桁DP

北海道大学　B3 mikoshiba

まずはウォーミングアップ

自然数Kに対して，K以下の非負整数はいくつありますか？

非負整数Kに対して，K以下の非負整数はいくつありますか？

- $0,1,...K$の$K+1$コあるなあ，とわかります.

非負整数Kに対して，K以下の非負整数はいくつありますか？

視点を変えて，上の桁から見てみましょう

- 例として$K = 2345$を考えてみます.

非負整数Kに対して，K以下の非負整数はいくつありますか？

視点を変えて，上の桁から見てみましょう

- 例として$K = 2345$を考えてみます.

- 1桁目を見てK以下の形をしている整数は$0\square\square\square$, $1\square\square\square$, $2\square\square\square$です.

- この個数は3コ

非負整数Kに対して，K以下の非負整数はいくつありますか？

視点を変えて，上の桁から見てみましょう

- 例として$K = 2345$を考えてみます.

- 2桁目を見てK以下の形をしている整数は$00\square\square, 01\square\square, 02\square\square \ldots 23\square\square$です.

- この個数は24コ

非負整数Kに対して，K以下の非負整数はいくつありますか？

視点を変えて，上の桁から見てみましょう

- 例として$K = 2345$を考えてみます.

- 3桁目を見てK以下の形をしている整数は$000\square, 001\square, 002\square \ldots 234\square$ です.

- この個数は235コ

非負整数Kに対して，K以下の非負整数はいくつありますか？

視点を変えて，上の桁から見てみましょう

- 例として$K = 2345$を考えてみます.

- 4桁目を見てK以下の形をしている整数は$0000, 0001, 0002, \ldots 2345$ です.

- この個数は2346コ

非負整数Kに対して，K以下の非負整数はいくつありますか？

視点を変えて，上の桁から見てみましょう

- 例として$K = 2345$を考えてみます.

- 4桁目を見てK以下の形をしている整数は$0000, 0001, 0002, \ldots 2345$ です.

- この個数は2346コ

このように，いろんな問題に対して「上の桁から見ていこう」という視点を持ってみることにします

問題）自然数K以下の非負整数であり，10進数で表して「1」が含まれる数字はいくつありますか？

制約）$K \leq 10^{18}$

- 例：$K = 11$のとき1,10,11,12の4個あります.

問題）自然数K以下の非負整数であり，10進数で表して「1」が含まれる数字はいくつありますか？

制約）$K \leq 10^{18}$

- 上の桁をi桁見た時にいくつ「1」を含む非負整数があるか？という視点をもってみます

問題）自然数K以下の非負整数であり，10進数で表して「1」が含まれる数字はいくつありますか？

制約）$K \leq 10^{18}$

- 上の桁をi桁見た時にいくつ「1」を含む非負整数があるか？という視点をもってみます

- とりあえず思いつくDP

- $dp[i+1][has] := K$という数の上からi桁目まで見たときに，「1」を含まない/含むものの個数(has = 0 or 1)

問題）自然数K以下の非負整数であり，10進数で表して「1」が含まれる数字はいくつありますか？

制約）$K \leq 10^{18}$

- 上の桁をi桁見た時にいくつ「1」を含む非負整数があるか？という視点をもってみます

- とりあえず思いつくDP

- $dp[i+1][has] := K$という数の上からi桁目まで見たときに，「1」を含まない/含むものの個数(has = 0 or 1)

このDPの遷移を考えてみます.

問題）自然数K以下の非負整数であり，10進数で表して「1」が含まれる数字はいくつありますか？

制約）$K \leq 10^{18}$

- $i-1$桁目まで見て「1」がないとき，i桁目に入れる数字dが「1」かどうかで2つの状態に遷移します

よって　$dp[i][0] \longrightarrow dp[i+1][0 \, || \, d == 1]$

- $i-1$桁目まで見て「1」があるとき，，i桁目に入れる数字dはどれにしても$dp[i+1][1]$に推移します.

よって　$dp[i][1] \longrightarrow dp[i+1][1]$　　($dp[i+1][1 \, || \, d == 1]$と書いても同じ)

問題）自然数K以下の非負整数であり，10進数で表して「1」が含まれる数字はいくつありますか？

制約）$K \leq 10^{18}$

- $i-1$桁目まで見て「1」がないとき，i桁目に入れる数字dが「1」かどうかで2つの状態に遷移します

よって　$dp[i][0] \longrightarrow dp[i+1][0 \,||\, d == 1]$

- $i-1$桁目まで見て「1」があるとき，，i桁目に入れる数字dはどれにしても$dp[i+1][1]$に推移します.

よって　$dp[i][1] \longrightarrow dp[i+1][1]$　　($dp[i+1][1 \,||\, d == 1]$と書いても同じ)

