Lista 4

tags: `AiSD`

Zadanie 1

tablica $\{a_{ij}\}$

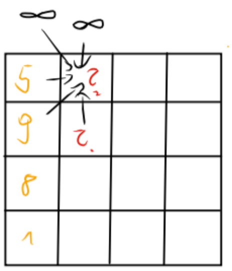
5	1	1	1
9	1	8	6
8	1	9	1
1	1	7	3

Tablica T

Nowy problem:

Dopuszczalny ruchy

2. Przy wyznaczeniu kolejnej kolumny napotkamy na problem, bo mamy za mało informacji jak wpisać cokolwiek w jakąkolwiek komórkę.



:::info

- Wkażdej komórce trzymamy najkrótsze ścieżki do tego miejsca

1. Wypełniamy pierwszą kolumnę liczbami z tabelicy T
2. Złożoność to $O(n^2)$ :::

Zadanie 3

:::info Zadanie można rozwiązać trywialnie, tj. spamiętać wyniki silnii. To jest w porządku dla małych liczb, ale da dużych da się lepiej. :::

:::success Pokażmy najpierw pewną zależność: $$ x = \frac{a}{b} $$ $$ bx = a $$ $$ xb \equiv_p a \space\space /\cdot b^{-1} $$ $$ x \equiv_p ab^{-1} $$ :::

Wiemy, że $\binom{n}{k} = \frac{n!}{(n-k)!k!}$. Zauważmy, że

$$\binom{n}{k}(n-k)!k! = n!$$

$$\binom{n}{k}(n-k)!k! \equiv_p n! \quad /((n-k)!k!)^{-1}$$

istnieje, bo p jest liczbą pierwszą

$$\binom{n}{k} \equiv_p n!(n-k)!^{-1}(k!)^{-1}$$

:::danger ***Pewna ważna własność:*** ab mod n = [(a mod n)(b mod n)] mod n :::

We have:

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

Taking both sides modulo m yields:

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

Multiply both sides by $i^{-1} \cdot (m \bmod i)^{-1}$ yields

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

which simplifies to:

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

Aby zoptymalizować ilość wykonywanych obliczeń silni, stworzymy tablicę (o nazwie fact) o wielkości $N=max(n_{i})$, dla $i = 1, ..., r$, do której wpiszemy wszystkie obliczone silnie%p za jednym razem co kosztować nas będzie $O(N)$.

```
 fact[0] = 1;
for(i = 1; i<=N; i++)
   fact[i] = fact[i-1] * i % p;
```

Ponieważ z polecenia zadania wiemy, że $k\leq n$ to silnie k mamy już zapisane w tablicy fact.

Powyższy lemat wykorzystamy do obliczenia $(k!)^{-1}$ i ${((n-k)!)^{-1}}$.

Korzystamy z faktu, że $x!^{-1} \equiv_p (x-1)!^{-1} \cdot x^{-1}$

```
inv[1] = 1;
for(int i = 2; i <= N; i++)
{
    inv[i] = (-(p/i) * inv[p%i]) % p
}

inv_fact[0] = 1;
for(i = 1; i<=N; i++)
    inv_fact[i] = (inv_fact[i-1] * inv[i]) % p;
```

:::info fact <- tablica silnii od 0 do max(n1, n2...) mod p inv <- modularne odwrotności od 1 do max(n1, n2, ...) inv_fact <- odwrotności silnii $k!^{-1}$ :::

Wypisujemy wynik

```
 for i = 1, 2, ... r:
        print(((fact[n_i] * inv_fact[k_i] % p) * inv_fact[n_i - k_i]) % p)
```

Złożoność to $O(max), bo każda pętla wykona się max razy