Programación dinámica 2

mat-151

Programación dinámica: **producto de una cadena de matrices**

Programación dinámica: **producto de una cadena de matrices**

- Tenemos una secuencia (cadena) $<A_1, A_2, ..., A_n>$ de **n matrices** a ser multiplicadas y queremos calcular el producto: $A_1 A_2 ... A_n$.

Programación dinámica: **producto de una cadena de matrices**

- Tenemos una secuencia (cadena) $<A_1, A_2, ..., A_n>$ de **n matrices** a ser multiplicadas y queremos calcular el producto: $A_1A_2...A_n$.
- Esto se puede resolver con una subrutina de multiplicación de pares de matrices una vez que se ha decidido el orden en que serán multiplicadas.

Programación dinámica: **producto de una cadena de matrices**

- Tenemos una secuencia (cadena) $<A_1, A_2, ..., A_n>$ de n matrices a ser multiplicadas y queremos calcular el producto: $A_1A_2...A_n$.
- Esto se puede resolver con una subrutina de multiplicación de pares de matrices una vez que se ha decidido el orden en que serán multiplicadas.
- La multiplicación de matrices es asociativa, así que cualquier orden dará el mismo resultado.

Programación dinámica: **producto de una cadena de matrices**

- Tenemos una secuencia (cadena) $<A_1, A_2, ..., A_n>$ de n matrices a ser multiplicadas y queremos calcular el producto: $A_1A_2...A_n$.
- Esto se puede resolver con una subrutina de multiplicación de pares de matrices una vez que se ha decidido el orden en que serán multiplicadas.
- La multiplicación de matrices es asociativa, así que cualquier orden dará el mismo resultado.
- Por ejemplo, si tenemos la cadena <A,B,C,D>, el producto ABCD se puede realizar de 5 formas distintas:

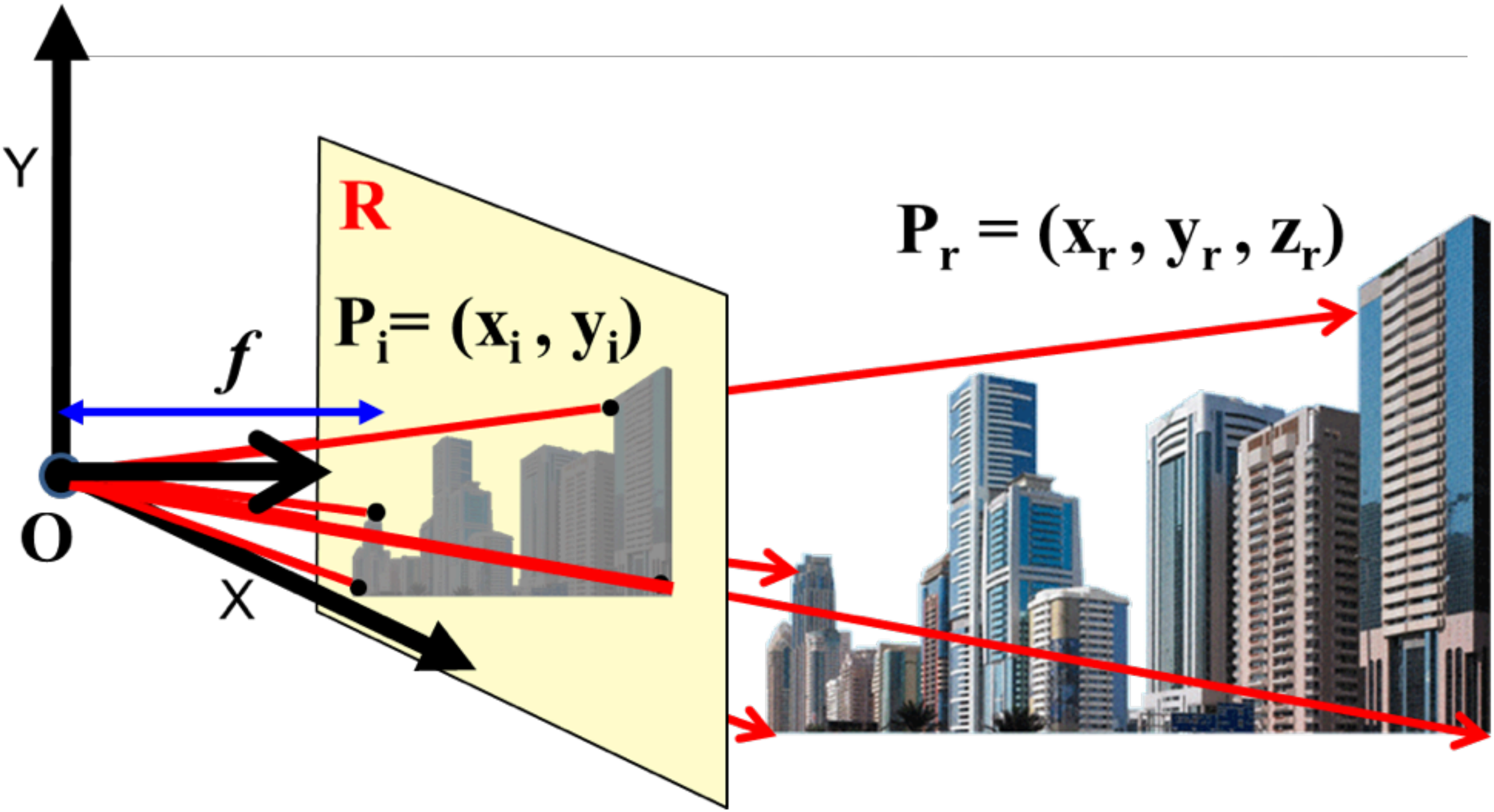
Programación dinámica: **producto de una cadena de matrices**

- Tenemos una secuencia (cadena) $<A_1, A_2, ..., A_n>$ de **n matrices** a ser multiplicadas y queremos calcular el producto: $A_1A_2...A_n$.
- Esto se puede resolver con una subrutina de multiplicación de pares de matrices una vez que se ha decidido el orden en que serán multiplicadas.
- La multiplicación de matrices es **asociativa**, así que cualquier orden dará el mismo resultado.
- Por ejemplo, si tenemos la cadena <A,B,C,D>, el producto ABCD se puede realizar de 5 formas distintas:
  - (A(B(CD))), (A((BC)D)), ((AB)(CD)), ((A(BC))D), (((AB)C)D).

