text{NWD}(k, d \bmod k)$

Algorytm Euklidesa

1 **input:** $m, n \in \mathbb{N}_0, m + n > 0$

2 **output:** $d = \text{NWD}(m, n)$

3 $d := m$ n $m \bmod n$

4 $k := n$

5 # $\text{NWD}(d, k) = \text{NWD}(m, n)$

6 **while** $k \neq 0$ **do**

7 $(d, k) := (k, d \bmod k)$

8 **end**

$$\text{nowe}(k) = \left\lfloor \frac{d}{k} \right\rfloor \leq k-1$$

$\text{nowe}(k) = \lfloor d \bmod k \rfloor$

$$\Rightarrow \text{NWD}(\text{nowe}(d), \text{nowe}(k)) = \text{NWD}(d, k)$$

$$\Rightarrow k = 0 \quad \text{NWD}(d, 0) = d$$

Algorytm Euklidesa: dowód poprawności

```
1  input: m,n
2  output: d
3  d := m
4  k := n
5  while k  $\neq$  0 do
6    (d,k) := (k,d mod k)
7  end
```

Algorytm Euklidesa: dowód poprawności

```
1  input: m,n
2  output: d
3  d := m
4  k := n
5  while k  $\neq$  0 do
6    (d,k) := (k,d mod k)
7  end
```

⇒ Pętla się kończy, ponieważ
 $0 \leq d \bmod k < k$.

Algorytm Euklidesa: dowód poprawności

1 **input:** m, n

2 **output:** d

3 $d := m$

4 $k := n$

5 **while** $k \neq 0$ **do**

6 $(d, k) := (k, d \bmod k)$

7 **end**

⇒ Pętla się kończy, ponieważ
 $0 \leq d \bmod k < k$.

⇒ $\text{NWD}(d, k) = \text{NWD}(m, n)$ jest
niezmiennikiem pętli.

Algorytm Euklidesa: dowód poprawności

$$k \leq \max \left(m, n \right) \rightarrow \left(n, m \bmod n \right) \rightarrow \dots \rightarrow \dots$$

1 **input:** m, n

2 **output:** d

3 $d := m$

4 $k := n$

5 **while** $k \neq 0$ **do**

6 $(d, k) := (k, d \bmod k)$

7 **end**

⇒ Pętla się kończy, ponieważ
 $0 \leq d \bmod k < k$.

⇒ $\text{NWD}(d, k) = \text{NWD}(m, n)$ jest
niezmiennikiem pętli.

⇒ Po zakończeniu mamy $k = 0$
i $\text{NWD}(d, k) = \text{NWD}(m, n)$, skąd
 $d = \text{NWD}(d, 0) = \text{NWD}(m, n)$.

$$\text{nowe}(k) \leq k - 1$$

$$\leq \max\{m, n\}$$

Algorytm Euklidesa: dowód poprawności

1 **input:** m, n

2 **output:** d

3 $d := m$

4 $k := n$

5 **while** $k \neq 0$ **do**

6 $(d, k) := (k, d \bmod k)$

7 **end**

→ Pętla się kończy, ponieważ
 $0 \leq d \bmod k < k$.













→ $\text{NWD}(d, k) = \text{NWD}(m, n)$ jest
niezmiennikiem pętli.

→ Po zakończeniu mamy $k = 0$
i $\text{NWD}(d, k) = \text{NWD}(m, n)$, skąd
 $d = \text{NWD}(d, 0) = \text{NWD}(m, n)$.

Algorytm wykonuje co najwyżej $\max\{m, n\} + 1$ obrotów pętli.

$\min\{m, n\}$

Algorytm Euklidesa: przykłady

$m = 45, n = 12$	$m = 20, n = 63$	$m = 12, n = 6$
(d, k)	(d, k)	(d, k)
 (45,12)  (12,9)  (9,3)  (3 ,0)	 (20,63)  (63,20)  (20,3)  (3,2)  (2,1)  (1 ,0)	 (12,6)  (6 ,0)

Algorytm Euklidesa: przykłady

$m = 45, n = 12$	$m = 20, n = 63$	$m = 12, n = 6$
(d, k)	(d, k)	(d, k)
$(45, 12)$	$(20, 63)$	$(12, 6)$
$(12, 9)$	$(63, 20)$	$(\mathbf{6}, 0)$
$(9, 3)$	$(20, 3)$	
$(\mathbf{3}, 0)$	$(3, 2)$	
	$(2, 1)$	
	$(\mathbf{1}, 0)$	

Wygląda na to, że obrotów pętli jest istotnie mniej niż n . Ile?

Algorytm Euklidesa: złożoność

Twierdzenie

Algorytm Euklidesa wykonuje co najwyżej

$$2 \log_2(m + n) + 1$$

przebiegów pętli.

Algorytm Euklidesa: złożoność

Twierdzenie

Algorytm Euklidesa wykonuje co najwyżej

$$2 \log_2(m + n) + 1$$

przebiegów pętli.

