Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Tema 14. Esquema Programación Dinámica

Algorítmica y Complejidad

1

Introducción

Problemas y Subproblemas

Objetivo 1 **Estado Inicial 1**

Problema

*Buscar las "**mejores decisiones**" para conseguir un **objetivo** a partir de un **estado inicial***

Tomamos una decisión

Objetivo 2 **Estado Inicial 2**

Problema

Este problema es más fácil de resolver que el anterior

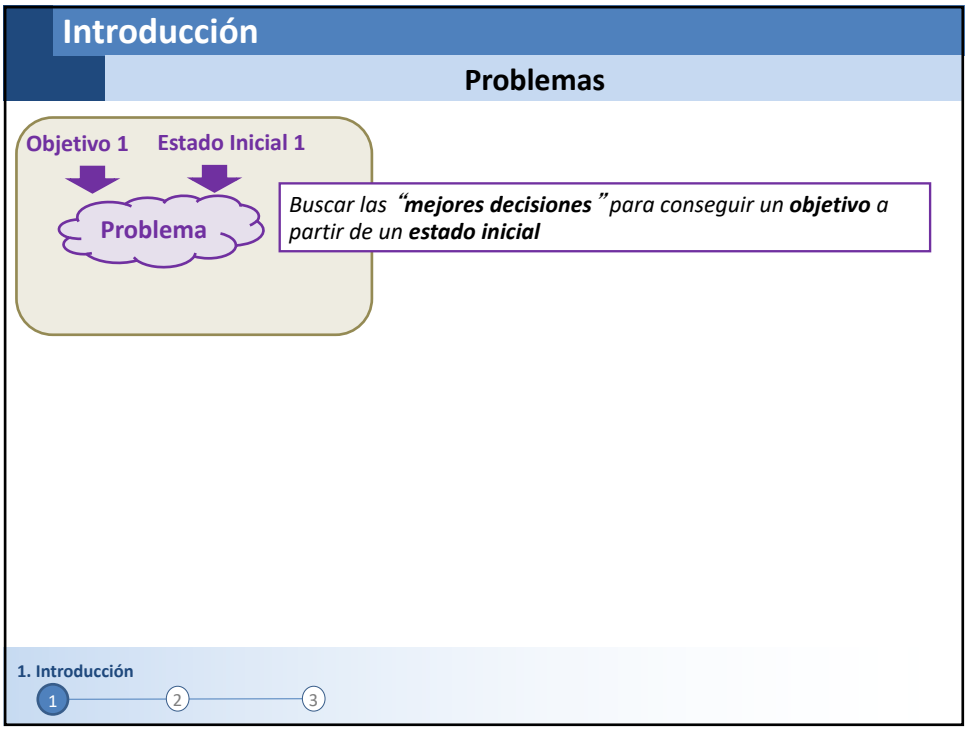
...

