Epiphany 10.0
ACM-NIT, Surat

Editorial

A. Three piles of coin

- It can be solved by making three equations.
- Suppose we are performing x operations by taking 2 coins from the 1st pile, y operations by taking 2 coins from 2nd pile and z operations by taking 2 coins from the 3rd pile.
- So we can make three equations:
    - $2*x+y+z = A$
    - $x+2*y+z = B$
    - $x+y+2*z = C$
- By solving these 3 equations we can find 4 conditions and using them we can check whether we can reach the final state or not.
    - $(A+B+C)\%4 = 0$
    - $3*A >= B+C$
    - $3*B >= A+C$
    - $3*C >= A+B$
- Time Complexity = O(t)

B. Schrute's Potion

- Let’s break this question down to an easier one:
- Let array B be the difference array of length n-1 where B[i] = A[i+1] -A[i], so if we can make all the elements of A equal after some operations then we should be able to make all elements of B equal to 0 after some operations.

Now lets see what will happen to array B after one operation on Index i, i+1, i+2 on array A,

A[i] = A[i] +1 then B[i-1] = A[i] - A[i-1] = A[i] - A[i-1] +1 = B[i-1] + 1.
A[i+1] = A[i+1] +1 then B[i] = A[i+1] - A[i] = B[i].
A[i+2] = A[i+2] +1 then B[i+1] = A[i+2] - A[i+1] = B[i+1].
A[i+3] = A[i+3] then B[i+2] = A[i+3] - A[i+2] - 1 = A[i+3] - A[i+2] -1 = B[i+2] - 1.

- So we can see in one operation we can change B[i-1]=B[i-1]+1 and B[i+2] = B[i+2] - 1.

- So the new question is whether it is possible to make all elements of B equal to 0 after performing a certain number of such (B[i] = B[i] + 1, B[i+3] = B[i+3] - 1) operations.

- So let’s run a loop from left to right and while running this loop we will try to make B[i] = 0 by performing the operation on A at index i+1,i+2,i+3 abs(B[i]) times, that will make B[i]=0 and B[i+3] = B[i+3] + B[i].
- Note that we can only increment B[i] so if B[i] is greater than 0 we can’t make it 0 by this operation, the only way to decrement positive B[i] to 0 is to increment all elements of A till index i by B[i] and that is only possible if i+1 (total number of elements till i) is divisible by 3 and if not the answer is “No”.

- After traversing the array the only possible non zero elements will be the last 3 elements of B. Now we have to check if we can make these 0 or not and the only way to decrement or increment the last 3 elements is if to check they have total elements on the left or right of index i in A is divisible by 3 respectively.

C. The Imitation Game

- First of all, we observe that the Enigma Matrix is a Symmetric matrix. Now we want to find the element at the position (i,j) for that let us assume that $i <= j \; and \; X = 1$, the element at position (i,j) will be $\binom{i+j}{i}$.

- Since it is a symmetric matrix then we can also apply this in the case when $i > j$ by replacing i with j.

- So, the answer will be $X * \binom{i+j}{min(i,j)}$.

and don't forget to mod this value.

Proof:

- Let us try to prove why this works, for that let us start with

$${}^{n}C_0 + {}^{n+1}C_1 + {}^{n+2}C_2 + \dots + {}^{n+r}C_r$$

Given expression

$$\left({}^{n+1}C_0 + {}^{n+1}C_1\right) + {}^{n+2}C_2 + {}^{n+3}C_3 + \ldots\ldots + {}^{n+r}C_r$$

$$\left[\because {}^{n}C_0 = {}^{n+1}C_0\right]$$

$$= \left({}^{n+2}C_1 + {}^{n+2}C_2\right) + {}^{n+3}C_3 + \ldots\ldots + {}^{n+r}C_r$$

$$= \left({}^{n+3}C_2 + {}^{n+3}C_3\right) + \ldots\ldots + {}^{n+r}C_r$$

$$= \left({}^{n+4}C_3 + {}^{n+4}C_4\right) + \ldots\ldots + {}^{n+r}C_r$$

$$= {}^{n+r}C_{r-1} + {}^{n+r}C_r = {}^{n+r+1}C_r$$

- So, how is this helpful, if we closely look at this by putting $n + r + 1 = i + j$ and $r = i$ (as we assume $i <= j$).
We find that this expression is the same as the enigma formula definition.

D. D-Controllable Numbers

- This problem can be solved using digit dp.

- Suppose, count of D-controllable numbers between L and R = g(0, R)-g(0, L), where g(0, N) counts D-controllable numbers between 0 and N.

- Now let's count g(0, N).

  - Simple idea is to traverse from 0 to 9 for all digits and check for every possibility and count the number.

- Also, this number must be less than the given N and digit D must not be present at even positions and also take care of leading zeros.

- We can make this type of recurrence relation for this.

- DP states:-
    - n -> remaining digits to be filled
    - Tight -> to not exceed the limit(N)
    - Leading -> checking for leading zeros
    - Check -> 1 if it is D-controllable till now
    - Pos -> 1 if we are filling odd position else 0

- Dp(n,tight,leading,check,pos) =
  $\sum_{i=0}^{ub}$ dp(n-1 , tight&(i == ub) , leading (i == 0) ,   check & (((leading & (i == 0)) == 0 &&& pos == 0 && i == d) ? 0 : 1) , ((leading & (i == 0)) == 0) ? (pos^1) : pos)

  Here ‘ub’ is defined such that the number is not exceeding N.

- Time Complexity: O(t*(log10(r)+1)*9*16)

E. Alice and Bob

Let's represent Head as 1 and Tail as 0, then flipping of coins can be represented as (current_state+1)%2.

Basic DP/ brute force solution will be

```
vector<int>dp(n,1);//initial array A filled by 1
for(int p=0;p<k;p++)
{
    // copy aux array in new array to preserve value of aux array
    vector<int>newdp=AUX;
    for(int i=0;i<n;i++){
        for(int j=0;j<=i;j++)
            //flipping i coins starting from index i
            newdp[(i+j)%n] = (newdp[(i+j)%n] + dp[i])%2;
    }
    dp = newdp;
}
```

At end If count of ones is greater than count of zeros then BOB will win.Check this for all 4 cases of AUX array.

Now we can use "Matrix exponentiation" to fasten this DP. There are 2*N states required to build a matrix.(N states representing DP array , another N states representing AUX array).

In this method (2N)^3 operation will be required for Matrix Multiplication.We will do these operations logK times.
So overall time complexity will be (2N)^3 * logK.

F. Santa's GIFT

Let's solve two cases.

First case is when graph is the bipartite graph

Then the sum of weights of the left part should be equal to the sum of weights of the right part (because each edge will bring an equal contribution to the sums of both part).

We will leave any spanning tree of this graph, then for it the solution is unique (take the edges entering the leafs, their weights are uniquely determined, subtract weights from weights of the second ends of this edges, delete the leafs, recursively) this solution can be found with dfs, then in the end, the root will has weight 0 (because sum of weights of the left part equal to the sum of weights of the right part) Thus, the answer exists when the sum of the weights of the left part is equal to the sum of the weights of the right part.

Second case when graph has the odd cycle.

We find an odd cycle, root the tree for any of its vertices, solve the tree. Then, we add to the weights of the edges of the cycle adjacent to the root minus its weight divided by 2 (it is even, because it is the sum of the weights of all vertices (with different signs) equal to the sum of the degrees of vertices by modulo 2). , and for all others we alternate the signs with which we add this value, then for all vertices except the root the sum does not change, but for the root we get the required value.