Dla przypomnienia

$$\log_2(m + n) = a \quad \Longleftrightarrow \quad 2^a = m + n.$$

Algorytm Euklidesa: złożoność

Twierdzenie

Algorytm Euklidesa wykonuje co najwyżej

$$2 \log_2(m + n) + 1$$

przebiegów pętli.

Dla przypomnienia

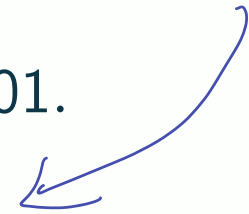
$$\log_2(m + n) = a \quad \Longleftrightarrow \quad 2^a = m + n.$$

Przykładowo, jeżeli $m, n \sim 2^{100}$, to

$$\log_2(m + n) \sim \log_2(2 \cdot 2^{100}) = \log_2(2^{101}) = 101.$$

$$\sim 2 \cdot 101 + 1 \approx \boxed{203}$$

2^{100}



Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geq n > 0$, to

$$\text{nowe}(k) \leq k - 1$$

$$\text{NWD}(m, n) = \text{NWD}(n, m \bmod n)$$

$$n + m \bmod n < \frac{2}{3}(m + n).$$

Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geq n > 0$, to

$$n + m \bmod n < \frac{2}{3}(m + n).$$

$$m = (m \operatorname{div} n)n + m \bmod n$$

Dowód. Mamy

.3

-2n

$$n + m \bmod n < \frac{2}{3}(m + n) \iff 3n + 3(m \bmod n) < 2m + 2n$$


Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geq n > 0$, to

$$n + m \bmod n < \frac{2}{3}(m + n).$$

Dowód. Mamy

$$\begin{aligned} n + m \bmod n < \frac{2}{3}(m + n) &\iff 3n + 3(m \bmod n) < 2m + 2n \\ &\iff n + 3[m - (m \operatorname{div} n)n] < 2m \end{aligned}$$


Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geq n > 0$, to

$$n + m \bmod n < \frac{2}{3}(m + n).$$

Dowód. Mamy

$$n + m \bmod n < \frac{2}{3}(m + n) \iff 3n + 3(m \bmod n) < 2m + 2n$$

$$\overset{m}{n} + \cancel{3m} - 3(m \operatorname{div} n)n < \cancel{2m}$$

$$\iff n + \underbrace{3[m - (m \operatorname{div} n)n]} < 2m$$

$$\iff m - (m \operatorname{div} n)n < 2n(m \operatorname{div} n) - n$$

$$n[2(m \operatorname{div} n) - 1]$$

Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geq n > 0$, to

$$n + m \bmod n < \frac{2}{3}(m + n).$$

Dowód. Mamy

$$n + m \bmod n < \frac{2}{3}(m + n) \iff 3n + 3(m \bmod n) < 2m + 2n$$

$$\iff n + 3[m - (m \operatorname{div} n)n] < 2m$$

$$\iff m - (m \operatorname{div} n)n < 2n(m \operatorname{div} n) - n$$

$$\iff m \bmod n < n[2(m \operatorname{div} n) - 1],$$

$m \bmod n$

$\leq n-1$

≥ 1

$$n \cdot \geq 2 \cdot 1 - 1 = 1$$

Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geq n > 0$, to

$$n + m \bmod n < \frac{2}{3}(m + n).$$

Dowód. Mamy

$$\begin{aligned} n + m \bmod n < \frac{2}{3}(m + n) &\iff 3n + 3(m \bmod n) < 2m + 2n \\ &\iff n + 3[m - (m \operatorname{div} n)n] < 2m \\ &\iff m - (m \operatorname{div} n)n < 2n(m \operatorname{div} n) - n \\ &\iff m \bmod n < n[2(m \operatorname{div} n) - 1], \end{aligned}$$

a ostatnia nierówność jest prawdziwa, ponieważ $m \bmod n < n$
i $m \operatorname{div} n \geq 1$.

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

$$\text{liczba obrotów} \leq 2 \log_2(m + n) + 1.$$

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

liczba obrotów $\leq 2 \log_2(m + n) + 1$.

Dowód. Można założyć (dlaczego?), że $m \geq n$. Przy każdym obrocie pętli

$$m < n \rightarrow (m, n) \rightarrow (n, m)$$

```
1  while k ≠ 0 do
2      (d, k) := (k, d mod k)
3  end
```

mamy $\text{nowe}(d) + \text{nowe}(k) < \frac{2}{3}(d + k)$.

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa ≥ 1

liczba obrotów $\leq 2 \log_2(m + n) + 1$

$1 \leq \text{ostatek}(d) \rightarrow \text{ostatek}(k) < \dots$

Dowód. Można założyć (dlaczego?), że $m \geq n$. Przy każdym obrocie pętli

```
1  while k  $\neq$  0 do
2      (d,k) := (k, d mod k)
3  end
```

i -razy

mamy $\text{nowe}(d) + \text{nowe}(k) < \frac{2}{3}(d + k)$. Zatem, jeżeli obrotów było i , to (dlaczego?) $1 \leq (\frac{2}{3})^i(m + n)$.