Objetivo *N* **Estado Inicial *N***

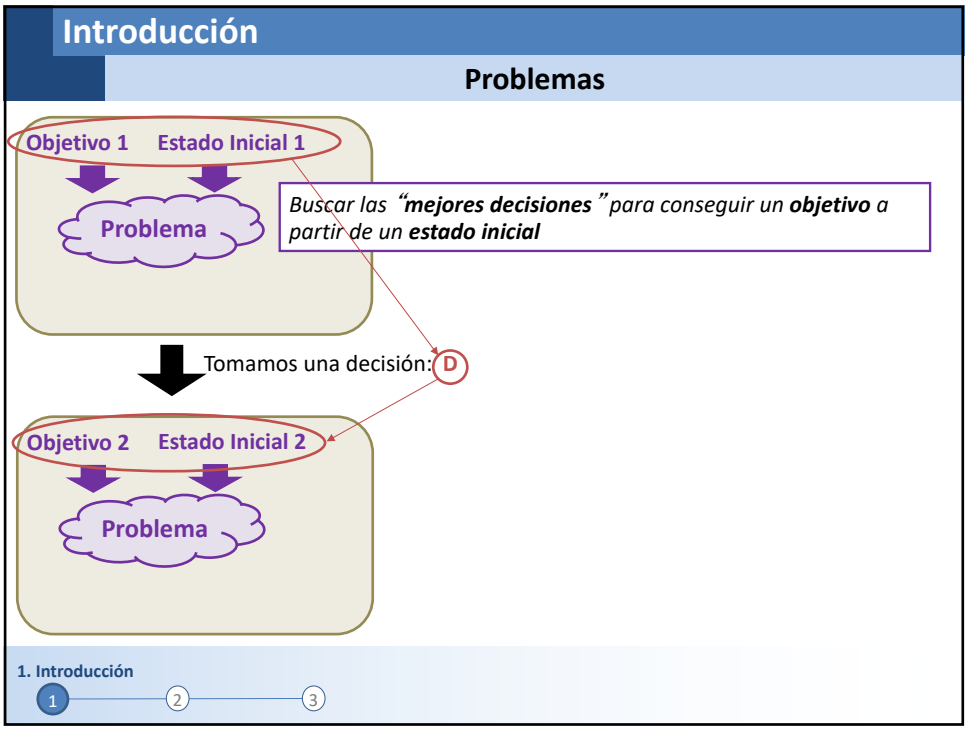
Problema

*Este problema es **inmediato** de resolver*

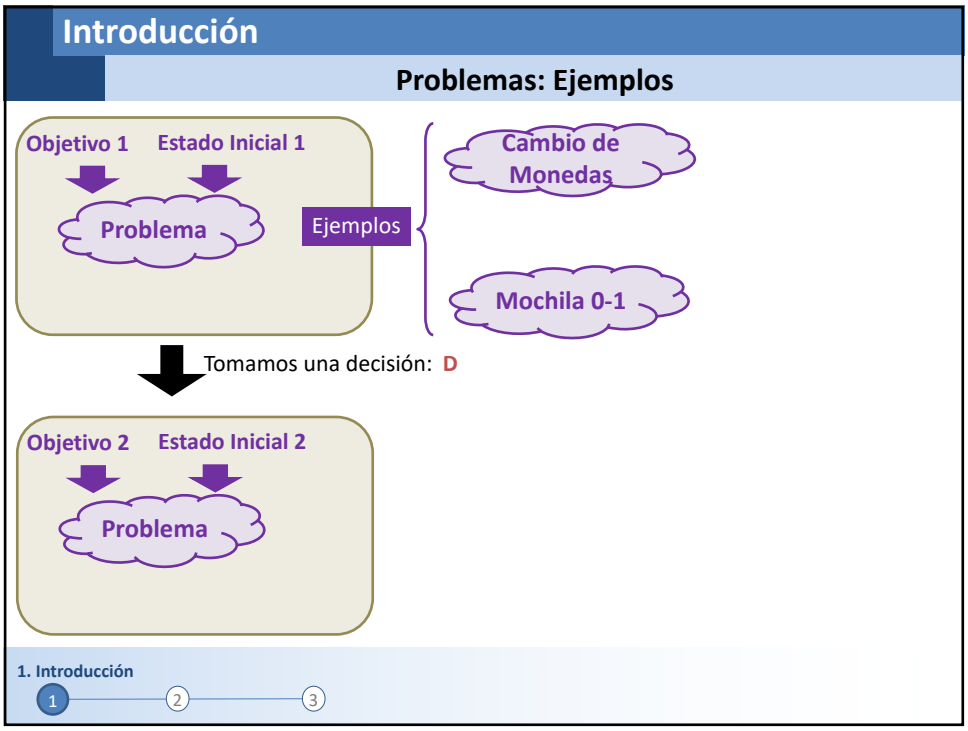
2


Introducción
Problemas
Objetivo 1
Estado Inicial 1
Problema
Buscar las "mejores decisiones" para conseguir un objetivo a partir de un estado inicial
1. Introducción
1
2
3

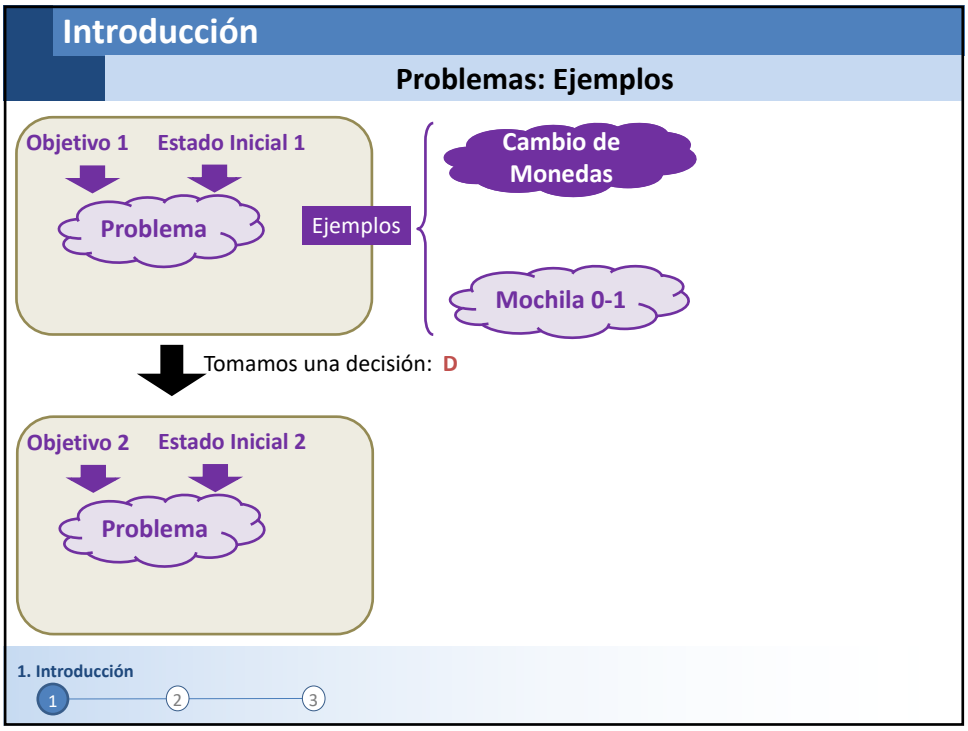
3


Introducción
Problemas
Objetivo 1
Estado Inicial 1
Problema
Buscar las "mejores decisiones" para conseguir un objetivo a partir de un estado inicial
Tomamos una decisión:
D
Objetivo 2
Estado Inicial 2
Problema
1. Introducción
1
2
3

