# Christmas Party / Karácsonyi parti

A feladat lényege, hogy meghatározzuk, hányféleképpen lehet szétosztani a karácsonyi ajándékokat úgy, hogy minden gyerek kap egy ajándékot, de nem a sajátját. Ez a feladat arra kérdez rá, hogy hány olyan permutáció létezik egy halmazban, ahol egy elem sem marad a helyén. Azaz, ha van n gyerek, akkor hányféleképpen lehet őket úgy elrendezni, hogy egyik gyerek sem kapja vissza a saját ajándékát.

**Megoldás lépései:**

1. **Kombinatorikai képlet**: A permutációk számát egy rekurzív képlettel számolhatjuk ki:

$$D(n)=(n-1)\times(D(n-1)+D(n-2))$$

   - Az első elem két lehetséges helyen lehet (azaz, ne a saját helyén legyen).
   - Az n elem megfelelő elrendezése a hátralévő n-1 elem
2. **Alapértékek**:
   - D(1)=0D(1) = 0D(1)=0, mert ha csak egy gyerek van, akkor ő biztosan a saját ajándékát kapja.
   - D(2)=1D(2) = 1D(2)=1, mert két gyerek esetén csak egy lehetséges permutáció van, hogy ne a saját ajándékukat kapják.
3. **Moduláris aritmetika**: A számítások nagymértékűek lehetnek, így minden eredményt $10^9$+7-el kell leosztani, hogy elkerüljük a túlcsordulást és a számok kezelhetetlenségét.
4. **Dinamika**: Az optimális megoldás érdekében egy dinamikus programozás (DP) alapú megoldást alkalmazunk, ahol először kiszámoljuk a kisebb n-ek lehetőségeit, majd azokat felhasználva számoljuk ki a nagyobb n-eket.

**Python megoldás:**

```
MOD = 10**9 + 7


def derangements(n):
    # Ha csak 1 gyerek van, nincs érvényes permutáció (mert mindenkinek saját ajándékot adna)
    if n == 1:
        return 0
    # Ha két gyerek van, akkor egyféleképpen lehet szétosztani
    if n == 2:
        return 1

    # Létrehozzuk a DP tömböt, hogy kiszámoljuk a derangementeket
    dp = [0] * (n + 1)
```

```
    # Alapértékek
    dp[1] = 0
    dp[2] = 1

    # Kitöltjük a DP tömböt a képlettel
    for i in range(3, n + 1):
        dp[i] = (i - 1) * (dp[i - 1] + dp[i - 2]) % MOD

    return dp[n]

# Bemenet beolvasása
n = int(input())

# Eredmény kiírása
print(derangements(n))
```

**Magyarázat:**

1. **MOD konstans**: A MOD értéke $10^9+7$, amit minden számításhoz használunk, hogy az eredményeket a modulo értékével számoljuk.
2. **Függvény derangements(n)**:
   - Először két alapértelmet adunk meg:
     - D(1)=0D(1) = 0D(1)=0 (mivel egyetlen gyerek van, így nem lehet az ajándékot másnak adni).
     - D(2)=1D(2) = 1D(2)=1 (két gyerek esetén pontosan egy lehetséges mód van, hogy ne kapják vissza a saját ajándékukat).
   - Ezután dinamikusan számoljuk ki minden i értékre, hogy hányféleképpen lehet elrendezni az i elemű halmazt úgy, hogy egyik elem sem marad a helyén.

     A rekurzív képletet alkalmazzuk:

$$D(i)=(i-1)\times(D(i-1)+D(i-2)) \bmod 10^9+7$$

Ez azt jelenti, hogy ha az i-edik elem nem marad a helyén, akkor az i - 1 és i - 2 számú derangementek kombinációját kell figyelembe venni.

3. **Bemenet és kimenet**:

  - Az első sorban beolvassuk a gyerekek számát, n.
  - Majd a derangements(n) függvény segítségével kiszámoljuk, hányféleképpen lehet szétosztani az ajándékokat úgy, hogy egyik gyerek se kapja vissza a saját ajándékát.
  - Az eredményt a modulo $10^9+7$ értékével kiírjuk.

**Példa:**

**Bememenet:**

4

**Kimenet:**

9

**Miért működik?**

- A probléma egy klasszikus derangement probléma, ahol a cél, hogy meghatározzuk, hány olyan permutáció létezik, ahol egy elem sem marad a saját helyén.
- A dinamikus programozás (DP) segítségével minden n-re megtartjuk a legjobb megoldásokat a kisebb problémákból, és azokat építjük be a nagyobb problémákba, így optimalizálva a számításokat.

**Idő- és memória komplexitás:**

- **Idő komplexitás**: O(n), mivel egy egyszerű iterációval kiszámoljuk minden egyes derangementet az n-től kezdve.
- **Memória komplexitás**: O(n), mivel a DP tömböt használjuk a számítások tárolására.

Ez a megoldás megfelel az $n \leq 10^6$ korlátnak is.

CSES - Christmas Party - Results