Multiplicación de matrices simétricas

La traspuesta de una matriz $A_{n\times n}$ es definida como una matriz $A^T_{n\times n}$ tal que $A^T[i,j] = A[j,i]$ para cada $i,j \in \{1,\ldots,n\}$

Además, una matriz A es simétrica si $A = A^T$

Multiplicación de matrices simétricas

La traspuesta de una matriz $A_{n\times n}$ es definida como una matriz $A^T_{n\times n}$ tal que $A^T[i,j] = A[j,i]$ para cada $i,j \in \{1,\ldots,n\}$

Además, una matriz A es simétrica si $A = A^T$

Como en las matrices simétricas hay elementos repetidos, es natural preguntar si la multiplicación de dos matrices simétricas puede hacerse más rápido.

Multiplicación de matrices simétricas

La traspuesta de una matriz $A_{n\times n}$ es definida como una matriz $A^T_{n\times n}$ tal que $A^T[i,j] = A[j,i]$ para cada $i,j \in \{1,\ldots,n\}$

Además, una matriz A es simétrica si $A = A^T$

Como en las matrices simétricas hay elementos repetidos, es natural preguntar si la multiplicación de dos matrices simétricas puede hacerse más rápido.

- ¿Es el resultado de la multiplicación de dos matrices simétricas una matriz simétrica?

Multiplicación de matrices simétricas

¿Es posible tener un algoritmo para multiplicar matrices simétricas que realice $f(n)$ sumas y multiplicaciones de números enteros, donde $f(n) \in o(n^2)$?

Multiplicación de matrices simétricas

¿Es posible tener un algoritmo para multiplicar matrices simétricas que realice $f(n)$ sumas y multiplicaciones de números enteros, donde $f(n) \in o(n^2)$?

Usando la técnica de la mejor estrategia del adversario es posible dar una respuesta negativa a esta pregunta.

- La demostración sigue la misma idea que para el caso general de matrices cuadradas

Multiplicación de matrices simétricas

¿Es posible tener un algoritmo para multiplicar matrices simétricas que realice $f(n)$ sumas y multiplicaciones de números enteros, donde $f(n) \in o(n^2)$?

Usando la técnica de la mejor estrategia del adversario es posible dar una respuesta negativa a esta pregunta.

- La demostración sigue la misma idea que para el caso general de matrices cuadradas

Pero en este caso es posible utilizar la técnica de reducción para dar de manera más simple una respuesta negativa a la pregunta.

Una cota inferior para la multiplicación de matrices simétricas

Vamos a reducir el problema de multiplicar matrices cuadradas al problema de multiplicar matrices simétricas.

Una cota inferior para la multiplicación de matrices simétricas

Vamos a reducir el problema de multiplicar matrices cuadradas al problema de multiplicar matrices simétricas.

Sean $A_{n\times n}$ y $B_{n\times n}$ dos matrices cuadradas (no necesariamente simétricas)

Una cota inferior para la multiplicación de matrices simétricas

Vamos a reducir el problema de multiplicar matrices cuadradas al problema de multiplicar matrices simétricas.

Sean $A_{n\times n}$ y $B_{n\times n}$ dos matrices cuadradas (no necesariamente simétricas)

Vamos a mostrar cómo calcular $A \cdot B$ utilizando un algoritmo arbitrario $\mathcal{A}$ para la multiplicación de matrices simétricas.

Una cota inferior para la multiplicación de matrices simétricas

Defina las matrices $C_{2 \cdot n \times 2 \cdot n}$ y $D_{2 \cdot n \times 2 \cdot n}$ de la siguiente forma:

$$C = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 0 & B^T \\ B & 0 \end{pmatrix}$$

donde 0 es una matriz de $n \times n$ que contiene el valor 0 en cada posición.

Una cota inferior para la multiplicación de matrices simétricas

Defina las matrices $C_{2 \cdot n \times 2 \cdot n}$ y $D_{2 \cdot n \times 2 \cdot n}$ de la siguiente forma:

$$C = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 0 & B^T \\ B & 0 \end{pmatrix}$$

donde 0 es una matriz de $n \times n$ que contiene el valor 0 en cada posición.

Las matrices C y D son simétricas.