$$\begin{aligned} \text{nowe}(\text{nowe}(d)) + \text{nowe}(\text{nowe}(k)) &< \frac{2}{3}(\text{nowe}(d) + \text{nowe}(k)) \\ &< \frac{2}{3} \left[\frac{2}{3}(d + k) \right] = \left(\frac{2}{3} \right)^2 (d + k) \end{aligned}$$

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

$$\text{liczba obrotów} \leq 2 \log_2(m+n) + 1.$$

Dowód. Można założyć (dlaczego?), że $m \geq n$. Przy każdym obrocie pętli

```
1  while k ≠ 0 do
2      (d,k) := (k,d mod k)
3  end
```

mamy $\text{nowe}(d) + \text{nowe}(k) < \frac{2}{3}(d+k)$. Zatem, jeżeli obrotów było i , to (dlaczego?) $1 \leq (\frac{2}{3})^i(m+n)$. Stąd $(\frac{3}{2})^i \leq m+n$, czyli

$$i \cdot \frac{1}{2} < i \log_2 \frac{3}{2} \leq \log_2(m+n).$$

\Downarrow
 $i \leq 2 \log_2(m+n)^{\frac{1}{2}}$

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

$$\text{liczba obrotów} \leq 2 \log_2(m + n) + 1.$$

Dowód. Można założyć (dlaczego?), że $m \geq n$. Przy każdym obrocie pętli

```
1   while k  $\neq$  0 do  
2       (d,k) := (k,d mod k)  
3   end
```

mamy $\text{nowe}(d) + \text{nowe}(k) < \frac{2}{3}(d + k)$. Zatem, jeżeli obrotów było i , to (dlaczego?) $1 \leq (\frac{2}{3})^i(m + n)$. Stąd $(\frac{3}{2})^i \leq m + n$, czyli

$$2i \log_2 \frac{3}{2} \leq 2 \log_2(m + n).$$

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

$$\text{liczba obrotów} \leq 2 \log_2(m + n) + 1.$$

Dowód. Można założyć (dlaczego?), że $m \geq n$. Przy każdym obrocie pętli

```
1   while k  $\neq$  0 do  
2       (d,k) := (k,d mod k)  
3   end
```

mamy $\text{nowe}(d) + \text{nowe}(k) < \frac{2}{3}(d + k)$. Zatem, jeżeli obrotów było i , to (dlaczego?) $1 \leq (\frac{2}{3})^i(m + n)$. Stąd $(\frac{3}{2})^i \leq m + n$, czyli

$$i \leq 2i \log_2 \frac{3}{2} \leq 2 \log_2(m + n).$$

Rozszerzony algorytm Euklidesa

```

1  input:  $m, n \in \mathbb{N}_0, m + n > 0$ 
2  output:  $d = \text{NWD}(m, n)$ 
3   $d := m$ 
4   $k := n$ 
5  while  $k \neq 0$  do
6       $(d, k) := (k, d \bmod k)$ 
7  end
    
```

135 120

15

135 40

5

$$15 = \cancel{135} - 3 \cdot \cancel{40}$$

$$(3) \cdot 135 - (10) \cdot 40$$

$$\forall_{s, t \in \mathbb{Z}} \boxed{\text{NWD}(m, n)} = \boxed{s} \cdot m + \boxed{t} \cdot n$$

$$m, n \in \mathbb{N}_0, \quad m+n > 0$$

rozklad na cynniki (HOLNE!)
 $\text{NWD}(m, n)$ \swarrow
algorithm Euklidesa (SLYBKIE!)

$$\text{NWD}(m, n) = \text{NWD}(n, m \bmod n)$$

$$= \dots = \text{NWD}(d, 0)$$

$$\leq 2 \log_2(m+n) + 1 \quad d = \text{NWD}(m, n)$$

$$\Rightarrow O(\log_2(m+n))$$

$$\bigvee_{s, t \in \mathbb{Z}} \text{NWD}(m, n) = d = \underline{s \cdot m + t \cdot n}$$

Rozszerzony algorytm Euklidesa

```
1  input:  $m, n \in \mathbb{N}_0, m + n > 0$   
2  output:  $d = \text{NWD}(m, n)$   
3   $d := m$   
4   $k := n$   
5  while  $k \neq 0$  do  
6       $(d, k) := (k, d \bmod k)$   
7  end
```

$$\begin{cases} n = q \cdot d + r \\ r = n - q \cdot d \end{cases}$$

Rozszerzony algorytm Euklidesa

/ , %

1 **input:** $m, n \in \mathbb{N}_0, m + n > 0$

2 **output:** $d = \text{NWD}(m, n)$

3 $d := m$

4 $k := n$

5 **while** $k \neq 0$ **do**

6 $\# d = (d \text{ div } k)k + d \text{ mod } k$

7 $q := d \text{ div } k$

8 $(d, k) := (k, d - q * k)$

9 **end**

dzielenie d z resztką przez k

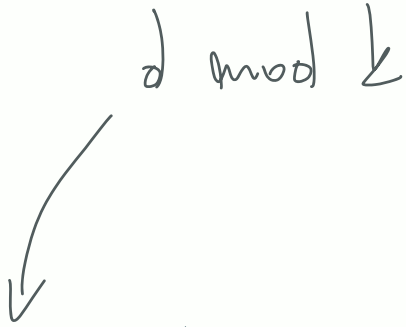
$$\{ u = q \cdot d + r \}$$

$d \text{ mod } k$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4      q := d div k
5      (d,k) := (k, d - q * k)
6  end
```

$d \bmod k \downarrow$



Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k,d - q * k)
6  end
```

