MDM Lista 4

Weronika Jakimowicz

ZAD 1.

Na początku warto zauważyć, że

$$\mathrm{lcm}(n,m) = \frac{nm}{\gcd(n,m)}.$$

Jeśli liczby są wystarczająco duże, może okazać się, że iloczyn nm przekracza górny zakres liczb całkowitych języka z jakiego korzystamy. Żeby temu zapobiedz, możemy podzielić większą z nich przez gcd(n,m) i dopiero wynik pomnożyć przez mniejszą z liczb.
Algorytm napisany w języku Python.

```python
# funkcja obliczajaca gcd na podstawie algorytmu Euklidesa
def gcd(n, m):
    if m == 0: return n
    else: return gcd(m, n % m)


def lcm(n, m):
    div = gcd(n, m)

    # wybranie wiekszej z liczb n,m
    mx = m
    mn = n

    if n > m:
        mn = m
        mx = n

    # dziele wieksza liczbe, zeby na pewno po pomnozeniu nie wyjsc poza zakres
    mx = mx / div

    # tak naprawde zwracam (n*m)/gcd(n,m)
    return mn * mx
```

ZAD 2.

Zauważmy, że

$$\gcd(a,b,c,d) = \gcd(\gcd(a,b),c,d) = \gcd(\gcd(a,b),\gcd(c,d))$$

czyli listę liczb całkowitych m_i możemy za każdym razem dzielić na pół aż dojdziemy do momentu kiedy mamy listy 2 lub 1 elementowe. Zakładamy, że $\gcd(a) = a$.
Poniższy algorytm, napisany w Pythonie, jest analogiczny do merge sort, gdzie dzielimy listę na podlisty o podobnym rozmiarze i wykonujemy na nich operacje, po czym łączymy je z powrotem w całość.

```python
# implementacja algorytmu Euklidesa jak w Zad 1.
def euclid(n, m):
    if m == 0: return n
    else: return euclid(m, n % m)

def gcd(k, M):
    if k == 1: return M[0] # gcd(a) = a
    if k == 2: return euclid(M[0], M[1]) # mamy liste dwuelementowa

```

```
    else:
        # na poczatku rozbijamy liste na dwie podlisty: L i R
        L = []
        R = []

        i = 0
        while i < k//2:
            L.append(M[i])
            i += 1
        while i < k:
            R.append(M[i])
            i += 1

        # gcd(M) to gcd(L, R) - analogicznie jak w merge sort
        return gcd(2, [gcd(k//2, L), gcd(k-k//2, R)])
```

ZAD 3.

Zacznijmy od pomysłu nie w pełni wydajnego. Chcemy znaleźć wyznaczniki takie, że

$$x_1m_1 + x_2m_2 + \ldots + x_km_k = \gcd(m_1, m_2, \ldots, m_k).$$

Możemy zająć się najpierw problemem pośrednim i znaleźć

$$x_1m_1 + y_1(m_2 + m_3 + \ldots + m_k) = \gcd(m_1, \gcd(m_2, \ldots, m_k)) = \gcd(m_1, \ldots, m_2).$$

Po odpowiedniej liczbie takich modyfikacji powinniśmy otrzymać

$$x_1m_1 + y_1(m_2 + y_2(m_3 + \ldots + y_{k-2}(m_{k-1} + y_{k-1}m_k))) = \gcd(m_1, \gcd(m_2, \gcd(\ldots \gcd(m_k)))).$$

Rozwiązanie tego problemu od najbardziej zagnieżdżonych do najmniej zagnieżdżonych nawiasów korzysta z rozszerzonego algorytmu Euklidesa. W jego oryginalnej wersji szukamy a, b takich, że

$$ax + by = \gcd(x, y).$$

Czyli za każdym razem chcemy doprowadzić do sytuacji, gdzie mamy dwie liczby x, y. Pierwszy pomysł po zaimplementowaniu będzie przechodził przez k liczb, co nie jest bardzo efektowne. Zastanówmy się, czy możemy to przyśpieszyć.
Rozważmy przypadek dla $k = 5$.