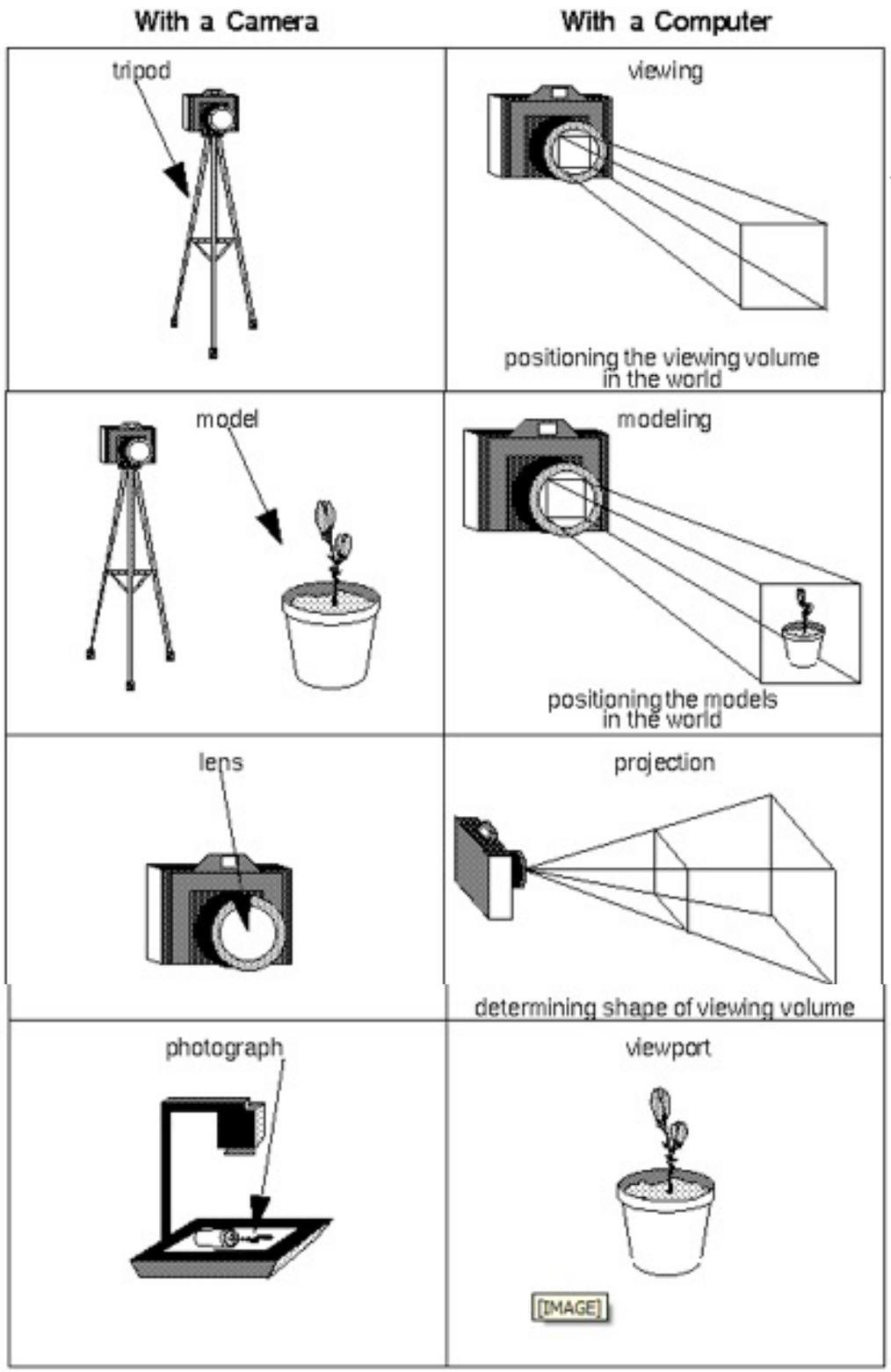
Programación dinámica: **producto de una cadena de matrices**

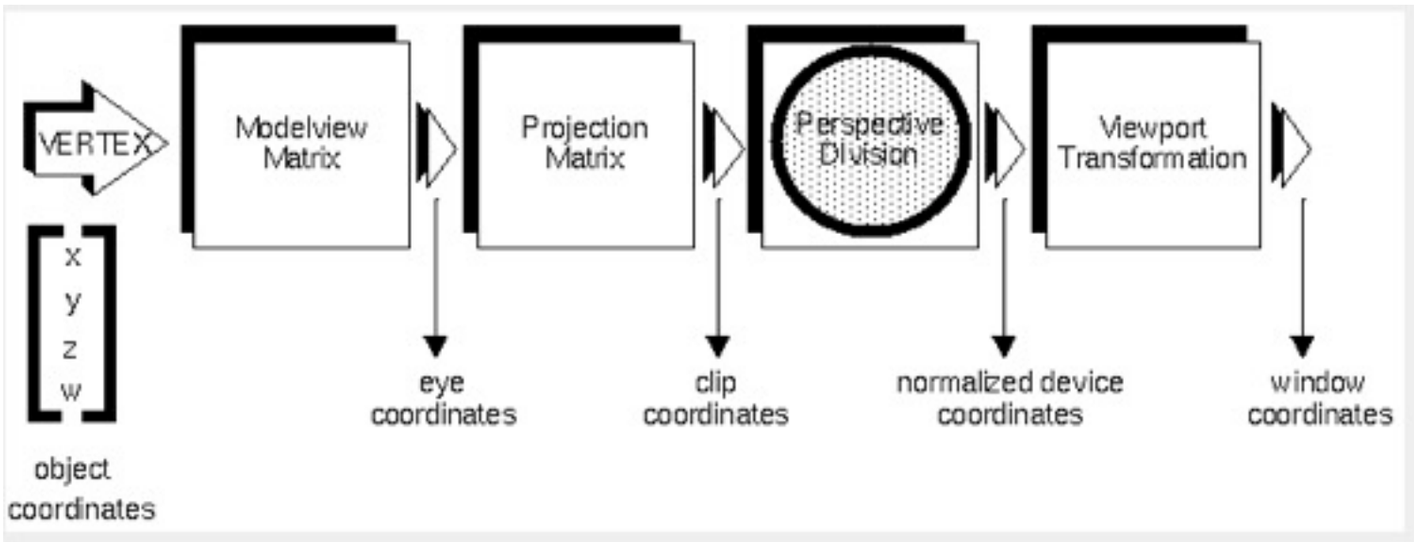
- Tenemos una secuencia (cadena) $<A_1, A_2, ..., A_n>$ de n matrices a ser multiplicadas y queremos calcular el producto: $A_1A_2...A_n$.
- Esto se puede resolver con una subrutina de multiplicación de pares de matrices una vez que se ha decidido el orden en que serán multiplicadas.
- La multiplicación de matrices es asociativa, así que cualquier orden dará el mismo resultado.
- Por ejemplo, si tenemos la cadena <A,B,C,D>, el producto ABCD se puede realizar de 5 formas distintas:
  - (A(B(CD))), (A((BC)D)), ((AB)(CD)), ((A(BC))D), (((AB)C)D).
- Sin embargo, el orden tendrá un impacto dramático en el costo de realizar el producto.

