数位dp

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<ll, ll> PLL;
const int N = 50, INF = 0x3f3f3f3f, MOD = 10007;
int l, r;
int f[N][12];
void init(){
    //  初始化f数组，通常利用动态规划思想预处理出一些信息
}

int dp(int n){
    int ans = 0, last = 0;  //  ans统计答案，last记录前缀信息
    if(!n) return 1;    //  边界情况
    vector<int> nums;
    while(n) nums.push_back(n % 10), n /= 10;   //  逐位放到数组
    for(int i = nums.size() - 1; i >= 0; i--){
        int x = nums[i];
        //  核心逻辑
        //  1. for(int j = xxx; j < x; j++) 枚举左边的分支
        //  2.  if(xxx) ans += f[i + 1][j] 根据该位置和last的关系，判断是否能取j
        //  3. if(xxx) last = xxx   根据题意更新last/break;
        if(!i) ans++;   //  最右边的分支
    }

    return ans;
}

int main(){
    init();
    cin >> l >> r;
    cout << dp(r) - dp(l - 1) << '\n';
    return 0;
}
```