$$x_1m_1 + x_2m_2 + x_3m_3 + x_4m_4 + x_5m_5 = \gcd(m_1, \ldots, m_5)$$

$$z_1(m_1 + m_2 + m_3) + y_1(m_4 + m_5) = \gcd(\gcd(m_1, m_2, m_3), \gcd(m_4, m_5))$$

$$z_1(z_2(m_1 + m_2) + m_3) + y_1(m_4 + m_5) = \gcd(\gcd(\gcd(m_1, m_2), m_3), \gcd(m_4, m_5))$$

Zauważmy, że możemy nadal dzielić listę na połowe i wywoływać rozszerzony algorytm Euklidesa kiedy już dojdziemy do 2 elementów. W przypadku jednoelementowego odcinka możemy zwracać ten jedyny element.

```
def ext_gcd(a, b):
    R = [a, b]
    S = [1, 0]
    T = [0, 1]

    while R[1] != 0:
        q = R[0] // R[1]
        R = [R[1], R[0] - q * R[1]]
        S = [S[1], S[0] - q * S[1]]
        T = [T[1], T[0] - q * T[1]]

    ret = [S[0], T[0]]

    return ret

def zad(k, M):
```

```
    if k == 1: return [M[0]]
    if k == 2:
        return ext_gcd(M[0], M[1])

    else:
        L = []
        R = []

        i = 0

        # dziele liste na dwie podlisty
        while i < k//2:
            L.append(M[i])
            i += 1
        while i < k:
            R.append(M[i])
            i += 1

        # wykonuje na nich rekurencyjnie to samo dzialanie
        coef_L = zad(k//2, L)
        coef_R = zad(k-k//2, R)

        new_L = []
        new_R = []

        # mnoze wczesniej otrzymane wspolczynniki przez aktualnie otrzymane, a potem sumuje
        i = 0
        while i < len(L):
            new_L.append(coef_L[i] * L[i])
            i += 1

        i = 0
        while i < len(R):
            new_R.append(coef_R[i] * R[i])
            i += 1

        sum_L = sum(new_L)
        sum_R = sum(new_R)

        # wspolczynniki przy sumie podlist
        vec = ext_gcd(sum_L, sum_R)

        i = 0
        while i < len(L):
            coef_L[i] *= vec[0]
            i += 1
        i = 0
        while i < len(R):
            coef_R[i] *= vec[1]
            i += 1

        # zwroc wspolczynniki lewej podlisty zlaczone ze wspolczynnikami prawej podlisty
        ret = coef_L + coef_R

        return ret
```

ZAD 4.

Jeśli a,b są parzyste, to obie są podzielne przez 2, więc 2 wystąpi co najmniej raz w rozkładzie gcd(a,b) na czynniki pierwsze. Możemy więc zapisać, że istnieją $a', b' \in \mathbb{N}$ takie,

że $a = 2a', b = 2b'$ oraz

$$\gcd(a, b) = 2\gcd(a', b').$$

```python
def binary_gcd(a, b):
    if a == 0:
        return b

    # XOR sprawdza, czy tylko jedna liczba jest parzysta
    bol = (a % 2 == 0) ^ (b % 2 == 0)

    if bol:
        if a % 2 == 1: return binary_gcd(b, a)
        return binary_gcd(a//2, b)

    # jezeli obie sa nieparzyste
    elif (a+b) % 2 == 0:
        if a > b: return binary_gcd(a-b, b)
        return binary_gcd(b-a, a)

    # jezeli obie sa parzyste, to na pewno obie dzieli 2, wiec mozna zwrocic 2 * gcd(a/2, b/2)
    else:
        return 2 * binary_gcd(a//2, b//2)
```

Zauważmy, że jeśli chociaż jedna z liczb a, b jest parzysta, to zmniejszamy je 2 razy. Jeżeli obie są nieparzyste, to $a - b$ jest parzyste i wtedy co drugi krok zmniejszamy liczby o polowe. Czyli co najwyżej co dwa obroty zmniejszamy liczby dwukrotnie, więc wykona się $2\log_2 \max(a, b)$ operacji, co daje nam złożoność $O(\log_2 \max(a, b))$.