Una cota inferior para la multiplicación de matrices simétricas

Tenemos que:

$$C \cdot D = \begin{pmatrix} A \cdot B & 0 \\ 0 & A^T \cdot B^T \end{pmatrix}$$

Por lo tanto a partir de $C \cdot D$ podemos obtener $A \cdot B$

Una cota inferior para la multiplicación de matrices simétricas

Tenemos que:

$$C \cdot D = \begin{pmatrix} A \cdot B & 0 \\ 0 & A^T \cdot B^T \end{pmatrix}$$

Por lo tanto a partir de $C \cdot D$ podemos obtener $A \cdot B$

Recuerde que la multiplicación de dos matrices cuadradas de $n \times n$ requiere de al menos n^2 sumas y multiplicaciones de enteros.

Una cota inferior para la multiplicación de matrices simétricas

Tenemos que:

$$C \cdot D = \begin{pmatrix} A \cdot B & 0 \\ 0 & A^T \cdot B^T \end{pmatrix}$$

Por lo tanto a partir de $C \cdot D$ podemos obtener $A \cdot B$

Recuerde que la multiplicación de dos matrices cuadradas de $n \times n$ requiere de al menos n^2 sumas y multiplicaciones de enteros.

Entonces, dado que el cálculo de la reducción no utiliza sumas ni multiplicaciones de números enteros, obtenemos que $n^2 \leq t_{\mathcal{A}}(2 \cdot n)$

- Puesto que C y D son matrices de $2 \cdot n \times 2 \cdot n$

Una cota inferior para la multiplicación de matrices simétricas

Si suponemos que $t_{\mathcal{A}}(n) \in o(n^2)$, entonces:

$$(\forall c \in \mathbb{R}_0^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(n) \leq c \cdot n^2)$$

Una cota inferior para la multiplicación de matrices simétricas

Si suponemos que $t_{\mathcal{A}}(n) \in o(n^2)$, entonces:

$$(\forall c \in \mathbb{R}_0^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(n) \leq c \cdot n^2)$$

Tenemos entonces que:

$$(\forall c \in \mathbb{R}_0^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(2 \cdot n) \leq 4 \cdot c \cdot n^2)$$

Una cota inferior para la multiplicación de matrices simétricas

Si suponemos que $t_{\mathcal{A}}(n) \in o(n^2)$, entonces:

$$(\forall c \in \mathbb{R}_0^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(n) \leq c \cdot n^2)$$

Tenemos entonces que:

$$(\forall c \in \mathbb{R}_0^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(2 \cdot n) \leq 4 \cdot c \cdot n^2)$$

En particular si consideramos $c = \frac{1}{8}$ obtenemos que:

$$(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(2 \cdot n) \leq \frac{1}{2} \cdot n^2)$$

Una cota inferior para la multiplicación de matrices simétricas

Si suponemos que $t_{\mathcal{A}}(n) \in o(n^2)$, entonces:

$$(\forall c \in \mathbb{R}_0^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(n) \leq c \cdot n^2)$$

Tenemos entonces que:

$$(\forall c \in \mathbb{R}_0^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(2 \cdot n) \leq 4 \cdot c \cdot n^2)$$

En particular si consideramos $c = \frac{1}{8}$ obtenemos que:

$$(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(t_{\mathcal{A}}(2 \cdot n) \leq \frac{1}{2} \cdot n^2)$$

Como tenemos que $n^2 \leq t_{\mathcal{A}}(2 \cdot n)$ concluimos que:

$$(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(n^2 \leq \frac{1}{2} \cdot n^2)$$

Lo cual nos lleva a una contradicción. □

Programación dinámica: un primer ingrediente

Al igual que dividir para conquistar, la técnica de programación dinámica resuelve un problema dividiéndolo en sub-problemas más pequeños.

Programación dinámica: un primer ingrediente

Al igual que dividir para conquistar, la técnica de programación dinámica resuelve un problema dividiéndolo en sub-problemas más pequeños.

Pero a diferencia de dividir para conquistar, en este caso se espera que los sub-problemas estén traslapados.

- De esta forma se reduce el número de sub-problemas a resolver, de hecho se espera que este número sea pequeño (al menos polinomial)

Contando el número de caminos en un grafo

Dado un grafo $G = (N, A)$, recuerde que una secuencia $a_1, \ldots, a_\ell$ de elementos en N es un camino en G si:

1. $\ell \geq 2$
2. $(a_i, a_{i+1}) \in A$ para cada $i \in \{1, \ldots, \ell - 1\}$

Decimos que un camino $a_1, \ldots, a_\ell$ va desde a_1 a a_ℓ, y definimos su largo como (ℓ - 1), vale decir, el número de arcos en el camino.