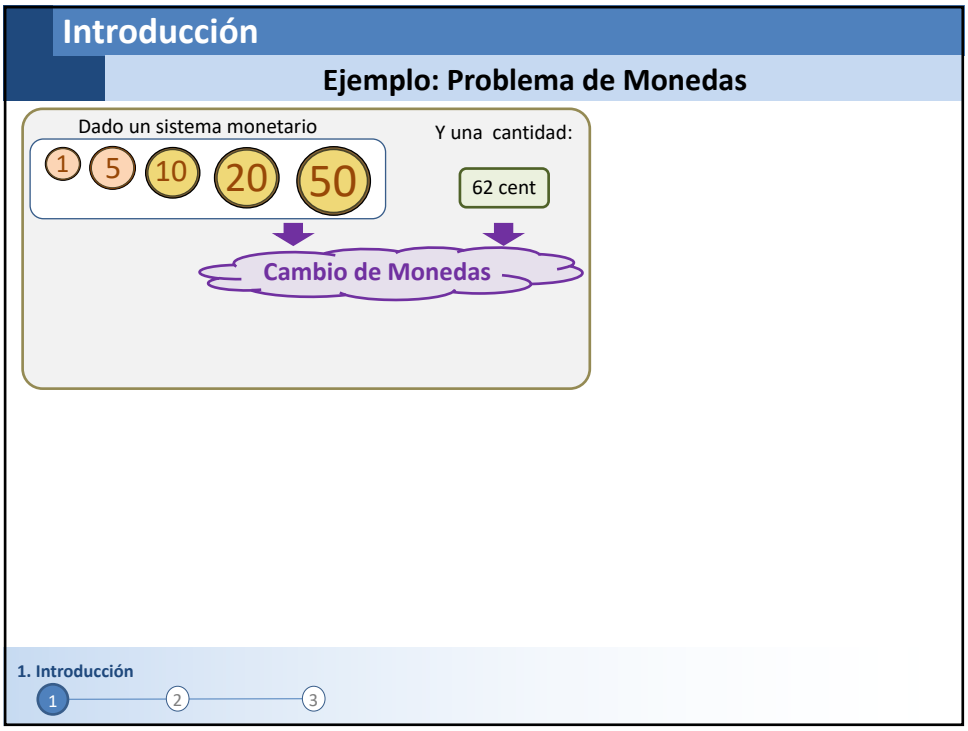
4


Introducción
Problemas: Ejemplos
Objetivo 1
Estado Inicial 1
Problema
Ejemplos
Cambio de Monedas
Mochila 0-1
Tomamos una decisión: D
Objetivo 2
Estado Inicial 2
Problema
1. Introducción
1
2
3

