Q:- ADJACENT BIT COUNT

$0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1$

$0+1+1+0+0+1+0+0 = 3$

$X_1 * X_2 + X_2 * X_3 \text{ ------- } X_{n-1} * X_n$

$\boxed{n,\ k} \leftarrow$ input

2 d

$\boxed{n \le 100}$

$\boxed{t \le 10^3} \rightarrow$ testcases

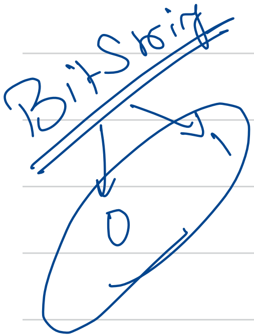
Return how many bit string of length n you can make such that the adj BC is K.

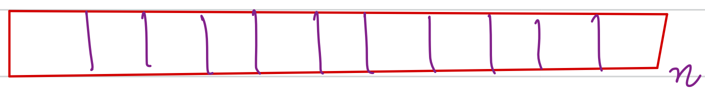
$n = 5$, $K = 2$ $\Big\} \rightarrow 6$

11100 10111
01110 11101
00111 11011

6 string of length n=5 while adj (BC = K (2))

→ Counting Problem → Count the total no. of strings of length n, and $\underbrace{\text{AdjBitCount}}$ $= k$ → bits at adjacent indexes





zeroth index

$\boxed{x_i * x_{i+1}}$ → 1

$x_i == 1$

$x_{i+1} == 1$

$0 \times 1 = 0$

$0 \times 0 = 0$

$1 \times 0 = 0$

Case 1

fixed with value zero

no adj BC found

zeroth index

first index

Case 2

zeroth index

choose → 1 → K−1 adj BC in n−1 length string

choose → 0 → K adj BC in n−1 len string

→ that if zeroth char is 0, and we were finding all string with adj BC = K & length n, now I need to find all strings with adj BC = K & length = n-1

$f(n, k, 0) = f(n-1, k, 0) + f(n-1, k, 1)$

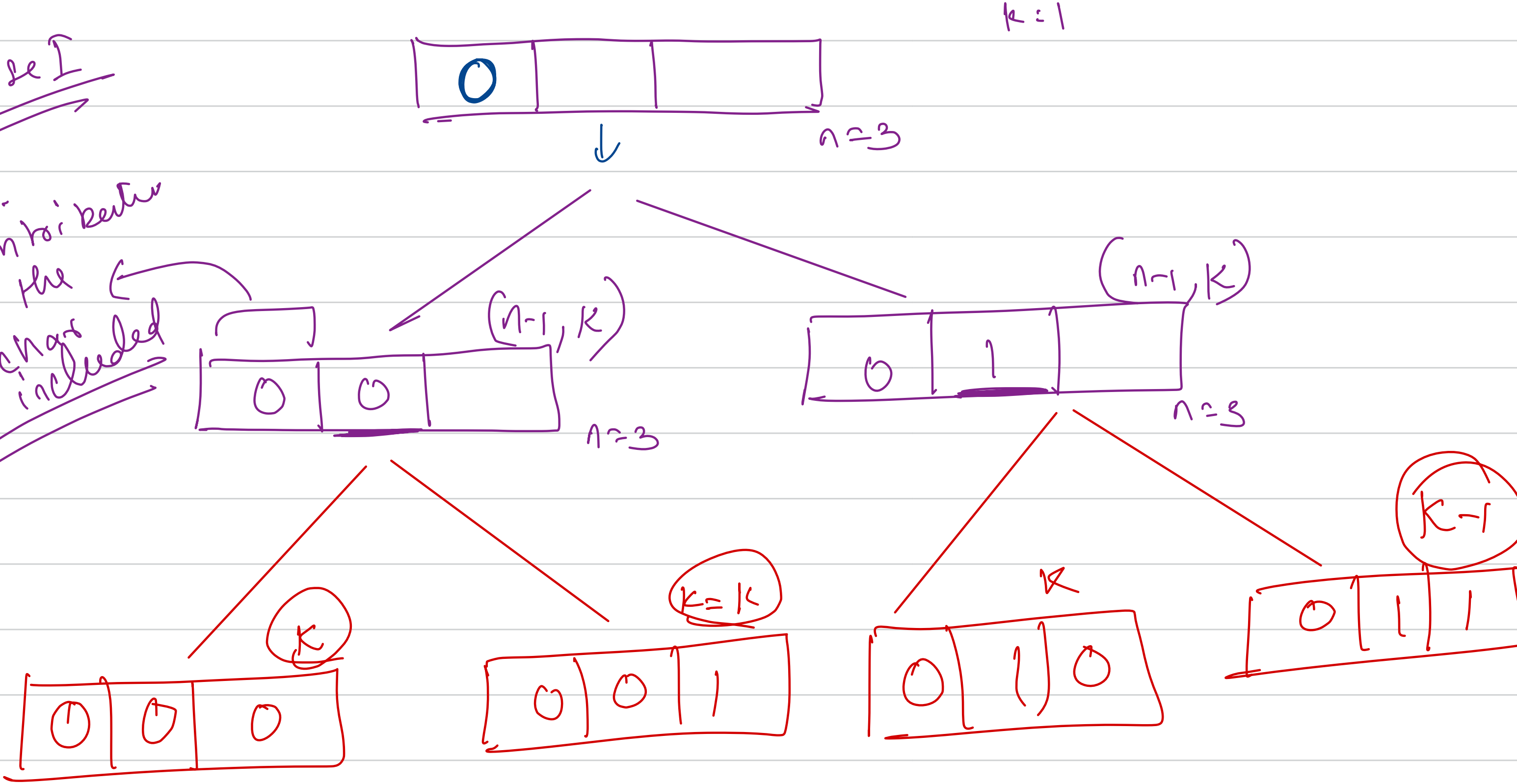
func that returns the no. of strings with adj BC = k, length n and starting from 0.