Contando el número de caminos en un grafo

Dado un grafo $G = (N, A)$, un par de nodos b, c en N y un número ℓ, queremos desarrollar un algoritmo que cuente el número de caminos desde b a c en G cuyo largo es igual a ℓ

Contando el número de caminos en un grafo

Dado un grafo $G = (N, A)$, un par de nodos b, c en N y un número ℓ, queremos desarrollar un algoritmo que cuente el número de caminos desde b a c en G cuyo largo es igual a ℓ

Suponemos que $N = \{1, \ldots, n\}$, $1 \leq \ell \leq n$ y representamos G a través de su matriz de adyacencia M:

- si $(i, j) \in A$, entonces $M[i, j] = 1$, en caso contrario $M[i, j] = 0$

Contando el número de caminos en un grafo

Dado un grafo $G = (N, A)$, un par de nodos b, c en N y un número ℓ, queremos desarrollar un algoritmo que cuente el número de caminos desde b a c en G cuyo largo es igual a ℓ

Suponemos que $N = \{1, \ldots, n\}$, $1 \leq \ell \leq n$ y representamos G a través de su matriz de adyacencia M:

- si $(i, j) \in A$, entonces $M[i, j] = 1$, en caso contrario $M[i, j] = 0$

Queremos entonces definir la función **ContarCaminos**(M, b, c, ℓ)

Una primera definición de **ContarCaminos**

ContarCaminos$(M[1 \ldots n][1 \ldots n],\ b,\ c,\ \ell)$
    if $\ell = 1$ **then return** $M[b, c]$
    else
        $aux := 0$
        for $i := 1$ **to** n **do**
            $aux := aux + M[b, i] \cdot$ **ContarCaminos**$(M,\ i,\ c,\ \ell - 1)$
        return aux

Una primera definición de **ContarCaminos**

ContarCaminos$(M[1 \ldots n][1 \ldots n],\ b,\ c,\ \ell)$
    if $\ell = 1$ **then return** $M[b, c]$
    else
        $aux := 0$
        for $i := 1$ **to** n **do**
            $aux := aux + M[b, i] \cdot$ **ContarCaminos**$(M,\ i,\ c,\ \ell - 1)$
        return aux

Observe que usamos la notación $C[1 \ldots m][1 \ldots n]$ para indicar que la matriz C tiene m filas y n columnas.

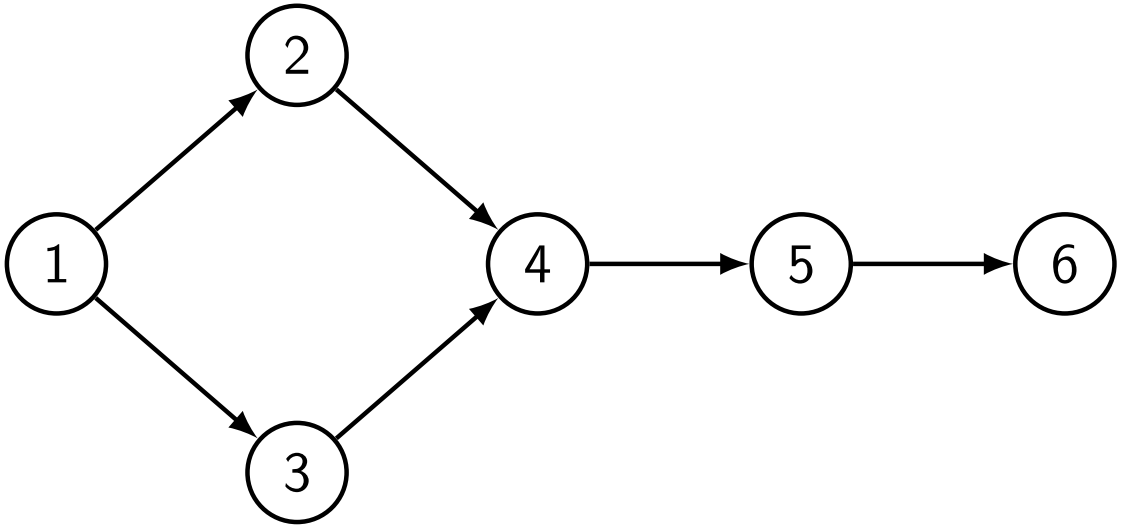
Una segunda definición de **ContarCaminos**

Podemos reducir el número de llamadas recursivas:

