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)])
```