Matematyka dyskretna

Teoria liczb cz. 1

Adam Gregosiewicz

24 listopada 2022 r.

Podzielność

Definicja

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

$$a = kb.$$

Piszemy wtedy

$$b|a.$$

Inaczej mówiąc, mamy

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

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

- $\bigwedge_{n\in\mathbb{Z}} n|0,$
- $\bigwedge_{n\in\mathbb{Z}} n|n,$
- $\bigwedge_{n\in\mathbb{Z}} \pm 1|n,$
- $\bigwedge_{n\in\mathbb{Z}} 0|n \Rightarrow n = 0.$

Własności relacji podzielności

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.

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

$$n = qd + r \qquad \text{oraz} \qquad 0 \leqslant r < d.$$

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

$$n = qd + r \qquad \text{oraz} \qquad 0 \leqslant r < d.$$

Dowód

- Istnienie.
- Jedyność.

Przykłady

- $14 = 2 \cdot 6 + 2,$
- $31 = 4 \cdot 7 + 3,$
- $55 = 3 \cdot 15 + 10.$

Podłoga i sufit

- Funkcja **podłoga**

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

- Funkcja **sufit**

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

Podłoga i sufit

$\rightsquigarrow$ Funkcja **podłoga**

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

$\rightsquigarrow$ Funkcja **sufit**

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

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

$$\lfloor x \rfloor \leqslant x < \lfloor x + 1 \rfloor, \qquad x \leqslant \lceil x \rceil < x + 1.$$

Dzielenie z resztą: dowód

- **Istnienie**. Niech

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

Dzielenie z resztą: dowód

⇝ **Istnienie**. Niech

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

Wtedy oczywiście

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

oraz

$$q \leqslant \frac{n}{d} < q + 1 \quad \Rightarrow \quad qd \leqslant n < qd + d \quad \Rightarrow \quad 0 \leqslant r < d.$$

Dzielenie z resztą: dowód

$\rightsquigarrow$ **Jedyność**. Załóżmy, że dla q, q', r i r' zachodzi

$$n = qd + r = q'd + r', \qquad 0 \leqslant r, r' < d.$$

Dzielenie z resztą: dowód

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

$$n = qd + r = q'd + r', \qquad 0 \leqslant r, r' < d.$$

Wtedy

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

Dzielenie z resztą: dowód

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

$$n = qd + r = q'd + r', \qquad 0 \leqslant r, r' < d.$$

Wtedy

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

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

Dielenie z resztą: dowód

Oznaczenia

Jeżeli

$$n = qd + r, \qquad 0 \leqslant r < d,$$

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

$$q = n \operatorname{div} d, \qquad r = n \bmod d.$$

Wzory

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

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

Wzory

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

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

Jako wniosek otrzymujemy

$$n = (n \text{ div } d)d + n \text{ mod } d, \qquad 0 \leqslant n \text{ mod } d < d.$$

Algorytm dzielenia

```
input: n ⩾ 0, d > 0
output: q, r ∈ ℤ, n = qd + r, 0 ⩽ r < d
q := 0
r := n
while r ⩾ d do
    q := q + 1
    r := r - d
end
```

Algorytm dzielenia

```
input: n ⩾ 0, d > 0
output: q, r ∈ ℤ, n = qd + r, 0 ⩽ r < d
q := 0
r := n
while r ⩾ d do
    q := q + 1
    r := r - d
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	$r \geqslant n$
0	0	31	1
1	1	24	1
2	2	17	1
3	3	10	1
4	**4**	**3**	0

Niezmienniki pętli

Zdanie *p* jest **niezmiennikiem** pętli

```
while q do
    S
end
```

jeżeli spełniony jest następujący warunek:

Niezmienniki pętli

Zdanie p jest **niezmiennikiem** pętli

```
while q do
    S
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.

Niezmienniki pętli

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

Załóżmy, że p jest niezmiennikiem pętli

```
while q do
    S
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

```
input: n ⩾ 0, d > 0
output: q, r ∈ ℤ, n = qd + r, 0 ⩽ r < d
q := 0
r := n

while r ⩾ d do
    q := q + 1
    r := r - d
end
```

Algorytm dzielenia

```
input: n ⩾ 0, d > 0
output: q, r ∈ ℤ, n = qd + r
q := 0
r := n
# niezmiennik: qd + r = n, r ⩾ 0
while r ⩾ d do
    q := q + 1
    r := r - d
end
```

Algorytm dzielenia: dowód poprawności

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

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

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

$$0 \cdot d + n = n \qquad \wedge \qquad n \geqslant 0.$$

- Algorytm się zatrzymuje, ponieważ

$$r - d < r.$$

- Na mocy twierdzenia o niezmiennikach po zakończeniu mamy

$$qd + r = n \qquad \wedge \qquad r \geqslant 0 \qquad \wedge \qquad \neg(r \geqslant d)$$

czyli

$$qd + r = n \qquad \wedge \qquad 0 \leqslant r < d.$$

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

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

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

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

Jak znaleźć $\mathrm{NWD}(m, n)$?

NWD

Zadanie

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

NWD

Zadanie

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

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

NWD

Zadanie

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

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

Stąd

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

NWD – wersja naiwna

