Proyecto final de la asignatura Diseño y Análisis de Algoritmos

Autor :
Amanda Noris Hernández

4to Año Ciencias de la Computación
Universidad de La Habana

September 23, 2024

1 Bicolorings Problem

Se le proporciona una cuadrícula que consta de 2 filas y n columnas. Cada celda de esta cuadrícula debe estar coloreada en blanco o negro.

Dos celdas se consideran vecinas si tienen un borde común y comparten el mismo color. Dos celdas A y B pertenecen al mismo componente si son vecinas, o si hay un vecino de A que pertenece al mismo componente que B.

Llamemos hermosa a un bicoloración si tiene exactamente k componentes.

Cuente el número de bicoloraciones hermosas. El número puede ser lo suficientemente grande, así que imprima el módulo de respuesta 998244353.

Input La única línea contiene dos números enteros n y k ($1 \leq n \leq 1000, 1 \leq k \leq 2n$): el número de columnas de una cuadrícula y el número de componentes necesarios.

Output Imprime un solo número entero: el número de bicoloraciones hermosas módulo 998244353.

2 Solución con fuerza bruta

Para resolver este problema mediante fuerza bruta, se pueden generar todas las posibles configuraciones de colores en la cuadrícula de 2 filas y n columnas. Dado que cada celda puede estar coloreada de blanco o negro, el número total de configuraciones posibles es 2^{2n}, ya que hay dos filas y n columnas, y cada celda tiene 2 opciones de color.

Una vez que se han generado todas las configuraciones posibles, para cada configuración se puede contar el número de componentes conectados que tiene la cuadrícula. Después, se filtran aquellas configuraciones que tienen exactamente k componentes.

Pasos:

1. Generar todas las configuraciones posibles: Cada celda de la cuadrícula puede ser negra o blanca. En total hay 2^{2n} configuraciones posibles.
2. Contar el número de componentes conectados en cada configuración.
3. Filtrar las configuraciones que tengan exactamente k componentes.
4. Imprimir el número total de configuraciones que cumplan la condición, módulo 998244353.

2.1 Complejidad temporal

La complejidad temporal es aproximadamente $O(2^{2n} \times n)$, ya que hay 2^{2n} configuraciones posibles y, para cada configuración, se cuenta el número de componentes en tiempo $O(n)$ usando una búsqueda en profundidad.

3 Solución con programación dinámica

El objetivo es contar cuántas formas hay de colorear una cuadrícula de 2 filas y n columnas de tal manera que haya exactamente k componentes conectados, donde dos celdas pertenecen al mismo componente si comparten un borde común y están coloreadas con el mismo color.

Este problema tiene subestructura óptima porque la cantidad de configuraciones para una cuadrícula de tamaño n depende de las configuraciones de cuadrículas de tamaño menor. Al establecer relaciones de recurrencia que solo dependen de los resultados anteriores, podemos utilizar programación dinámica para resolverlo de manera eficiente.

Utilizaremos 2 ciclos for para llevar a cabo las transiciones. El primer for itera sobre el número de columnas a construir, desde 2 hasta n, ya que la primera columna ya está inicializada antes de este bucle. Ahora, queremos construir las configuraciones para las columnas adicionales.Para ello el segundo for itera sobre el número de componentes posibles, desde 1 hasta $2*i$ porque en una cuadrícula de 2 filas y i columnas, el número máximo de componentes es $2 * i$ (en el peor de los casos, cada celda podría ser un componente separado). Dentro del segundo bucle, se calculan las transiciones de las configuraciones anteriores a la configuración actual, almacenando el resultado en la tabla de DP.

Definición del estado dinámico

El enfoque dinámico se basa en la definición del estado $dp[i][j][x]$, donde:

- i es el número de columnas procesadas hasta el momento (de 1 a n).
- j es el número de componentes ya formados hasta la columna i.
- x es una máscara que indica el color de la última columna (0 para WW, 1 para WB, 2 para BW, 3 para BB).