(assume this works fine)

$K = 1$ $\quad n = 3$

Case I

Contribution of the 0th char is included



$$f(n, k, 1) = f(n-1, k-1, 1) + f(n-1, k, 0)$$

func^n returns count of all strings of len = n and adj BC = k starting with 1

$f(n, k, first)$

$n \rightarrow$ len of string

$k \rightarrow$ adj BC reqd

first $\rightarrow$ the zeroth char of n length string which we call

first

optimal → most favourable

→ at max 3D array

$$\underbrace{f(n, k, first)}_{\text{Bigger problem}} = \begin{cases} \underbrace{f(n-1, K, 1)} + \underbrace{f(n-1, k, 0)} & \text{if } first == 0 \\ \text{Smaller Subproblems} \\ \overbrace{f(n-1, K-1, 1)} + \overbrace{f(n-1, K, 0)} & \text{if } first == 1 \end{cases}$$

recurrence relation

How many parameters are reqd to uniquely identify a subproblem ?? ⟹ 3 → max dimension of space reqd if we have previously computed any subproblem, and we encounter it again, then how about save it somewhere.

i) overlapping subproblem

$$f(n, k, 1)$$

$$f(n-1, k-1, 1) \qquad f(n-1, k, 0)$$

$$f(n-2, k-2, 1) \quad f(n-2, k-1, 0) \quad f(n-2, k, 1) \quad f(n-2, k, 0)$$

$$f(n-3, k-3, 1) \quad f(n-3, k-2, 0) \quad \boxed{f(n-3, k-1, 1)} \quad f(n-3, k-1, 0) \quad \boxed{f(n-3, k-1, 1)} \quad f(n-3, k, 0)$$

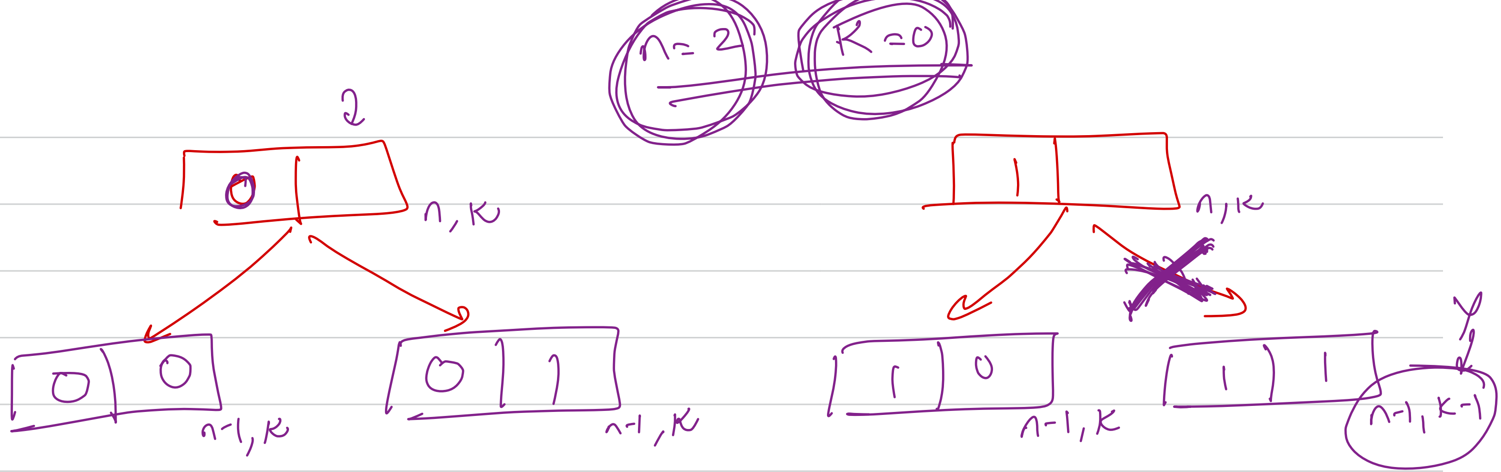
2) Optimal Substructure => If we have optimal ans for a smaller subproblem and they also contribute optimally to generate ans of bigger problem then it shows optimal Substructure

Store common / repeating / overlapping subproblem.

array → dimensions

$$f(n) = f(n-1) + f(n-2)$$

at max → 1D array



n = 2
K = 0
2
0
n, K
1
n, K
0 0
n-1, K
0 1
n-1, K
1 0
n-1, K
1 1
n-1, K-1
y

$n = 1$ | $K = 0$

one length string

$\boxed{0}$ $K=0$ → 0

$\boxed{1}$ $K=0$ → 0

one length string → $K = 0$ → $1 + 0$

$\boxed{0} \to 0$

$\boxed{1} \to 0$

1D, 2D, 3D, Probability, Expectation, Symmetric tree, Binary search, Combinatorics, Convex hulls etc.

Optimize Bitmasking, Tree dp

Q:1 Count the no. of binary strings of length n, such that they don't have any consecutive

Ones.

$n=3$

ans → 5

000
001
010
100
101

$n=1 \to 2$
$n=2 \to 3$
$n=3 \to 5$
$n=4 \to 8$

Fibonacci

$n=2 \to$ 00, 10, 01

↓

3

$n=1 \to$ 0, 1

↓

2

$n=4 \to$ 0000, 0001, 0010, 0100, 1000, 1001, 1010, 0101

↓

8

$f(n) = f(n-1) + f(n-2)$

@isanketsingh