```
input: m,n ∈ ℕ₀, m + n > 0
output: d = NWD(m,n)
d := 1, k := 2
while k ⩽ m ∧ k ⩽ n do
    if k|m ∧ k|n do
        d := d * k
        m := m/k
        n := n/k
    else
        k := k+1
    end
end
```

NWD – wersja naiwna

```
input: m,n ∈ ℕ₀, m + n > 0
output: d = NWD(m,n)
d := 1, k := 2
while k ⩽ m ∧ k ⩽ n do
    if k|m ∧ k|n do
        d := d * k
        m := m/k
        n := n/k
    else
        k := k+1
    end
end
```

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

Algorytm Euklidesa

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

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* $(m \bmod n)$.

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

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 $(m \bmod n)$.

Dowód.

- Wystarczy sprawdzić (dlaczego?), że

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

- Powyższa równoważność wynika (dlaczego?) z równości

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

Algorytm Euklidesa

Wniosek

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

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

Algorytm Euklidesa

Wniosek

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

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

Przykłady:

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

Algorytm Euklidesa

Wniosek

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

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

Przykłady:

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

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

Algorytm Euklidesa

```
input: m,n ∈ ℕ₀, m + n > 0
output: d = NWD(m,n)
d := m
k := n

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

Algorytm Euklidesa

```
input: m,n ∈ ℕ₀, m + n > 0
output: d = NWD(m,n)
d := m
k := n
# NWD(d,k) = NWD(m,n)
while k ≠ 0 do
    (d,k) := (k,d mod k)
end
```

Algorytm Euklidesa: dowód poprawności

```
input: m,n
output: d
d := m
k := n
while k ≠ 0 do
 (d,k) := (k,d mod k)
end
```

Algorytm Euklidesa: dowód poprawności

```
input: m,n
output: d
d := m
k := n
while k ≠ 0 do
 (d,k) := (k,d mod k)
end
```

$\rightsquigarrow$ Pętla się kończy, ponieważ $0 \leqslant d \bmod k < k$.

Algorytm Euklidesa: dowód poprawności

```
input: m,n
output: d
d := m
k := n
while k ≠ 0 do
 (d,k) := (k,d mod k)
end
```

- ⇝ Pętla się kończy, ponieważ $0 \leqslant d \bmod k < k$.
- ⇝ $\mathrm{NWD}(d,k) = \mathrm{NWD}(m,n)$ jest niezmiennikiem pętli.

Algorytm Euklidesa: dowód poprawności

```
input: m,n
output: d
d := m
k := n
while k ≠ 0 do
 (d,k) := (k,d mod k)
end
```

- Pętla się kończy, ponieważ $0 \leqslant d \bmod k < k$.
- $\mathrm{NWD}(d,k) = \mathrm{NWD}(m,n)$ jest niezmiennikiem pętli.
- Po zakończeniu mamy $k = 0$ i $\mathrm{NWD}(d,k) = \mathrm{NWD}(m,n)$, skąd $d = \mathrm{NWD}(d,0) = \mathrm{NWD}(m,n)$.

Algorytm Euklidesa: dowód poprawności

```
input: m,n
output: d
d := m
k := n
while k ≠ 0 do
  (d,k) := (k,d mod k)
end
```

- $\rightsquigarrow$ Pętla się kończy, ponieważ $0 \leqslant d \bmod k < k$.
- $\rightsquigarrow$ $\mathrm{NWD}(d, k) = \mathrm{NWD}(m, n)$ jest niezmiennikiem pętli.
- $\rightsquigarrow$ Po zakończeniu mamy $k = 0$ i $\mathrm{NWD}(d, k) = \mathrm{NWD}(m, n)$, skąd $d = \mathrm{NWD}(d, 0) = \mathrm{NWD}(m, n)$.

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

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)	(**6**,0)
(9,3)	(20,3)	
(**3**,0)	(3,2)	
	(2,1)	
	(**1**,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)	(**6**,0)
(9,3)	(20,3)	
(**3**,0)	(3,2)	
	(2,1)	
	(**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

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

Algorytm Euklidesa: złożoność

Twierdzenie

Algorytm Euklidesa wykonuje co najwyżej

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

przebiegów pętli.

