flash_7's blog

Digit DP

By **flash_7**, history, 18 months ago,

Wrote this article a long ago but during solving a problem recently thought of sharing this article publicly. Hope it will help some contestants to understand the idea clearly.

Digit dp is a very easy technique and also useful to solve many dynamic programming problems. Seeing the name “Digit DP” it’s easy to guess that we are going to do something using the digits. Yes we are actually going to play with digits. Let’s explain the concept using a classical problem.

Problem

How many numbers **x** are there in the range **a** to **b**, where the digit **d** occurs exactly **k** times in **x**? There may have several solutions including number theory or combinatorics, but let’s see how we can solve this problem using digit dp.

Solve for range (zero to a)

Using digit dp we always focus on building a number satisfying all the conditions. If we finally manage to build that number then we say, yes we have got one ;-) But how we’ll build that number? For the time being let’s say **a** is zero. So we need to find the total numbers which are not greater than **b** and also satisfy the given conditions.

→ Top rated

#	User	Rating
1	tourist	3624
2	Um_nik	3396
3	V--o_o--V	3309

Building a sequence of digits

Let’s consider the number as a sequence of digits. Let’s name the sequence **sq**. Initially **sq** is empty. We’ll try to add new digits from left to right to build the sequence. In each recursive call we’ll place a digit in our current position and will call recursively to add a digit in the next position. But can we place any of the digits from **0** to **9** in our current position? Of course not, because we need to make sure that the number is not getting larger than **b**.

Information we need to place a digit at the current position

Let’s say during the building of the sequence, currently we are at position **pos**. We have already placed some digits in position from **1** to **pos-1**. So now we are trying to place a digit at current position **pos**. If we knew the whole sequence we have build so far till position **pos-1** then we could easily find out which digits we can place now. But how?

You can see that, in the sequence **sq** the left most digit is actually the most significant digit. And the significance get decreased from left to right. So if there exist any position **t** (1<=t<pos) where sq[t] < b[t] then we can place any digit in our current position. Because the sequence has already become smaller than **b** no matter which digit we place in the later positions. Note, b[t] means the digit at position **t** at number **b**.

But if there was no **t** that satisfy that condition then at position **pos**, we can’t place any digit greater than b[pos]. Because then the number will become larger than **b**.

Do we really need the whole sequence?

Now imagine, do we really need that whole sequence to find if a valid **t** exist? If we placed any digit in our previous position which was smaller than its corresponding digit in **b** then couldn’t we just pass the information somehow so that we can use it later? Yes, using an extra parameter **f1**(true/false) in our function we can handle that. Whenever we place a digit at position **t** which is smaller than b[t] we can make **f1** = **1** for the next recursive call. So whenever we are at any position later, we don’t actually need the whole sequence. Using the value of **f1** we can know if the sequence have already become smaller than **b**.

Extra condition

4	Petr	3297
5	wxhtxdy	3293
6	mnbvmar	3255
7	LHiC	3250
8	TLE	3186
9	Vn_nV	3182
10	dotorya	3165

Countries | Cities | Organizations View all →

→ Top contributors

#	User	Contrib.
1	Radewoosh	205
2	Errichto	181
3	neal	159
4	Ashishgup	158
5	PikMike	157
6	Petr	156
7	majk	155
8	rng_58	154
9	Um_nik	153
10	300iq	151

View all →

→ Find user

Handle:

Find

→ Recent actions

So far we focused on building the sequence **sq**, but we have forgotten that there is an extra condition which is, digit **d** will have to occur exactly **k** times in sequence **sq**. We need another parameter **cnt**. **cnt** is basically the number of times we have placed digit **d** so far in our sequence **sq**. Whenever we place digit **d** in our sequence **sq** we just increment **cnt** in our next recursive call.

In the base case when we have built the whole sequence we just need to check if **cnt** is equal to **k**. If it is then we return **1**, otherwise we return **0**.

Final DP States

If we have understood everything so far then it's easy to see that we need total three states for DP memoization. At which position we are, if the number has already become smaller than **b** and the frequency of digit **d** till now.

Solve for range (a to b)

Using the above approach we can find the total valid numbers in the range **0** to **b**. But in the original problem the range was actually **a** to **b**. How to handle that? Well, first we can find the result for range **0** to **b** and then just remove the result for range **0** to **a-1**. Then what we are left off is actually the result from range **a** to **b**.

How to solve for range a to b in a single recursion?