$$d = 135 \mid k = 40$$

Rozszerzony algorytm Euklidesa

$$q = 135 \text{ div } 40 = 3$$

```
1  d := m
2  k := n
3  while k ≠ 0 do
4    q := d div k
5    (d,k) := (k, d - q * k)
6  end
```

$$\begin{array}{c|c} d = 135 & k = 40 \\ \hline d = 40 & k = 135 - \boxed{3} \cdot 40 \end{array}$$

Rozszerzony algorytm Euklidesa

$$q = 40 \div 15 = 2$$

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k,d - q * k)
6  end
```

$d = 135$		$k = 40$	
<hr/>			
$d = 40$		$k = 135 - 3 \cdot 40$	$= 15$
$d = 15$		$k = 40 - 2 \cdot 15$	

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4      q := d div k
5      (d,k) := (k, d - q * k)
6  end
```

$d = 135$	$k = 40$
<hr/>	
$d = 40$	$k = 135 - 3 \cdot 40$
$d = 15$	$k = 40 - 2 \cdot 15$
$d = 10$	$k = 15 - 1 \cdot 10$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k, d - q * k)
6  end
```

$d = 135$	$k = 40$
$d = 40$	$k = 135 - 3 \cdot 40$
$d = 15$	$k = 40 - 2 \cdot 15$
$d = 10$	$k = 15 - 1 \cdot 10$
<u>$d = 5$</u>	$k = 10 - 2 \cdot 5 = 0$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k, d - q * k)
6  end
```

$d = 135$	$k = 40$
$d = 40$	$k = 135 - 3 \cdot 40$
$d = 15$	$k = 40 - 2 \cdot 15$
$d = 10$	$k = 15 - 1 \cdot 10$
$d = 5$	$k = 10 - 2 \cdot 5$

5 =

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k, d - q * k)
6  end
```

$d = 135$	$k = 40$
$d = 40$	$k = 135 - 3 \cdot 40$
$d = 15$	$k = 40 - 2 \cdot 15$
$d = 10$	$k = 15 - 1 \cdot 10$
$d = 5$	$k = 10 - 2 \cdot 5$

$5 = 15 - 1 \cdot 10$

$\text{? } 40 - 2 \cdot 15$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4      q := d div k
5      (d,k) := (k, d - q * k)
6  end
```

$d = 135$	$k = 40$
$d = 40$	$k = 135 - 3 \cdot 40$
$d = 15$	$k = 40 - 2 \cdot 15$
$d = 10$	$k = 15 - 1 \cdot 10$
$d = 5$	$k = 10 - 2 \cdot 5$

$$5 = 15 - 1 \cdot 10 = 15 - 1 \cdot (40 - 2 \cdot 15)$$

Rozszerzony algorytm Euklidesa

```

1   d := m
2   k := n
3   while k  $\neq$  0 do
4       q := d div k
5       (d,k) := (k,d - q * k)
6   end

```

$d = 135$	$k = 40$
$d = 40$	$k = 135 - 3 \cdot 40$
$d = 15$	$k = 40 - 2 \cdot 15$
$d = 10$	$k = 15 - 1 \cdot 10$
$d = 5$	$k = 10 - 2 \cdot 5$

$$5 = 15 - 1 \cdot 10 = 15 - 1 \cdot (40 - 2 \cdot 15) =$$
$$= 3 \cdot 15 - 1 \cdot 40$$

\Downarrow
 $135 - 3 \cdot 40$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k, d - q * k)
6  end
```

$d = 135$	$k = 40$
$d = 40$	$k = 135 - 3 \cdot 40$
$d = 15$	$k = 40 - 2 \cdot 15$
$d = 10$	$k = 15 - 1 \cdot 10$
$d = 5$	$k = 10 - 2 \cdot 5$

$$\begin{aligned} 5 &= 15 - 1 \cdot 10 = 15 - 1 \cdot (40 - 2 \cdot 15) = \\ &= 3 \cdot 15 - 1 \cdot 40 = 3 \cdot (135 - 3 \cdot 40) - 1 \cdot 40 \end{aligned}$$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k, d - q * k)
6  end
```

$d = 135$	$k = 40$
$d = 40$	$k = 135 - 3 \cdot 40$
$d = 15$	$k = 40 - 2 \cdot 15$
$d = 10$	$k = 15 - 1 \cdot 10$
$d = 5$	$k = 10 - 2 \cdot 5$

