Universidad Nacional de Ingeniería
UNI-Norte

III Unidad
Programación Dinámica

Investigación de Operaciones II

Maestro
Ing. Julio Rito Vargas Avilés

I semestre 2009

Método general

La **programación dinámica** se suele utilizar en problemas de **optimización**, donde una solución está formada por una serie de decisiones.

Igual que la técnica **divide y vencerás**, resuelve el problema original combinando las soluciones para subproblemas más pequeños.

Sin embargo, la programación dinámica no utiliza recursividad, sino que almacena los resultados de los subproblemas en una **tabla**, calculando primero las soluciones para los problemas pequeños.

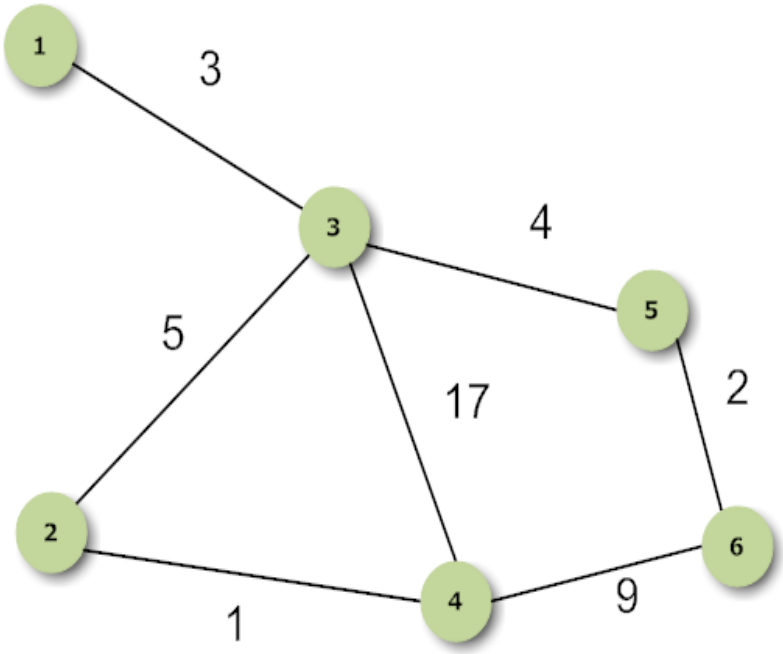
Con esto se pretende evitar la repetición de cálculos para problemas más pequeños.

- **La Programación Dinámica (PD) intenta mejorar la eficiencia del cálculo de problemas descomponiéndolos en subproblemas de menor tamaño, más fáciles de de resolver.**

- **La PD está basada en el principio de Optimalidad de Bellman:**

"Cualquier subsecuencia de decisiones de una secuencia óptima de decisiones que resuelve un problema, también debe ser óptima respecto al suproblema resuelto."

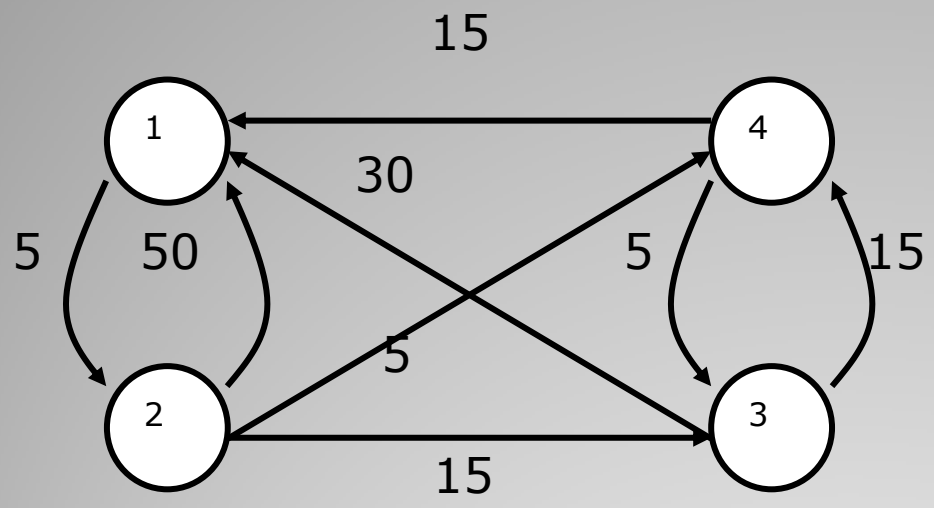
- **La PD resuelve el problema en etapas (problemas multietápicos).**
- **En cada etapa interviene una variable de optimización.**
- **Los cálculos de las diferentes etapas se enlazan de forma recursiva para generar la solución óptima.**
- **La PD se aplica en problemas como calendarización (scheduling), edición de cadenas, almacenamiento e inventario.**



