flash_7's blog

Digit DP

By **flash_7**, history, 5 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.

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

→ Pay attention

Before contest
Codeforces Round #461 (Div. 2)
5 days

Like 32 people like this. Sign Up to see what your friends like.

→ JacobianDet

Rating: **1267**
Contribution: **0**

- Settings
- Blog
- Teams
- Submissions
- Favourites
- Groups
- Talks
- Contests

JacobianDet

→ Top rated

#	User	Rating
1	tourist	3496
2	Petr	3298
3	OO0OOO00O0OOO0O0...O	3294
4	Um_nik	3248
5	W4yneb0t	3218
6	Syloviaely	3168
7	izrak	3109
8	anta	3106
9	Radewoosh	3084
10	HYPERHYPERHYPERC...R	3071

Countries | Cities | Organizations View all →

→ Top contributors

#	User	Contrib.
1	tourist	184
2	rng_58	173
3	csacademy	166
4	Petr	158
5	Swistakk	155
6	Errichto	149
7	lewin	148
7	matthew99	148
9	Zlobober	141
9	adamant	141

View all →

→ Favourite groups

#	Name
1	ACM-OI

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

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.

Problem List

1. Investigation
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

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

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

+123 flash_7 5 months ago 23

View all →

→ Find user

Handle:

Find

→ Recent actions

BledDest → Educational Codeforces Round 37 [Rated for Div. 2]

Luqman → Teams going to ACM ICPC World Finals 2018

gKseni → Design competition for Codeforces T-shirts

MemorySlices → Invitation to CodeChef February Long Challenge 2018!

okaybody10 → How can I practice effectively?

f2lk6wf90d → Find value of sum of a set of rows in a submatrix

Shaad221B → Default Submission Language

sinhashubham95 → Invitation to February Easy'18 on Hackerearth

flash_7 → Burnside Lemma

mn-9 → average value of a list in python

Nerevar → Codeforces Beta Round #10

AminAnvari → Segment Tree Problems

ArtDitel → AIM Tech Mini Marathon 1

Madiyar → __gcd

moamen_ahmed → codeforces is very slow

jinlifu1999 → Codeforces Round #460 (Div. 2) Editorial

flash_7 → Digit DP

Underdog_eagle → All of the good tutorials found on codeforces

ivan100sic → User editorial for Codeforces Round #135 (Div. 2)

A.GATZ → Geometry problem — bounding polygon with minimal area

arjun95 → how to find total memory consumption of c/c++ program

kardeepak → Suffix Arrays Help Please

hissain.khan → 2D Segment tree — Implementation

PraveenDhinwa → Invitation to ACM-ICPC Amritapuri Onsite Replay Contest on CodeChef

fan_of_stoyan_malinin → http://codeforces.com/contest/13/problem/C

Detailed →

Comments (23)

Write comment?

5 months ago, # | ← Rev. 2 +6

minimario

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 $i \leq X$ and $rev(i) \leq X$, 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

5 months ago, # ^ | 0

flash_7

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.

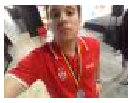
Now we'll solve the second part where we need to find the number of Len digit number which has no leading or trailing zeroes. I'll explain this with an example. Let's L = 372967524. Let's divide this L into 3 parts. P1 = 372, P2 = 9, P3 = 67524. Here P2 has only one digit.

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 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.

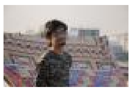
→ Reply


VladaMG98

5 months ago, # | 0

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

→ Reply



5 months ago, # | ← Rev. 2 +11

I think you can add this problem. I solved it using Digit DP. I really liked this sam29

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

5 months ago, # ^ | ☆ 0

Nice problem. Thank you :)
→ Reply

flash_7

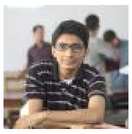
5 months ago, # | ☆ 0

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

Ashik13

5 months ago, # | ☆ 0

Nice explanation. Keep up the good work :) .
→ Reply


BishalG

5 months ago, # | ☆ 0

One may find this interesting.
→ Reply

flash_7

4 months ago, # ^ | ☆ ← Rev. 2 0

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

GrV_GamE

4 months ago, # | ☆ 0

you can also add this
→ Reply

xxxcxxx

4 months ago, # | ☆ 0

How to solve LIDS ?
→ Reply

flash_7

4 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.

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

4 months ago, # ^ | ☆ 0

Thank you.
→ Reply

demonsbane

4 months ago, # | ☆ ← Rev. 3 +6

Some additional problems:-

Digit Sum

RAONE

LUCIFER

NUMTSN

GONE

Chef and special numbers
→ Reply

flash_7

4 months ago, # | ☆ 0

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

joaquimnt_

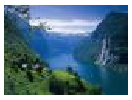
4 months ago, # ^ | ☆ 0

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


Target2018

3 months ago, # | ☆ 0

How to solve "palindromic numbers"?
→ Reply

egor.okhterov

3 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 $f\colon S \longrightarrow S_2$ that will make the size of S_2 manageable?
→ Reply

new, 3 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.

Talk_less

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?

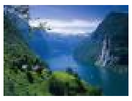
→ Reply

SajidZakaria

new, 2 months ago, # | ☆ +5

Vaiya can you please sort the problems according to difficulty?

→ Reply



Target2018

new, 2 months ago, # ^ | ☆ ← Rev. 2 0

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

- 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

System_test_failed

new, 13 hours ago, # | ☆ 0

You can add this problem

→ Reply

Stakinvario

new, 8 hours ago, # | ☆ ← Rev. 2 0

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

→ Reply