Programación Competitiva

Competencia #2

SEMANA i

acm International Collegiate Programming Contest

Problema #1: Divisibility by 8

- Criterio de divisibilidad de 8, es que los últimos tres dígitos son divisibles por 8.
- Elige 3 dígitos tales que sean divisibles entre 8.
- **$O(n^3)$**

Problema #2 Unusual sequences

- Verificar si Y es divisible entre X, en caso de ser divisible dividir Y entre X.
- El problema se reescribe a dado un número de cuántas maneras distintas se puede sumar a ese número usando al menos un 1, ya que eso producira el GCD correcto.

- Planteamos un dp[N][2] donde N indica que la suma de todos los terminos tiene que llegar a N y la otra dimension indica si ya se uso un 1 o no.
- El dp seria $s_{i,1} = \Sigma^{i-2}_{j=1} s_{j,1} + s_{i-1,0}$ y $s_{i,0} = \Sigma^{i-1}_{j=1} s_{j,0}$ con $s_{0,1}=0$ y $s_{0,0}=1$
- Notar que $s_{i,0}$ se puede convertir a 2^{i-1}.
- Dado eso $s_{i,1}$ se convierte a 2^{i-1} - fib(i-1), donde fib(i) = fib(i-1)+fib(i-2) y fib(0)=0 y fib(1)=1.
- Se puede obtener el N-esimo termino de fibonacci con matrix exponentiation.

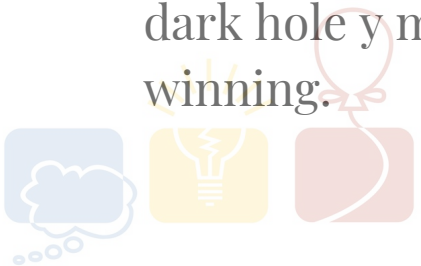
Problema #3 Jon and Orbs

- Hacer un dp con la probabilidad de encontrar exactamente X Orbs diferentes en tiempo n, entonces queda n*k
- Notar que las queries piden menos de 50%, entonces N no debería de ser muy grande.

- Pre calcular el dp, y responder los queries usando una búsqueda binaria

Problema #4 Berzerk

- Marcar cada nodo con un counter de la cantidad de transiciones que pueden llegar al nodo. Dos valores por nodo, uno de Rick y uno de Morty
- Hacer un bfs del dark hole a todos los nodos de los cuales Rick y Morty pueden llegar al dark hole y marcarlos como winning.



- Para cada nodo marcado como winning, visitar todos los nodos que lo pueden visitar y reducir el counter en 1 (Para Rick y para Morty.)
- Cuando el contador llegue a 0 meter a la queue del bfs y ponerle valor losing.
- De cada losing marcar como winning los nodos que llegan a ese estado y meter a la queue.
- Si el nodo nunca se visitó en el bfs es empate.

Problema #5 Functions again

- Pre Calcular las sumas de las diferencias alternando entre positivo y negativo y llamar al arreglo S.
- Tener dos variables: una para las posiciones pares y otra para las impares. y guardar el mínimo y máximo visitado hasta el momento, respectivamente.

- **La respuesta está dada por:**
- **El máximo entre S[i] - even y odd - S[i] para cada i.**

Problema #6 Reberland Linguistics

- Haces un DP[N][2] donde la N representa la posición en el string y la segunda dimensión indica si el último suffix fue de tamaño 2 o 3.
- La transicion es a N-2 y N-3 si el nuevo sufijo es diferente al previo.

- **Mantener un mapa donde se guardan todos los sufijos.**
- **La respuesta es la cantidad de sufijos en el mapa.**
- **Considerar qué te detienes cuando N es menor a 4.**

Problema #7 Kyoya and Colored Balls

- Para cumplir la condición de sacarlas en orden sacas primero todas las del color 1, luego las del 2, etc.
- Para cada color que se saca, la combinatoria está dada por la cantidad de pelotas que se han sacado y la cantidad de pelotas que se van a sacar - 1. (Dejas el color actual hasta el final para mantener el orden)

- **La combinatoria está dada por $(S + C_i - 1)! / (S! * (C_i - 1)!)$.**
- **S es la suma de todos los counts de los colores que ya se han sacado.**
- **Considerar módulos.**

Problema #8 Sereja and Suffixes

- Iterar sobre el arreglo al reverso y usar un map para contar la cantidad de numeros diferentes desde n hasta j.
- Guardar las respuestas y responder las queries en O(1).