ContarCaminos($M[1 \ldots n][1 \ldots n]$, b, c, ℓ)
    if $\ell = 1$ **then return** $M[b, c]$
    else
        $aux := 0$
        for $i := 1$ **to** n **do**
            if $M[b, i] = 1$ **then**
                $aux := aux +$ **ContarCaminos**(M, i, c, $\ell - 1$)
        return aux

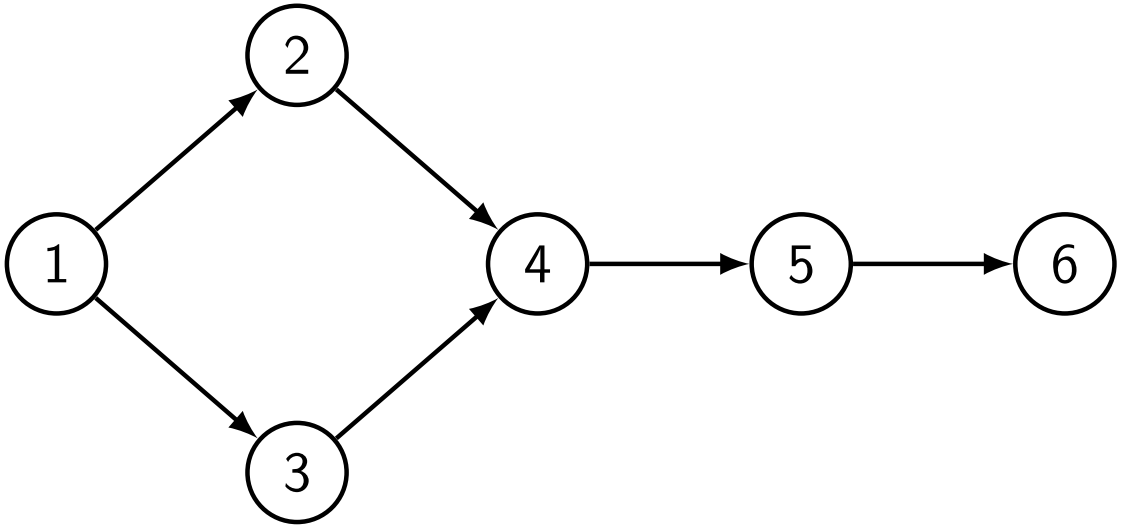
Llamadas repetidas en **ContarCaminos**

Considere el siguiente grafo G (representado por una matriz M):