Dla przypomnienia

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

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

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

Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geqslant n > 0$, *to*

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

Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geqslant n > 0$*, to*

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

Dowód. Mamy

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

Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geqslant n > 0$, *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 $m \geqslant n > 0$*, to*

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

Dowód. Mamy

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

Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geqslant n > 0$, *to*

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

Dowód. Mamy

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

Algorytm Euklidesa: złożoność

Twierdzenie

Jeżeli $m \geqslant n > 0$, *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 $m \operatorname{div} n \geqslant 1$.

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

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

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

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

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

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

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

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

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

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

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

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

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

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

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

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

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

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

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

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

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

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

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

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

Algorytm Euklidesa: złożoność

Złożoność Algorytmu Euklidesa

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

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

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

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

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

Rozszerzony algorytm Euklidesa

```
input: m,n ∈ ℕ0, m + n > 0
output: d = NWD(m,n)
d := m
k := n
while k ≠ 0 do
    (d,k) := (k,d mod k)
end
```

Rozszerzony algorytm Euklidesa

```
input: m,n ∈ ℕ0, m + n > 0
output: d = NWD(m,n)
d := m
k := n
while k ≠ 0 do
    # d = (d div k)k + d mod k
    q := d div k
    (d,k) := (k,d - q * k)
end
```

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

$d = 135$	$k = 40$

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

$d = 135$	$k = 40$
$d = 40$	$k = 135 - 3 \cdot 40$

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

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

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

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

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

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

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
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

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

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

$$5 = 15 - 1 \cdot 10$$

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
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

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

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

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

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
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

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

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

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

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

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

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

$d_0 = 135$	$k_0 = 40$	q_i

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
end
```

$d_0 = 135$	$k_0 = 40$	q_i
$d_1 = 40$	$k_1 = 135 - 3 \cdot 40$	$q_1 = 3$

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
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

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
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

```
d := m
k := n
while k ≠ 0 do
  q := d div k
  (d,k) := (k,d - q * k)
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

```
d := m
k := n
while k ≠ 0 do
  q := d div k
 (d,k) := (k,d - q * k)
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

```
d := m
k := n
while k ≠ 0 do
  q := d div k
 (d,k) := (k,d - q * k)
end
```

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

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

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
 (d,k) := (k,d - q * k)
end
```

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

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

Rozszerzony algorytm Euklidesa

```
d := m
k := n
while k ≠ 0 do
  q := d div k
 (d,k) := (k,d - q * k)
end
```

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

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

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

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

Rozszerzony algorytm Euklidesa

- Mamy
$$d_0 = m$$
oraz
$$q_i = d_{i-1} \text{ div } d_i, \qquad d_{i+1} = d_{i-1} - q_i \cdot d_i$$
dla $i = 1, 2, \ldots, j$ (j - liczba obrotów pętli).

Rozszerzony algorytm Euklidesa

- Mamy
$$d_0 = m$$
oraz
$$q_i = d_{i-1} \text{ div } d_i, \qquad d_{i+1} = d_{i-1} - q_i \cdot d_i$$
dla $i = 1, 2, \ldots, j$ (j - liczba obrotów pętli).
- Dodatkowo wiemy, że
$$d_j = \text{NWD}(m, n).$$

Rozszerzony algorytm Euklidesa

- Mamy
$$d_0 = m$$
oraz
$$q_i = d_{i-1} \text{ div } d_i, \qquad d_{i+1} = d_{i-1} - q_i \cdot d_i$$
dla $i = 1, 2, \ldots, j$ (j - liczba obrotów pętli).
- Dodatkowo wiemy, że
$$d_j = \mathrm{NWD}(m, n).$$
- Chcemy
$$d_j = s \cdot m + t \cdot n.$$

Rozszerzony algorytm Euklidesa

- Mamy
$$d_0 = m$$
oraz
$$q_i = d_{i-1} \text{ div } d_i, \qquad d_{i+1} = d_{i-1} - q_i \cdot d_i$$
dla $i = 1, 2, \ldots, j$ (j - liczba obrotów pętli).
- Dodatkowo wiemy, że
$$d_j = \mathrm{NWD}(m, n).$$
- Chcemy
$$d_j = s \cdot m + t \cdot n.$$
- Skonstruujemy takie ciągi (s_i), (t_i), że
$$d_i = s_i \cdot m + t_i \cdot n, \qquad i = 0, 1, \ldots, j.$$

Rozszerzony algorytm Euklidesa

- Na początku chcemy, aby

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

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

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 $i \geqslant 1$ przyjmujemy

$$s_{i+1} \coloneqq s_{i-1} - q_i s_i, \qquad t_{i+1} \coloneqq t_{i-1} - q_i t_i.$$