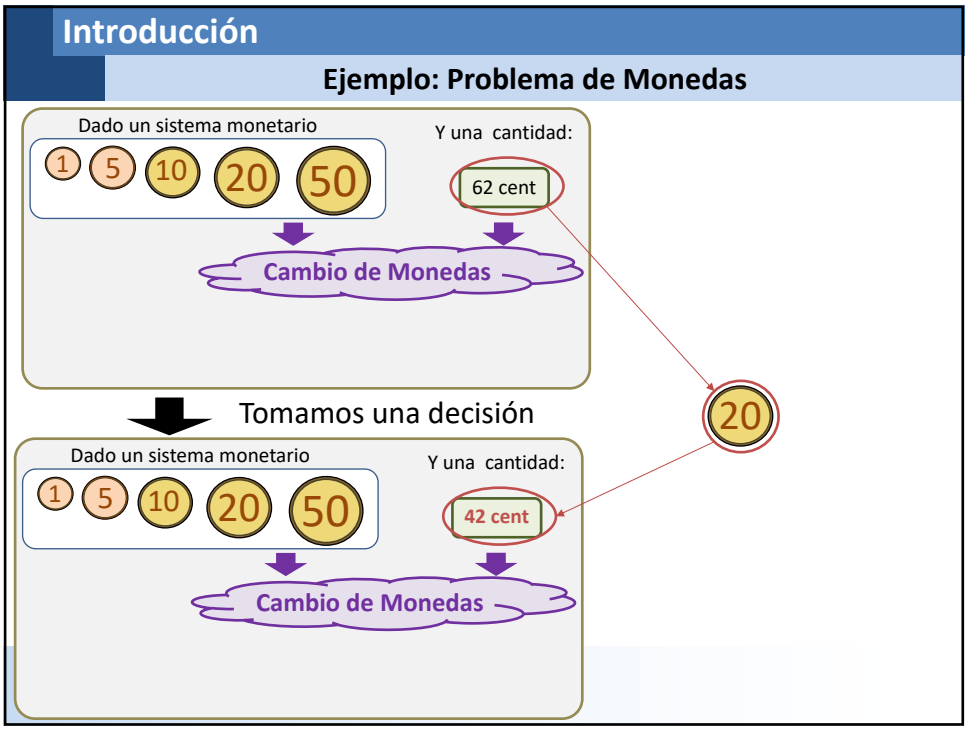
5


Introducción
Problemas: Ejemplos
Objetivo 1
Estado Inicial 1
Problema
Ejemplos
Cambio de Monedas
Mochila 0-1
Tomamos una decisión: D
Objetivo 2
Estado Inicial 2
Problema
1. Introducción
1
2
3

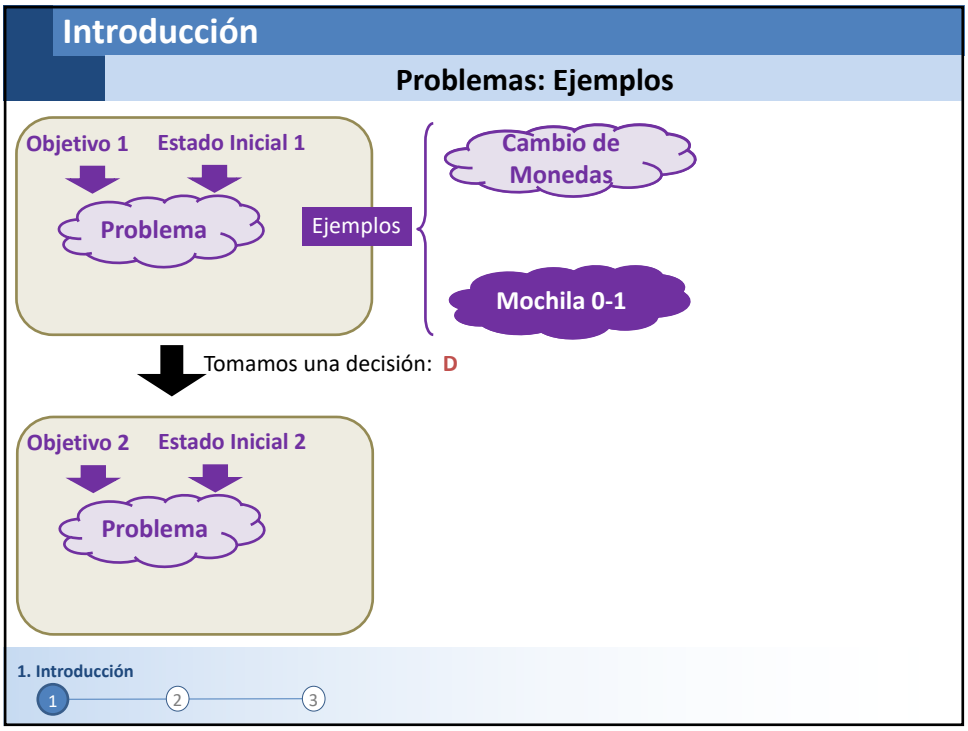
6


Introducción
Ejemplo: Problema de Monedas
Dado un sistema monetario
1
5
10
20
50
Y una cantidad:
62 cent
Cambio de Monedas
1. Introducción
1
2
3