	1	2	3	4	5	6
1	0	8	3	9	7	9
2	8	0	5	1	9	10
3	3	5	0	6	4	6
4	9	1	6	0	10	9
5	7	9	4	10	0	2
6	9	10	6	9	2	0

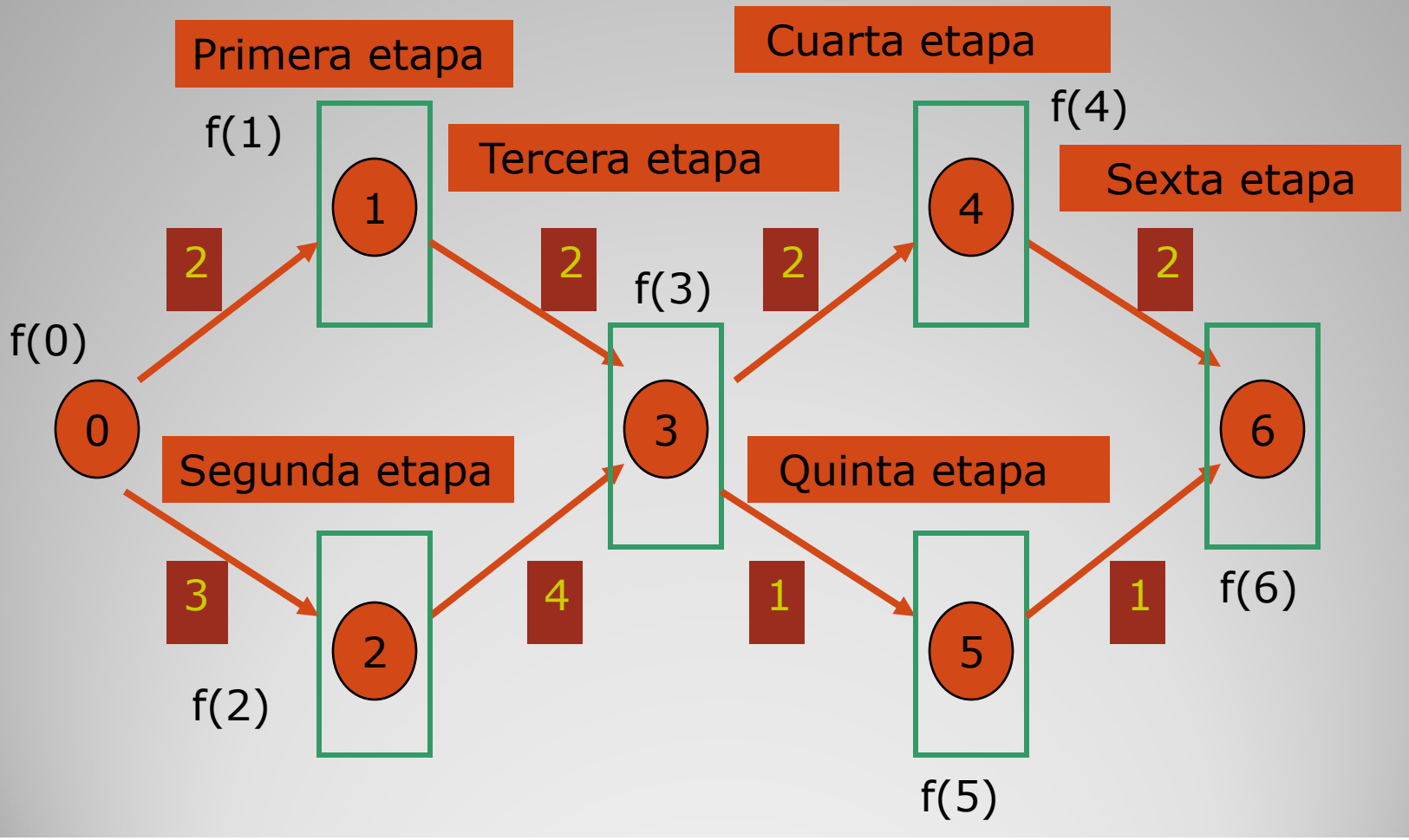
Vamos a analizar la técnica de programación dinámica a través de un ejemplo.