ZAD 5.

ZAD 6.

Zapis liczby m w układzie kolejnych liczb pierwszych to zapis postaci

$$m = \prod_{i=1}^{\infty} q_i^{m_i}$$

gdzie q_i to kolejne liczby pierwsze. Oczywiście, dla skończonych liczb naturalnych m od pewnego momentu k dla każdego $i > k$ będzie $m_i = 0$.

a) $k = \gcd(m, n) \iff k_i = \min(m_i, n_i)$ dla każdego $i = 1, 2...$

$\Rightarrow$

Skoro $k = \gcd(m, n)$, to k jest największe takim, że $k|m$ oraz $k|n$.Podzielność obu liczb jest oczywista. Weźmy dowolną liczbę pierwszą q_i. Wtedy, ponieważ q_i jest pierwsze, to musi zachodzić

$$q_i^{k_i} | m \wedge q_i^{k_i} | n.$$

Znowu, dla m i n sprowadza się to do podzielnośći q^{m_i} oraz q^{n_i} przez q^{k_i}. W takim razie k_i jest największym takim, że

$$k_i \leq m_i \wedge k_i \leq n_i,$$

więc $k_i = \min(m_i, n_i)$.

$\Leftarrow$

Wiemy, że dla każdego $i \in \mathbb{N}$ zachodzi $k_i = \min(m_i, n_i)$. Chcemy pokazać, że wtedy k dzieli jednocześnie n i m oraz, że jest to największy taki dzielnik.
To, że dzieli jest proste:

$$\frac{q^{m_i}}{q^{k_i}} = q^{m_i - k_i} \geq 1$$

$$\frac{q^{n_i}}{q^{k_i}} = q^{n_i - k_i} \geq 1$$

i obie te liczby są naturalne, bo jeśli $m_i \geq n_i$, to $q^{n_i - k_i} = 1$, natomiast

$$(\exists\, a \in \mathbb{N})\; m_i - k_i = a$$

co daje $q^{m_i - k_i} = q^a \in \mathbb{N}$.
Załóżmy teraz, że istnieje liczba $p > k$ taka, że $p|m$ i $p|n$. Wtedy istnieje chociaż jedno i takie, że $p_i > k_i$. Niech więc $p_i = k_i + d$, gdzie $d \in \mathbb{N}$, $d \geq 1$. Znowu załóżmy, że $m_i \geq n_i$.

$$\frac{q_i^{n_i}}{q_i^{p_i}} = \frac{q^{n_i}}{q_i^{k_i+d}} = \frac{q^{n_i}}{q_i^{n_i+d}} = q^{n_i - n_i - d} = q^d$$

ale wtedy $q^{p_i} \nmid q^{n_i}$, czyli $p \nmid n$ i mamy sprzeczność. W takim razie k jest największą liczbą dzielącą jednocześnie n i m.

b) $k = \mathrm{lcm}(m,n) \iff k_i = \max(m_i, n_i)$ dla każdego $i = 1, 2 \ldots$

$\Rightarrow$
Skoro $k = \mathrm{lcm}(m,n)$ to k jest najmniejsze takie, że $m|k$ i $n|k$. Weźmy dowolną liczbę pierwszą q_i. Tak samo jak w poprzednim podpunkcie, wystarczy, żeby potęgi liczby pierwszej były przez siebie podzielne:

$$q_i^{m_i} | q_i^{k_i} \;\wedge\; q_i^{n_i} | q_i^{k_i},$$

czyli $k_i \geq m_i$ oraz $k_i \geq n_i$. Ponieważ k ma być najmniejszą wspólną wielokrotnością, to szukamy najmniejszego takiego k_i, czyli

$$k_i = \max(m_i, n_i).$$

$\Leftarrow$
Po pierwsze, chcemy pokazać że $m|k$ oraz $n|k$. Wystarczy, że pokażemy dla dowolnej liczby pierwszej q_i, że $q_i^{m_i} | q_i^{k_i}$ i $q_i^{n_i} | q_i^{k_i}$.