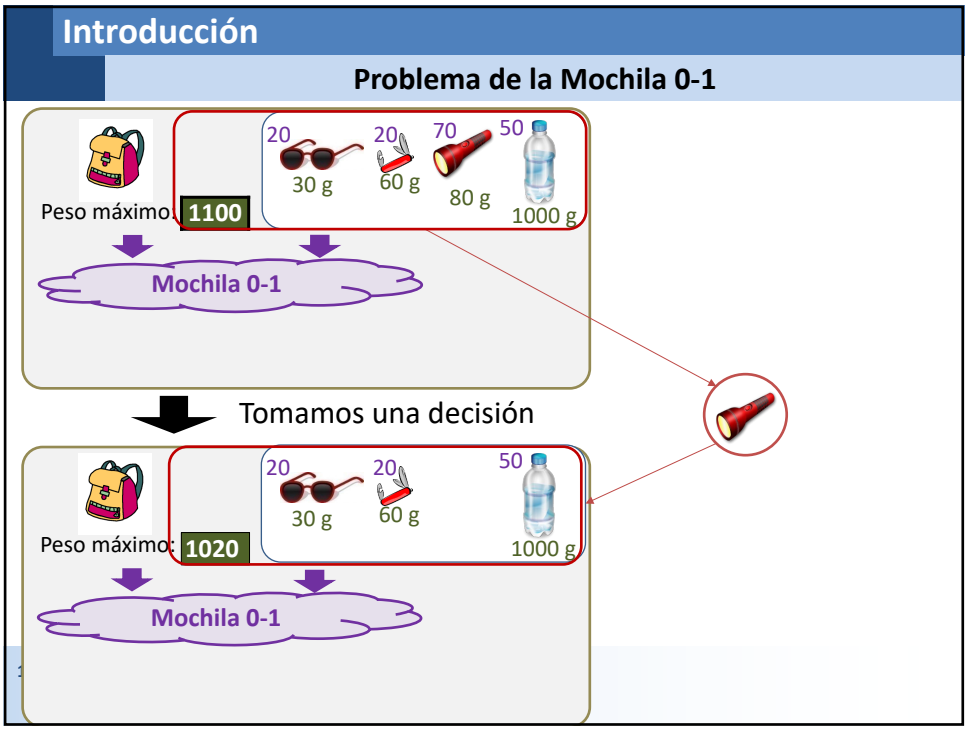
7


Introducción
Ejemplo: Problema de Monedas
Dado un sistema monetario
1
5
10
20
50
Y una cantidad:
62 cent
Cambio de Monedas
Tomamos una decisión
20
Dado un sistema monetario
1
5
10
20
50
Y una cantidad:
42 cent
Cambio de Monedas

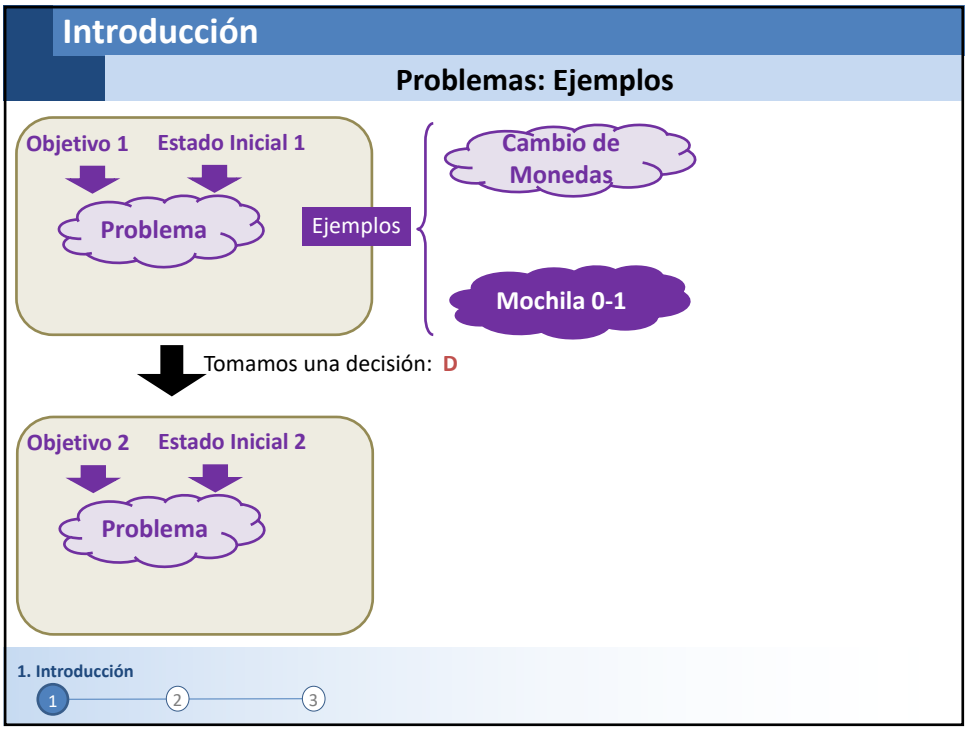
8


Introducción
Problemas: Ejemplos
Objetivo 1
Estado Inicial 1
Problema
Ejemplos
Cambio de Monedas
Mochila 0-1
Tomamos una decisión: D
Objetivo 2
Estado Inicial 2
Problema
1. Introducción

