

[Understanding Digit DP]

Me → Hey chat gpt am a Veteran leetcode but
I have forgotten how to do digit dp can
You please teach me.

Chatgpt → Sure. Let's understand first what is the crux of
Digit DP, well digit dp is used for processing
info b/w a range of very large numbers.

Let's say like counting 10^{18} so brute force does not
work for counting and looping over each number
from $(1 \text{ to } 10^{18})$.

We convert the ~~no~~ number into string

$[10^9 \rightarrow "1000000000"]$

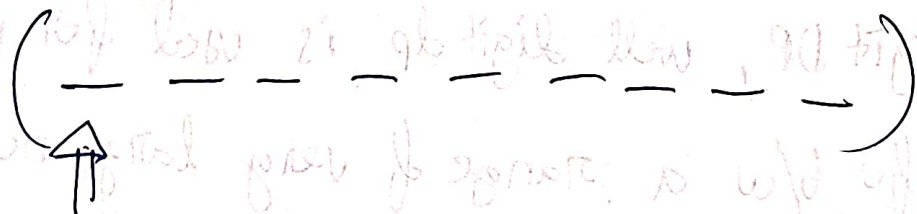
Now on this string evaluations can be done

Why do we need digit dp?

- * Count numbers without repeated digits
- * Count numbers where digits increase/decrease.
- * Count numbers divisible by a number.

State \Rightarrow $[pos, tight, \text{variable depending on the problem}]$

Remember we have converted the number into string right



pos \rightarrow represents the current digit index in the number.

tight \Rightarrow tight is a boolean (0,1).

tight represents whether there ~~was~~ is a restriction on the current pos or not.

Let's say $R = \text{"|1|0000000000"}$

\rightarrow At this position there is a restriction, you can only place "0" or "1",

So there is a restriction here.

209 = restriction to tight because number 209 is greater than 0

Let's say some num = 09 | 209 \rightarrow 1=tight \rightarrow then there's no restriction at this point right?

You can place anything from "0", "1", "2" ---- to "9"

with (tight = 0)

Example Say we are building numbers ≤ 4581

$pos = 0$, We can pick $0, 1, 2, 3$ or 4 (must be ≤ 4)

* if we pick 4 , we are still restricted ($tight = 1$), next (pick ≤ 5)

* if we pick ~~4~~ 3 , we can pick anything next ($tight = 0$).

Q) So when is $tight$ changing?

If we pick maximum allowed digit at a position = pos

and $tight = 1$ at that pos .

Then next pos will have $tight = 1$

~~But if~~

$next_tight = (tight == 1 \ \& \ number_chosen == limit) ? 1 : 0$

Lastly we come to last variable, this

depends on the problem statement

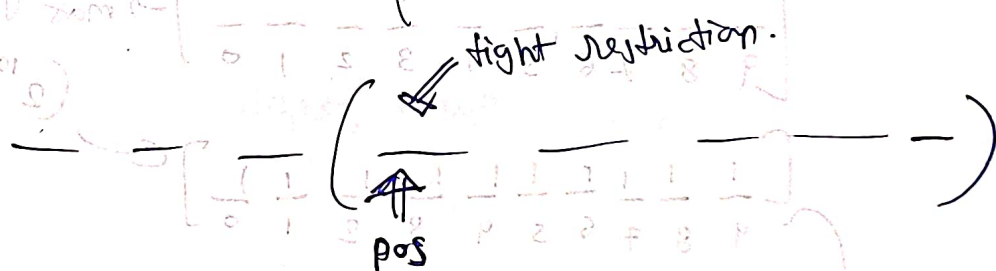
lets say the problem states, count the number of ~~elements~~ nos from (L to R) whose sum of digits is = x .

then you have to maintain function right in which you have to keep track of sum.

String nums = " - - - ";

int func (pos, tight, sum)

↳ this function tells me how many numbers I can form or what number combinations.



if { current position = pos } → precalculated in front. Sum
 { current sum = (sum) } → then how many numbers form their sum as D.
 { tight At pos = tight }

the sum which I will get from [len --- pos] will

be $[D - \text{sum}]$ those many count.

Lets look at a sample problem.

Q) Given two positive integers a and b, return the count of numbers having unique digits in the range $[a, b]$ (inclusive).

We will use digit dp in this.

Lets first find the count for $[1 \text{ to } a]$

pos \rightarrow position

tight \rightarrow restriction on that position

mask \rightarrow it will be mask in our case

$\left[\overline{9} \ \overline{8} \ \overline{7} \ \overline{6} \ \overline{5} \ \overline{4} \ \overline{3} \ \overline{2} \ \overline{1} \ \overline{0} \right] \rightarrow$ max value it will be $(2^{10} - 1)$

$\left[\frac{1}{9} \ \frac{1}{8} \ \frac{1}{7} \ \frac{1}{6} \ \frac{1}{5} \ \frac{1}{4} \ \frac{1}{3} \ \frac{1}{2} \ \frac{1}{1} \ \frac{1}{0} \right]$

mask which tells me what digit has been already picked.

int $f(\text{int pos}, \text{int tight}, \text{int mask}) \{$

if (pos == num.length) $\{$ return 1; $\}$

\Rightarrow we have reached a valid number with unique combo.

P.T.O

```
int limit = tight == 1 ? nums[pos] : 9;
```

```
ans = 0;
```

```
for (i = 0 to limit) {
```

```
    if ((mask & (1 << limit)) == 1) {
```

digit has already been picked

continue;

```
}
```

mark that digit as picked

```
newmask = mask | (1 << limit);
```

```
newtight = (tight == 1 && i == limit) ? 0 :
```

```
ans += f(pos+1, newtight, newmask);
```

```
}
```

```
return ans;
```

```
}
```

Let's look at another sample problem,

for fine tune understanding

Problem: Count Numbers with Sum of Digits Divisible by D.

my attempt:-

In my mind the attempt was like

pos \rightarrow for position

tight \rightarrow for restriction

sum \rightarrow to store sum.

```
int f ( int pos, int tight, int sum ) {
```

```
    if ( pos == num.length ) {
```

```
        return sum % D == 0 ? 1 : 0;
```

```
    }
```

```
    limit = tight == 1 ? num[pos] : 9;
```

```
    for ( i = 0 to limit ) {
```

```
        newtight = ( tight == 1 && i == limit ) ? 1 : 0;
```

```
        ans += f ( pos+1, newtight, sum + i );
```

```
    }
```

```
}
```


to this approach

chatgpt, gave me a space optimisation.

~~sum~~ ~~its instead of storing~~

I was using a cache[pos][tight][sum].

Chatgpt told me ~~at first~~ why not store (sum % mod) instead of the whole sum.

In the beginning my brain was rejecting the thought. Brain was like, why complicate the thought process when it can be already done using the current way of storing.

But its an optimisation. It was a mindset shift. I have noted down everything in my diary →

↓
This event is related to me going to the gym during interview preparation