Calcular los caminos más cortos entre todos los pares de nodos en un grafo (red) no dirigido



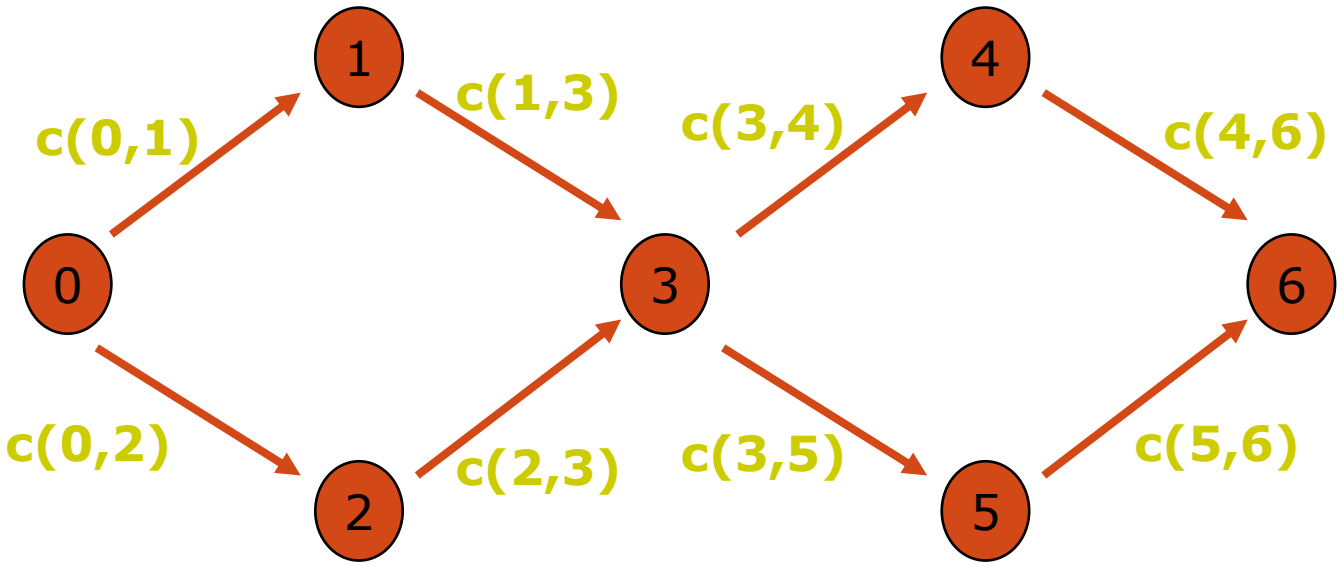
D	1	2	3	4
1	0	5	15	10
2	20	0	10	5
3	30	35	0	15
4	15	20	5	0