Inicialización de la DP

Los casos con los que se inicia nuestra dinámica, y que cubren todas las posibilidades, son:

Caso 1 Hay 1 forma de tener una configuración con la primera columna siendo WW (blanca en ambas filas) y teniendo exactamente 1 componente.

$$dp[1, 1, 0] = 1$$

Caso 2: Hay 1 forma de tener una configuración con la primera columna siendo WB (blanca en la fila superior y negra en la inferior) y teniendo 2 componentes. Aquí, WB se considera un nuevo componente.

$$dp[1, 2, 1] = 1$$

Caso 3: Hay 1 forma de tener una configuración con la primera columna siendo BW (negra en la fila superior y blanca en la inferior) y teniendo 2 componentes. Este también crea un nuevo componente.

$$dp[1, 2, 2] = 1$$

Caso 4: Hay 1 forma de tener una configuración con la primera columna siendo BB (negra en ambas filas) y teniendo exactamente 1 componente.

$$dp[1, 1, 3] = 1$$

Transiciones de estado

Cada una de las transiciones está diseñada para calcular cuántas configuraciones pueden llevar a una determinada configuración de la columna i. De esta manera cubrimos todos los posibles casos y nos aseguramos de que la solución sea correcta.

Transición para WW ($dp[i, j, 0]$): WW no crea un nuevo componente. Puede venir de:

- WW (mismo componente, j se mantiene)
- WB (mismo componente, j se mantiene)
- BW (mismo componente, j se mantiene)
- BB (nuevo componente, $j-1$)

Transición para WB ($dp[i,j,1]$): WB crea un nuevo componente. Puede venir de:

- WW (nuevo componente, $j-1$)
- WB (mismo componente, j)
- BW (nuevos componentes, $j-2$)
- BB (nuevo componente, $j-1$)

Transición para BW ($dp[i,j,2]$): BW también crea un nuevo componente. Puede venir de:

- WW (nuevo componente, $j-1$)
- WB (nuevos componentes, $j-2$)
- BW (mismo componente, j)
- BB (nuevo componente, $j-1$)

Transición para BB ($dp[i,j,3]$): BB no crea un nuevo componente. Puede venir de:

- WW (nuevo componente, $j-1$)
- WB (mismo componente, j)
- BW (mismo componente, j)
- BB (mismo componente, j)

Resultado

El resultado de ejecutar el algoritmo nos dará que la cantidad de bicolores hermosos que se pueden formar con k componentes y n columnas y determinada coloración en la última columna estará almacenado en $dp[n, k, x]$ para cada uno de los valores de x. Por tanto la respuesta al problema sería la suma de todas estas posibles configuraciones, o sea:

$$dp[n, k, 0] + dp[n, k, 1] + dp[n, k, 2] + dp[n, k, 3]$$

3.1 Demostración de correctitud de la solución

Para la primera columna ($i = 1$), el algoritmo inicializa correctamente el estado dinámico para todas las máscaras posibles según lo explicado anteriormente.

Supongamos que para una columna i el estado $dp[i][j][x]$ es correcto y contiene el número de maneras de colorear las primeras i columnas con exactamente j componentes, dada la configuración x en la última columna. Al procesar la columna $i + 1$, el número de componentes nuevos se actualiza correctamente en la transición de estados ya que asegura que todos los casos posibles están cubiertos. Por lo tanto, si el estado es correcto para i, también lo será para $i + 1$.

3.2 Complejidad temporal

La complejidad temporal del algoritmo se puede calcular de la siguiente manera:

El bucle externo se ejecuta $O(n)$.

El bucle interno, para cada iteración del externo, se ejecuta hasta $O(n)$ en el peor de los casos (hasta $2n$).

Dentro del bucle interno, las operaciones son $O(1)$.

Por lo tanto, la complejidad total se puede expresar como:

$$O(n \cdot n) = O(n^2)$$

4 Implementación

Tanto la implementación del algoritmo de fuerza bruta como el de programación dinamica se pueden entontrar en los archivos adjuntos. Los casos de prueba con los que se comprobó la correctitud de este código son los proporcionados por la plataforma Codeforces.