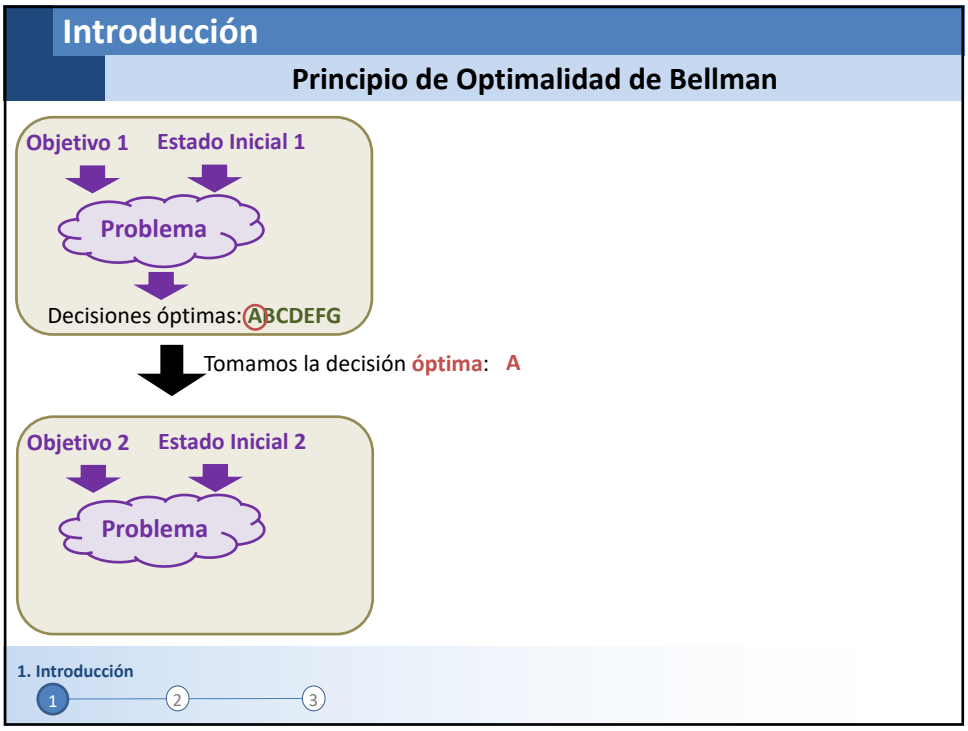
9


Introducción
Problema de la Mochila 0-1
Peso máximo: 1100
20
30 g
20
60 g
70
80 g
50
1000 g
Mochila 0-1
Tomamos una decisión
Peso máximo: 1020
20
30 g
20
60 g
50
1000 g
Mochila 0-1

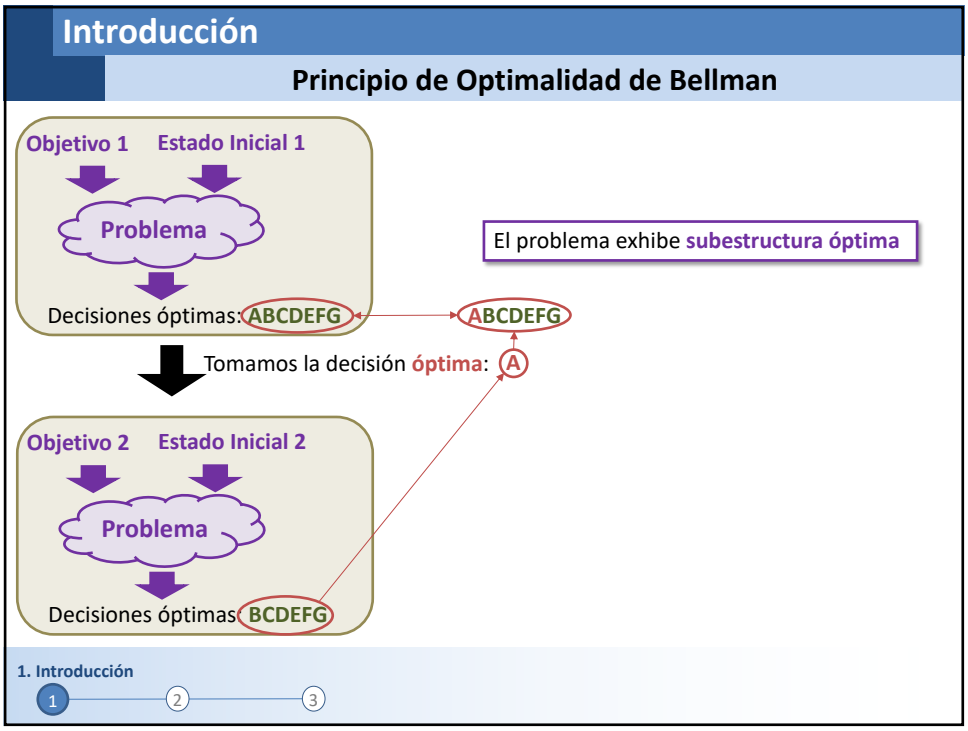
10


Introducción
Problemas: Ejemplos
Objetivo 1
Estado Inicial 1
Problema
Ejemplos
Cambio de Monedas
Mochila 0-1
Tomamos una decisión: D
Objetivo 2
Estado Inicial 2
Problema
1. Introducción
1
2
3