In the above approach we used an extra parameter **f1** which helped us to make sure the sequence is not getting larger than **b**. Can’t we do the similar thing so that the sequence does not become smaller than **a**? Yes of course. For that, we need to maintain an extra parameter **f2** which will say if there exist a position **t** such that sq[t] > a[t]. Depending on the value of **f2** we can select the digits in our current position so that the sequence does not become smaller than **a**. Note: We also have to maintain the condition for **f1** parallely so that the sequence remains valid.

Please check this to find the sample code of our initial approach.

Problem List

1. Investigation

sheaf → [Timus 1766] How to apply Gaussian elimination?

Atreus → COCI 2018/2019 Round 5

rng_58 → Yahoo Programming Contest 2019

kun → The Editorial of the First Codeforces Global Round

athin → Invitation to TOKI Regular Open Contest #5

silent__killer1 → Simple thing turns out to be difficult for me..:(

codefresher → Please help me to solve 769C "Cycle In Maze"

nitesh_gupta → CodeCraft-19 and Codeforces Round #537 (Div. 2) Editorial

Laggy → ICPC World Finals 2019 Team Ratings — What is your prediction?

Ninjo → Solve Algorithmic Problems | new youtube channel

Akikaze → Codeforces Round #538 (Div. 2)

KAN → Codeforces Global Round 1

Hazyknight → A brief tutorial of problem G of Hello 2019.

duckladydinh → How would you configure VIM during a contest?

Wani4ka → B. Bayan

LuckyPaper → How to stop being addicted to cp

prophet_ov_darkness → [GYM] 2018 Battle of Brains Replay

Nickir → 403 error

Arpa → HackerEarth Hourstorm #8

LovesProgramming → Help with modulo function needed .

Vovuh → Codeforces Round #531 (Div. 3) Editorial

Anadi → Codeforces Round #519 by Botan Investments — announcement

2. LIDS
3. Magic Numbers
4. Palindromic Numbers
5. Chef and Digits
6. Maximum Product
7. Cantor
8. Digit Count
9. Logan and DIGIT IMMUNE numbers
10. Sanvi and Magical Numbers
11. Sum of Digits
12. Digit Sum
13. Ra-One Numbers
14. LUCIFER Number
15. 369 Numbers
16. Chef and special numbers
17. Perfect Number
18. The Great Ninja War

Is there some other problems? Some suggestions are really welcomed :)

dynamic programming, **number theory**, **digit dp**

+123 flash_7 18 months ago 58

Comments (58)

Write comment?

minimario

18 months ago, # | ← Rev. 2 +6

I've had this problem for a while, and most probably it's solvable with Digit DP. Maybe it's not, but I haven't found a solution:

Given a integer X, find the number of integers i in $[l, r]$ such that $i \le X$ and $rev(i) \le X$, where rev(i)

ojuz → A blog post for people who want to connect their account with oj.uz

harshit2202 → Another DP Problem

SahilPranjal → Help me problem

Detailed →

Given a integer X, find the number of integers i in $[l, r]$ such that $i \le X$ and $rev(i) \le X$, where rev(i) is the number formed by reversing the digits of i. For example, rev(1560) = 651 and rev(156) = 651 (not 6510, 65100, etc)

Unfortunately I can remember neither the source nor the limits, but probably $X < 10^{18}$ or something.

→ Reply

flash_7

18 months ago, # ^ | ☆ 0

I have found a solution but not fully sure about it.

First of all if X < l then the answer is zero. And i must be <= L where L = min(r, X). Now let’s solve the problem for range (0, L). Let's Len = number of digits in L. Now we can divide the solution in two parts.

In the first part, for each k (1 <= k < Len) we'll try to find the number of valid i which has k digit in it's representation. We can do this easily because if i <= X then rev(i) <= will also hold. As total number of digits is smaller. So we can just ignore that rev(i) <= X condition for this part.

Let's sum1 is the summation of result for each k (1 <= k < Len). Now each of the number x in sum1 will contribute to the final result twice. Because for each x we can add some trailing zeroes in x to make it a Len digit number. As we are adding some trailing zeroes the conditions i <= X and rev(i) <= X will still hold.

Now we’ll solve the second part where we need to find the number of Len digit number which has no leading or trailing zeroes. I'll explain this with an example. Let's L = 372967524. Let's divide this L into 3 parts. P1 = 372, P2 = 9, P3 = 67524. Here P2 has only one digit.

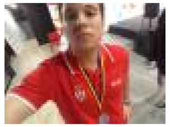
Let's build a number i whose first 3 digit is fixed (same as P1). And at the 4th position we place a smaller digit than P2 (for example 8). So i now looks like this 3 7 2 8 _ _ _ _ _. So i has already become less than X. It doesn't matter what we place in the remaining 5 positions. Condition i <= X will always satisfy. So now we focus on selecting 5 digits for the remaining position such that rev(i) <= X satisfy.