EJEMPLO



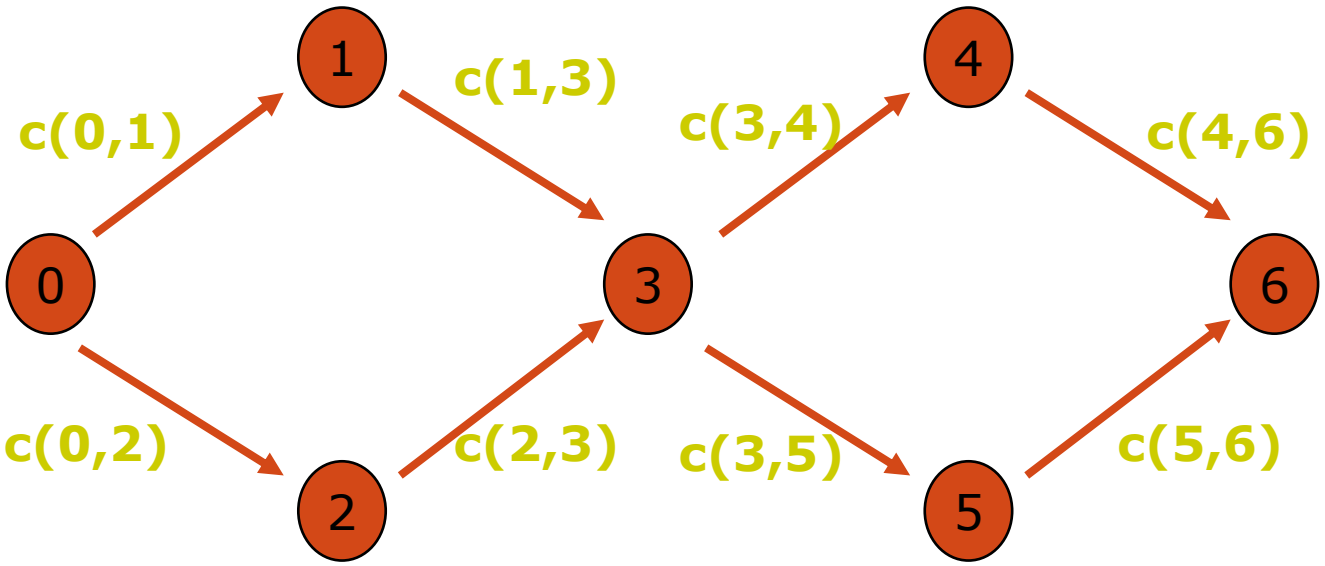
¿Cual es el camino más corto desde 0 a 6?