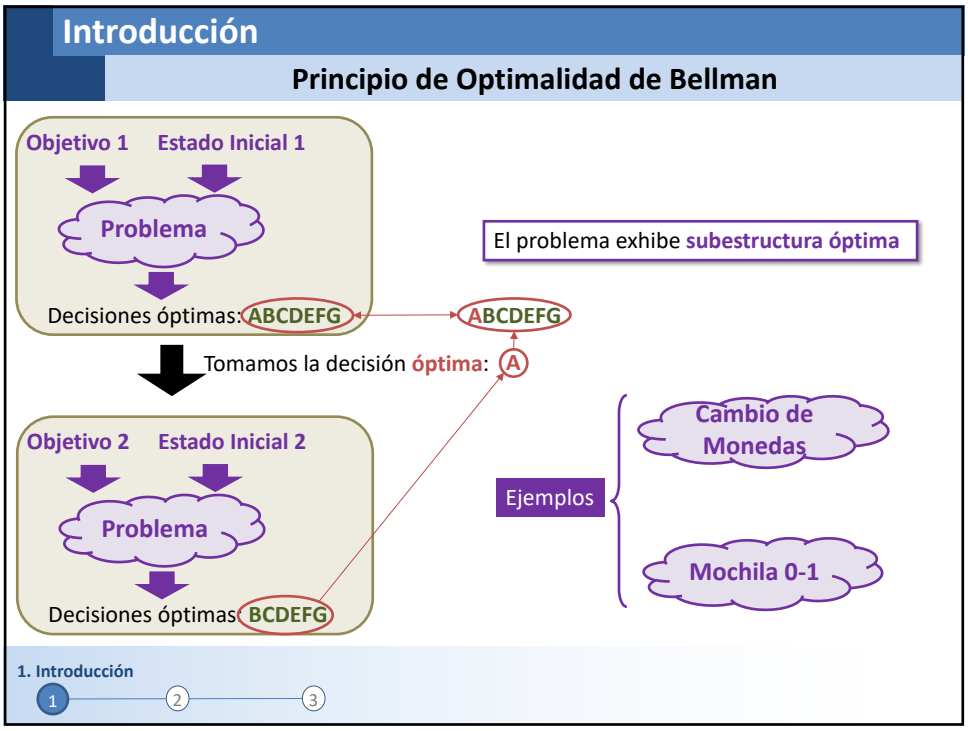
11


Introducción
Principio de Optimalidad de Bellman
Objetivo 1
Estado Inicial 1
Problema
Decisiones óptimas: ABCDEFG
Tomamos la decisión óptima: A
Objetivo 2
Estado Inicial 2
Problema
1. Introducción
1
2
3

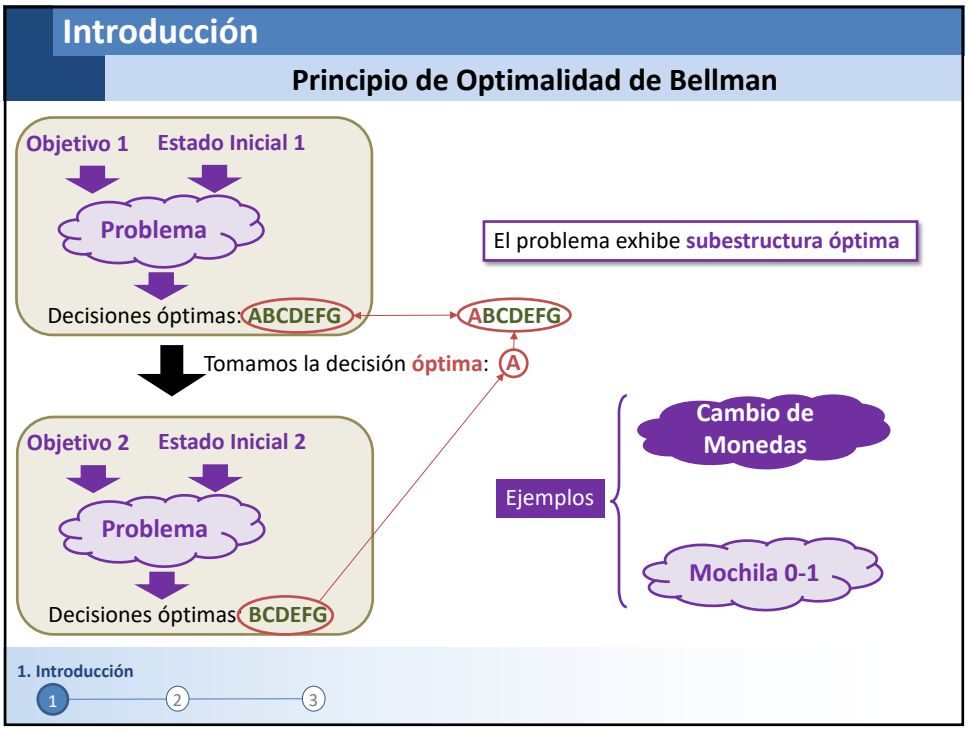
12


Introducción
Principio de Optimalidad de Bellman
Objetivo 1
Estado Inicial 1
Problema
Decisiones óptimas: ABCDEFG
El problema exhibe subestructura óptima
ABCDEFG
Tomamos la decisión óptima: A
Objetivo 2
Estado Inicial 2
Problema
Decisiones óptimas: BCDEFG
1. Introducción
1
2
3