本当に具体的にかけるのでしょうか？

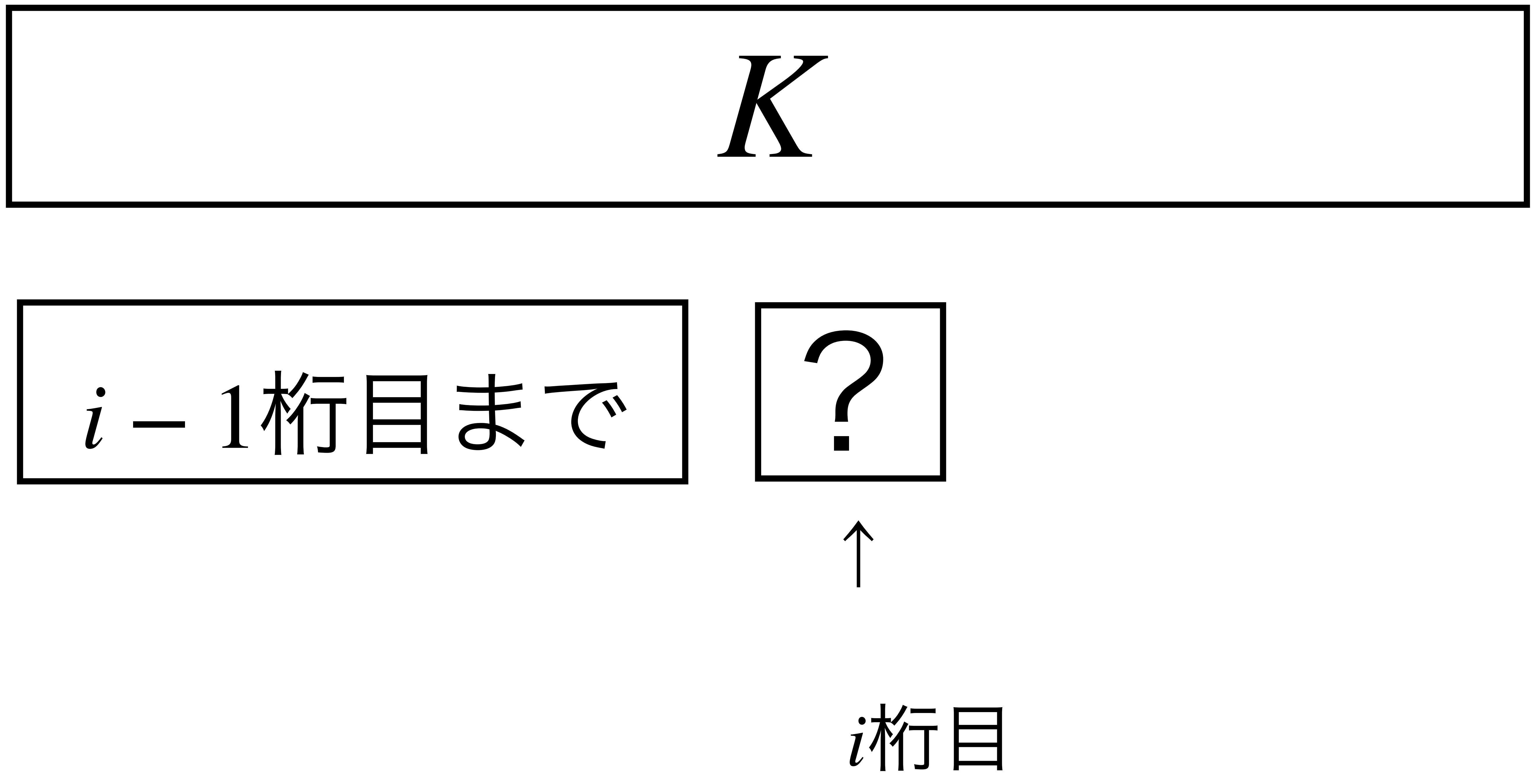
$dp[i][1] \longrightarrow dp[i+1][1]$の遷移は本当に書けるのでしょうか？

$i-1$桁目まで見て「1」があるとき，i桁目にどのような数字をとってでも$dp[i+1][1]$に推移します(?)

$dp[i][1] \longrightarrow dp[i+1][1]$の遷移は本当に書けるのでしょうか？

$i-1$桁目まで見て「1」があるとき，i桁目にどのような数字をとっても$dp[i+1][1]$に推移します(?)

- i桁目にどのような数字dが入れられますか？（0~9?）



例として$K = 2145$で3桁目に入る数字を考えてみます.

- 3桁目にどのような数字dが入れられますか？（0~9?）

2　1　4　5

2桁目まで　？

↑

3桁目

例として$K = 2145$で3桁目に入る数字を考えてみます.

2桁目までが「21」のときを考えてみます

- 3桁目にどのような数字dが入れられますか？（0~9?）

2	1	4	5

2	1	？

↑

3桁目

例として$K = 2145$で3桁目に入る数字を考えてみます.

2桁目までが「21」のときを考えてみます

- 3桁目にどのような数字dが入れられますか？（0~9?）

2	1	4	5

2	1	?

0 ~ 4じゃないとK以下にならないことがわかります.

↑

3桁目

例として$K = 2145$で3桁目に入る数字を考えてみます.

2桁目までが「14」のときを考えてみます

- 3桁目にどのような数字dが入れられますか？（0~9?）

2	1	4	5

1	4	？

↑

3桁目

例として$K = 2145$で3桁目に入る数字を考えてみます.

2桁目までが「14」のときを考えてみます

- 3桁目にどのような数字dが入れられますか？（0~9?）

2　1　4　5

1　4　?

↑

3桁目

0~9どれをいれてもK以下になります！

一般的に考えてみます

- i桁目にどのような数字dが入れられますか？（0~9?）

K

$i-1$桁目まで　？

↑

i桁目

一般的に考えてみます

- i桁目にどのような数字dが入れられますか？（0~9?）

$$\boxed{K}$$

$\boxed{i-1\text{桁目まで}}$ $\boxed{?}$

↑

i桁目

Kと$i-1$桁目まで同じ $\Longrightarrow$ i桁めに0~K_iが入れらます

Kと$i-1$桁目よりもちっちゃい⇒i桁めに0~9が入れられます

補足）「K_i := Kの上からi桁目の数字」　と表現することにします.

一般的に考えてみます

- i桁目にどのような数字dが入れられますか？（0~9?）

$$\boxed{\quad K \quad}$$

$\boxed{i-1\text{桁目まで}}$ $\boxed{?}$

↑

i桁目

Kと$i-1$桁目まで同じ $\Longrightarrow$ i桁めに0~K_iが入れらます

Kと$i-1$桁目よりもちっちゃい⇒i桁めに0~9が入れられます

以上から，Kと$i-1$桁目まで同じかちっちゃいかを

状態として持っておくとうまくいきそうです

問題）自然数K以下の非負整数であり，10進数で表して「1」が含まれる数字はいくつありますか？

制約）$K \leq 10^{18}$

- DPを持ち直してみます.

問題）自然数K以下の非負整数であり，10進数で表して「1」が含まれる数字はいくつありますか？

制約）$K \leq 10^{18}$

- DPを持ち直してみます.