NWD

$$\begin{aligned} 5 &= 15 - 1 \cdot 10 = 15 - 1 \cdot (40 - 2 \cdot 15) = \\ &= 3 \cdot 15 - 1 \cdot 40 = 3 \cdot (135 - 3 \cdot 40) - 1 \cdot 40 \\ &= 3 \cdot 135 - 10 \cdot 40 \end{aligned}$$

s m t n

$$\text{NWD}(m,n) = s \cdot m + t \cdot n$$

Rozszerzony algorytm Euklidesa

Twierdzenie

Dla dowolnych liczb $m, n \in \mathbb{N}_0$, które nie są jednocześnie równe zero, istnieją takie liczby całkowite s i t , że

$$\text{NWD}(m, n) = s \cdot m + t \cdot n.$$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4      q := d div k
5      (d,k) := (k,d - q * k)
6  end
```

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k,d - q * k)
6  end
```

$$\begin{array}{c|c|c} d_0 = 135 & k_0 = 40 & q_i \\ \hline \end{array}$$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k,d - q * k)
6  end
```

$d_0 = 135$		$k_0 = 40$		q_i
<hr/>				
$d_1 = 40$		$k_1 = 135 - 3 \cdot 40$		$q_1 = 3$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k,d - q * k)
6  end
```

$d_0 = 135$	$k_0 = 40$	q_i
$d_1 = 40$	$k_1 = 135 - 3 \cdot 40$	$q_1 = 3$
$d_2 = 15$	$k_2 = 40 - 2 \cdot 15$	$q_2 = 2$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k,d - q * k)
6  end
```

$d_0 = 135$	$k_0 = 40$	q_i
$d_1 = 40$	$k_1 = 135 - 3 \cdot 40$	$q_1 = 3$
$d_2 = 15$	$k_2 = 40 - 2 \cdot 15$	$q_2 = 2$
$d_3 = 10$	$k_3 = 15 - 1 \cdot 10$	$q_3 = 1$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4    q := d div k
5    (d,k) := (k, d - q * k)
6  end
```

$d_0 = 135$	$k_0 = 40$	q_i
$d_1 = 40$	$k_1 = 135 - 3 \cdot 40$	$q_1 = 3$
$d_2 = 15$	$k_2 = 40 - 2 \cdot 15$	$q_2 = 2$
$d_3 = 10$	$k_3 = 15 - 1 \cdot 10$	$q_3 = 1$
$d_4 = 5$	$k_4 = 10 - 2 \cdot 5$	$q_4 = 2$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4      q := d div k
5      (d,k) := (k,d - q * k)
6  end
```

$d_0 = 135$	$k_0 = 40$	q_i
$d_1 = 40$	$k_1 = 15$	$q_1 = 3$
$d_2 = 15$	$k_2 = 10$	$q_2 = 2$
$d_3 = 10$	$k_3 = 5$	$q_3 = 1$
$d_4 = 5$	$k_4 = 0$	$q_4 = 2$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k ≠ 0 do
4    q := d div k
5    (d, k) := (k, d - q * k)
6  end
```

$\rightsquigarrow d_i = k_{i-1}, q_i = d_{i-1} \text{ div } d_i,$

$d_{i-1} \text{ div } k_{i-1} = d_i$

$d_0 = 135$	$k_0 = 40$	q_i
$d_1 = 40$	$k_1 = 15$	$q_1 = 3$
$d_2 = 15$	$k_2 = 10$	$q_2 = 2$
$d_3 = 10$	$k_3 = 5$	$q_3 = 1$
$d_4 = 5$	$k_4 = 0$	$q_4 = 2$

Rozszerzony algorytm Euklidesa

1 $d := m$

2 $k := n$

3 **while** $k \neq 0$ **do**

4 $q := d \text{ div } k$

5 $(d, k) := (k, d - q * k)$

6 **end**

$d_0 = 135$	$k_0 = 40$	q_i
$d_1 = 40$	$k_1 = 15$	$q_1 = 3$
$d_2 = 15$	$k_2 = 10$	$q_2 = 2$
$d_3 = 10$	$k_3 = 5$	$q_3 = 1$
$d_4 = 5$	$k_4 = 0$	$q_4 = 2$

$$\boxed{135} - \boxed{3} \cdot \boxed{40} =$$

$$k_i \quad d_{i-1} - q_i \cdot k_{i-1}$$

$$\rightsquigarrow d_i = k_{i-1}, q_i = d_{i-1} \text{ div } d_i,$$

$$\rightsquigarrow \boxed{d_{i+1}} = k_i = d_{i-1} - q_i \cdot k_{i-1} = \boxed{d_{i-1} - q_i \cdot d_i}.$$

$$\parallel$$

$$d_i$$

Rozszerzony algorytm Euklidesa

```
1  d := m
2  k := n
3  while k  $\neq$  0 do
4      q := d div k
5      (d,k) := (k, d - q * k)
6  end
```

$d_0 = 135$	$k_0 = 40$	q_i
$d_1 = 40$	$k_1 = 15$	$q_1 = 3$
$d_2 = 15$	$k_2 = 10$	$q_2 = 2$
$d_3 = 10$	$k_3 = 5$	$q_3 = 1$
$d_4 = 5$	$k_4 = 0$	$q_4 = 2$

$\rightsquigarrow d_i = k_{i-1}, q_i = d_{i-1} \text{ div } d_i,$

$\rightsquigarrow d_{i+1} = k_i = d_{i-1} - q_i \cdot k_{i-1} = d_{i-1} - q_i \cdot d_i.$

Zatem dla $i = 1, 2, \dots, j$ mamy

$$q_i = d_{i-1} \text{ div } d_i, \quad d_{i+1} = d_{i-1} - q_i \cdot d_i.$$

Rozszerzony algorytm Euklidesa

⇒ Mamy

$$d_0 = m \quad d_1 = n$$

oraz

$$q_i = d_{i-1} \operatorname{div} d_i, \quad d_{i+1} = d_{i-1} - q_i \cdot d_i$$

dla $i = 1, 2, \dots, j$ (j - liczba obrotów pętli).

