

A. Bài 1:

Tìm số hoàn hảo thứ K

Ý tưởng tìm kiếm nhị phân

Tìm kiếm nhị phân từ 1 đến MAX .

Số hoàn hảo thứ K là số tự nhiên nhỏ nhất có ít nhất K số hoàn hảo bé hơn hoặc bằng nó.

Ví dụ: 37 là số thứ 3 vì nó có 3 số hoàn hảo bé hơn hoặc bằng nó là 19, 28, 37.

Khi đó chúng ta có thể sử dụng DP Digit để đếm số lượng số hoàn hảo bé hơn hoặc bằng N .

```
ll dp[i][sum][ok];
```

- i là vị trí hiện tại.
- sum là tổng các chữ số.
- ok là biến kiểm tra xem hiện tại đang bé hơn hay là bằng N .

```
ll L = 1, R = MAX;
while (L <= R) {
    ll mid = (L + R) / 2;
    if (cal(mid) >= K) {
        ans = mid;
        R = mid - 1;
    } else L = mid + 1;
}
```

Hạn chế

MAX bị giới hạn bởi long long. Do đó, nếu kết quả vượt quá long long thì tìm kiếm nhị phân sẽ khó khăn trong việc tính L , R , mid .

Ý tưởng DP cấu hình

Cấu hình cần tạo có dạng a_1, a_2, \dots, a_{300} với $a_i \in \{0, 9\}$ và $a_1 + a_2 + \dots + a_{300} = 10$

Nếu $a_1 = 1$ thì nó lớn hơn mọi cấu hình có $a_1 = 0 \rightarrow dp[2][10]$.

Nếu $a_1 = 2$ thì nó lớn hơn mọi cấu hình có $a_1 = \{0, 1\} \rightarrow dp[2][10] + dp[2][9]$.

Nếu $a_1 = 3$ thì nó lớn hơn mọi cấu hình có $a_1 = \{0, 1, 2\} \rightarrow dp[2][10] + dp[2][9] + dp[2][8]$.

...

Nếu $a_1 = 9$ thì nó lớn hơn mọi cấu hình có $a_1 = \{0, 1, 2, \dots, 8\} \rightarrow dp[2][10] + dp[2][9] + dp[2][8] + \dots + dp[2][1]$.

```
ll dp[i][sum];
```

```
int a[300];
ll K;

for (int i = 0, sum = 10; i < 300; ++i)
    for (int j = 0; ; ++j) {
        if (K > cal(i + 1, sum - j))
            K -= cal(i + 1, sum - j);
        else {
            a[i] = j;
            sum -= j;
            break;
        }
    }
```

Cách tính $dp[i][sum]$

```
ll cal(int i, int sum) {
    if (i == 300) return sum == 0;
    if (dp[i][sum] != -1) return dp[i][sum];
    ll res = 0;
    for (int j = 0; j <= min(9, sum); ++j) {
        res += cal(i + 1, sum - j);
        if (res > 1e18) res = 1e18;
    }
    return dp[i][sum] = res;
}
```

B. Bài 3:

Tìm $A + B + C = N$ thoả mãn A chia hết cho a , B chia hết cho b , C chia hết cho c .

Ý tưởng

Nhắc lại phép cộng tiểu học:

```
 123
+ 456
+ 789
-----
 1368
```

Nếu ta đặt $A = a_1a_2\dots a_9$, $B = b_1b_2\dots b_9$, $C = c_1c_2\dots c_9$ thì ta có:

```
  a1a2\dots a9
+  b1b2\dots b9
+  c1c2\dots c9
-----
  n1n2\dots n9
```

Vì phép cộng sẽ cộng từ hàng đơn vị trở lên, cho nên chúng ta cũng phải xây dựng từ hàng đơn vị trở lên.

Mỗi hàng, chúng ta sẽ thử cả a_i, b_i, c_i từ 0 đến 9 sao cho $a_i + b_i + c_i + nho = n_i$ với nho là số nhớ từ hàng trước.

```
string n;
ll dp[10][3];

ll cal(int i, int carry) {
    if (i == 0) return carry == 0;
    if (dp[i][carry] != -1) return dp[i][carry];
    ll res = 0;
    for (int a = 0; a <= 9; ++a)
        for (int b = 0; b <= 9; ++b)
            for (int c = 0; c <= 9; ++c) {
                int sum = a + b + c + carry;
                if (sum % 10 != n[i - 1] - '0') continue;
                res += cal(i - 1, sum / 10);
            }
    return dp[i][carry] = res;
}

cal(n.size(), 0)
```

Bài toán điển hình cho dạng này là:

```
1?3?5??
+ ??4??8
```

??7??39

Dếm số cách thay thế dấu ? bằng các chữ số từ 0 đến 9 sao cho phép cộng thỏa mãn.

Quay lại bài toán gốc:

- A chia hết cho a
- B chia hết cho b
- C chia hết cho c

Nhiều bạn sẽ nghĩ đến $dp[i][nho][du1][du2][du3]$ với $du1 = A \% a$, $du2 = B \% b$, $du3 = C \% c$.

Độ phức tạp sẽ là $O(10 * 3 * 31 * 31 * 31 * 10^3 * T) \approx 8.10^8 * T$ với T là số test.

Ăn được $T \leq 3$ test.

Với $T \leq 1000$ thì sao?

Phân tích $N = a * x + b * y + c * z$. Khi đó ta sẽ xây dựng x, y, z thay vì A, B, C .

```
x1x2...x9    * a
+ y1y2...y9    * b
+ z1z2...z9    * c
-----
n1n2...n9
```

Chúng ta sẽ không cần số dư cho A, B, C nữa nhưng nhớ sẽ tăng lên.

```
string n;
ll dp[10][100]; // max(carry) = 31 * 9 * 3 / 10 ~ 83

ll cal(int i, int carry) {
    if (i == 0) return carry == 0;
    if (dp[i][carry] != -1) return dp[i][carry];
    ll res = 0;
    for (int x = 0; x <= 9; ++x)
        for (int y = 0; y <= 9; ++y)
            for (int z = 0; z <= 9; ++z) {
                int sum = x * a + y * b + z * c + carry;
                if (sum % 10 != n[i - 1] - '0') continue;
                res += cal(i - 1, sum / 10);
            }
    return dp[i][carry] = res;
}

cal(n.size(), 0)
```

Độ phức tạp sẽ còn $O(10 * 100 * 10^3 * T) \approx 10^6 * T$ với $T \leq 1000$ là được.