¿Dónde usamos esto? Ejemplo de graficación

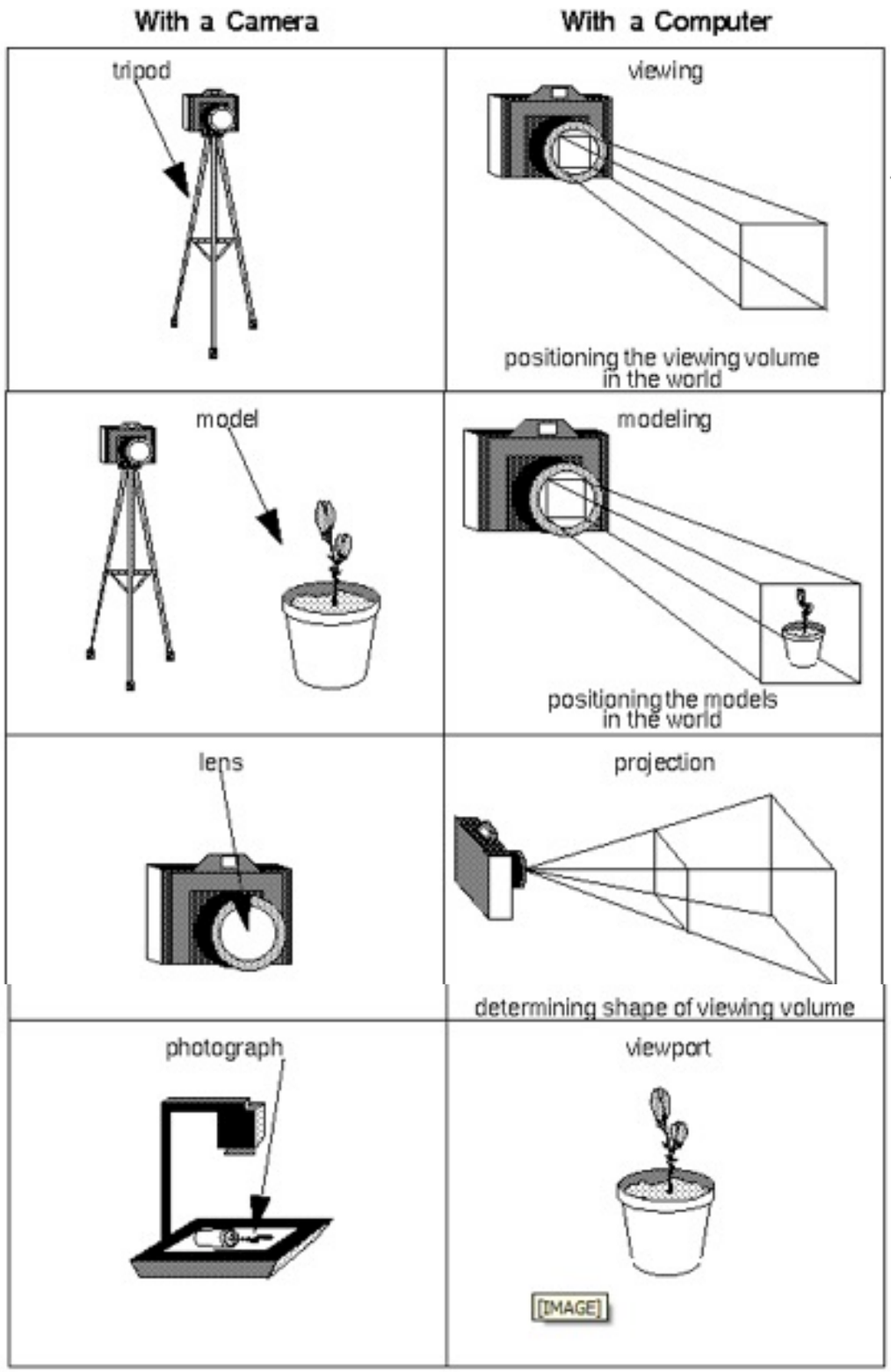


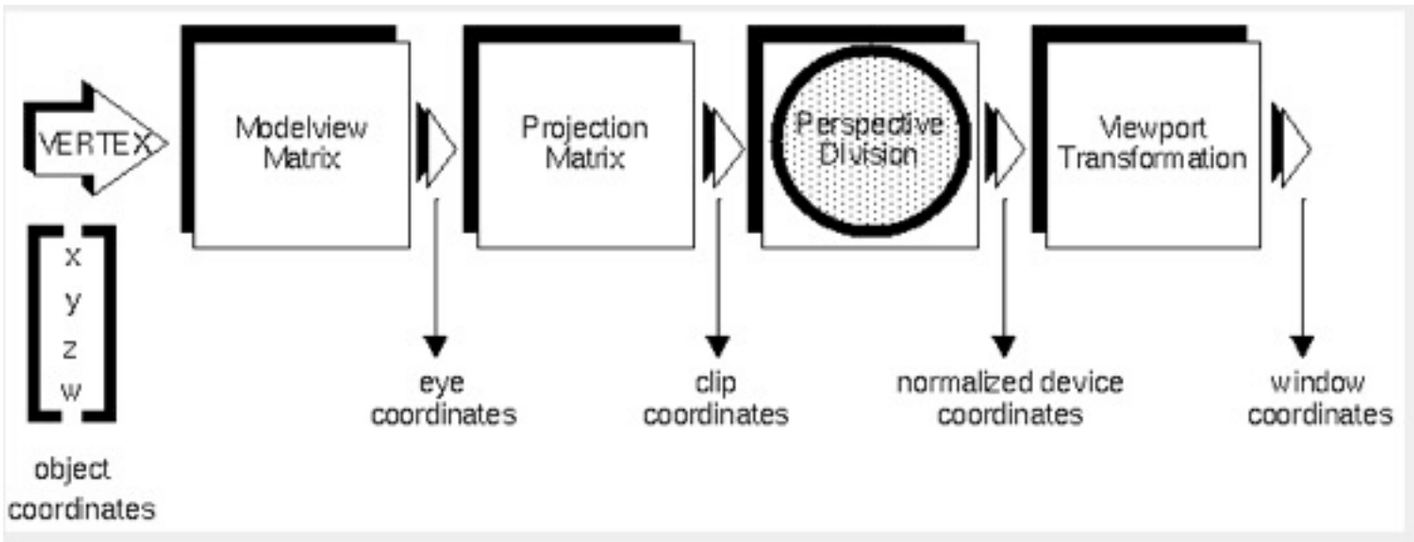
¿Dónde usamos esto? Ejemplo de graficación





¿Dónde usamos esto? Ejemplo de graficación





Programación dinámica: **producto de una cadena de matrices**

Programación dinámica: **producto de una cadena de matrices**

- Consideramos el costo de multiplicar 2 matrices:

Programación dinámica: **producto de una cadena de matrices**

- Consideramos el costo de multiplicar 2 matrices:

```
MATRIX-MULTIPLY(A,B)

1   if columns[A] ≠ rows[B]
2      then error "dimensiones incompatibles"
3      else for i←1 to rows[A]
4         do for j←1 to columns[B]
5            do C[i,j] ← 0
6               for k←1 to columns[A]
7                  do C[i,j] ← C[i,j] + A[i,k]B[k,j]
8      return C
```

Programación dinámica: **producto de una cadena de matrices**

- Consideramos el costo de multiplicar 2 matrices:

```
MATRIX-MULTIPLY(A,B)

1   if columns[A] ≠ rows[B]
2      then error "dimensiones incompatibles"
3      else for i←1 to rows[A]
4         do for j←1 to columns[B]
5            do C[i,j] ← 0
6               for k←1 to columns[A]
7                  do C[i,j] ← C[i,j] + A[i,k]B[k,j]
8      return C
```

- El número de columnas de A debe ser igual al número de renglones de B.

Programación dinámica: **producto de una cadena de matrices**

- Consideramos el costo de multiplicar 2 matrices:

```
MATRIX-MULTIPLY(A,B)

1   if columns[A] ≠ rows[B]
2      then error "dimensiones incompatibles"
3      else for i←1 to rows[A]
4         do for j←1 to columns[B]
5            do C[i,j] ← 0
6               for k←1 to columns[A]
7                  do C[i,j] ← C[i,j] + A[i,k]B[k,j]
8      return C
```

- El número de columnas de A debe ser igual al número de renglones de B.
- Si A es una matriz de p **x** q y B es de q **x** r, entonces C es de dimensión p **x** r.

Programación dinámica: **producto de una cadena de matrices**

- Consideramos el costo de multiplicar 2 matrices:

```
MATRIX-MULTIPLY(A,B)

1   if columns[A] ≠ rows[B]
2      then error "dimensiones incompatibles"
3      else for i←1 to rows[A]
4         do for j←1 to columns[B]
5            do C[i,j] ← 0
6               for k←1 to columns[A]
7                  do C[i,j] ← C[i,j] + A[i,k]B[k,j]
8      return C
```

- El número de columnas de A debe ser igual al número de renglones de B.
- Si A es una matriz de p **x** q y B es de q **x** r, entonces C es de dimensión p **x** r.
- El tiempo de cálculo de C es dominado por la multiplicación de escalares en la línea 7 que es pqr.

Programación dinámica: **producto de una cadena de matrices**

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:
- $<A_1,A_2,A_3>$, 10x100, 100x5, 5x50

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:
- $\langle A_1, A_2, A_3 \rangle$, 10x100, 100x5, 5x50
- $((A_1 A_2) A_3)$

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:
- $<A_1,A_2,A_3>$, 10x100, 100x5, 5x50
- $((A_1A_2)A_3)$
- $(A_1(A_2A_3)$

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:
- $<A_1,A_2,A_3>$, 10x100, 100x5, 5x50
- $((A_1A_2)A_3)$
- $(A_1(A_2A_3)$

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:
- <A_1,A_2,A_3>, 10x100, 100x5, 5x50
- ((A_1A_2)A_3)
- (A_1(A_2A_3)

- Dada una cadena **<A_1,A_2,...,A_n>** de **n** matrices, donde **i=1,2,...,n** la matriz **A_i** tiene dimensiones p_{i-1}x p_i, poner los paréntesis del producto minimizando el número de multiplicaciones escalares.