- $dp[i+1][0][has] :=$ i桁目まで見てKと同じで，「1」を含む/含まないK以下のものの数

- $dp[i+1][1][has] :=$ i桁目まで見てKよりちっちゃくて，「1」を含む/含まないK以下のものの数

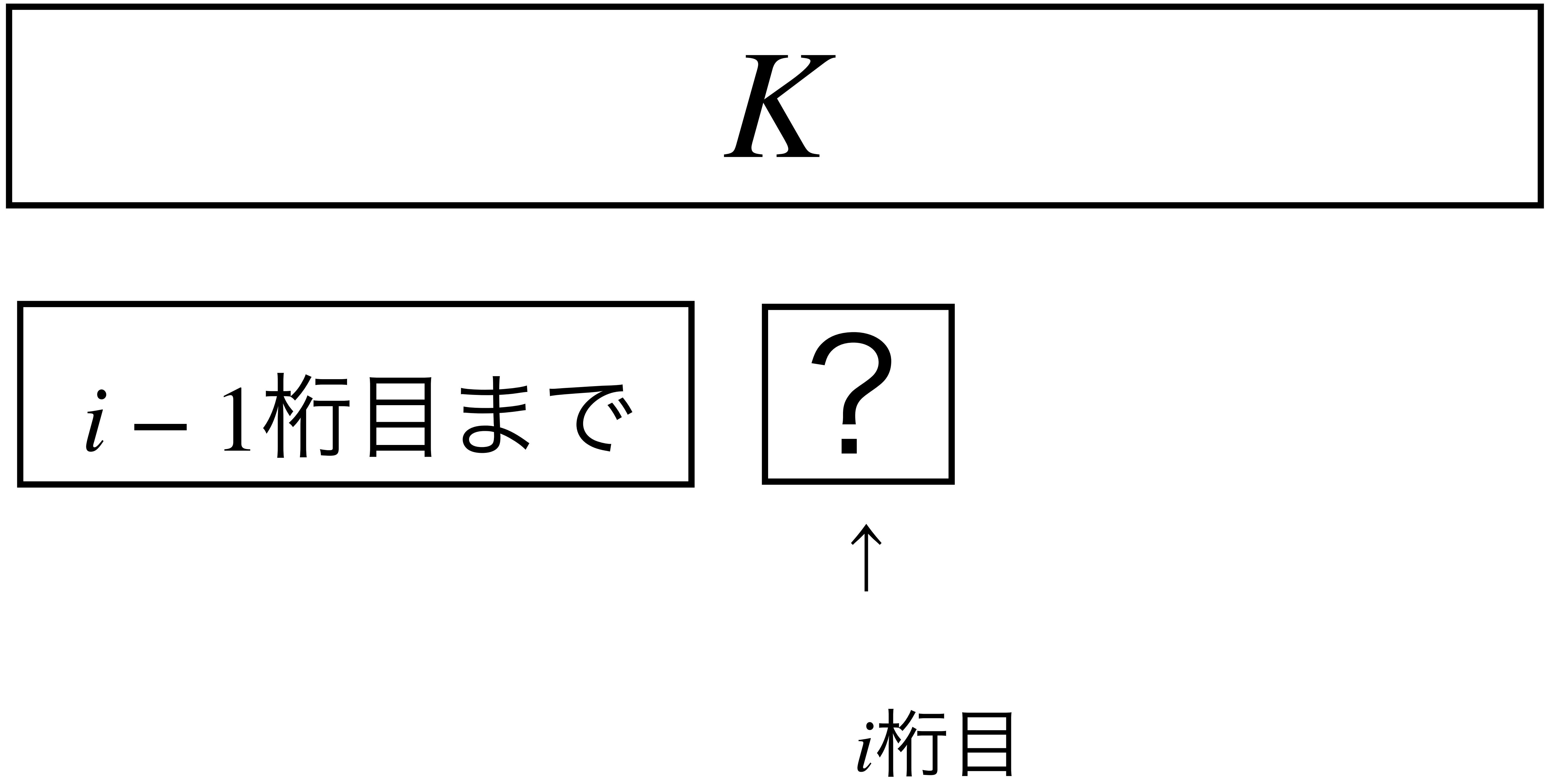
$dp[i][0][has]$からの遷移を考えてみます.

$dp[i][0][has] := i-1$桁目まで見てKと同じで, 「1」を含む/含まないK以下のものの数

$dp[i][0][has]$からの遷移を考えてみます.

$dp[i][0][has]$:= $i-1$桁目まで見てKと同じで，「1」を含む/含まないK以下のものの数

- i桁目にどのような数字dが入れられますか？（0~9?）



$dp[i][0][has]$からの遷移を考えてみます.

$dp[i][0][has] := i-1$桁目まで見てKと同じで，「1」を含む/含まないK以下のものの数

- i桁目にどのような数字dが入れられますか？（0~9?）

K

$i-1$桁目まで

?

↑

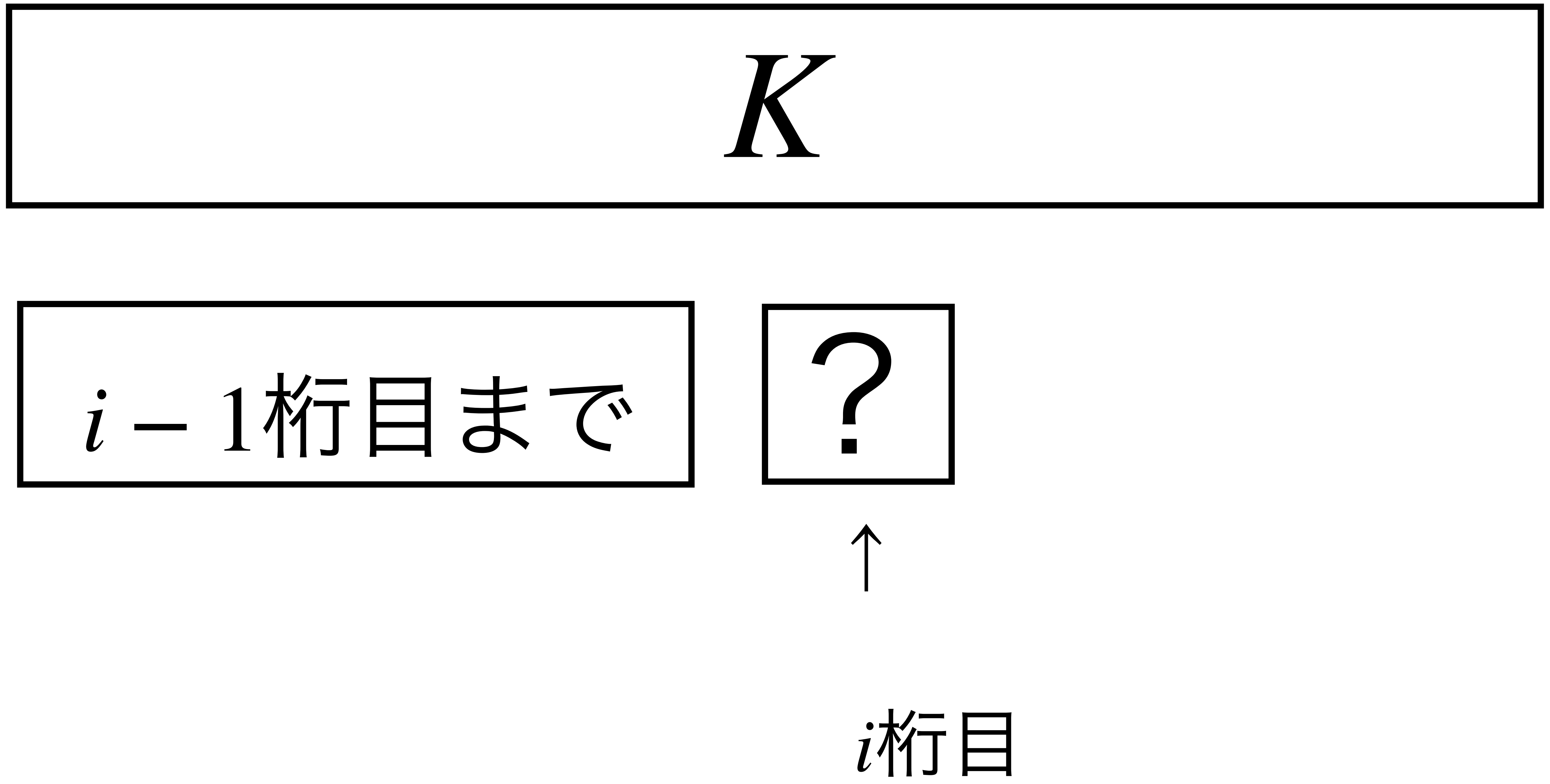
i桁目

$0 \sim K_i$がとれますね！

$dp[i][0][has]$からの遷移を考えてみます.