The reverse of the trailing 5 digits of i is actually the 5 leading digits of rev(i). As we don't need to worry about i <= L condition any more, so now we'll do digit DP to find this first 5

leading digits of rev(i), comparing to the first 5 digits in X. We need to handle one more leading digits of rev(i), comparing to the first 5 digits in X. We need to handle one more case here. As we have already placed the leading 4 digits in i whose reverse is actually the trailing 4 digits of rev(i), so after choosing the leading 5 digits in rev(i) we just need to check if the whole rev(i) satisfy the condition (<= X) or not.

As i said above, we'll have to divide L in 3 parts. For each digit in L we'll select it as P2. And the remaining two parts as P1 and P2. Then we repeat the same approach each time. The summation of all these results(including the one in part1) is our final answer.

→ Reply



VladaMG98

18 months ago, # | ☆ ▲ 0 ▼

Check BOI(Baltic) 2013 Numbers, i really liked that one.

→ Reply

sam29

18 months ago, # | ☆ ← Rev. 2 ▲ +11 ▼

I think you can add this problem. I solved it using Digit DP. I really liked this problem.

UPD: Another interesting problem on DIGIT DP https://www.hackerrank.com/contests/morgan-stanley-codeathon-2017/challenges/dreamplay-and-clubbing/problem

→ Reply

flash_7

18 months ago, # ^ | ☆ ▲ 0 ▼

Nice problem. Thank you :)

→ Reply

DebSourav33

3 months ago, # ^ | ☆ ▲ 0 ▼

Beautiful problem! I used digit dp, bitmask, binary search- all in 1 problem!!

→ Reply

18 months ago, # | ☆ ▲ 0 ▼

*Auto comment: topic has been updated by **flash_7** (previous revision, new revision, compare).*

→ Reply

→ Reply

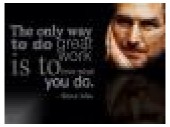
flash_7

18 months ago, # | ☆ 0

Nice explanation. Keep up the good work :) .

→ Reply

Ashik13



17 months ago, # | ☆ 0

One may find this interesting.

→ Reply

BishalG

16 months ago, # ^ | ☆ ← Rev. 2 0

Solved it. A very nice problem. Thanks Vai :)

→ Reply

flash_7

16 months ago, # | ☆ 0

you can also add this

→ Reply

GrV_GamE

16 months ago, # | ☆ 0

How to solve LIDS ?

→ Reply

xxxcxxx

16 months ago, # ^ | ☆ ← Rev. 2 0

The maximum length of LIDS can be at max 10. So we can check for each length k, in how many ways we can make a number whose LIDS is k. We can then print the result we found for the maximum k flash_7

for the maximum k.

I’m describing the states we need to find the number of ways we can build an LIDS of length k.

1. At which position we are.
2. Has the sequence already become less than y? (0 or 1)
3. Has the sequence already become greater than x? (0 or 1)
4. Did we place at least one non zero digit so far? (To handle the first digit of LIDS)
5. Last digit we have placed in the sequence.
6. Number of total digit in the LIDS sequence.

Basically we build two sequence parallelly. One is definitely the whole sequence, another one is the LIDS sequence (Which is the sub sequence of the original sequence). When ever we select a digit we want to place, it becomes the last digit of the original sequence till now. But we can also choose if we want to consider that digit in our LIDS sequence or not.

→ Reply

xxxcxxx

16 months ago, # ^ | ☆ ▲ 0 ▼

Thank you.

→ Reply

sm247

5 months ago, # ^ | ☆ ▲ 0 ▼

Why do need the fourth state ??@**flash_7**

→ Reply

flash_7

5 months ago, # ^ | ☆ ▲ 0 ▼

To make sure we don't place any leading zeroes. Because if we build a number of length k with some leading zeroes then that number is not really a number of length k.

→ Reply

demonsbane

16 months ago, # |

← Rev. 3 +6

Some additional problems:-

Digit Sum

RAONE

LUCIFER

NUMTSN

GONE

Chef and special numbers

→ Reply

flash_7

16 months ago, # |

0

*Auto comment: topic has been updated by **flash_7** (previous revision, new revision, compare).*

→ Reply

joaquimnt_

16 months ago, # ^ |

0

I just wrote a topic about a doubt in one question here: http://codeforces.com/blog/entry/55105

I will read your topic now. Hope that helps me :)

Some questions that i believe that are solvable by Digit DP:

https://www.urionlinejudge.com.br/judge/pt/problems/view/1138
https://www.urionlinejudge.com.br/judge/pt/problems/view/2013
https://www.urionlinejudge.com.br/judge/pt/problems/view/1492

→ Reply

15 months ago, # |

0

How to solve "palindromic numbers"?