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 $i \geqslant 1$ przyjmujemy

$$s_{i+1} \coloneqq s_{i-1} - q_i s_i, \qquad t_{i+1} \coloneqq t_{i-1} - q_i t_i.$$

- Wtedy, jeżeli

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

to

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

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 $i \geqslant 1$ przyjmujemy

$$s_{i+1} \coloneqq s_{i-1} - q_i s_i, \qquad t_{i+1} \coloneqq t_{i-1} - q_i t_i.$$

- Wtedy, jeżeli

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

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

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

Rozszerzony algorytm Euklidesa

```
input: m,n ∈ ℕ₀, m + n > 0
output: d = NWD(m,n)
d := m
k := n
while k ≠ 0 do
    q := d div k
    (d,k) := (k,d - q * k)
end
```

Rozszerzony algorytm Euklidesa

```
input: m,n ∈ ℕ₀, m + n > 0
output: d = NWD(m,n)
d := m
d′ := n
s := 1, s′ := 0
t := 0, t′ := 1

while d′ ≠ 0 do
    q := d div d′
    (d,d′) := (d′,d-q * d′)
    (s,s′) := (s′,s-q * s′)
    (t,t′) := (t′,t-q * t′)
end
```

Rozszerzony algorytm Euklidesa

```
input: m,n ∈ ℕ_0, m + n > 0
output: d = NWD(m,n)
d := m
d' := n
s := 1, s' := 0
t := 0, t' := 1
# d = sm + tn, d' = s'm + t'n
while d' ≠ 0 do
    q := d div d'
    (d,d') := (d',d-q * d')
    (s,s') := (s',s-q * s')
    (t,t') := (t',t-q * t')
end
```

Rozszerzony algorytm Euklidesa

```
input: m,n
output: d = NWD(m,n)
d := m
d′ := n
s := 1, s′ := 0
t := 0, t′ := 1
while d′ ≠ 0 do
  q := d div d′
  (d,d′) := (d′,d-q * d′)
  (s,s′) := (s′,s-q * s′)
  (t,t′) := (t′,t-q * t′)
end
```

i	d_i	q_i	s_i	t_i

Rozszerzony algorytm Euklidesa

```
input: m,n
output: d = NWD(m,n)
d := m
d′ := n
s := 1, s′ := 0
t := 0, t′ := 1
while d′ ≠ 0 do
  q := d div d′
  (d,d′) := (d′,d-q * d′)
  (s,s′) := (s′,s-q * s′)
  (t,t′) := (t′,t-q * t′)
end
```

i	d_i	q_i	s_i	t_i
0	135		1	0

Rozszerzony algorytm Euklidesa

```
input: m,n
output: d = NWD(m,n)
d := m
d′ := n
s := 1, s′ := 0
t := 0, t′ := 1
while d′ ≠ 0 do
  q := d div d′
  (d,d′) := (d′,d-q * d′)
  (s,s′) := (s′,s-q * s′)
  (t,t′) := (t′,t-q * t′)
end
```

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

Rozszerzony algorytm Euklidesa

```
input: m,n
output: d = NWD(m,n)
d := m
d' := n
s := 1, s' := 0
t := 0, t' := 1
while d' ≠ 0 do
  q := d div d'
  (d,d') := (d',d-q * d')
  (s,s') := (s',s-q * s')
  (t,t') := (t',t-q * t')
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

```
input: m,n
output: d = NWD(m,n)
d := m
d′ := n
s := 1, s′ := 0
t := 0, t′ := 1
while d′ ≠ 0 do
  q := d div d′
  (d,d′) := (d′,d-q * d′)
  (s,s′) := (s′,s-q * s′)
  (t,t′) := (t′,t-q * t′)
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

```
input: m,n
output: d = NWD(m,n)
d := m
d′ := n
s := 1, s′ := 0
t := 0, t′ := 1
while d′ ≠ 0 do
  q := d div d′
  (d,d′) := (d′,d-q * d′)
  (s,s′) := (s′,s-q * s′)
  (t,t′) := (t′,t-q * t′)
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

```
input: m,n
output: d = NWD(m,n)
d := m
d′ := n
s := 1, s′ := 0
t := 0, t′ := 1
while d′ ≠ 0 do
  q := d div d′
  (d,d′) := (d′,d-q * d′)
  (s,s′) := (s′,s-q * s′)
  (t,t′) := (t′,t-q * t′)
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

```
input: m,n
output: d = NWD(m,n)
d := m
d′ := n
s := 1, s′ := 0
t := 0, t′ := 1
while d′ ≠ 0 do
  q := d div d′
  (d,d′) := (d′,d-q * d′)
  (s,s′) := (s′,s-q * s′)
  (t,t′) := (t′,t-q * t′)
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			

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