C. Classy Numbers

time limit per test: 3 seconds
memory limit per test: 256 megabytes

Let's call some positive integer *classy* if its decimal representation contains no more than 3 non-zero digits. For example, numbers 4, 200000, 10203 are *classy* and numbers 4231, 102306, 7277420000 are not.

You are given a segment $[L; R]$. Count the number of *classy* integers x such that $L \le x \le R$.

Each testcase contains several segments, for each of them you are required to solve the problem separately.

Input
The first line contains a single integer T ($1 \le T \le 10^4$) — the number of segments in a testcase.

Each of the next T lines contains two integers L_i and R_i ($1 \le L_i \le R_i \le 10^{18}$).

Output
Print T lines — the i-th line should contain the number of *classy* integers on a segment $[L_i; R_i]$.

Example

input Copy

```
4
1 1000
1024 1024
65536 65536
999999 1000001
```

output Copy

```
1000
1
0
2
```

→ **Problem tags**

combinatorics dp *1900

No tag edit access

Problem Breakdown

Lets understand the problem first

A number is classy if:

It has at most 3 non-zero digits
(0, 1, 2, or 3 non-zero digits allowed)

Examples:

- 4 → 1 non-zero → classy
- 200000 → 1 non-zero → classy
- 10203 → 3 non-zero → classy
- 4231 → 4 non-zero → NOT classy
- 7277420000 → many non-zero → NOT classy

We need to answer up to T = 10,000 queries, each with:

- L, R (1 ≤ L ≤ R ≤ 1e18)

Goal:

- Count classy numbers in [L, R].

This means we need something fast — brute force is impossible.

Digit DP is perfect.

Digit DP State

We design:

```
dfs(pos, nonZeroCount, tight, leadingZero)
```

Where:

- pos : digit index (0..18)
- nonZeroCount : how many non-zero digits we have placed so far (must be ≤ 3)
- tight : whether we match the prefix of X
- leadingZero : whether number hasn't started yet (still 000... prefix)

Why leadingZero?

Because leading zeros do not count as non-zero digits.

Base Case

If pos == len:

```
return (nonZeroCount <= 3) ? 1 : 0
```

Transition

At pos, digit can be:

```
for d = 0 .. (tight ? digits[pos] : 9):
    nextTight = tight && (d == limit)
    nextLeadingZero = leadingZero && (d == 0)
    nextNonZero = nonZeroCount + (d != 0 ? 1 : 0)
    if nextNonZero <= 3:
        ans += dfs(...)
```

Code

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

vector<int> digits;
ll dp[20][4][2][2];   // pos, nonZeroCount, tight, leadingZero
int len;

ll dfs(int pos, int nz, int tight, int leadingZero) {
    if (pos == len) {
        return (nz <= 3) ? 1 : 0;
    }

    ll &res = dp[pos][nz][tight][leadingZero];
    if (res != -1) return res;

    res = 0;
    int limit = tight ? digits[pos] : 9;

    for (int d = 0; d <= limit; d++) {
        int nextTight = tight && (d == limit);
        int nextLZ = leadingZero && (d == 0);
        int nextNZ = nz + ((d == 0 && nextLZ) ? 0 : (d != 0));

        // simpler: if (leadingZero && d==0) --> nextNZ = nz
        if (leadingZero && d == 0) nextNZ = nz;

        if (nextNZ <= 3) {
            res += dfs(pos + 1, nextNZ, nextTight, nextLZ);
        }
    }
    return res;
}

ll solveUpTo(ll x) {
    if (x < 0) return 0;
    digits.clear();

    if (x == 0) digits.push_back(0);
    while (x > 0) {
        digits.push_back(x % 10);
        x /= 10;
    }
    reverse(digits.begin(), digits.end());
    len = digits.size();

    memset(dp, -1, sizeof(dp));
    return dfs(0, 0, 1, 1);
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int T;
    cin >> T;

    while (T--) {
        ll L, R;
        cin >> L >> R;
        cout << solveUpTo(R) - solveUpTo(L - 1) << "\n";
    }
}
```