Rozszerzony algorytm Euklidesa

⇒ Mamy

$$d_0 = m$$

oraz

$$q_i = d_{i-1} \operatorname{div} d_i, \quad d_{i+1} = d_{i-1} - q_i \cdot d_i$$

dla $i = 1, 2, \dots, j$ (j - liczba obrotów pętli).

⇒ Dodatkowo wiemy, że

$$d_j = \operatorname{NWD}(m, n).$$

$$\begin{array}{r} d_i \quad k_i \\ \hline d_0 \quad | \\ d_1 \quad | \\ \vdots \quad | \\ d_j \quad 0 \end{array}$$

Rozszerzony algorytm Euklidesa

⇒ Mamy

$$d_0 = m$$

oraz

$$q_i = d_{i-1} \operatorname{div} d_i, \quad d_{i+1} = d_{i-1} - q_i \cdot d_i$$

dla $i = 1, 2, \dots, j$ (j - liczba obrotów pętli).

⇒ Dodatkowo wiemy, że

$$d_j = \text{NWD}(m, n).$$

⇒ Chcemy

$$d_j = \overset{?}{s} \cdot m + \overset{?}{t} \cdot n.$$

Rozszerzony algorytm Euklidesa

⇒ Mamy

$$d_0 = m$$

oraz

$$q_i = d_{i-1} \operatorname{div} d_i, \quad d_{i+1} = d_{i-1} - q_i \cdot d_i$$

dla $i = 1, 2, \dots, j$ (j - liczba obrotów pętli).

⇒ Dodatkowo wiemy, że

$$i=1 \Rightarrow$$

$$d_2 = d_0 - q_1 \cdot d_1$$
$$q_1 = d_0 \operatorname{div} d_1$$

$$d_j = \text{NWD}(m, n).$$

⇒ Chcemy

$$d_j = s \cdot m + t \cdot n.$$

⇒ Skonstruujemy takie ciągi (s_i) , (t_i) , że

$$d_i = s_i \cdot m + t_i \cdot n, \quad i = 0, 1, \dots, j.$$

Rozszerzony algorytm Euklidesa

~> Na początku chcemy, aby

$$m = d_0 = s_0 \cdot m + t_0 \cdot n, \quad m = 1 \cdot m + 0 \cdot n$$

więc przyjmujemy $s_0 = 1, t_0 = 0$.

$$d_1 = n = 0 \cdot m + 1 \cdot n$$

Rozszerzony algorytm Euklidesa

⇒ Na początku chcemy, aby

$$m = d_0 = s_0 \cdot m + t_0 \cdot n,$$

więc przyjmujemy $s_0 = 1$, $t_0 = 0$.

⇒ Dalej chcemy, aby

$$n = d_1 = s_1 \cdot m + t_1 \cdot n,$$

więc przyjmujemy $s_1 = 0$, $t_1 = 1$.

Rozszerzony algorytm Euklidesa

⇒ Na początku chcemy, aby

$$m = d_0 = s_0 \cdot m + t_0 \cdot n,$$

więc przyjmujemy $s_0 = 1, t_0 = 0$.

⇒ Dalej chcemy, aby

$$n = d_1 = s_1 \cdot m + t_1 \cdot n,$$

więc przyjmujemy $s_1 = 0, t_1 = 1$.

⇒ Następnie, skoro wiemy, że $d_2 = d_0 - q_1 d_1$, to

$$\begin{aligned} d_2 &= s_0 \cdot m + t_0 \cdot n - q_1 (s_1 \cdot m + t_1 \cdot n) = \\ &= \underbrace{(s_0 - q_1 s_1)}_{s_2} m + \underbrace{(t_0 - q_1 t_1)}_{t_2} n. \end{aligned}$$

Rozszerzony algorytm Euklidesa

⇒ Następnie, skoro wiemy, że $d_2 = d_0 - q_1 d_1$, to

$$d_2 = \underbrace{(s_0 - q_1 s_1)}_{s_2} m + \underbrace{(t_0 - q_1 t_1)}_{t_2} n.$$

Rozszerzony algorytm Euklidesa

~> Następnie, skoro wiemy, że $d_2 = d_0 - q_1 d_1$, to

$$d_2 = \underbrace{(s_0 - q_1 s_1)}_{s_2} m + \underbrace{(t_0 - q_1 t_1)}_{t_2} n.$$