Llamadas repetidas en **ContarCaminos**

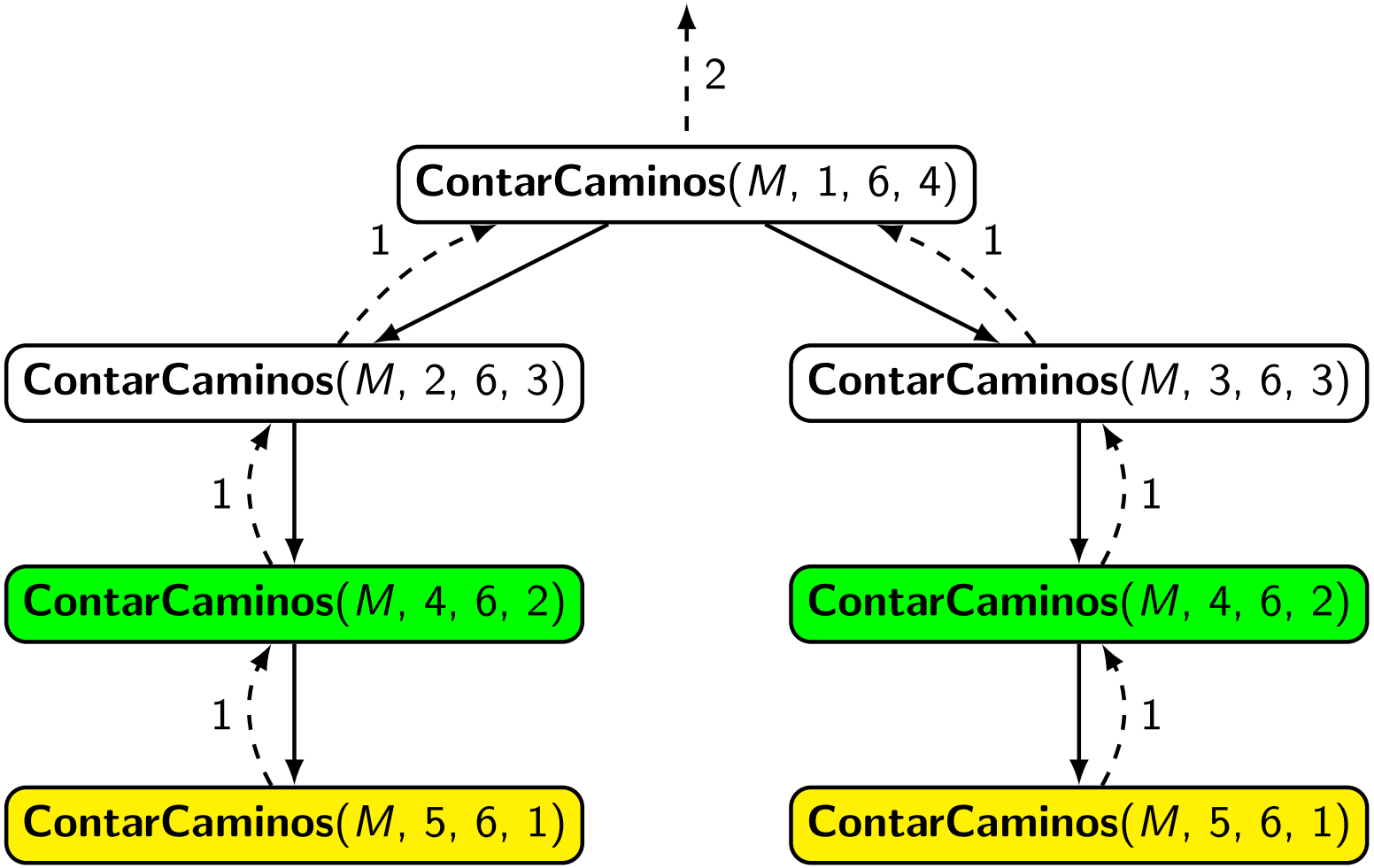
Considere el siguiente grafo G (representado por una matriz M):



Suponga que queremos contar el número de caminos en G desde el nodo 1 al nodo 6 y cuyo largo sea 4

Llamadas repetidas en **ContarCaminos**

Tenemos las siguientes llamadas recursivas:



Llamadas repetidas en **ContarCaminos**

Ejercicio

Demuestre que **ContarCaminos**($M[1 \ldots n][1 \ldots n]$, b, c, ℓ) realiza $\frac{n^\ell - n}{n-1}$ llamadas recursivas en el peor caso, para $n \geq 2$

- ¿Para qué grafos obtenemos el peor caso?

Llamadas repetidas en **ContarCaminos**

Ejercicio

Demuestre que **ContarCaminos**($M[1 \ldots n][1 \ldots n]$, b, c, ℓ) realiza $\frac{n^\ell - n}{n-1}$ llamadas recursivas en el peor caso, para $n \geq 2$

- ¿Para qué grafos obtenemos el peor caso?

El algoritmo realiza muchas llamadas recursivas repetidas.

Llamadas repetidas en **ContarCaminos**

Ejercicio

Demuestre que **ContarCaminos**$(M[1 \ldots n][1 \ldots n],\ b,\ c,\ \ell)$ realiza $\dfrac{n^\ell - n}{n-1}$ llamadas recursivas en el peor caso, para $n \geq 2$

- ¿Para qué grafos obtenemos el peor caso?

El algoritmo realiza muchas llamadas recursivas repetidas.

- Un número exponencial dado que sólo podemos tener $n^2 \cdot \ell$ llamadas distintas en la ejecución de **ContarCaminos**$(M[1 \ldots n][1 \ldots n],\ b,\ c,\ \ell)$

Llamadas repetidas en **ContarCaminos**

Ejercicio

Demuestre que **ContarCaminos**$(M[1 \ldots n][1 \ldots n],\ b,\ c,\ \ell)$ realiza $\dfrac{n^\ell - n}{n-1}$ llamadas recursivas en el peor caso, para $n \geq 2$

- ¿Para qué grafos obtenemos el peor caso?

El algoritmo realiza muchas llamadas recursivas repetidas.

- Un número exponencial dado que sólo podemos tener $n^2 \cdot \ell$ llamadas distintas en la ejecución de **ContarCaminos**$(M[1 \ldots n][1 \ldots n],\ b,\ c,\ \ell)$

Puesto que tenemos llamadas traslapadas y un espacio pequeño de sub-problemas este es un problema adecuado para programación dinámica.