FORMULACION DEL PROBLEMA



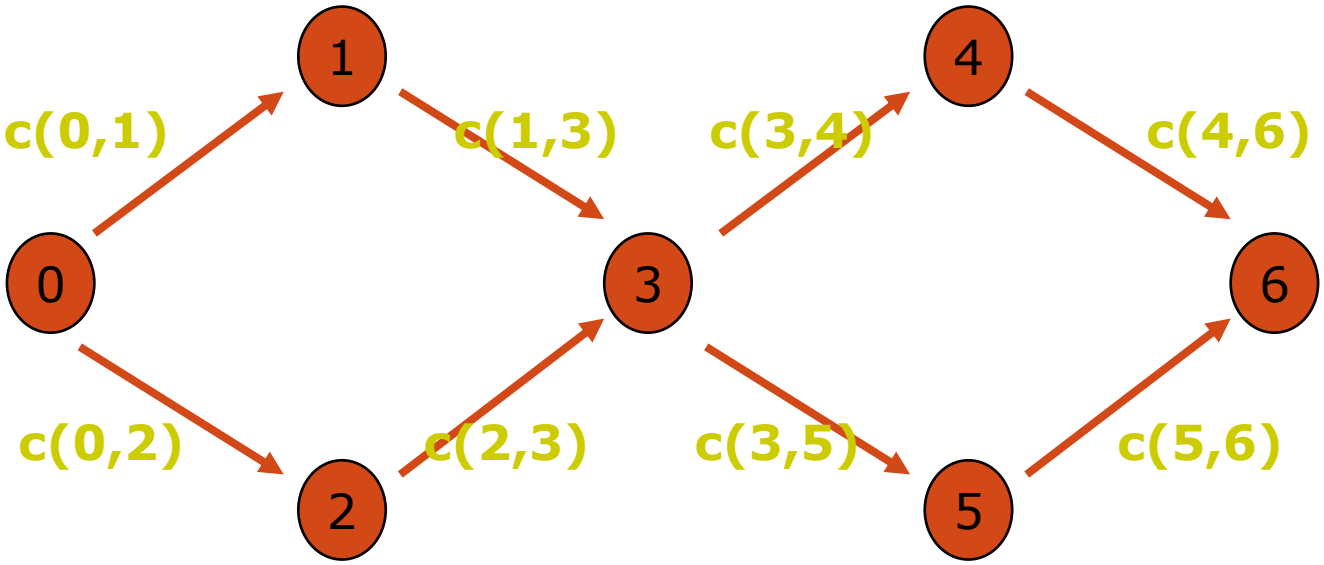
0 = inicio, n−1 = meta, 0 < x <= n−1 nodos intermedios
Si ∃ camino entonces c(i,j) = valor ∀ i < j
Si ∄ camino entonces c(i,j) = ∞ ∀ i < j
f(x) denota el camino más corto desde 0 al nodo x
f(n−1) es la solución del problema

FORMULACION DEL PROBLEMA



f(0) = 0
f(1) = min { f(0) + c(0,1)}=2
f(2) = min { f(1) + c(1,2) , f(0) + c(0,2)}=3
f(3) = min { f(2) + c(2,3) , f(1) + c(1,3) , f(0) + c(0,3)}=4
f(4) = min { f(3) + c(3,4) , f(1) + c(1,4)}=6
f(5)= min { f(3) + c(3,5) , f(2) + c(2,5),f(4)+ c(4,5)}=5
f(6) = min {f(5)+ c(5,6), f(4) + c(4,6), f(3) + c(3,6)}=6

FORMULACION DEL PROBLEMA



$$f(x) = \begin{cases} 0 \quad \text{si } x = 0 \\ \min\limits_{0 \le j < x} \{f(j) + c(j,x)\} & 1 \le x \le n-1 \end{cases}$$