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:
- $<A_1,A_2,A_3>$, 10x100, 100x5, 5x50
- $((A_1A_2)A_3)$
- $(A_1(A_2A_3)$

- Dada una cadena **$<A_1,A_2,...,A_n>$** de **n** matrices, donde **i=1,2,...,n** la matriz **A_i** tiene dimensiones p_{i-1}x p_i, poner los paréntesis del producto minimizando el número de multiplicaciones escalares.
- No buscamos multiplicar las matrices sino minimizar el costo de la multiplicación en general.

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:
- $<A_1,A_2,A_3>$, 10x100, 100x5, 5x50
- $((A_1A_2)A_3) = (10 \cdot 100 \cdot 5) + (10 \cdot 5 \cdot 50) = 5000 + 2500 = 7{,}500$
- $(A_1(A_2A_3)$

- Dada una cadena $\mathbf{<A_1,A_2,\ldots,A_n>}$ de **n** matrices, donde **i=1,2,…,n** la matriz $\mathbf{A_i}$ tiene dimensiones p_{i-1}x p_i, poner los paréntesis del producto minimizando el número de multiplicaciones escalares.
- No buscamos multiplicar las matrices sino minimizar el costo de la multiplicación en general.

Programación dinámica: **producto de una cadena de matrices**

- Por ejemplo:
- <A_1,A_2,A_3>, 10x100, 100x5, 5x50
- $((A_1A_2)A_3) = (10 \cdot 100 \cdot 5) + (10 \cdot 5 \cdot 50) = 5000 + 2500 = 7{,}500$
- $(A_1(A_2A_3) = (100 \cdot 5 \cdot 50) + (10 \cdot 100 \cdot 50) = 25000 + 50000 = 75{,}000$

- Dada una cadena **<A_1,A_2,...,A_n>** de **n** matrices, donde **i=1,2,...,n** la matriz **A_i** tiene dimensiones p_{i-1}x p_i, poner los paréntesis del producto minimizando el número de multiplicaciones escalares.
- No buscamos multiplicar las matrices sino minimizar el costo de la multiplicación en general.

Programación dinámica: **producto de una cadena de matrices**

Programación dinámica: **producto de una cadena de matrices**

- Paso 1: estructura de una parentesización óptima.

Programación dinámica: **producto de una cadena de matrices**

- Paso 1: estructura de una parentesización óptima.
  - Sea $\mathbf{A_{i...j}}$ ($i \leq j$) la matriz que resulta de evaluar la multiplicación de $A_iA_{i+1}...A_j$.

Programación dinámica: **producto de una cadena de matrices**

- Paso 1: estructura de una parentesización óptima.
  - Sea $\mathbf{A_{i...j}}$ ($i \leq j$) la matriz que resulta de evaluar la multiplicación de $A_iA_{i+1}...A_j$.
  - Para **i<j** cualquier parentesización del producto $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1} para un entero en el rango $i \leq k < j$.

Programación dinámica: **producto de una cadena de matrices**

- Paso 1: estructura de una parentesización óptima.
  - Sea $\mathbf{A_{i...j}}$ ($i \leq j$) la matriz que resulta de evaluar la multiplicación de $A_iA_{i+1}...A_j$.
  - Para **i<j** cualquier parentesización del producto $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1} para un entero en el rango $i \leq k < j$.
  - Esto es, para un valor de **k** calculamos primero los productos $A_{i...k}$ y $A_{k+1...j}$ y luego los multiplicamos para tener el producto final $A_{i...j}$.

Programación dinámica: **producto de una cadena de matrices**

- Paso 1: estructura de una parentesización óptima.
  - Sea $\mathbf{A_{i...j}}$ ($i \leq j$) la matriz que resulta de evaluar la multiplicación de $A_iA_{i+1}...A_j$.
  - Para **i<j** cualquier parentesización del producto $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1} para un entero en el rango $i \leq k < j$.
  - Esto es, para un valor de **k** calculamos primero los productos $A_{i...k}$ y $A_{k+1...j}$ y luego los multiplicamos para tener el producto final $A_{i...j}$.
  - El costo de esta parentesización es la suma de:

Programación dinámica: **producto de una cadena de matrices**

- Paso 1: estructura de una parentesización óptima.
  - Sea $\mathbf{A_{i...j}}$ ($i \leq j$) la matriz que resulta de evaluar la multiplicación de $A_iA_{i+1}...A_j$.
  - Para **i<j** cualquier parentesización del producto $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1} para un entero en el rango $i \leq k < j$.
  - Esto es, para un valor de **k** calculamos primero los productos $A_{i...k}$ y $A_{k+1...j}$ y luego los multiplicamos para tener el producto final $A_{i...j}$.
  - El costo de esta parentesización es la suma de:
    - el costo de calcular la matriz $A_{i...k}$, más

Programación dinámica: **producto de una cadena de matrices**

- Paso 1: estructura de una parentesización óptima.
  - Sea $\mathbf{A_{i...j}}$ ($i \leq j$) la matriz que resulta de evaluar la multiplicación de $A_iA_{i+1}...A_j$.
  - Para **i<j** cualquier parentesización del producto $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1} para un entero en el rango $i \leq k < j$.
  - Esto es, para un valor de **k** calculamos primero los productos $A_{i...k}$ y $A_{k+1...j}$ y luego los multiplicamos para tener el producto final $A_{i...j}$.
  - El costo de esta parentesización es la suma de:
    - el costo de calcular la matriz $A_{i...k}$, más
    - el costo de calcular la matriz $A_{k+1...j}$, más

Programación dinámica: **producto de una cadena de matrices**

- Paso 1: estructura de una parentesización óptima.
  - Sea $\mathbf{A_{i...j}}$ ($i \leq j$) la matriz que resulta de evaluar la multiplicación de $A_iA_{i+1}...A_j$.
  - Para **i<j** cualquier parentesización del producto $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1} para un entero en el rango $i \leq k < j$.
  - Esto es, para un valor de **k** calculamos primero los productos $A_{i...k}$ y $A_{k+1...j}$ y luego los multiplicamos para tener el producto final $A_{i...j}$.
  - El costo de esta parentesización es la suma de:
    - el costo de calcular la matriz $A_{i...k}$, más
    - el costo de calcular la matriz $A_{k+1...j}$, más
    - el costo de multiplicar ambas matrices.

Programación dinámica: **producto de una cadena de matrices**

Programación dinámica: **producto de una cadena de matrices**

- La subestructura óptima de este problema es la siguiente:

Programación dinámica: **producto de una cadena de matrices**

- La subestructura óptima de este problema es la siguiente:
  - Supongase que una parentesización óptima de $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1},

Programación dinámica: **producto de una cadena de matrices**

- La subestructura óptima de este problema es la siguiente:
  - Supongase que una parentesización óptima de $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1},
  - La parentesización de la cadena $A_iA_{i+1}...A_k$ dentro la parentesización óptima de $A_iA_{i+1}...A_j$ debe ser la óptima.

Programación dinámica: **producto de una cadena de matrices**

- La subestructura óptima de este problema es la siguiente:
  - Supongase que una parentesización óptima de $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1},
  - La parentesización de la cadena $A_iA_{i+1}...A_k$ dentro la parentesización óptima de $A_iA_{i+1}...A_j$ debe ser la óptima.
  - Mismo razonamiento para $A_{k+1}A_{k+2}...A_j$.

Programación dinámica: **producto de una cadena de matrices**

- La subestructura óptima de este problema es la siguiente:
  - Supongase que una parentesización óptima de $A_iA_{i+1}...A_j$ divide el producto entre A_k y A_{k+1},
  - La parentesización de la cadena $A_iA_{i+1}...A_k$ dentro la parentesización óptima de $A_iA_{i+1}...A_j$ debe ser la óptima.
  - Mismo razonamiento para $A_{k+1}A_{k+2}...A_j$.
  - Problema: encontrar el valor óptimo del valor de división k.