~> Ogólnie, dla $i \geq 1$ przyjmujemy

$$s_{i+1} := s_{i-1} - q_i s_i, \quad t_{i+1} := t_{i-1} - q_i t_i.$$

Rozszerzony algorytm Euklidesa

~> Następnie, skoro wiemy, że $d_2 = d_0 - q_1 d_1$, to

$$d_2 = \underbrace{(s_0 - q_1 s_1)}_{s_2} m + \underbrace{(t_0 - q_1 t_1)}_{t_2} n.$$

~> Ogólnie, dla $i \geq 1$ przyjmujemy

$$s_{i+1} := s_{i-1} - q_i s_i, \quad t_{i+1} := t_{i-1} - q_i t_i.$$

~> Wtedy, jeżeli

$$d_i = s_i \cdot m + t_i \cdot n \quad \text{oraz} \quad d_{i-1} = s_{i-1} \cdot m + t_{i-1} \cdot n,$$

to

$$d_{i+1} = d_{i-1} - q_i d_i = s_{i+1} \cdot m + t_{i+1} \cdot n.$$

$$d_0, d_1 \rightsquigarrow d_2$$

$$d_{i-1}, d_i \rightsquigarrow d_{i+1}$$

Rozszerzony algorytm Euklidesa

⇒ Następnie, skoro wiemy, że $d_2 = d_0 - q_1 d_1$, to

$$d_2 = \underbrace{(s_0 - q_1 s_1)}_{s_2} m + \underbrace{(t_0 - q_1 t_1)}_{t_2} n.$$

⇒ Ogólnie, dla $i \geq 1$ przyjmujemy

$$s_{i+1} := s_{i-1} - q_i s_i, \quad t_{i+1} := t_{i-1} - q_i t_i.$$

⇒ Wtedy, jeżeli

$$d_i = s_i \cdot m + t_i \cdot n \quad \text{oraz} \quad d_{i-1} = s_{i-1} \cdot m + t_{i-1} \cdot n,$$

to

$$d_{i+1} = d_{i-1} - q_i d_i = s_{i+1} \cdot m + t_{i+1} \cdot n.$$

⇒ Z zasady indukcji otrzymujemy

$$d_{i+1} = s_{i+1} \cdot m + t_{i+1} \cdot n, \quad i = 1, 2, \dots, j-1.$$

$$\text{NWD}(m, n) = d_j' = s_j \cdot m + t_j \cdot n$$

Rozszerzony algorytm Euklidesa

```
1  input:  $m, n \in \mathbb{N}_0, m + n > 0$   
2  output:  $d = \text{NWD}(m, n)$   
3   $d := m$   
4   $k := n$   
5  while  $k \neq 0$  do  
6       $q := d \text{ div } k$   
7       $(d, k) := (k, d - q * k)$   
8  end
```

Rozszerzony algorytm Euklidesa

```
1  input:  $m, n \in \mathbb{N}_0, m + n > 0$ 
2  output:  $d = \text{NWD}(m, n)$ 
3   $d := m$             $d \approx d_{i-1}$ 
4   $d' := n$            $d' \approx d_i$ 
5   $s := 1, s' := 0$ 
6   $t := 0, t' := 1$ 
7
8  while  $d' \neq 0$  do
9       $q := d \text{ div } d'$ 
10      $(d, d') := (d', d - q * d')$ 
11      $(s, s') := (s', s - q * s')$ 
12      $(t, t') := (t', t - q * t')$ 
13 end
```

Rozszerzony algorytm Euklidesa

1 **input:** $m, n \in \mathbb{N}_0, m + n > 0$

2 **output:** $d = \text{NWD}(m, n)$

3 $d := m$

4 $d' := n$

5 $s := 1, s' := 0$

6 $t := 0, t' := 1$

7 $\# d = sm + tn, d' = s'm + t'n$

8 **while** $d' \neq 0$ **do**

9 $q := d \text{ div } d'$

10 $(d, d') := (d', d - q * d')$

11 $(s, s') := (s', s - q * s')$

12 $(t, t') := (t', t - q * t')$

13 **end**

$$\text{nowe}(d) = \text{nowe}(s) \cdot m + \text{nowe}(t) \cdot n$$

\vdots
 $\text{NWD}(m, n)$

Rozszerzony algorytm Euklidesa

```
1  input: m,n
2  output: d = NWD(m,n)
3  d := m
4  d' := n
5  s := 1, s' := 0
6  t := 0, t' := 1
7  while d'  $\neq$  0 do
8      q := d div k d'
9      (d,d') := (d',d-q * d')
10     (s,s') := (s',s-q * s')
11     (t,t') := (t',t-q * t')
12 end
```

i	d_i	q_i	s_i	t_i
-----	-------	-------	-------	-------

Rozszerzony algorytm Euklidesa

```
1  input: m,n
2  output: d = NWD(m,n)
3  d := m
4  d' := n
5  s := 1, s' := 0
6  t := 0, t' := 1
7  while d' ≠ 0 do
8    q := d div d'
9    (d,d') := (d',d-q * d')
10   (s,s') := (s',s-q * s')
11   (t,t') := (t',t-q * t')
12 end
```

i	d_i	q_i	s_i	t_i
0	135		1	0

40

Rozszerzony algorytm Euklidesa

```

1  input:  m,n
2  output: d = NWD(m,n)
3  d := m
4  d' := n
5  s := 1, s' := 0
6  t := 0, t' := 1
7  while d'  $\neq$  0 do
8      q := d div k
9      (d,d') := (d',d-q * d')
10     (s,s') := (s',s-q * s')
11     (t,t') := (t',t-q * t')
12 end