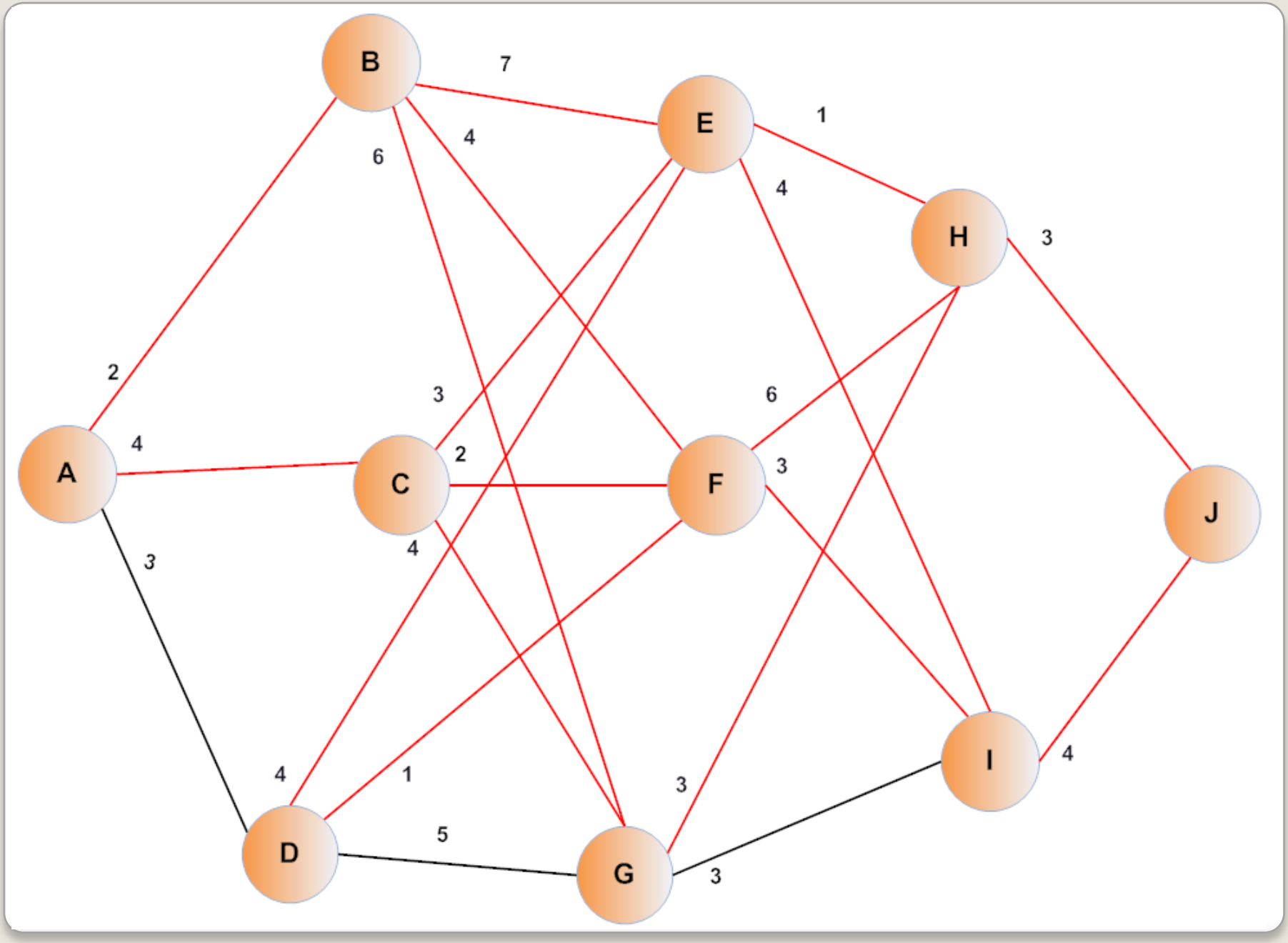
El problema de la diligencia:

El problema de la diligencia se construyó especialmente para ilustrar las características e introducir la terminología de la programación dinámica. Este paradigma se refiere a un cazafortunas mítico de Missouri que quiere ir al oeste a sumergirse en la fiebre del oro que surgió en California a mediados del siglo XIX. Tiene que hacer un viaje en diligencia a través de territorios sin ley, donde existen serios peligros de ser atacado por merodeadores. Aún cuando su punto de partida y su destino son fijos, tiene muchas opciones en cuanto a que estados o territorios debe elegir como puntos intermedios. En la gráfica siguiente se muestran las rutas posibles, en donde cada estado se presenta mediante un círculo con una letra; además, en el diagrama, la dirección del viaje es siempre de izquierda a derecha. Como se puede observar, se requieren cuatro etapas - jornadas en diligencias para viajar desde un punto de partida en el estado A (Missouri) a su destino en el estado J (California). Este cazafortunas es un hombre prudente preocupado por su seguridad. Después de reflexionar un poco ideó una manera bastante ingeniosa para determinar la ruta más segura.