Programación dinámica: **producto de una cadena de matrices**

Programación dinámica: **producto de una cadena de matrices**

- Paso 2: solución recursiva.

Programación dinámica: **producto de una cadena de matrices**

- Paso 2: solución recursiva.
  - Subproblema: determinar el costo mínimo de la parentesización de $A_iA_{i+1}...A_j$ para $1 \leq i \leq j \leq n$.

Programación dinámica: **producto de una cadena de matrices**

- Paso 2: solución recursiva.
  - Subproblema: determinar el costo mínimo de la parentesización de $A_iA_{i+1}...A_j$ para $1 \leq i \leq j \leq n$.
  - Sea m[i,j] el número mínimo de multiplicaciones de escalares necesarias para calcular el producto Ai...j. Para el problema entero el costo de la operación más eficiente para calcular A1...n es m[1,n].

Programación dinámica: **producto de una cadena de matrices**

- Paso 2: solución recursiva.
  - Subproblema: determinar el costo mínimo de la parentesización de $A_iA_{i+1}...A_j$ para $1 \leq i \leq j \leq n$.
  - Sea m[i,j] el número mínimo de multiplicaciones de escalares necesarias para calcular el producto Ai...j. Para el problema entero el costo de la operación más eficiente para calcular A1...n es m[1,n].
  - ¿Qué pasa con m[i,j] cuando i=j?

Programación dinámica: **producto de una cadena de matrices**

- Paso 2: solución recursiva.
  - Subproblema: determinar el costo mínimo de la parentesización de $A_iA_{i+1}...A_j$ para $1 \leq i \leq j \leq n$.
  - Sea m[i,j] el número mínimo de multiplicaciones de escalares necesarias para calcular el producto Ai...j. Para el problema entero el costo de la operación más eficiente para calcular A1...n es m[1,n].
  - ¿Qué pasa con m[i,j] cuando i=j?
    - El subproblema consta de una sola matriz Ai...i=i, no hay multiplicaciones y m[i,i]=0 para i=1,2,...,n.

Programación dinámica: **producto de una cadena de matrices**

- Paso 2: solución recursiva.
  - Subproblema: determinar el costo mínimo de la parentesización de $A_iA_{i+1}...A_j$ para $1 \leq i \leq j \leq n$.
  - Sea m[i,j] el número mínimo de multiplicaciones de escalares necesarias para calcular el producto Ai...j. Para el problema entero el costo de la operación más eficiente para calcular A1...n es m[1,n].
  - ¿Qué pasa con m[i,j] cuando i=j?
    - El subproblema consta de una sola matriz Ai...i=i, no hay multiplicaciones y m[i,i]=0 para i=1,2,...,n.
  - Para m[i,j] cuando i<j: (A_i tiene dimensiones $p_{i-1} \times p_i$ para i=1,2,...,n.)

Programación dinámica: **producto de una cadena de matrices**

- Paso 2: solución recursiva.
  - Subproblema: determinar el costo mínimo de la parentesización de $A_iA_{i+1}...A_j$ para $1 \leq i \leq j \leq n$.
  - Sea m[i,j] el número mínimo de multiplicaciones de escalares necesarias para calcular el producto Ai...j. Para el problema entero el costo de la operación más eficiente para calcular A1...n es m[1,n].
  - ¿Qué pasa con m[i,j] cuando i=j?
    - El subproblema consta de una sola matriz Ai...i=i, no hay multiplicaciones y m[i,i]=0 para i=1,2,...,n.
  - Para m[i,j] cuando i<j: (A_i tiene dimensiones $p_{i-1} \times p_i$ para i=1,2,...,n.)
    - $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_kp_j$.

Programación dinámica: **producto de una cadena de matrices**

Programación dinámica: **producto de una cadena de matrices**

- No conocemos k pero solo hay j-i posibilidades a examinar.

Programación dinámica: **producto de una cadena de matrices**

- No conocemos k pero solo hay j-i posibilidades a examinar.
- La definición recursiva es entonces:

Programación dinámica: **producto de una cadena de matrices**

- No conocemos k pero solo hay j-i posibilidades a examinar.
- La definición recursiva es entonces:

$$m[i,j] = \begin{cases} 0 & \text{si } i = j, \\ \min\limits_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1} p_k p_j] & \text{si } i < j. \end{cases}$$

Programación dinámica: **producto de una cadena de matrices**

- No conocemos k pero solo hay j-i posibilidades a examinar.
- La definición recursiva es entonces:

$$m[i,j] = \begin{cases} 0 & \text{si } i = j, \\ \min\limits_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1} p_k p_j] & \text{si } i < j. \end{cases}$$

- y llamaremos s[i,j] al valor de k en el que dividimos $A_i A_{i+1} \ldots A_j$ para obtener una parentesización óptima.

Programación dinámica: **producto de una cadena de matrices**

Programación dinámica: **producto de una cadena de matrices**

- Paso 3: cálculo de los costos óptimos

Programación dinámica: **producto de una cadena de matrices**

- Paso 3: cálculo de los costos óptimos
  - Implementación con programación dinámica bottom-up.

Programación dinámica: **producto de una cadena de matrices**

- Paso 3: cálculo de los costos óptimos
  - Implementación con programación dinámica bottom-up.
  - A_i tiene dimensiones $p_{i-1} \times p_i$ para i=1,2,...,n.

Programación dinámica: **producto de una cadena de matrices**

- Paso 3: cálculo de los costos óptimos
  - Implementación con programación dinámica bottom-up.
  - A_i tiene dimensiones $p_{i-1} \times p_i$ para i=1,2,...,n.
  - La entrada es una secuencia $p<p_0,p_1,...p_n>$, donde length[p] = n+1.

Programación dinámica: **producto de una cadena de matrices**

- Paso 3: cálculo de los costos óptimos
  - Implementación con programación dinámica bottom-up.
  - A_i tiene dimensiones $p_{i-1} \times p_i$ para i=1,2,...,n.
  - La entrada es una secuencia $p<p_0,p_1,...p_n>$, donde length[p] = n+1.
  - Se utiliza la tabla auxiliar m[1...n, 1...n] para almacenar los m[i,j] costos.

Programación dinámica: **producto de una cadena de matrices**

- Paso 3: cálculo de los costos óptimos
  - Implementación con programación dinámica bottom-up.
  - A_i tiene dimensiones $p_{i-1} \times p_i$ para i=1,2,...,n.
  - La entrada es una secuencia $p<p_0,p_1,...p_n>$, donde length[p] = n+1.
  - Se utiliza la tabla auxiliar m[1...n, 1...n] para almacenar los m[i,j] costos.
  - Se utiliza la tabla auxiliar s[1...n, 1...n] que almacene el índice de k que produjo los valores óptimos.