```

i	d_i	q_i	s_i	t_i
0	135		1	0
1	40	3	0	1

Diagram illustrating the state of the algorithm after the first iteration. The table shows the values of d_i , q_i , s_i , and t_i for $i=0$ and $i=1$. The values are: $d_0=135$, $d_1=40$, $q_1=3$, $s_0=1$, $s_1=0$, $t_0=0$, and $t_1=1$. The diagram shows the state of the algorithm after the first iteration, with the values of d_i , q_i , s_i , and t_i for $i=0$ and $i=1$ shown in the table. The values are: $d_0=135$, $d_1=40$, $q_1=3$, $s_0=1$, $s_1=0$, $t_0=0$, and $t_1=1$. The diagram shows the state of the algorithm after the first iteration, with the values of d_i , q_i , s_i , and t_i for $i=0$ and $i=1$ shown in the table. The values are: $d_0=135$, $d_1=40$, $q_1=3$, $s_0=1$, $s_1=0$, $t_0=0$, and $t_1=1$.

15	2	1	-3
10	1	-2	7
5	2	3	-10
0			

Rozszerzony algorytm Euklidesa

```
1  input: m,n
2  output: d = NWD(m,n)
3  d := m
4  d' := n
5  s := 1, s' := 0
6  t := 0, t' := 1
7  while d'  $\neq$  0 do
8      q := d div d'
9      (d,d') := (d',d-q * d')
10     (s,s') := (s',s-q * s')
11     (t,t') := (t',t-q * t')
12 end
```

i	d_i	q_i	s_i	t_i
0	135		1	0
1	40	3	0	1
2	15	2	1	-3

Rozszerzony algorytm Euklidesa

```
1  input: m,n
2  output: d = NWD(m,n)
3  d := m
4  d' := n
5  s := 1, s' := 0
6  t := 0, t' := 1
7  while d'  $\neq$  0 do
8      q := d div d'
9      (d,d') := (d',d-q * d')
10     (s,s') := (s',s-q * s')
11     (t,t') := (t',t-q * t')
12 end
```

i	d_i	q_i	s_i	t_i
0	135		1	0
1	40	3	0	1
2	15	2	1	-3
3	10	1	-2	7

Rozszerzony algorytm Euklidesa

```
1  input: m,n
2  output: d = NWD(m,n)
3  d := m
4  d' := n
5  s := 1, s' := 0
6  t := 0, t' := 1
7  while d'  $\neq$  0 do
8      q := d div d'
9      (d,d') := (d',d-q * d')
10     (s,s') := (s',s-q * s')
11     (t,t') := (t',t-q * t')
12 end
```

i	d_i	q_i	s_i	t_i
0	135		1	0
1	40	3	0	1
2	15	2	1	-3
3	10	1	-2	7
4	5	2	3	-10

Rozszerzony algorytm Euklidesa

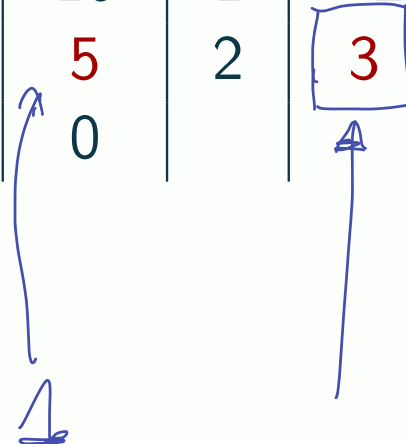
```
1  input: m,n
2  output: d = NWD(m,n)
3  d := m
4  d' := n
5  s := 1, s' := 0
6  t := 0, t' := 1
7  while d'  $\neq$  0 do
8      q := d div d'
9      (d,d') := (d',d-q * d')
10     (s,s') := (s',s-q * s')
11     (t,t') := (t',t-q * t')
12 end
```

i	d_i	q_i	s_i	t_i
0	135		1	0
1	40	3	0	1
2	15	2	1	-3
3	10	1	-2	7
4	5	2	3	-10
5	0			

Rozszerzony algorytm Euklidesa

```
1  input: m,n
2  output: d = NWD(m,n)
3  d := m
4  d' := n
5  s := 1, s' := 0
6  t := 0, t' := 1
7  while d' ≠ 0 do
8      q := d div d'
9      (d,d') := (d', d - q * d')
10     (s,s') := (s', s - q * s')
11     (t,t') := (t', t - q * t')
12 end
```

i	d_i	q_i	s_i	t_i
0	135		1	0
1	40	3	0	1
2	15	2	1	-3
3	10	1	-2	7
4	5	2	3	-10
5	0			



$$\text{NWD}(135, 40) = 5 = 3 \cdot 135 + (-10) \cdot 40.$$