$$\frac{q^{k_i}}{q^{m_i}} = q^{k_i - m_i} \geq 1$$
$$\frac{q^{k_i}}{q^{n_i}} = q^{k_i - n_i} \geq 1$$

Teraz, jeżeli BSO $m_i \geq n_i$, to $m_i = k_i$ i

$$q^{k_i - m_i} = 1 \in \mathbb{N}$$
$$q^{k_i - n_i} = q^{m_i - n_i} \in \mathbb{N}$$

a więc $k|m$ i $k|n$.
Załóżmy, nie wprost, że istnieje liczba p mniejsza niż k, która jest wielokrotnością m i n. Wtedy na chociaż jednym miejscu, niech będzie to i, mamy $p_i < k_i$, a więc istnieje $d \in \mathbb{N}$ takie, że $p_i = k_i - d$. Mamy wtedy

$$\frac{q^{p_i}}{q^{m_i}} = \frac{q^{k_i - d}}{q^{m_i}} = \frac{q^{m_i - d}}{q^{m_i}} = q^{m_i - d - m_i} = q_i^{-d}$$

co nie jest liczbą naturalną, a więc $m \nmid p$ i mamy sprzeczność. W takim razie $k = \mathrm{lcm}(m,n)$.

ZAD 7.

a) $xz \equiv yz \mod mz \iff x \equiv y \mod m$ dla $z \neq 0$

$\Leftarrow$
Skoro $x \equiv y \mod m$, to znaczy, że istnieje $k \in \mathbb{Z}$ takie, że

$$x = km + y.$$

Jeżeli teraz pomnożymy obie strony przez z, to dostaniemy

$$xz = kmz + yz.$$

To oznacza, że xz jest podzielne przez mz reszta yz, czyli

$$xz \equiv yz \mod mz.$$

$\Rightarrow$

Jeżeli $xz \equiv yz \mod mz$, to dla pewnego $k \in \mathbb{Z}$ zachodzi

$$xz = kmz + yz$$

skoro $z \neq 0$, to możemy obustronnie podzielić przez z

$$x = km + y.$$

Z tego z kolei wynika, że

$$x \equiv y \mod m$$

b) $xz \equiv yz \mod m \iff x \equiv y \mod \frac{m}{\gcd(z,m)}$ dla $x, y, z, m \in \mathbb{Z}$

$\Leftarrow$

Z założenia, że $x \equiv y \mod \frac{m}{\gcd(z,m)}$, to istnieje $k \in \mathbb{Z}$ takie, że

$$x = k\frac{m}{\gcd(z,m)} + y.$$

Jeżeli pomnożymy obie strony przez z, to dostaniemy

$$xz = km\frac{z}{\gcd(z,m)} + yz$$

I zauważmy, że $\frac{z}{\gcd(z,m)} \in \mathbb{Z}$, czyli istnieje $p \in \mathbb{Z}$ takie, że

$$xz = pm + yz$$

a więc

$$xz \equiv yz \mod m$$

$\Rightarrow$

Zakładamy, że $xz \equiv yz \mod m$, czyli istnieje $k \in \mathbb{Z}$ takie, że

$$xz = km + yz$$

Podzielmy teraz obustronnie przez $\gcd(z,m)$, dostajemy:

$$x\frac{z}{\gcd(z,m)} = \frac{k}{\gcd(z,m)}m + y\frac{z}{\gcd(z,m)}.$$

Zauważmy, że istnieje pewno $p \in \mathbb{Z}$ takie, że

$$z = p \cdot \gcd(z,m)$$

oraz $\gcd(p,m) = 1$. Dalej mamy

$$xp = k\frac{m}{\gcd(z,m)} + yp \quad (☕)$$

$$p(x-y) = k\frac{m}{\gcd(z,m)}$$

Ponieważ prawa strona równania jest liczbą całkowitą podzielną przez p, to również lewa strona musi być podzielna przez p. Zauważmy, że $p \nmid m$, a więc również $p \nmid \frac{m}{\gcd(z,m)}$. W takim razie $p \mid k$ i wtedy $\frac{k}{p} \in \mathbb{Z}$. Czyli jeśli podzielimy obie strony równania (☕) przez p, dostajemy

