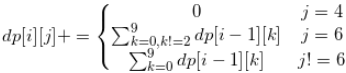
# 数位dp详解及模板

*数位dp一般应用于：*   
****求出在给定区间****[A,B]****内，符合条件****P(i)****的数****i****的个数.条件****P(i)****一般与数的大小无关，而与数的组成有关.****   
比如说在[HDU2089](http://acm.hdu.edu.cn/showproblem.php?pid=2089" \t "https://blog.csdn.net/bestsort/article/details/_blank)中， 让求区间内数中不有4和62的数字个数之和

****对于此类问题，我们一般设****dp****数组****dp[i][j]****，表示i位数，最高位是j的数，不含有**62**和**4**的数有多少个****   
对于上述不含有62和4的要求，递推式如下



换成代码就是：

if(j==4)

dp[i][j] = 0;

else{

for(int k=0;k<=9;k++){

if(j==6&&k==2)

continue;

dp[i][j] += dp[i-1][k];

}

}

到此，我们就能求对于所有能被10整除的满足条件的数的个数了

而对于任意数x,只需要将其拆成x=x1∗100+x2∗101+x3∗102+......的形式即可

然后求出x和y的个数，相减便是答案

#include <map>

#include <queue>

#include <cmath>

#include <cstdio>

#include <string>

#include <cstring>

#include <iostream>

#include <algorithm>

#define IN freopen("in.txt","r",stdin)

#define OUT freopen("out.txt","w",stdout)

#define IO do{\

ios::sync\_with\_stdio(false);\

cin.tie(0);\

cout.tie(0);}while(0)

using namespace std;

typedef long long ll;

const int maxn = 1e4+10;

const int MAXN = 1e6+10;

const int INF = 0x3f3f3f3f;

const int inf = 0x3f;

const double EPS = 1e-7;

const double Pi = acos(-1);

const int MOD = 1e9+7;

int dp[15][15];

int num[15];

void init()

{

dp[0][0] = 1;

for(int i=1; i<=9; i++)

for(int j=0; j<=9; j++)

{

if(j==4)

dp[i][j] = 0;

else

for(int k=0; k<=9; k++)

{

if(j==6&&k==2)

continue;

dp[i][j] += dp[i-1][k];

}

}

}

int ask(int x)

{

ll ans = 0;

int cnt = 0;

while(x)

num[++cnt] = x%10,x/=10;

num[cnt+1] = 0;

for(int i=cnt; i>=1; i--)

{

for(int j=0; j<num[i]; j++)

{

if(j==4||(j==2&&num[i+1]==6))

continue;

ans += dp[i][j];

}

if(num[i] == 4||(num[i]==2&&num[i+1]==6))

break;

}

return ans;

}

int main()

{

IO;

//IN;

int n,m;

init();

while(cin >> m >>n &&(n||m))

cout <<ask(n+1)-ask(m)<<endl;

return 0;

}

[HDU 3555](http://acm.hdu.edu.cn/showproblem.php?pid=3555" \t "https://blog.csdn.net/qq_37957829/article/details/_blank)

#include <bits/stdc++.h>

using namespace std;

typedef long long LL ;

int digit[20];

LL dp[20][2];

LL dfs(int len ,bool if6,bool limit) {

if(len==0) return 1ll;

if(!limit && dp[len][if6]) return dp[len][if6];

int upperBound = (limit?digit[len]:9);

LL sum=0;

for(int i=0;i<=upperBound;i++) {

if((if6 && i==2 )|| i==4) continue;

sum+=dfs(len-1,i==6,limit && i==upperBound);

}

if(!limit) dp[len][if6]=sum;

return sum;

}

LL solve(LL n){

memset(dp,0,sizeof(dp));

int k=0;

while(n){

digit[++k]=n%10;

n/=10;

}

return dfs(k,false,true); ///对于最高位而言，就相当于他的上一位有限制

}

int main()

{

int T;

//scanf("%d",&T);

LL n,m;

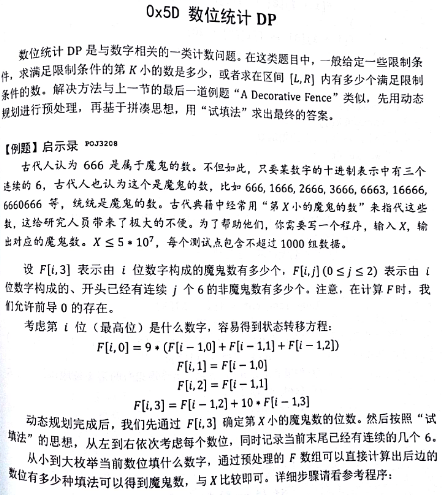
while(scanf("%lld%lld",&n,&m)==2 && !(n==0 && m==0)) {

printf("%lld\n",solve(m)-solve(n-1));

}

return 0;

}



#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

int t,n,m,k;

long long f[21][4];

void work(){

f[0][0]=1;

for(int i=0;i<20;i++){

for(int j=0;j<3;j++){

f[i+1][j+1]+=f[i][j];

f[i+1][0]+=f[i][j]\*9;

}

f[i+1][3]+=f[i][3]\*10;

}

}

int main(){

work();

scanf("%d",&t);

while(t){

t--;

scanf("%d",&n);

for(m=3;f[m][3]<n;m++);

k=0;

for(int i=m;i;i--){

for(int j=0;j<=9;j++){

long long tmp=f[i-1][3];

if(k==3||j==6)

for(int p=max(0,3-k-(j==6));p<3;p++)

tmp+=f[i-1][p];

if(tmp<n) n-=tmp;

else{

if(k<3&&j==6) k++;

if(k<3&&j!=6) k=0;

printf("%d",j);

break;

}

}

}

printf("\n");

}

return 0;

}