Counting Tilings

A feladat az, hogy megszámoljuk, hányféleképpen lehet kitölteni egy n×m-es rácsot 1× 2 és 2×1 méretű csempékkel.

Bemenet
Az egyetlen sor két egész számot tartalmaz: n és m.

Kimenet
Nyomtass ki egy egész számot: a lehetséges kitöltési módok számát 10^9+7-es maradék szerint.

Korlátok
$1 \leq n \leq 10$
$1 \leq m \leq 1000$

Példa
Bemenet:
4 7

Kimenet:
781

Probléma elemzése:
Egy n×m-es rácsot kell lefedni 1×2 és 2×1 méretű csempékkel. A feladat az, hogy kiszámítsuk, hányféleképpen lehet lefedni a rácsot úgy, hogy minden mező teljesen lefedett legyen.

Algoritmus kidolgozása
Maszkok és bitmask reprezentáció: A rács egy oszlopát bináris számként (maszkként) kezeljük, ahol 1 jelzi, hogy a mező már le van fedve, 0 pedig azt, hogy a mező még nincs lefedve. Minden oszlopban a csempézés aktuális állapotát a maszk írja le.

Rekurzív megoldás dinamikus programozással:
A dp[i][mask] tárolja az i-edik oszlopig tartó rács lefedési módjainak számát, ha az i-edik oszlop aktuális állapota a "mask". A mask jelzi az i-edik oszlop celláinak lefedettségét.

Következő oszlop maszkjainak generálása: Az aktuális oszlopból generáljuk a következő oszlop állapotait (maszkokat), figyelembe véve a 1×2 és 2×1 csempéket. Csak olyan maszkok generálhatók, amelyek érvényesek a rács szélessége (n) szerint.

Memoizáció: Az dp[i][mask]-ben tároljuk az előzőleg kiszámolt eredményeket, hogy ne kelljen újraszámolni őket.

Lépések:

1. Maszkok generálása: A következő oszlop összes lehetséges maszkját az aktuális oszlop maszkja alapján generáljuk. Például, ha az aktuális oszlop maszkja 101 (binárisan), akkor a következő maszk lehet 010, amelyet egy 1×21 \times 2-es csempe generált.
2. Rekurzív lefedési számítás: A maszkok alapján számítjuk ki a lefedési módok számát az aktuális oszlop és maszk alapján.
3. Memoizáció: Ha egy adott oszlop-maszk kombinációra már van eredmény, akkor azt közvetlenül visszaadjuk a memoizációs táblából.

Hatékonyság:
Az algoritmus időbonyolultsága $O(m \times 2^n)$, mivel minden oszlopot (m) és annak összes állapotát (2^n) vizsgáljuk. A térbonyolultság $O(m \times 2^n)$, mivel a DP-táblában tároljuk az oszlopok és maszkok kombinációit.

Python megvalósítás

https://github.com/pipdom/L_Algoritmusok_es_adatszerkezetek/blob/main/tilings.py

CSES teszt eredmények

Submission details

Task:	Counting Tilings
Sender:	pipdom
Submission time:	2024-12-15 10:33:41 +0200
Language:	Python3 (PyPy3)
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.04 s	»
#2	ACCEPTED	0.04 s	»
#3	ACCEPTED	0.04 s	»
#4	ACCEPTED	0.04 s	»
#5	ACCEPTED	0.04 s	»
#6	ACCEPTED	0.04 s	»
#7	ACCEPTED	0.05 s	»
#8	ACCEPTED	0.05 s	»
#9	ACCEPTED	0.07 s	»
#10	ACCEPTED	0.09 s	»
#11	ACCEPTED	0.35 s	»
#12	ACCEPTED	0.36 s	»
#13	ACCEPTED	0.40 s	»
#14	ACCEPTED	0.71 s	»
#15	ACCEPTED	0.74 s	»