倒计时(countdown)

- 一开始你有一个非负整数n。
- 每一次操作，你可以从当前的数上减去当前的数某个数位上的数值。当当前数变成0时，倒计时结束。
- 求让倒计时结束的最少操作次数。

倒计时(countdown)

- 15%：n<=10^4
- 25%：n<=10^6
- 直接暴力dp

倒计时(countdown)

- 50%：n<=10^9
- 分段打表or其它

倒计时(countdown)

- 100%：n<=10^18
- 首先我们不难有这样的一个思路，令f[i]表示将i减到0所需最少操作数，然后枚举减去哪一位上的数进行转移，复杂度是O(nlogn)的。
- 要更进一步，我们就得利用更多的性质：对于任意的a>b，有f[a]>=f[b]。
- 如果这个是对的，那我们每次肯定是减去数值最大的那一位。

倒计时(countdown)

- 证明?
- 我们用归纳法，假设在0..n以内这个性质是成立的，要证明在0..n+1以内这个性质仍然成立，那么我们只要证明f[n]<=f[n+1]即可，换而言之，（另g(x)表示x数位上最大的那位的数值）我们要证明的是f(n-g(n))<=f(n+1-g(n+1))，即n-g(n)<=n+1-g(n+1)，那么只要证g(n+1)-g(n)<=1即可，而这是显然的。

倒计时(countdown)

- 然后我们得到了我们的策略，即每次减去最大的数。我们不妨继续dp，用f[i][j]表示前面的数位中最大的一位是i，然后要把j减至不大于0的最少操作数，那么转移只要不断把当前最高非零位消去1即可。考虑有用的状态总数,i只有[0,9],而j要么是n的某段后缀，要么形如10^k-t(0<=t<=9)，所以有用的状态数是不多的，那么我们只要记录有用的状态进行转移即可。

倒计时(countdown)

- 至于具体实现可以使用记忆化搜索或者分2类表示状态进行dp。