$dp[i][0][has] := i-1$桁目まで見てKと同じで, 「1」を含む/含まないK以下のものの数

- i桁目にどのような数字dが入れられますか？（0~9?）



i桁目に$d = 0 \sim K[i]-1$を入れるなら

$dp[i+1][1][has \,||\, d == 1]$に遷移

i桁目に$d = K[i]$を入れるときのみ

$dp[i+1][1][has \,||\, d == 1]$に遷移

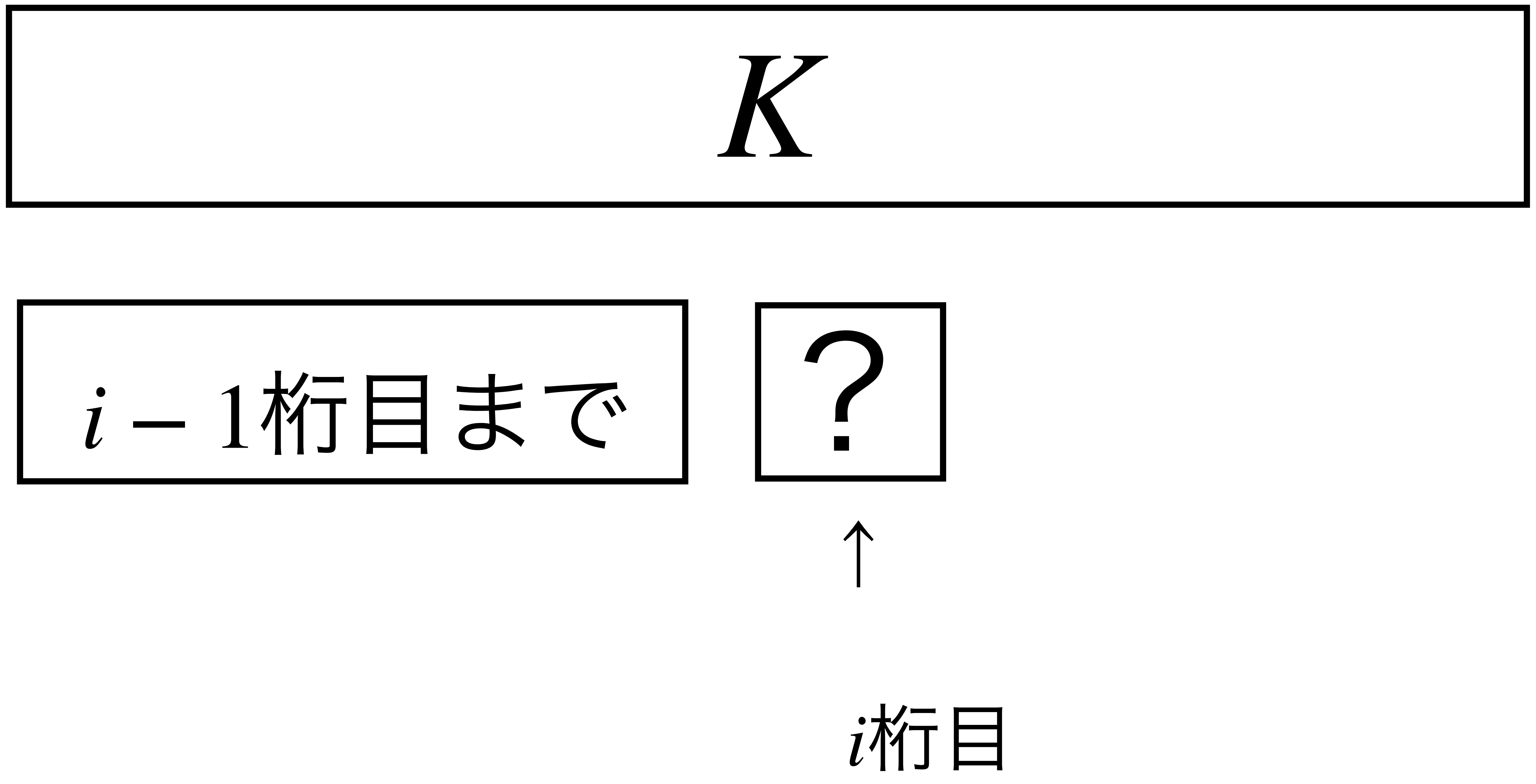
$dp[i][1][has]$からの遷移を考えてみます.

- $dp[i][1][has] := i-1$桁目まで見てKよりちっちゃくて, 「1」を含む/含まないK以下のものの数

$dp[i][1][has]$からの遷移を考えてみます.

- $dp[i][1][has] := i-1$桁目まで見てKよりちっちゃくて，「1」を含む/含まないK以下のものの数

- i桁目にどのような数字dが入れられますか？（0~9?）



$dp[i][1][has]$からの遷移を考えてみます.

- $dp[i][1][has]$:= $i-1$桁目まで見てKよりちっちゃくて, 「1」を含む/含まないK以下のものの数

- i桁目にどのような数字dが入れられますか？（0~9?）

K

$\boxed{i-1\text{桁目まで}}$ $\boxed{?}$

↑

i桁目

$0 \sim 9$どれを入れてもK以下になります！

よってi桁目に$0 \sim 9$どれをいれても

$dp[i+1][1][has \mid\mid d == 1]$に遷移します.