El problema de la diligencia:

Se ofrecen pólizas de seguro de vida a los pasajeros. Como el costo de la póliza de cualquier jornada en la diligencia está basada en una evaluación cuidadosa de la seguridad del recorrido, la ruta más segura debe ser aquella cuya póliza de seguro tenga el menor costo total. El costo de la póliza estándar del viaje, del estado i al estado j, que se denota como c(i,j) es:

Cual es la ruta que minimiza el costo total?


B
E
H
A
C
F
J
D
G
I
7
1
6
4
4
3
2
3
6
4
2
3
4
3
4
1
3
5
3
4

Usaremos el Software WinQB para resolver el problema de la Diligencia.

Para lo cual procedemos de la siguiente manera:

1. Hacemos clic en la opción Dynamic Programming
2. En el menú de Dynamic Programming, hacemos clic en **File** (archivo) y luego en **New Problem** (nuevo problema)**.**
3. Stagecoach(Shortest Routes) Problem – así mismo ponga el título del problema e ingrese el número de nodos que tiene el problema.
4. Al hacer clic en ok, se le mostrará una matriz 10x10, cambie los nombres de los nodos (menú edit). (A,B,C,…J)
5. Introduzca los datos de costo de la póliza, por cada arco.
6. Para obtener la solución hacer clic en el menú Solve and analyze. en la opción solve and display steps.
7. Para obtener la solución resumén, haga clic en el submenú de Results (Show solution summary)

Solution Steps for Problema Diligencia: Stagecoach-Shortest Route Problem

	Etapa Stage	Del nodo From Input State	al nodo To Output State	Distancia Distance	Distancia a J Distance to J	Status
1	**1**	**A**	**C**	**4**	11	Optimal
2	2	B	E	7	11	
3	**2**	**C**	**E**	**3**	7	Optimal
4	2	D	E	4	8	
5	**3**	**E**	**H**	**1**	4	Optimal
6	3	F	I	3	7	
7	3	G	H	3	6	
8	**4**	**H**	**J**	**3**	3	Optimal
9	4	I	J	4	4	

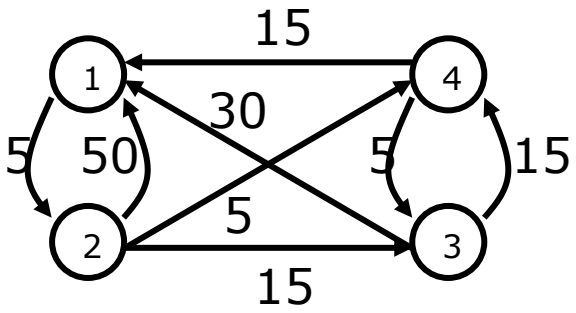
From A To J Minimum Distance = 11

Stage	From Input State	To Output State	Distance	Cumulative Distance	Distance to J
1	A	C	4	4	11
2	C	E	3	7	7
3	E	H	1	8	4
4	H	J	3	11	3

From A To J Min. Distance = 11

Ejemplo Algoritmo Floyd

```
Funcion Floyd(L[1..n,1..n]):matriz[1..n,1..n]
 matriz D[1..n,1..n]
 D=L
P=0
 para k=1 hasta n hacer
        para i=1 hasta n hacer
            para j=1 hasta n hacer
                si D[i,k]+D[k,j]<D[i,j] entonces
                        D[i,j]=D[i,k]+D[k,j]
                        P[i,j]=k
                 finsi
                        finpara
        finpara
 finpara
 Devolver D
```



D	1	2	3	4
1	0	5	15	10
2	20	0	10	5
3	30	35	0	15
4	15	20	5	0

k	i	j
4	2	3

D[2,4]+D[4,3]<D[2,3] ?
5+5 < 15
D[2,3]= D[2,4]+D[4,3]
P[4,2]=4

P	1	2	3	4
1	0	0	4	2
2	4	0	4	0
3	0	1	0	0
4	0	1	0	0

RESUELA ENCONTRANDO LOS CAMINOS MAS CORTOS, USANDO PROGRAMACIÓN DINÁMICA