Programación dinámica: producto de una cadena de matrices

p = [30, 35, 15, 5, 10, 20, 25]

m(i,j) , i columna , j renglón

	1	2	3	4	5	6
6						0
5					0	
4				0		
3			0			
2		0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,2] = \min\{m[1,1] + m[2,2] + p_0 p_1 p_2 = 0 + 0 + 30 \cdot 35 \cdot 15 = 15,750$$
$$s[1,2] = k = 1$$

m(i,j) , i columna , j renglón

	1	2	3	4	5	6
6						0
5					0	
4				0		
3			0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,2] = \min\{m[1,1] + m[2,2] + p_0 p_1 p_2 = 0 + 0 + 30 \cdot 35 \cdot 15 = 15,750$$
$$s[1,2] = k = 1$$

$$m[2,3] = \min\{m[2,2] + m[3,3] + p_1 p_2 p_3 = 0 + 0 + 35 \cdot 15 \cdot 5 = 2625$$
$$s[2,3] = k = 2$$

m(i,j) , i columna , j renglón

	1	2	3	4	5	6
6						0
5					0	
4				0		
3		2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,2] = \min\{m[1,1] + m[2,2] + p_0 p_1 p_2 = 0 + 0 + 30 \cdot 35 \cdot 15 = 15,750$$
$$s[1,2] = k = 1$$

$$m[2,3] = \min\{m[2,2] + m[3,3] + p_1 p_2 p_3 = 0 + 0 + 35 \cdot 15 \cdot 5 = 2625$$
$$s[2,3] = k = 2$$

$$m[3,4] = \min\{m[3,3] + m[4,4] + p_2 p_3 p_4 = 0 + 0 + 15 \cdot 5 \cdot 10 = 750$$
$$s[3,4] = k = 3$$

m(i,j) , i columna , j renglón

	1	2	3	4	5	6
6						0
5					0	
4			750	0		
3		2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,2] = \min\{m[1,1] + m[2,2] + p_0 p_1 p_2 = 0 + 0 + 30 \cdot 35 \cdot 15 = 15,750$$
$$s[1,2] = k = 1$$

$$m[2,3] = \min\{m[2,2] + m[3,3] + p_1 p_2 p_3 = 0 + 0 + 35 \cdot 15 \cdot 5 = 2625$$
$$s[2,3] = k = 2$$

$$m[3,4] = \min\{m[3,3] + m[4,4] + p_2 p_3 p_4 = 0 + 0 + 15 \cdot 5 \cdot 10 = 750$$
$$s[3,4] = k = 3$$

$$m[4,5] = \min\{m[4,4] + m[5,5] + p_3 p_4 p_5 = 0 + 0 + 5 \cdot 10 \cdot 20 = 1,000$$
$$s[4,5] = k = 4$$

m(i,j) , i columna , j renglón

	1	2	3	4	5	6
6						0
5				1,000	0	
4			750	0		
3		2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,2] = \min\{m[1,1] + m[2,2] + p_0 p_1 p_2 = 0 + 0 + 30 \cdot 35 \cdot 15 = 15,750$$
$$s[1,2] = k = 1$$

$$m[2,3] = \min\{m[2,2] + m[3,3] + p_1 p_2 p_3 = 0 + 0 + 35 \cdot 15 \cdot 5 = 2625$$
$$s[2,3] = k = 2$$

$$m[3,4] = \min\{m[3,3] + m[4,4] + p_2 p_3 p_4 = 0 + 0 + 15 \cdot 5 \cdot 10 = 750$$
$$s[3,4] = k = 3$$

$$m[4,5] = \min\{m[4,4] + m[5,5] + p_3 p_4 p_5 = 0 + 0 + 5 \cdot 10 \cdot 20 = 1,000$$
$$s[4,5] = k = 4$$

$$m[5,6] = \min\{m[5,5] + m[6,6] + p_4 p_5 p_6 = 0 + 0 + 10 \cdot 20 \cdot 25 = 5,000$$
$$s[5,6] = k = 5$$

m(i,j) , i columna , j renglón

	1	2	3	4	5	6
6					5,000	0
5				1,000	0	
4			750	0		
3		2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

	1	2	3	4	5	6
6					5,000	0
5				1,000	0	
4			750	0		
3		2,625	0			
2	15,750	0				
1	0					

Programación dinámica: producto de una cadena de matrices

$$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + p_0 p_1 p_3 = 0 + 2,625 + 30 \cdot 35 \cdot 5 = 7,875 \\ m[1,2] + m[3,3] + p_0 p_2 p_3 = 15,750 + 0 + 30 \cdot 15 \cdot 5 = 18,000 \end{cases}$$

$$s[1,3] = 1$$

p = [30, 35, 15, 5, 10, 20, 25]

	1	2	3	4	5	6
6					5,000	0
5				1,000	0	
4			750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: producto de una cadena de matrices

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + p_0 p_1 p_3 = 0 + 2,625 + 30 \cdot 35 \cdot 5 = 7,875 \\ m[1,2] + m[3,3] + p_0 p_2 p_3 = 15,750 + 0 + 30 \cdot 15 \cdot 5 = 18,000 \end{cases}$$

$$s[1,3] = 1$$

$$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] + p_1 p_2 p_4 = 0 + 750 + 35 \cdot 15 \cdot 10 = 6,000 \\ m[2,3] + m[4,4] + p_1 p_3 p_4 = 2,625 + 0 + 35 \cdot 5 \cdot 10 = 4,375 \end{cases}$$

$$s[2,4] = 3$$

	1	2	3	4	5	6
6					5,000	0
5				1,000	0	
4		4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + p_0 p_1 p_3 = 0 + 2,625 + 30 \cdot 35 \cdot 5 = 7,875 \\ m[1,2] + m[3,3] + p_0 p_2 p_3 = 15,750 + 0 + 30 \cdot 15 \cdot 5 = 18,000 \end{cases}$$

$$s[1,3] = 1$$

$$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] + p_1 p_2 p_4 = 0 + 750 + 35 \cdot 15 \cdot 10 = 6,000 \\ m[2,3] + m[4,4] + p_1 p_3 p_4 = 2,625 + 0 + 35 \cdot 5 \cdot 10 = 4,375 \end{cases}$$

$$s[2,4] = 3$$

$$m[3,5] = \min \begin{cases} m[3,3] + m[4,5] + p_2 p_3 p_5 = 0 + 1000 + 15 \cdot 5 \cdot 20 = 2,500 \\ m[3,4] + m[5,5] + p_2 p_4 p_5 = 750 + 0 + 15 \cdot 10 \cdot 20 = 3,750 \end{cases}$$

$$s[3,5] = 3$$

	1	2	3	4	5	6
6					5,000	0
5			2,500	1,000	0	
4		4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: producto de una cadena de matrices

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + p_0 p_1 p_3 = 0 + 2,625 + 30 \cdot 35 \cdot 5 = 7,875 \\ m[1,2] + m[3,3] + p_0 p_2 p_3 = 15,750 + 0 + 30 \cdot 15 \cdot 5 = 18,000 \end{cases}$$

$$s[1,3] = 1$$

$$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] + p_1 p_2 p_4 = 0 + 750 + 35 \cdot 15 \cdot 10 = 6,000 \\ m[2,3] + m[4,4] + p_1 p_3 p_4 = 2,625 + 0 + 35 \cdot 5 \cdot 10 = 4,375 \end{cases}$$

$$s[2,4] = 3$$

$$m[3,5] = \min \begin{cases} m[3,3] + m[4,5] + p_2 p_3 p_5 = 0 + 1000 + 15 \cdot 5 \cdot 20 = 2,500 \\ m[3,4] + m[5,5] + p_2 p_4 p_5 = 750 + 0 + 15 \cdot 10 \cdot 20 = 3,750 \end{cases}$$

$$s[3,5] = 3$$

$$m[4,6] = \min \begin{cases} m[4,4] + m[5,6] + p_3 p_4 p_6 = 0 + 5,000 + 5 \cdot 10 \cdot 25 = 6,250 \\ m[4,5] + m[6,6] + p_3 p_5 p_6 = 1,000 + 0 + 5 \cdot 20 \cdot 25 = 3,500 \end{cases}$$

$$s[4,6] = 5$$

	1	2	3	4	5	6
6				3,500	5,000	0
5			2,500	1,000	0	
4		4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: producto de una cadena de matrices

p = [30, 35, 15, 5, 10, 20, 25]