遷移のまとめと時間計算量

制約）$K \leq 10^{18}$

$dp[i][0][has]$

$i-1$桁目まで見てKと同じで,

「1」を含む/含まないK以下のものの数

i桁目にdという数を入れる

$dp[i+1][0][has \;||\; d == 1]$

i桁目まで見てKと同じで,

「1」を含む/含まないK以下のものの数

$dp[i][1][has]$

$i-1$桁目まで見てKよりちっちゃくて,

「1」を含む/含まないK以下のものの数

$dp[i+1][1][has \;||\; d == 1]$

$i-1$桁目まで見てKよりちっちゃくて,

「1」を含む/含まないK以下のものの数

遷移のまとめと時間計算量

制約） $K \leq 10^{18}$

$dp[i][0][has]$ $\xrightarrow{i\text{桁目に}d\text{という数を入れる}}$ $dp[i+1][0][has \mid\mid d == 1]$

$i-1$桁目まで見てKと同じで,

「1」を含む/含まないK以下のものの数

i桁目まで見てKと同じで,

「1」を含む/含まないK以下のものの数

$dp[i][0][has] \rightarrow dp[i+1][1][has \mid\mid d == 1]$

$dp[i][1][has] \rightarrow dp[i+1][1][has \mid\mid d == 1]$

$i-1$桁目まで見てKよりちっちゃくて,

「1」を含む/含まないK以下のものの数

$i-1$桁目まで見てKよりちっちゃくて,

「1」を含む/含まないK以下のものの数

状態の数が最大$log_{10}K \cdot 2 \cdot 2$個で$O(log_{10}K)$, 遷移が最大10個で$O(1)$. よって時間計算量は$O(log_{10}K)$です.

めっちゃ小さいですね！

コード（C++）

配るDPが考えやすいです.

```
#include <iostream>
#include <string>

std::string K;
int N; //Kのケタ数

const int max_N = 20;
long long dp[max_N][2][2]; //dp[i][isSmaller][has]

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';

        int d = D;
        //not smaller <- not smaller
        if(d != 1) dp[i + 1][0][0] += dp[i][0][0];
        if(d == 1) dp[i + 1][0][1] += dp[i][0][0];
        dp[i + 1][0][1] += dp[i][0][1];

        //smaller     <- not smaller
        for(d = 0; d < D; d++){
            if(d != 1) dp[i + 1][1][0] += dp[i][0][0];
            if(d == 1) dp[i + 1][1][1] += dp[i][0][0];
        }
        for(d = 0; d < D; d++) dp[i + 1][1][1] += dp[i][0][1];

        //smaller     <- smaller
        for(d = 0; d < 10; d++){
            if(d != 1) dp[i + 1][1][0] += dp[i][1][0];
            if(d == 1) dp[i + 1][1][1] += dp[i][1][0];
        }
        for(d = 0; d < 10; d++) dp[i + 1][1][1] += dp[i][1][1];
    }

    printf("%lld\n", dp[N][1][1] + dp[N][0][1]);
}
```

$i-1$桁までKとぴったり → i桁までKとぴったり　の遷移

$i-1$桁までKとぴったり → i桁まででKよりちっちゃい

の遷移

$i-1$桁まででKよりちっちゃい → i桁まででKよりちっちゃい

の遷移

ここまでのまとめ

- 桁を上からみてDPをするときは，与えられた数（さっきの問題ではKだった）よりちっちゃいかどうかを管理するとよさそう

コーヒーブレイク

コードを簡潔に

```
#include <iostream>
#include <string>

std::string K;
int N; //Kのケタ数

const int max_N = 20;
long long dp[max_N][2][2]; //dp[i][isSmaller][has]

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';

        int d = D;
        //not smaller <- not smaller
        if(d != 1) dp[i + 1][0][0] += dp[i][0][0];
        if(d == 1) dp[i + 1][0][1] += dp[i][0][0];
        dp[i + 1][0][1] += dp[i][0][1];

        //smaller     <- not smaller
        for(d = 0; d < D; d++){
            if(d != 1) dp[i + 1][1][0] += dp[i][0][0];
            if(d == 1) dp[i + 1][1][1] += dp[i][0][0];
        }
        for(d = 0; d < D; d++) dp[i + 1][1][1] += dp[i][0][1];

        //smaller     <- smaller
        for(d = 0; d < 10; d++){
            if(d != 1) dp[i + 1][1][0] += dp[i][1][0];
            if(d == 1) dp[i + 1][1][1] += dp[i][1][0];
        }
        for(d = 0; d < 10; d++) dp[i + 1][1][1] += dp[i][1][1];
    }

    printf("%lld\n", dp[N][1][1] + dp[N][0][1]);
}
```

```cpp
#include <iostream>
#include <string>

std::string K;
int N; //Kのケタ数

const int max_N = 20;
long long dp[max_N][2][2]; //dp[i][isSmaller][has]

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';

        int d = D;
        //not smaller <- not smaller
        for(int has = 0; has < 2; has++){
            dp[i + 1][0][has || (d == 1)] += dp[i][0][has];
        }

        //smaller     <- not smaller
        for(int has = 0; has < 2; has++){
            for(d = 0; d < D; d++){
                dp[i + 1][1][has || (d == 1)] += dp[i][0][has];
            }
        }

        //smaller     <- smaller
        for(int has = 0; has < 2; has++){
            for(d = 0; d < 10; d++){
                dp[i + 1][1][has || (d == 1)] += dp[i][1][has];
            }
        }
    }

    printf("%lld\n", dp[N][1][1] + dp[N][0][1]);
}
```

コードを簡潔に

$$dp[*][*][has] \longrightarrow dp[*][*][has || (d == 1)]$$

のように条件を管理する書き方はしばしば便利です.

```cpp
#include <iostream>
#include <string>

std::string K;
int N; //Kのケタ数

const int max_N = 20;
long long dp[max_N][2][2]; //dp[i][isSmaller][has]

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';

        //not smaller <- not smaller
        //smaller     <- not smaller
        for(int has = 0; has < 2; has++){
            for(int d = 0; d <= D; d++){
                dp[i + 1][0 || (d < D)][has || (d == 1)] += dp[i][0][has];
            }
        }




        //smaller     <- smaller
        for(int has = 0; has < 2; has++){
            for(int d = 0; d < 10; d++){
                dp[i + 1][1][has || (d == 1)] += dp[i][1][has];
            }
        }
    }

    printf("%lld\n", dp[N][1][1] + dp[N][0][1]);
}
```