$$x = \frac{k}{p}\frac{m}{\gcd(z,m)} + y$$

i z tego wynika, że

$$x \equiv y \mod \frac{m}{\gcd(z,m)}$$

c) $x \equiv y \mod mz \implies x \equiv y \mod m$

Ponieważ zwykle operacja modulo n jest zdefiniowana dla n całkowitych, założę, że jednocześnie mz jak i m są liczbami całkowitymi. Dodatkowo, $z \in \mathbb{Z}$, bo jeśli $m = 18$, $z = \frac{1}{3}$ i $x = 10$:

$$10 \equiv 4 \mod 6$$

$$10 \equiv 10 \mod 18$$

to mamy sprzeczność.

Istnieje $k \in \mathbb{Z}$ takie, że $\gcd(a,b) = 1$ oraz

$$x = kmz + y$$

i wtedy również $kz \in \mathbb{Z}$, czyli $x \equiv y \mod m$.

ZAD 8.

a) $2^n - 1$ – liczba pierwsza $\implies$ n jest liczbą pierwszą

Załóżmy nie wprost, że istnieje n takie, że n nie jest liczbą pierwszą, ale $2^n - 1$ jest liczbą pierwszą. Ponieważ n nie jest pierwsze, to istnieją $k, m \in \mathbb{Z}$ większe od 1 takie, że $n = km$. Wtedy

$$2^n - 1 = 2^{km} - 1 = (2^k)^m - 1 = (2^k - 1)(2^{k(m-1)} + 2^{k(m-2)} + \ldots + 2^1 + 2^0)$$

Ponieważ $2^k > 2^1 = 2$, to mnożymy liczbę całkowitą $(2^{k(m-1)} + 2^{k(m-2)} + \ldots + 2^1 + 2^0)$ przez liczbę całkowitą różną od 1 $(2^k - 1)$. W takim razie $2^n - 1$ nie jest liczbą pierwszą i mamy sprzeczność.

b) $a^n - 1$ jest liczbą pierwszą, to $a = 2$

Wiemy, że $a^n - 1$ jest liczbą pierwszą, czyli nie ma dzielników całkowitych. Podobnie jak w poprzednim podpunkcie, rozpiszmy wyrażenie jako iloczyn sum:

$$a^n - 1 = (a - 1)(a^{n-1} + a^{n-2} + \ldots + a^1 + a^0).$$

W takim razie $a - 1$ musi być równe 1, wtedy $a = 2$, lub

$$a^{n-1} + a^{n-2} + \ldots + a + 1 = 1,$$

a więc

$$a^{n-1} + a^{n-2} + \ldots + a = 0$$

co dawałoby $a = 0$, co nie jest poprawne, bo $0^n - 1 = -1$ nie jest liczbą pierwszą.

c) $2^n + 1$ – liczba pierwsza $\implies (\exists\, k)\, 2^k = n$

Załóżmy nie wprost, że n nie jest potęgą liczby 2. W takim razie $n = km$ gdzie $k, m \in \mathbb{N}$ i m jest liczbą nieparzystą.

$$2^n + 1 = 2^{km} + 1 = (2^k)^m + 1$$

niech $a = 2^k$, wtedy

$$a^x + 1 = a^x - (-1)^x = (a + 1)(a^{x-1} + a^{x-2} + \ldots + a + 1)$$

Czyli 2^n+1 jest iloczynem $(a+1) = 2^y+1$ gdzie $y > 1$, więc 2^n+1 jest iloczynem dwóch liczb całkowitych i nie może być liczbą pierwszą.