Target2018

How to solve "palindromic numbers"?

→ Reply

egor.okhterov

15 months ago, # ^ | ☆ ← Rev. 3 0

Initially we have the space of all numbers $S = \{0, 1, 2, 3, ..., 10^{17}\}$.
This space is too big, so we want to reduce it.
From all that space we only need to consider the numbers with a special property (being palindrome) $P \subset S$.

Now let's consider another space where all of our original numbers are mapped into a different number $S_2 = \{f(0), f(1), f(2), f(3), ..., f(10^{17})\}$.

There is no point in cosidering this new space S_2 instead of original S if it has the same size.
Can you think of **any** mapping $f: S \longrightarrow S_2$ that will make the size of S_2 manageable?

→ Reply

Talk_less

15 months ago, # | ☆ 0

"So if there exist any position t (1<=t<pos) where sq[t] < b[t] then we can place any digit in our current position.

Because the sequence has already become smaller than b no matter which digit we place in the later positions" How is this true? If b = 5321 and current sq = 621, there exists a t such that sq[t] < b[t]. So in the current position pos, putting any digit will cause sq > b? Am I missing something?

→ Reply

SajidZakaria

14 months ago, # | ☆ +5

Vaiya can you please sort the problems according to difficulty?

→ Reply

14 months ago, # ^ | ☆ ← Rev. 4 0

I didn't solve them all but here goes my order

- Investigation

Target2018

- Investigation
- Sum of Digits
- Digit Sum
- Bomb
- Ra-One Numbers
- LUCIFER Number
- G-One Numbers
- Digit Count
- Round Numbers
- Fast Bit Calculations
- Magic Numbers
- Maximum Product

If you need help then you can look at this.

→ Reply

DollarAkshay

12 months ago, # ^ | ☆ -11

How do I solve Maximum Product ?

→ Reply

Target2018

12 months ago, # ^ | ☆ -10

Here's a gift :D

→ Reply

System_test_failed

12 months ago, # | ☆ +5

You can add this problem

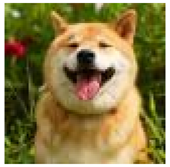
→ Reply

12 months ago, # | ☆ ← Rev. 2 +3

919B - Perfect Number also can be solved by digit DP and it's very amazing if n not greater 10^6

919B - Perfect Number also can be solved by digit DP and it's very amazing if n not greater 10^6

→ Reply



FlyToTheSky

bounce_back20

12 months ago, # | ☆ ▲ 0 ▼

how to solve http://codeforces.com/gym/100886/problem/G ? please tell me the state of the dp

→ Reply

Target2018

12 months ago, # ^ | ☆ ▲ 0 ▼

Take a look

→ Reply

bounce_back20

12 months ago, # ^ | ☆ ▲ 0 ▼

thanks !

→ Reply

bounce_back20

12 months ago, # ^ | ☆ ▲ -10 ▼

i have seen the solution , can you please explain the use of FLAG in dp state , how its helping?

→ Reply

Target2018

12 months ago, # ^ | ☆ ▲ -10 ▼

Check the answer in the blog

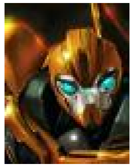
→ Reply

12 months ago, # | ☆ ▲ 0 ▼

Which way is more efficient? Double recursion call with 3 states or single recursion call with 4 states?

→ Reply

→ Reply



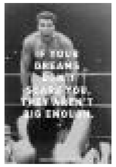
BumbleBee

khokharnikunj8

12 months ago, # | ☆ ▲ 0 ▼

Can Anyone please provide solution for this tutorial also ? I am noob. Sorry for that.

→ Reply



sinnersneversleep

11 months ago, # | ☆ ▲ 0 ▼

I am having really hard time thinking about the time complexity of this approach can anyone please help

→ Reply

ak07

7 months ago, # | ☆ ▲ 0 ▼

Can someone help me with the problem "Cantor". I am getting WA.I can't find my mistake. Link — https://ideone.com/ELiN2b

→ Reply

Target2018

5 months ago, # ^ | ☆ ▲ 0 ▼

Your link is dead here's my solution

→ Reply

ak07

5 months ago, # ^ | ☆ ▲ 0 ▼

Found my mistake! Thank you.