コードを簡潔に

$$dp[*][j][*] \rightarrow dp[*][\, j || (d < D)][*]$$

の書き方は使えると便利です.

```cpp
#include <iostream>
#include <string>

std::string K;
int N; //Kのケタ数

const int max_N = 20;
long long dp[max_N][2][2]; //dp[i][isSmaller][has]

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';

        //not smaller <- not smaller
        //smaller     <- not smaller
        //smaller     <- smaller
        for(int isSmall = 0; isSmall < 2; isSmall++){
            for(int has = 0; has < 2; has++){
                for(int d = 0; d < (isSmall ? 10 : D + 1); d++){
                    dp[i + 1][isSmall || (d < D)][has || (d == 1)] += dp[i][isSmall][has];
                }
            }
        }
    }




    printf("%lld\n", dp[N][1][1] + dp[N][0][1]);
}
```

```cpp
#include <iostream>
#include <string>

std::string K;
int N; //Kのケタ数

const int max_N = 20;
long long dp[max_N][2][2]; //dp[i][isSmaller][has]

#define rep(i, n) for(int i = 0; i < (int)(n); i++)

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';


        for(int isSmall = 0; isSmall < 2; isSmall++){
            for(int has = 0; has < 2; has++){
                for(int d = 0; d < (isSmall ? 10 : D + 1); d++){
                    dp[i + 1][isSmall || (d < D)][has || (d == 1)] += dp[i][isSmall][has];
                }
            }
        }
    }









    printf("%lld\n", dp[N][1][1] + dp[N][0][1]);
}
```

```cpp
#include <iostream>
#include <string>

std::string K;
int N; //Kのケタ数

const int max_N = 20;
long long dp[max_N][2][2]; //dp[i][isSmaller][has]

#define rep(i, n) for(int i = 0; i < (int)(n); i++)

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';


        rep(isSmall, 2) rep(has, 2) rep(d, isSmall ? 10 : D + 1){
            dp[i + 1][isSmall || (d < D)][has || (d == 1)] += dp[i][isSmall][has];
        }
    }








    printf("%lld\n", dp[N][1][1] + dp[N][0][1]);
}
```

ここまで簡潔に書くこともできます

コーヒーブレイクおわり

次からは問題を解いてみましょう！

4問あります！

区間$[A, B]$に十進数で書いて「4」か「9」が含まれる数字はいくつありますか？

出典：ABC007 D問題

制約： $1 \leq A \leq B \leq 10^{18}$

- 例）$A = 1, B = 9$　答えは4, 9の2つです.

区間$[A, B]$に十進数で書いて「4」か「9」が含まれる数字はいくつありますか？

制約：$1 \leq A \leq B \leq 10^{18}$

- 上の桁からみていくDPを考えてみます.

区間$[A, B]$に十進数で書いて「4」か「9」が含まれる数字はいくつありますか？

制約：$1 \leq A \leq B \leq 10^{18}$

- $dp[i+1][0][has]$:= i桁めまでみてBと同じで「４」か「９」を含む/含まない数字の個数

- $dp[i+1][1][has]$:= i桁めまでみてBより小さくて「４」か「９」を含む/含まない数字の個数

区間$[A, B]$に十進数で書いて「4」か「9」が含まれる数字はいくつありますか？

制約：$1 \leq A \leq B \leq 10^{18}$

- $dp[i+1][0][has]$:= i桁めまでみてBと同じで「４」か「９」を含む/含まない数字の個数

- $dp[i+1][1][has]$:= i桁めまでみてBより小さくて「４」か「９」を含む/含まない数字の個数

答えはこのときの$dp[N][0][1] + dp[N][1][1]$を$f(B)$として$f(B) - f(A-1)$です

遷移と時間計算量

$dp[i][0][has]$ $\xrightarrow{i\text{桁目に}d\text{という数を入れる}}$ $dp[i+1][0][has \mid\mid d == 4 \mid\mid d == 9]$

$i-1$桁目まで見てBと同じで,

「4」か「9」を含む/含まないB以下の
ものの数

i桁目まで見てBと同じで,

「4」か「9」を含む/含まないB以下の
ものの数

$dp[i][1][has]$ $\longrightarrow$ $dp[i+1][1][has \mid\mid d == 4 \mid\mid d == 9]$

$i-1$桁目まで見てBよりちっちゃくて,

「4」か「9」を含む/含まないB以下の
ものの数

i桁目まで見てBよりちっちゃくて,

「4」か「9」を含む/含まないB以下の
ものの数

時間計算量は$O(log_{10}B)$です

コード（C++）

```cpp
#include <iostream>
#include <string>

long long A, B;

const int max_N = 20;

long long dp[max_N][2][2];

long long f(std::string B){
    for(int i = 0; i < max_N; i++) {
        for(int j = 0; j < 2; j++){
            for(int k = 0; k < 2; k++){
                dp[i][j][k] = 0;
            }
        }
    }

    int N = B.length();

    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = B[i] - '0';
        for(int j = 0; j < 2; j++){
            for(int k = 0; k < 2; k++){
                for(int d = 0; d < (j ? 10 : D + 1); d++){
                    dp[i + 1][j || (d < D)][k || (d == 4) || (d == 9)] += dp[i][j][k];
                }
            }
        }
    }

    return dp[N][0][1] + dp[N][1][1];
}

void solve(){
    std::cout << f(std::to_string(B)) - f(std::to_string(A - 1)) << '\n';
    return;
}
```

1以上自然数K以下の数のうち，十進数で書いて桁の和がMの倍数であるものはいくつありますか？10^9+7で割った余りを求めてください.

制約） $1 \leq K < 10^{10000},\ 1 \leq M \leq 100$

出典：EDPC S問題

1以上自然数K以下の数のうち，十進数で書いて桁の和がMの倍数であるものはいくつありますか？10^9+7で割った余りを求めてください.

制約） $1 \leq K < 10^{10000}, 1 \leq M \leq 100$

- 上の桁からみていくDPを考えてみます.

1以上自然数K以下の数のうち，十進数で書いて桁の和がMの倍数であるものはいくつありますか？10^9+7で割った余りを求めてください.

制約） $1 \leq K < 10^{10000}$, $1 \leq M \leq 100$

- 上の桁からみていくDPを考えてみます.

- $dp[i+1][0][k]$:= i桁までみてKと同じで，桁の和をMで割ってあまりがkとなる

数字の数$(k = 0,1,2,...,9)$

- $dp[i+1][1][k]$:= i桁までみてKより小さくて，桁の和をMで割ってあまりがkとなる

数字の数$(k = 0,1,2,...,9)$

1以上自然数K以下の数のうち，十進数で書いて桁の和がMの倍数であるものはいくつありますか？10^9+7で割った余りを求めてください.

制約） $1 \leq K < 10^{10000}$, $1 \leq M \leq 100$

- 上の桁からみていくDPを考えてみます.

