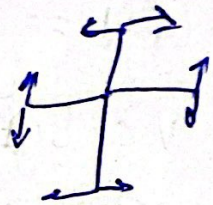
Problem Definition: Knight Dialer



- We have a Knight Piece, which can move
- We also have a Phone Pad

1	2	3
4	5	6
7	8	9
X	0	X

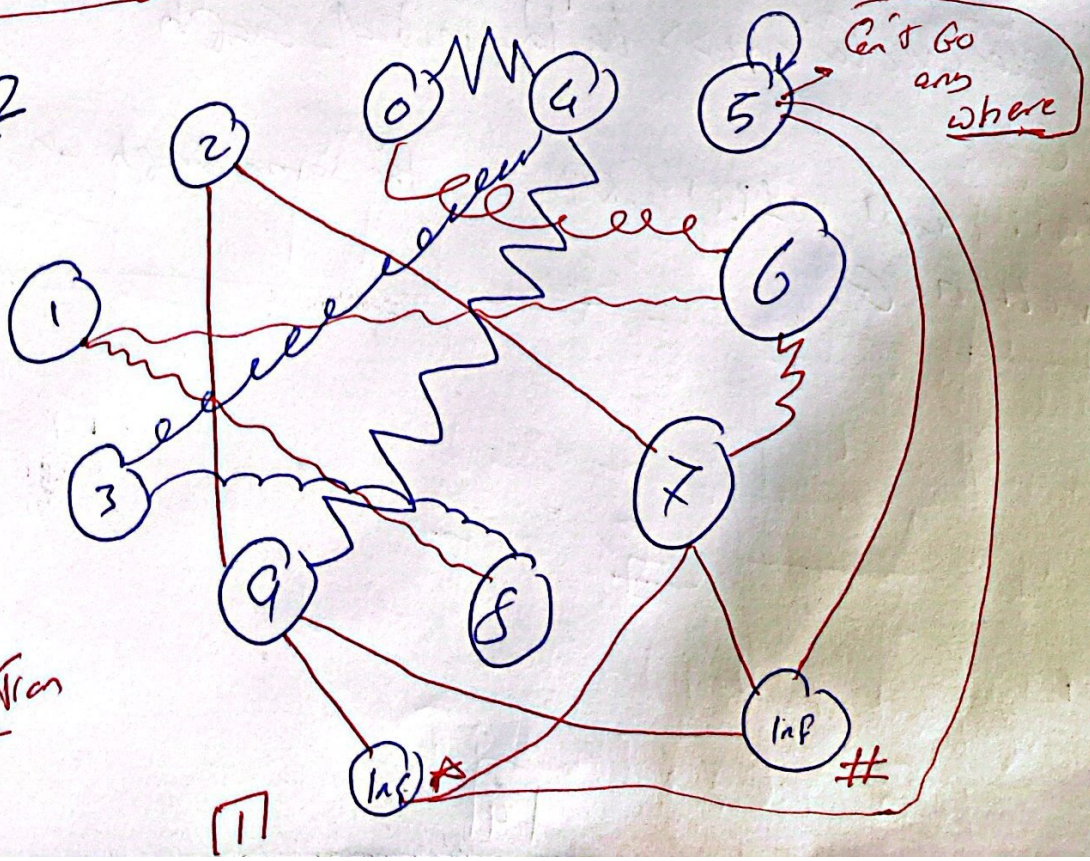
- the Knight can stand only on number
- we initially allowed to place the Knight on any numeric cell, then we need to do n-1 jumps to dial a number of length n
- all jumps must be valid jumps.

- we are asked to:
  - given an integer [n], we went to get how many distinct phone numbers of length [n] we can dial
  - so we should place the Knight over all digits then sum up the total result @ the end.
  - the answer we return should be modulo 10^9+7

Constrains:-
- $1 \le n \le 5000$

Test Cases:-
TI: @ n=1 we can dial only our digits [0, 1 — 9] : res = 10 ✓✓

TI: @ n=2



this Graph is constant so no solution

11

∴ Adjacency List :-

0	4,6
1	6,8
2	7,9
3	8,4
4	0,3,9
5	X
6	7,1,0
7	6,2
8	1,3
9	4,2
No we can start from each digit, then apply depth-limited dfs with d = n	

∴ @ n = 2

starting with depth limit = 2 & stop then apply back tracks

starting from 0 :- 0,4 | 0,6 — 2

starting from 1 :- 1,6 } 1,8 — 2

starting from 2) 2,7 } 2,9 — 2

3 — 3,8, 34 — 2

4 — 4,0 | 43 } 49 — 3

5 — 5 — don't count

6 — 67 61 60 — 3

7 — 76, 72 — 2

8 — 81, 83 — 2

9 — 94 92 — 2

∴ **Total Sum = 20** ✓✓

@ n = 3

هنا هيطلعلك مشكلة انك تدخل ف cycle

المفروض الرقم ده تعده 0404040404 مثلاً الحل لازم نلاقي طريقة نخلي الرقم ميطلعش + حل ال complexity متبقى fit !؟

انا شايف مشكلة ممكن نطبق DP ف الجزء ده لو ال x متبقى

@ n=3 for example:-

Starting from 0:-

040 | 043 | 049

067 | 061 | 060

those was previously computed when remaining depth = 1 @ curr dig = 6

Starting from 1:-

167 | 161 | 160

181 | 183

Starting from 2:-

276 | 272

294 | 292

Starting from 3:-

381, 383

340, 343, 349 → DP

Starting from 4

404, 406

467, 461, 460 → DP

ال DP اللي بعمله ال memo ال key بتاعها هو pair (digit, depth)

و ال value هو عدد ال combinations اللي ممكن تطلع منها

احنا كدا ماشيين صح. فاضل بس نكتب ال code ونفكر ف (corner case)

@ n=4 ✓✓

Starting from 0

0404 | 0406

0438 | 0434

→ Now let write the Algorithm

Note the coming Algorithm is just make it work algorithm with out applying the D

Algorithm: Knight_Dialer_DP_Solution(int n, int d, int node);

1. graph = build_initial_Graph(); Here call discuss it later

~~2. int res = Knight_Dialer_DP_Solution~~ 2. int res = 0

3. For i = 0 to 9:
   1. res += Knight_Dialer_DP_Solution(n, n-1, i) % mod

4. return res % mod;

---

2. vector<vector<int>> build_initial_Graph() {
   G = {
   {0}: {4, 6}, {1}: {6, 8} — the same Graph we discussed.
   return G;

3.

Knight_Dialer_DP (int d, int node#):

1. if (d == 0)
   1. return 1
2. res = 0
3. for each neighbour for G[node#]
   1. res += Knight_Dialer_DP (d-1, neighbour) % mod
4. return res % mod

Apply memoization

map <pair<int, int>, int> mp.

Kn_DP (d, node):

it = mp.find({d, node})

if (it)

return it -> second

{

mp[{d, node}] = res % mod

return mp[{d, node}];

Memoization

Accepte

first Time

VV.

[5]