Una tercera definición de **ContarCaminos**

Queremos calcular el número de caminos de largo ℓ desde un nodo b a un nodo c en un grafo G (representado por una matriz de adyacencia M)

Una tercera definición de **ContarCaminos**

Queremos calcular el número de caminos de largo ℓ desde un nodo b a un nodo c en un grafo G (representado por una matriz de adyacencia M)

Para evitar hacer llamadas recursivas repetidas, y así disminuir el número de llamadas recursivas, definimos una secuencia de matrices $H_1, \ldots, H_\ell$ tales que:

$$H_k[i,j] \text{ es el número de caminos de } i \text{ a } j \text{ de largo } k$$

De esta forma la respuesta a la pregunta inicial está en $H_\ell[b,c]$

Una tercera definición de **ContarCaminos**

Queremos calcular el número de caminos de largo ℓ desde un nodo b a un nodo c en un grafo G (representado por una matriz de adyacencia M)

Para evitar hacer llamadas recursivas repetidas, y así disminuir el número de llamadas recursivas, definimos una secuencia de matrices $H_1, \ldots, H_\ell$ tales que:

$H_k[i,j]$ es el número de caminos de i a j de largo k

De esta forma la respuesta a la pregunta inicial está en $H_\ell[b,c]$

La secuencia $H_1, \ldots, H_\ell$ se define de la siguiente forma:

1. $H_1 = M$
2. $H_{k+1} = M \cdot H_k$ para $k \in \{1, \ldots, \ell - 1\}$

Una tercera definición de **ContarCaminos**

La implementación recursiva de la idea descrita:

ContarTodosCaminos$(M[1 \ldots n][1 \ldots n], \ell)$
    if $\ell = 1$ **then return** M
    else
        $H :=$ **ContarTodosCaminos**$(M, \ell - 1)$
        return $M \cdot H$

ContarCaminos$(M[1 \ldots n][1 \ldots n], b, c, \ell)$
    $H :=$ **ContarTodosCaminos**(M, ℓ)
    return $H[b, c]$

La complejidad de **ContarCaminos**

Ejercicio

Demuestre que **ContarCaminos** en el peor caso es $O(\ell \cdot n^{\log_2(7)})$

- ¿Cuál es el tamaño de la entrada para **ContarCaminos**?
- ¿Qué operaciones básicas debemos considerar en el análisis de **ContarCaminos**?
- El algoritmo de Strassen es usado para obtener este resultado

Un primer desafío

Un camino $a_1, \ldots, a_\ell$ en un grafo G es simple si $a_i \neq a_j$ para $1 \leq i < j \leq \ell$

- Vale decir, no tiene nodo repetidos

Un primer desafío

Un camino $a_1, \ldots, a_\ell$ en un grafo G es simple si $a_i \neq a_j$ para $1 \leq i < j \leq \ell$

- Vale decir, no tiene nodo repetidos

Ejercicio

Suponga que G es un grafo representado por una matriz de adyacencia M y b, c son dos nodos distintos en G

Queremos implementar la función **ContarCaminosSimples**(M, b, c, ℓ) que retorna el número de caminos simples de largo ℓ desde b a c en G

Un primer desafío

Un camino $a_1, \ldots, a_\ell$ en un grafo G es simple si $a_i \neq a_j$ para $1 \leq i < j \leq \ell$

- Vale decir, no tiene nodo repetidos

Ejercicio

Suponga que G es un grafo representado por una matriz de adyacencia M y b, c son dos nodos distintos en G

Queremos implementar la función **ContarCaminosSimples**(M, b, c, ℓ) que retorna el número de caminos simples de largo ℓ desde b a c en G

- ¿Puede esta función ser implementada de manera eficiente usando programación dinámica?

Un primer desafío

Un camino $a_1, \ldots, a_\ell$ en un grafo G es simple si $a_i \neq a_j$ para $1 \leq i < j \leq \ell$

- Vale decir, no tiene nodo repetidos

Ejercicio

Suponga que G es un grafo representado por una matriz de adyacencia M y b, c son dos nodos distintos en G

Queremos implementar la función **ContarCaminosSimples**(M, b, c, ℓ) que retorna el número de caminos simples de largo ℓ desde b a c en G

- ¿Puede esta función ser implementada de manera eficiente usando programación dinámica?
- Si no puede ser implementada, ¿puede demostrar que éste es un problema difícil?