- $dp[i+1][0][k]$:= i桁までみてKと同じで，桁の和をMで割ってあまりがkとなる

数字の数$(k = 0,1,2,...,9)$

- $dp[i+1][1][k]$:= i桁までみてKより小さくて，桁の和をMで割ってあまりがkとなる

数字の数$(k = 0,1,2,...,9)$

答えは$dp[N][0][0]+dp[N][1][0]-1$

遷移と時間計算量

制約） $1 \leq K < 10^{10000}, 1 \leq M \leq 100$

$dp[i][0][k]$ $\xrightarrow{i\text{桁目に}d\text{という数を入れる}}$ $dp[i+1][0][(k+d)\,\%\,M]$

$i-1$桁目まで見てKと同じで,

桁の和をMで割ってあまりがkとなるものの数

i桁目まで見てKと同じで,

桁の和をMで割ってあまりがk'となるものの数

$dp[i][1][k]$ $\longrightarrow$ $dp[i+1][1][(k+d)\,\%\,M]$

$i-1$桁目まで見てKよりちっちゃくて,

桁の和をMで割ってあまりがkとなるものの数

i桁目まで見てKよりちっちゃくて,

桁の和をMで割ってあまりがk'となるものの数

時間計算量は$O(Mlog_{10}K)$です

コード
(C++)

```cpp
#include <iostream>
#include <string>

std::string K;
int N, M;

const int max_N = 10010;

long long dp[max_N][2][110];
const long long mod = 1000000007;

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';
        for(int j = 0; j < 2; j++){
            for(int k = 0; k < M; k++){
                for(int d = 0; d < (j ? 10 : D + 1); d++){
                    (dp[i + 1][j || (d < D)][(k + d) % M] += dp[i][j][k]) %= mod;
                }
            }
        }
    }

    std::cout << (dp[N][0][0] + dp[N][1][0] - 1 + mod) % mod << '\n';

    return;
}
```

時間計算量は

$O(M log_M K)$です

1から自然数Kまですべての整数を十進数で１回ずつ紙に書く書くとき1という数字は何回書かれますか？

出典：ABC029 D問題

制約） $1 \leq K < 10^9$

- 例）$K = 12$ このとき1, 10, 11, 12と5回「1」が出てくるので答えは5です.

1から自然数Kまですべての整数を十進数で１回ずつ紙に書く書くとき1という数字は何回書かれますか？

制約）$1 \leq K < 10^9$

．上の桁からみていくDPを考えてみます．　$N :=$ (Kのケタ数）とします.

1から自然数Kまですべての整数を十進数で１回ずつ紙に書く書くとき1という数字は何回書かれますか？

制約） $1 \leq K < 10^9$

- 上の桁からみていくDPを考えてみます． $N := (K$のケタ数）とします.

- $dp[i+1][0][k] :=$ i桁めをみたときにKの桁めまでと一致していてk個の1がある数字の個数$(k = 0,1,...,N)$

- $dp[i+1][1][k] :=$ i桁めをみたときにKの桁めまでより小さくてk個の1がある数字の個数$(k = 0,1,...,N)$

答えは $\displaystyle\sum_{k=1}^{N} (dp[N][0][k] + dp[N][1][k]) \cdot k$

遷移と時間計算量

制約） $1 \leq K < 10^9$

$dp[i][0][k]$

$i-1$桁目まで見てKと同じで,

k個の1があるものの数

i桁目にdという数を入れる

$dp[i+1][0][k+(d==1)]$

i桁目まで見てKと同じで,

k'個の1があるものの数

$dp[i][1][k]$

$i-1$桁目まで見てKよりちっちゃくて,

k個の1があるものの数

$dp[i+1][1][k+(d==1)]$

i桁目まで見てKよりちっちゃくて,

k'個の1があるものの数

時間計算量は$O(log_{10}^2 K)$です

コード(C++)

```cpp
#include <iostream>
#include <string>

std::string K;
int N;

const int max_N = 20;

long long dp[max_N][2][max_N + 1];

void solve(){
    dp[0][0][0] = 1;

    for(int i = 0; i < N; i++){
        int D = K[i] - '0';
        for(int j = 0; j < 2; j++){
            for(int k = 0; k <= max_N; k++){
                for(int d = 0; d < (j ? 10 : D + 1); d++){
                    dp[i + 1][j || (d < D)][k + (d == 1)] += dp[i][j][k];
                }
            }
        }
    }

    long long ans = 0;
    for(int k = 1; k <= 20; k++) ans += ( dp[N][0][k] + dp[N][1][k] ) * k;

    std::cout << ans << '\n';

    return;
}
```

M個の非負整数$A_1, A_2, \ldots, A_M$および非負整数Kが与えられます.

0以上Kの整数Xに対して, $f(X) := (X \oplus A_1) + (X \oplus A_2) + \ldots + (X \oplus A_M)$

とします.

$f(X)$の最大値を求めてください.

出典：ABC117 D問題

制約： $1 \leq M \leq 10^5, 0 \leq K \leq 10^{12}, 0 \leq A_i \leq 10^{12}$

- 例）$M = 3,\ K = 7,\ A = \{1,\ 6,\ 7\}$ のとき

  $f(4) = (4 \oplus 1) + (4 \oplus 6) + (4 \oplus 3) = 5 + 2 + 7$が最大です.

M個の非負整数$A_1, A_2, \ldots, A_M$および非負整数Kが与えられます.

0以上Kの整数Xに対して, $f(X) := (X \oplus A_1) + (X \oplus A_2) + \ldots + (X \oplus A_M)$

とします.

$f(X)$の最大値を求めてください.

制約： $1 \leq M \leq 10^5, 0 \leq K \leq 10^{12}, 0 \leq A_i \leq 10^{12}$

. 例）$M = 3,\ K = 7,\ A = \{1,\ 6,\ 7\}$ のとき

$f(4) = (4 \oplus 1) + (4 \oplus 6) + (4 \oplus 3) = 5 + 2 + 7$が最大です.

$A_1 = 1_{(10)} = 001_{(2)}$

$A_2 = 6_{(10)} = 110_{(2)}$

$X = \square\square\square_{(2)}$ の形で$X = 0\square\square_{(2)}$ ならfを計算する時に

$A_3 = 3_{(10)} = 011_{(2)}$

$1 \cdot 2^2 = 4$が得られます.

M個の非負整数$A_1, A_2, \ldots, A_M$および非負整数Kが与えられます.

0以上Kの整数Xに対して, $f(X) := (X \oplus A_1) + (X \oplus A_2) + \ldots + (X \oplus A_M)$

とします.

$f(X)$の最大値を求めてください.

制約： $1 \leq M \leq 10^5, 0 \leq K \leq 10^{12}, 0 \leq A_i \leq 10^{12}$

. 例）$M = 3,\ K = 7,\ A = \{1,\ 6,\ 7\}$ のとき

$f(4) = (4 \oplus 1) + (4 \oplus 6) + (4 \oplus 3) = 5 + 2 + 7$が最大です.

$A_1 = 1_{(10)} = 001_{(2)}$

$A_2 = 6_{(10)} = 110_{(2)}$　　$X = \square\square\square_{(2)}$ の形で$X = 1\square\square_{(2)}$ ならfを計算する時に

$A_3 = 3_{(10)} = 011_{(2)}$　　$2 \cdot 2^2 = 8$が得られます.