	1	2	3	4	5	6
6				3,500	5,000	0
5			2,500	1,000	0	
4		4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,4] = \min \begin{cases} m[1,1] + m[2,4] + p_0 p_1 p_4 = 0 + 4,375 + 30 \cdot 35 \cdot 10 = 14,875 \\ m[1,2] + m[3,4] + p_0 p_2 p_4 = 15,750 + 750 + 30 \cdot 15 \cdot 10 = 21,000 \\ m[1,3] + m[4,4] + p_0 p_3 p_4 = 7,875 + 0 + 30 \cdot 5 \cdot 10 = 9,375 \end{cases}$$

$$s[1,4] = 3$$

	1	2	3	4	5	6
6				3,500	5,000	0
5			2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,4] = \min \begin{cases} m[1,1] + m[2,4] + p_0 p_1 p_4 = 0 + 4,375 + 30 \cdot 35 \cdot 10 = 14,875 \\ m[1,2] + m[3,4] + p_0 p_2 p_4 = 15,750 + 750 + 30 \cdot 15 \cdot 10 = 21,000 \\ m[1,3] + m[4,4] + p_0 p_3 p_4 = 7,875 + 0 + 30 \cdot 5 \cdot 10 = 9,375 \end{cases}$$

$$s[1,4] = 3$$

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p_1 p_2 p_5 = 0 + 2,500 + 35 \cdot 15 \cdot 20 = 13,000 \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = 2,625 + 1,000 + 35 \cdot 5 \cdot 20 = 7,125 \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = 4,375 + 0 + 35 \cdot 10 \cdot 20 = 11,375 \end{cases}$$

$$s[2,5] = 3$$

	1	2	3	4	5	6
6				3,500	5,000	0
5		7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,4] = \min \begin{cases} m[1,1] + m[2,4] + p_0 p_1 p_4 = 0 + 4,375 + 30 \cdot 35 \cdot 10 = 14,875 \\ m[1,2] + m[3,4] + p_0 p_2 p_4 = 15,750 + 750 + 30 \cdot 15 \cdot 10 = 21,000 \\ m[1,3] + m[4,4] + p_0 p_3 p_4 = 7,875 + 0 + 30 \cdot 5 \cdot 10 = 9,375 \end{cases}$$

$$s[1,4] = 3$$

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p_1 p_2 p_5 = 0 + 2,500 + 35 \cdot 15 \cdot 20 = 13,000 \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = 2,625 + 1,000 + 35 \cdot 5 \cdot 20 = 7,125 \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = 4,375 + 0 + 35 \cdot 10 \cdot 20 = 11,375 \end{cases}$$

$$s[2,5] = 3$$

$$m[3,6] = \min \begin{cases} m[3,3] + m[4,6] + p_2 p_3 p_6 = 0 + 3,500 + 15 \cdot 5 \cdot 25 = 5,375 \\ m[3,4] + m[5,6] + p_2 p_4 p_6 = 750 + 5,000 + 15 \cdot 10 \cdot 25 = 9,500 \\ m[3,5] + m[6,6] + p_2 p_5 p_6 = 2,500 + 0 + 15 \cdot 20 \cdot 25 = 10,000 \end{cases}$$

$$s[3,6] = 3$$

	1	2	3	4	5	6
6			5,375	3,500	5,000	0
5		7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: producto de una cadena de matrices

p = [30, 35, 15, 5, 10, 20, 25]

	1	2	3	4	5	6
6			5,375	3,500	5,000	0
5		7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,5] = \min \begin{cases} m[1,1] + m[2,5] + p_0 p_1 p_5 = 0 + 7,125 + 30 \cdot 35 \cdot 20 = 28,125 \\ m[1,2] + m[3,5] + p_0 p_2 p_5 = 15,750 + 2500 + 30 \cdot 15 \cdot 20 = 27,250 \\ m[1,3] + m[4,5] + p_0 p_3 p_5 = 7,875 + 1,000 + 30 \cdot 5 \cdot 20 = 11,875 \\ m[1,4] + m[5,5] + p_0 p_4 p_5 = 9,375 + 0 + 30 \cdot 10 \cdot 20 = 15,375 \end{cases}$$

$$s[1,5] = 3$$

	1	2	3	4	5	6
6			5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,5] = \min \begin{cases} m[1,1] + m[2,5] + p_0 p_1 p_5 = 0 + 7,125 + 30 \cdot 35 \cdot 20 = 28,125 \\ m[1,2] + m[3,5] + p_0 p_2 p_5 = 15,750 + 2500 + 30 \cdot 15 \cdot 20 = 27,250 \\ m[1,3] + m[4,5] + p_0 p_3 p_5 = 7,875 + 1,000 + 30 \cdot 5 \cdot 20 = 11,875 \\ m[1,4] + m[5,5] + p_0 p_4 p_5 = 9,375 + 0 + 30 \cdot 10 \cdot 20 = 15,375 \end{cases}$$

$$s[1,5] = 3$$

$$m[2,6] = \min \begin{cases} m[2,2] + m[3,6] + p_1 p_2 p_6 = 0 + 5,375 + 35 \cdot 15 \cdot 25 = 18,500 \\ m[2,3] + m[4,6] + p_1 p_3 p_6 = 2,625 + 3,500 + 35 \cdot 5 \cdot 25 = 10,500 \\ m[2,4] + m[5,6] + p_1 p_4 p_6 = 4,375 + 5,000 + 35 \cdot 10 \cdot 25 = 18,125 \\ m[2,5] + m[6,6] + p_1 p_5 p_6 = 7,125 + 0 + 35 \cdot 20 \cdot 20 = 24,625 \end{cases}$$

$$s[2,6] = 3$$

	1	2	3	4	5	6
6		10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: producto de una cadena de matrices

p = [30, 35, 15, 5, 10, 20, 25]

	1	2	3	4	5	6
6		10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,6] = \min \begin{cases} m[1,1] + m[2,6] + p_0 p_1 p_6 = 0 + 10,500 + 30 \cdot 35 \cdot 25 = 36,750 \\ m[1,2] + m[3,6] + p_0 p_2 p_6 = 15,750 + 5,375 + 30 \cdot 15 \cdot 25 = 32,375 \\ m[1,3] + m[4,6] + p_0 p_3 p_6 = 7,875 + 3,500 + 30 \cdot 5 \cdot 25 = 15,125 \\ m[1,4] + m[5,6] + p_0 p_4 p_6 = 9,375 + 5,000 + 30 \cdot 10 \cdot 25 = 21,875 \\ m[1,5] + m[6,6] + p_0 p_5 p_6 = 11,875 + 0 + 30 \cdot 20 \cdot 25 = 26,875 \end{cases}$$

$$s[1,6] = 3$$

	1	2	3	4	5	6
6	15,125	10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,6] = \min \begin{cases} m[1,1] + m[2,6] + p_0 p_1 p_6 = 0 + 10,500 + 30 \cdot 35 \cdot 25 = 36,750 \\ m[1,2] + m[3,6] + p_0 p_2 p_6 = 15,750 + 5,375 + 30 \cdot 15 \cdot 25 = 32,375 \\ m[1,3] + m[4,6] + p_0 p_3 p_6 = 7,875 + 3,500 + 30 \cdot 5 \cdot 25 = 15,125 \\ m[1,4] + m[5,6] + p_0 p_4 p_6 = 9,375 + 5,000 + 30 \cdot 10 \cdot 25 = 21,875 \\ m[1,5] + m[6,6] + p_0 p_5 p_6 = 11,875 + 0 + 30 \cdot 20 \cdot 25 = 26,875 \end{cases}$$

$s[1,6] = 3$

	1	2	3	4	5	6
6	3	3	3	5	5	
5	3	3	3	4		
4	3	3	3			
3	1	2				
2	1					
1						

	1	2	3	4	5	6
6	15,125	10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **producto de una cadena de matrices**

p = [30, 35, 15, 5, 10, 20, 25]

$$m[1,6] = \min \begin{cases} m[1,1] + m[2,6] + p_0 p_1 p_6 = 0 + 10,500 + 30 \cdot 35 \cdot 25 = 36,750 \\ m[1,2] + m[3,6] + p_0 p_2 p_6 = 15,750 + 5,375 + 30 \cdot 15 \cdot 25 = 32,375 \\ m[1,3] + m[4,6] + p_0 p_3 p_6 = 7,875 + 3,500 + 30 \cdot 5 \cdot 25 = 15,125 \\ m[1,4] + m[5,6] + p_0 p_4 p_6 = 9,375 + 5,000 + 30 \cdot 10 \cdot 25 = 21,875 \\ m[1,5] + m[6,6] + p_0 p_5 p_6 = 11,875 + 0 + 30 \cdot 20 \cdot 25 = 26,875 \end{cases}$$

$s[1,6] = 3$

	1	2	3	4	5	6
6	3	3	3	5	5	
5	3	3	3	4		
4	3	3	3			
3	1	2				
2	1					
1						

$$((A_1(A_2A_3))(A_4A_5)A_6)$$

	1	2	3	4	5	6
6	15,125	10,500	5,375	3,500	5,000	0
5	11,875	7,125	2,500	1,000	0	
4	9,375	4,375	750	0		
3	7,875	2,625	0			
2	15,750	0				
1	0					

Programación dinámica: **elementos**

Programación dinámica: **elementos**

- El problema exhibe subestructura óptima.

