# Problem B. New Lottery Game

# 题目

# 注意

# 参考

参考：ACMonster和

<http://stackoverflow.com/questions/23472118/codejam-2014-how-to-solve-task-new-lottery-game>

本地文件：[ACMonster](1-doc/ACMonster_1_1.zip)和[Niklas B.](1-doc/string%20-%20CodeJam%202014%20%20How%20to%20solve%20task%20'New%20Lottery%20Game'%20%20-%20Stack%20Overflow.htm)

# Analysis

## set bits for A,B,K

#define MAX\_SIZE 31

int arr\_a[MAX\_SIZE];

int arr\_b[MAX\_SIZE];

int arr\_k[MAX\_SIZE];

for (int I = 30; I >= 1; --i)

{

arr\_a[i] = A % 2;

A /= 2;

// the same as B and K

}

## dp

Use Dynamic Programming

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 5 | 4 | 3 | 2 | 1 |
| A |  | 1 | 0 | 1 | 0 |
| B |  |  | 1 | 1 | 0 |
| K |  |  | 1 | 0 | 1 |

here show how A[i] changes.

A[4] == 0

A[3] can be 0 or 1

A[3]==0

A[2] can be 0 or 1

A[2]==0

A[1] can be 0 or 1

A[1]==0

A[1]==1

A[2]==1

A[1] can be 0 or 1

A[1]==0

A[1]==1

A[3]==1

A[2] can be 0 or 1

A[2]==0

A[1] can be 0 or 1

A[1]==0

A[1]==1

A[2]==1

A[1] can be 0 or 1

A[1]==0

A[1]==1

A[4] == 1,

A[3] can be 0 only

A[3]==0

A[2] can be 0 or 1

A[2]==0

A[1] can be 0 or 1

A[1]==0

A[1]==1

A[2]==1

A[1] can be 0 only

A[1]==0

int flag [MAX\_SIZE][2][2][2];

long long ans [MAX\_SIZE][2][2][2];

long long

**dp**(int n, bool A\_is\_upper\_bound,

bool B\_is\_upper\_bound,

bool K\_is\_upper\_bound)

{

// exit

if (-1 == n)

{

// Notice!

return !A\_is\_upper\_bound && !B\_is\_upper\_bound && !K\_is\_upper\_bound;

}

int current\_A\_max = arr\_a[n]; // current A[n]

int current\_B\_max = arr\_b[n]; // current B[n]

int current\_K\_max = arr\_k[n]; // current K[n];

if (!A\_is\_upper\_bound)

{

// A[0]—A[n-1] is not the upper boud

// so A[n] can be 0 or 1

current\_A\_max = 1;

}

if (!B\_is\_upper\_bound)

current\_B\_max =1;

if (!K\_is\_upper\_bound)

current\_K\_max =1;

long long res = 0;

for (int I = 0; I <= current\_A\_max; ++i)

{

for (int j = 0; j<= current\_B\_max; ++j)

{

int k = I & j;

if (k <= current\_K\_max)

{

res += dp(n+1,

A\_is\_upper\_bound && (I == current\_A\_max),

B\_is\_upper\_bound && (j == current\_B\_max),

K\_is\_upper\_bound && (k == current\_K\_max)

);

}

}

}

flag[MAX\_SIZE][A\_is\_upper\_bound][B\_is\_upper\_bound][K\_is\_upperbound] = true;

ans[MAX\_SIZE] [A\_is\_upper\_bound][B\_is\_upper\_bound][K\_is\_upperbound] = res;

return res;

}

int res = dp(1, true, true, true);

# 解法1

## 参考

## 耗时

## 分析

## 源码

# 解法2

## 参考

## 耗时

## 分析

## 源码

# extend

## A,B,K

I from 0-A including 0 and A

j from 0-B including 0 and B

I & j < K

please count the number