→ Reply

6 months ago, # | ☆ ▲ +3 ▼

what states are used in solving the problem(Cantor)

→ Reply

sm247

Target2018

5 months ago, # ^ | ☆ 0

I solved it using 3 states

→ Reply

flash_7

5 months ago, # | ☆ 0

*Auto comment: topic has been updated by **flash_7** (previous revision, new revision, compare).*

→ Reply

Gr8

5 months ago, # | ☆ +9

Is Classy Numbers also comes in this category?

→ Reply

zoombee

5 months ago, # ^ | ☆ +4

yes

→ Reply

khokharnikunj8

5 months ago, # ^ | ☆ +4

It was the simplest problem of DIGIT DP.

→ Reply

Testduk

5 months ago, # ^ | ☆ 0

could you explain the solution

→ Reply

khokharnikunj8

5 months ago, # ^ | ☆ 0

dp states for Classy numbers I used: index,prev_sum

here prev_sum denotes the number of non zero digits. check out my submission : Solution

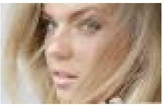
→ Reply

5 months ago, # ^ | ☆ ← Rev. 3 +4

Here is my solution using digit DP 42675923, it may help you. There are quite a lot of cases to take care of.

The main idea is that for each interval `[L, R]` the answer is `F(R) - F(L - 1)` . Where `F(x)` equals the number of *classy* integers from 0 to x.

How to you calculate `F(x)` ? Just using Dynamic Programming.



TheRedLegend

For a number **x**, the DP looks something like this:

`dp[i][j][k][0]` = number of integers **less than x** that are `i` digits long, end in `j` and they have `k` digits bigger than 0.

`dp[i][j][k][1]` = number of integers **equal to x** that are `i` digits long, end in `j` and they have `k` digits bigger than 0.

j is going to be between `0..9` and **k** in the range from `0..3` , because a classy number contains no more than 3 non-zero digits.

My dynamic has more states than the **khokharnikunj8**'s but I think it's more intuitive. Hope it's going to help you.

→ Reply

zoombee

5 months ago, # ^ | ☆ 0

https://www.hackerrank.com/topics/digit-dp this is a good tutorial

→ Reply

adityaraj28

5 months ago, # | ☆ ← Rev. 4 0

I tried to solve the INVESTIGATION Link: https://vjudge.net/problem/LightOJ-1068 problem given above, i wrote the following solution https://code.hackerearth.com/aedc54l I am getting a wrong answer for test case : 1 1000 4 , the answer is given 64 but i am getting 74, could someone please help

→ Reply

Target2018

5 months ago, # ^ | ☆ 0

You can look at this now, I wrote the solution here. Term final is the day after tomorrow, so can't look at your code now.

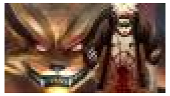
→ Reply

hey_there_delilah

5 months ago, # | ☆ 0

Superb explanation! Thanks.

→ Reply



ay2306

3 months ago, # | ☆ +1

Another really good question on digit dp: Link

→ Reply

prodipdatta7

3 months ago, # | ☆ ← Rev. 2 0

Another good problem in digit dp Balanced Number

→ Reply

CuteCoder07

3 months ago, # | ☆ ← Rev. 3 0

Why the for loop start from 0? let a = 100, d = 5, k = 1

Please correct me if I'm wrong, because one of the base cases is `pos==num.size()` , so there will will be state with, `050` chosen right? this is representing number `50` ? and is that `005`

means it's number `5` ? because we return 1 when `pos == num.size()` so we have to fill

means it's number `5` ?, because we return 1 when `pos == num.size()` so we have to fill each digit, this zero a bit difficult to understand.
→ Reply

risings

3 months ago, # ^ | ☆ ← Rev. 2 ▲ 0 ▼

I too have the same doubt. @flash_7 ,could you please explain this? If we want digit 0,two times then 005,050 are also counted but that shouldn't be the case.
→ Reply

3 months ago, # ^ | ☆ ▲ +1 ▼

Yes, you are correct. 005 means 5 and 050 means 50. So the way i have written my code, it'll consider all the numbers <= N including the ones whose number of digits is less than the number of digits in N. So it'll work perfectly.

flash_7

But when d = 0, my given code will produce an incorrect output. Because it'll count zero for the numbers who have leading zeroes. For example, for the number 005, it'll count zero two times. Which is wrong. Because we don't consider the leading zeroes in the decimal numbers. You have to handle this case separately.

How can we do that? We can use another boolean state, which will keep track if we have placed at least one non zero digits so far. If it's true in any state, then we'll count the zero in that state. Otherwise, we don't. Hope it helps you to understand.
→ Reply

whoami_shubham

3 months ago, # | ☆ ▲ +3 ▼

Thanks bro :) .
→ Reply