Programación dinámica: **elementos**

- El problema exhibe subestructura óptima.
- Patrón para encontrar la subestructura óptima:

Programación dinámica: **elementos**

- El problema exhibe subestructura óptima.
- Patrón para encontrar la subestructura óptima:
  i. mostrar que la solución consiste en una elección (la estación anterior en la línea de ensamblado, meter o no un objeto a la mochila, elegir un índice donde separar la cadena de matrices...). Esta elección deja uno o más subproblemas que resolver.

Programación dinámica: **elementos**

- El problema exhibe subestructura óptima.
- Patrón para encontrar la subestructura óptima:

  i. mostrar que la solución consiste en una elección (la estación anterior en la línea de ensamblado, meter o no un objeto a la mochila, elegir un índice donde separar la cadena de matrices...). Esta elección deja uno o más subproblemas que resolver.

  ii. suponer que, para un subproblema dado se nos da la solución óptima.

Programación dinámica: **elementos**

- El problema exhibe subestructura óptima.
- Patrón para encontrar la subestructura óptima:

  i. mostrar que la solución consiste en una elección (la estación anterior en la línea de ensamblado, meter o no un objeto a la mochila, elegir un índice donde separar la cadena de matrices...). Esta elección deja uno o más subproblemas que resolver.

  ii. suponer que, para un subproblema dado se nos da la solución óptima.

  iii. dada esta solución determinar la mejor manera de caracterizar los subproblemas.

Programación dinámica: **elementos**

- El problema exhibe subestructura óptima.
- Patrón para encontrar la subestructura óptima:

  i. mostrar que la solución consiste en una elección (la estación anterior en la línea de ensamblado, meter o no un objeto a la mochila, elegir un índice donde separar la cadena de matrices...). Esta elección deja uno o más subproblemas que resolver.

  ii. suponer que, para un subproblema dado se nos da la solución óptima.

  iii. dada esta solución determinar la mejor manera de caracterizar los subproblemas.

  iv. mostrar que las soluciones a los subproblemas del problema con resolución óptima son óptimas también o se presenta contradicción.

Programación dinámica: **elementos**

Programación dinámica: **elementos**

- La subestructura óptima varía de dos maneras:

Programación dinámica: **elementos**

- La subestructura óptima varía de dos maneras:
  - de acuerdo a cuántos subproblemas se usan en la solución óptima al problema original y,

Programación dinámica: **elementos**

- La subestructura óptima varía de dos maneras:
  - de acuerdo a cuántos subproblemas se usan en la solución óptima al problema original y,
  - de acuerdo a cuántas elecciones tenemos para determinar cuáles subproblemas usar en la solución óptima.

Programación dinámica: **elementos**

- La subestructura óptima varía de dos maneras:
  - de acuerdo a cuántos subproblemas se usan en la solución óptima al problema original y,
  - de acuerdo a cuántas elecciones tenemos para determinar cuáles subproblemas usar en la solución óptima.
- Linea de ensamblado: 1 subproblema, 2 elecciones (para ***F(j)***, resolver ***F(j-1)***, y decidir si llegar a ***j*** por arriba o por abajo)

Programación dinámica: **elementos**

- La subestructura óptima varía de dos maneras:
  - de acuerdo a cuántos subproblemas se usan en la solución óptima al problema original y,
  - de acuerdo a cuántas elecciones tenemos para determinar cuáles subproblemas usar en la solución óptima.
- Linea de ensamblado: 1 subproblema, 2 elecciones (para ***F(j)***, resolver ***F(j-1)***, y decidir si llegar a ***j*** por arriba o por abajo)
- Multiplicación de matrices: 2 subproblemas, ***j-i*** elecciones (para resolver ***m[i,j]***, resolver ***m[i,k]*** y ***m[k+1,j]***, con ***k = i,..,j-1***).

Programación dinámica: **elementos**

- La subestructura óptima varía de dos maneras:
  - de acuerdo a cuántos subproblemas se usan en la solución óptima al problema original y,
  - de acuerdo a cuántas elecciones tenemos para determinar cuáles subproblemas usar en la solución óptima.
- Linea de ensamblado: 1 subproblema, 2 elecciones (para ***F(j)***, resolver ***F(j-1)***, y decidir si llegar a ***j*** por arriba o por abajo)
- Multiplicación de matrices: 2 subproblemas, ***j-i*** elecciones (para resolver ***m[i,j]***, resolver ***m[i,k]*** y ***m[k+1,j]***, con ***k = i,..,j-1***).
- Bolsa del Ladrón: 1 subproblema, ***N*** elecciones (para resolver ***knapsack[c]***, resolver ***knapsack[c-size(O[i])]*** , con ***i*** = 1,...***N***).

Programación dinámica: **elementos**

Programación dinámica: **elementos**

- De manera informal, el tiempo de ejecución de un algoritmo de programación dinámica depende del producto de dos factores:

Programación dinámica: **elementos**

- De manera informal, el tiempo de ejecución de un algoritmo de programación dinámica depende del producto de dos factores:
  - el número total de subproblemas y

Programación dinámica: **elementos**

- De manera informal, el tiempo de ejecución de un algoritmo de programación dinámica depende del producto de dos factores:
  - el número total de subproblemas y
  - el número de opciones a revisar para cada subproblema.

Programación dinámica: **elementos**

- De manera informal, el tiempo de ejecución de un algoritmo de programación dinámica depende del producto de dos factores:
  - el número total de subproblemas y
  - el número de opciones a revisar para cada subproblema.
- Línea de ensamblado: Θ(n) subproblemas totales y 2 opciones que evaluar en cada uno = Θ(n).

Programación dinámica: **elementos**

- De manera informal, el tiempo de ejecución de un algoritmo de programación dinámica depende del producto de dos factores:
  - el número total de subproblemas y
  - el número de opciones a revisar para cada subproblema.

- Línea de ensamblado: $\Theta(n)$ subproblemas totales y 2 opciones que evaluar en cada uno = $\Theta(n)$.
- Multiplicación de matrices: $\Theta(n^2)$ subproblemas totales y a lo más n-1 opciones = $\Theta(n^3)$.

Programación dinámica: **elementos**

- De manera informal, el tiempo de ejecución de un algoritmo de programación dinámica depende del producto de dos factores:
  - el número total de subproblemas y
  - el número de opciones a revisar para cada subproblema.

- Línea de ensamblado: $\Theta(n)$ subproblemas totales y 2 opciones que evaluar en cada uno = $\Theta(n)$.
- Multiplicación de matrices: $\Theta(n^2)$ subproblemas totales y a lo más n-1 opciones = $\Theta(n^3)$.
- Bolsa del ladrón: numero de subproblemas totales depende de los tamaños de los objetos y la bolsa, a lo más ***n*** opciones por sub-problema.