Zad 5

zad 3

Policzymy trzy tablice dynamicznie: $\overbrace{\text{silnia}[\,]}^{\text{silnie mod } m}$, $\underbrace{\text{silniaINV}[\,]}_{\text{silnia odwrotności modulo } m}$, $\overbrace{\text{INV}[\,]}^{\text{odwrotności mod } m}$

Wtedy $\binom{m_i}{k_i}$ obliczymy w czasie stałym:

$$((\text{silnia}[n] * \text{silniaINV}[k]) \% m * \text{silniaINV}[n-k]) \% m$$

Teraz pokażę, jak obliczyć te tablice w $O(\max\{m_1, \dots, m_i\})$:

- silnia[]

$N = \max\{m_i\}$

```
silnia[0]=1
FOR i IN (1,N):
    silnia[i] = silnia[i-1] * i % M
```

- silniaINV[]

```
silniaINV[1] = 1
FOR i IN (2,N):
    SILNIAINV[i] = SILNIAINV[i-1] • INV[i] % M
```

$$(x!)^{-1} \equiv ((x-1)! \cdot x)^{-1} \equiv (x-1)!^{-1} x^{-1}$$

- INV[]

```
INV[1]
FOR i IN (2,N):
    INV[i] = m - (m // i) * inv[m % i] % m
```

bo:

$$m \bmod i = m - \left\lfloor \frac{m}{i} \right\rfloor \cdot i$$

$$\Downarrow$$

$$m \bmod i \equiv -\left\lfloor \frac{m}{i} \right\rfloor \cdot i \mod m$$

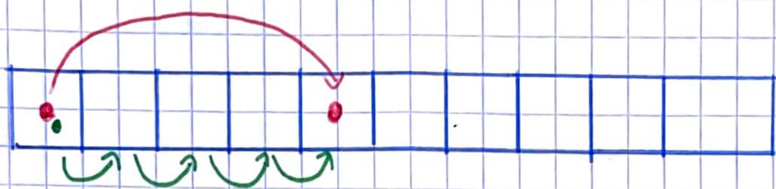
$$\Downarrow$$

$$(m \bmod i) \cdot i^{-1} \cdot (m \bmod i)^{-1} \equiv -\left\lfloor \frac{m}{i} \right\rfloor \cdot i \cdot i^{-1} \cdot (m \bmod i)^{-1}$$

$$\Downarrow$$

$$\underbrace{i^{-1} \equiv -\left\lfloor \frac{m}{i} \right\rfloor \cdot (m \bmod i)^{-1} \mod m}$$

Zad. 5



Rozpatrzmy 3 stany:

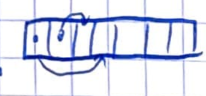
- O[i] pionki są na pozycjach $i-1, i$; O[i] – najmniejszy koszt dotarcia do tego stanu
  T[i] – ——— " ———
- T[i] oba są na pozycji i
- Pionki na pozycjach i, j $(i-j>1)$ ([i,i] nie ciągłe) , ten jest „ruchy" możemy tylko na jeden sposób dojść do O[i].

SUMY PREFIXOWE

$$O[i] = \min\left(\begin{array}{l} \min\limits_{2\le j<i} \left\{ O[j] + w_{j-1,i} + \overbrace{\sum_{k=j}^{i-2} w_{k,k+1}}^{\text{SUMY PREFIXOWE}} \right\}, \quad ' \\ \min\limits_{1\le j<i} \left\{ T[j] + w_{j,i} + \sum_{k=j}^{i-1} w_{k,k+1} \right\} \quad '' \end{array} \right)$$

$T[i] = O[i] + w_{i-1,i}$ (Ruchy pionków są pełne, zawsze na $i-1$ ktoś musi stać i wtedy każdy stan)

- I Decyzję ruch robimy zawsze lewym pionkiem, bo wpp. „zdublujemy" sytuację
- Nie idziemy na przemian bo dojdziemy w końcu do zapętlenia, które mielibyśmy



Nie mogą się spotkać w ...

Złożoność:

Czasowa: $O(n^2)$

Pamięciowa: $O(n)$

Zad. 7

$S = \sum a_i$

$T[i][j] = 1 \Leftrightarrow$ możemy podzielić zbiór na sumy i, j, $S-i-j$.
$(=0)$ w.p.p.

$T[0][0] = 1$

```
for k in range(1,n):        // próbujemy dołożyć a_k
    if a[k] < 0:
        sign = -1
    else:
        sign = 1

    for i in range(C·n·sign, -C·n·sign):
        for j in range(C·n·sign, -C·n·sign):
            if T[i][j]:
                T[i+a[k]][j] = 1
                T[i][j+a[k]] = 1


return T[S/3][S/3]
```

Pamięć: $O(C^2 n^2)$
Czas: $O(C^2 n^3)$