M個の非負整数$A_1, A_2, \ldots, A_M$および非負整数Kが与えられます.

0以上Kの整数Xに対して, $f(X) := (X \oplus A_1) + (X \oplus A_2) + \ldots + (X \oplus A_M)$

とします.

$f(X)$の最大値を求めてください.

制約： $1 \le M \le 10^5, 0 \le K \le 10^{12}, 0 \le A_i \le 10^{12}$

- $X = \underbrace{\square\square\ldots\square_{(2)}}_{N}$ とXを2進数で表して上の桁から0を入れるか1を入れるかを考えてみます.

M個の非負整数$A_1, A_2, \ldots, A_M$および非負整数Kが与えられます．
0以上Kの整数Xに対して，$f(X) := (X \oplus A_1) + (X \oplus A_2) + \ldots + (X \oplus A_M)$
とします．
$f(X)$の最大値を求めてください．

制約： $1 \le M \le 10^5, 0 \le K \le 10^{12}, 0 \le A_i \le 10^{12}$

- $X = \underbrace{\square\square\ldots\square_{(2)}}_{N}$ とXを2進数で表して上の桁から0を入れるか1を入れるかを考えてみます．
- $dp[i+1][0] :=$ Xの上からi桁目まで考えて0/1を入れた時に，Kと同じで得られる f のうち最大値
- $dp[i+1][1] :=$ Xの上からi桁目まで考えて0/1を入れた時に，Kより小さくてで得られる f のうち最大値

M個の非負整数$A_1, A_2, \ldots, A_M$および非負整数Kが与えられます.

0以上Kの整数Xに対して, $f(X) := (X \oplus A_1) + (X \oplus A_2) + \ldots + (X \oplus A_M)$

とします.

$f(X)$の最大値を求めてください.

制約: $1 \leq M \leq 10^5, 0 \leq K \leq 10^{12}, 0 \leq A_i \leq 10^{12}$

- $X = \underbrace{\square\square\ldots\square_{(2)}}_{N}$ とXを2進数で表して上の桁から0を入れるか1を入れるかを考えてみます.

- $dp[i+1][0] := X$の上からi桁目まで考えて0/1を入れた時に,Kと同じで得られるfのうち最大値

- $dp[i+1][1] := X$の上からi桁目まで考えて0/1を入れた時に,Kより小さくてで得られるfのうち最大値

答えは$max(dp[N][0], dp[N][1])$

$$\boxed{f(X) := (X \oplus A_1) + (X \oplus A_2) + \ldots + (X \oplus A_M)}$$

制約：$1 \le M \le 10^5, 0 \le K \le 10^{12}, 0 \le A_i \le 10^{12}$

- $dp[i][0] \longrightarrow dp[i+1][0]\ or\ dp[i+1][1],\ dp[i][1] \longrightarrow dp[i+1][1]$　と遷移します.

- Xの上からi桁目に0を入れたらいくらの値が得られるでしょうか？

- 各A_jに対して上からi桁目にビットが立っていたら出力 f として$\underbrace{100...0_{(2)}}_{N-i}$の値がプラスされます

$$\boxed{f(X) := (X \oplus A_1) + (X \oplus A_2) + \ldots + (X \oplus A_M)}$$

制約：$1 \le M \le 10^5, 0 \le K \le 10^{12}, 0 \le A_i \le 10^{12}$

- $dp[i][0] \longrightarrow dp[i+1][0]\ or\ dp[i+1][1],\ dp[i][1] \longrightarrow dp[i+1][1]$　と遷移します.
- Xの上からi桁目に1を入れたらいくらの値が得られるでしょうか？
- 各A_jに対して上からi桁目にビットが立っていなかったら出力 f として$\underbrace{100...0_{(2)}}_{N-i}$の値がプラスされます

遷移のまとめと時間計算量

制約： $1 \leq M \leq 10^5, 0 \leq K \leq 10^{12}, 0 \leq A_i \leq 10^{12}$

$dp[i][0] \xrightarrow{X\text{の}i\text{桁目に}d = 0\ or\ 1\text{を入れる}} dp[i+1][0]$

Xの$i-1$桁目まで見てKと同じで,

得られるfのうち最大値

Xのi桁目まで見てKと同じで,

得られるfのうち最大値

$dp[i][0] \longrightarrow dp[i+1][1]$

$dp[i][1] \longrightarrow dp[i+1][1]$

Xの$i-1$桁目まで見てKよりちっちゃくて,

得られるfのうち最大値

Xのi桁目まで見てKよりちっちゃくて,

得られるfのうち最大値

時間計算量は$O(M(log_2K + log_2max(A_i)))$

解答コード（C++）

```cpp
#include <iostream>
#include <algorithm>

int M; long long K;

const int max_M = 100010; long long A[max_M];

const int N = 50; //Kを２進数で表した時の桁数　の最大

long long dp[N + 1][2];

void solve(){
    for(int i = 0; i <= N; i++) for(int j = 0; j < 2; j++) dp[i][j] = -1;
    dp[0][0] = 0LL;                                                                                    //初期条件

    for(int i = 0; i < N; i++){
        long long mask = 1LL << (N - i - 1);                                                       //２の(N - i - 1)乗

        int bit_num = 0;                                                                               //上からi桁目にビットが立っているA_jの個数
        for(int j = 0; j < M; j++) if(A[j] & mask) bit_num++;

        long long value_zero = mask * bit_num;                                                    //Xの上からi桁目を0にした時にプラスされる値
        long long value_one = mask * (M - bit_num);                                             //Xの上からi桁目を1にした時にプラスされる値

        if(dp[i][0] != -1){ /* not smaller <- not smaller */
            if(K & mask) dp[i + 1][0] = std::max(dp[i + 1][0], dp[i][0] + value_one); // K[i]が１のとき　Xの上からi桁目に１を入れざるを得ない
            else         dp[i + 1][0] = std::max(dp[i + 1][0], dp[i][0] + value_zero);        // K[i]が0のとき　Xの上からi桁目に1を入れざるを得ない
        }

        if(dp[i][0] != -1){ /* smaller <- not smaller */
            if(K & mask) dp[i + 1][1] = std::max(dp[i + 1][1], dp[i][0] + value_zero); // K[i]が１のとき　Xの上からi桁目に0を入れるとnot smallerに遷移できる
        }

        if(dp[i][1] != -1){ /* smaller <- smaller */
                         dp[i + 1][1] = std::max(dp[i + 1][1], dp[i][1] + value_zero); // Xの上からi桁目に0, 1どれを入れてもsmallerに遷移する
                         dp[i + 1][1] = std::max(dp[i + 1][1], dp[i][1] + value_one);
        }
    }
    printf("%lld\n", std::max(dp[N][0], dp[N][1]));
}
```

END