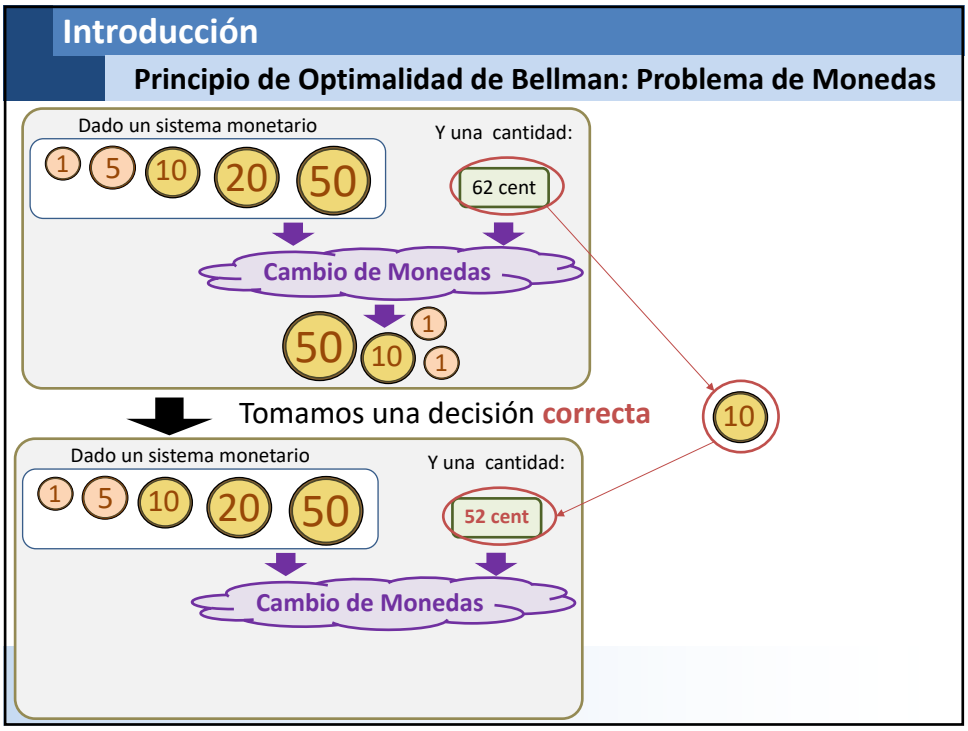
13


Introducción
Principio de Optimalidad de Bellman
Objetivo 1
Estado Inicial 1
Problema
Decisiones óptimas: ABCDEFG
El problema exhibe subestructura óptima
ABCDEFG
Tomamos la decisión óptima: A
Objetivo 2
Estado Inicial 2
Problema
Decisiones óptimas: BCDEFG
Ejemplos
Cambio de Monedas
Mochila 0-1
1. Introducción
1
2
3

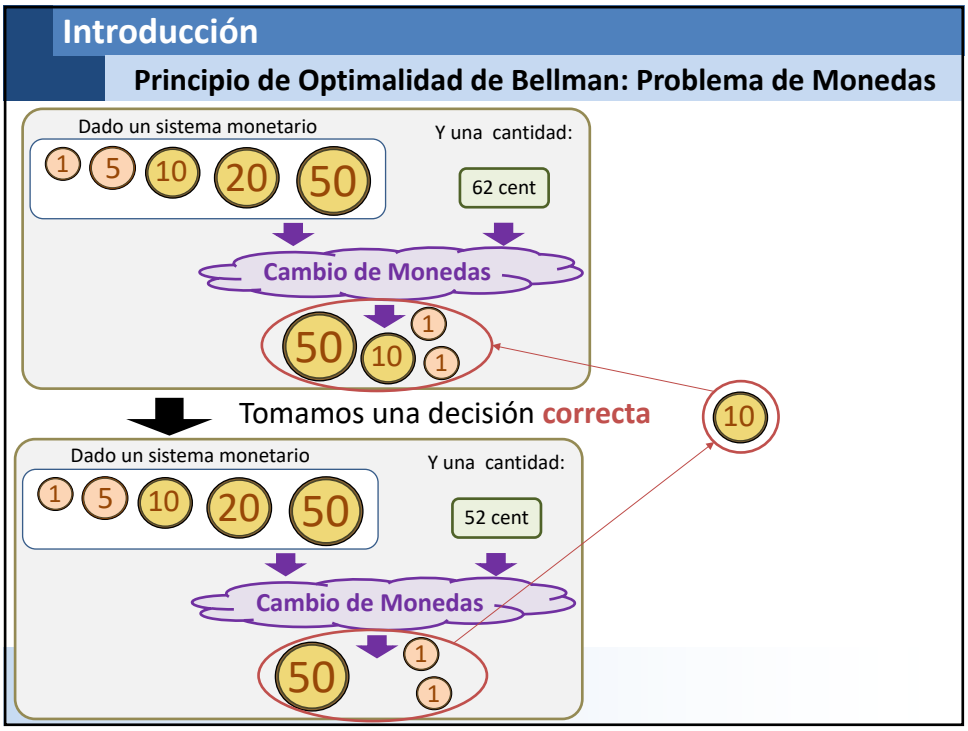
14


Introducción
Principio de Optimalidad de Bellman
Objetivo 1
Estado Inicial 1
Problema
Decisiones óptimas: ABCDEFG
ABCDEFG
El problema exhibe subestructura óptima
Tomamos la decisión óptima: A
Objetivo 2
Estado Inicial 2
Problema
Decisiones óptimas: BCDEFG
Ejemplos
Cambio de Monedas
Mochila 0-1
1. Introducción
1
2
3

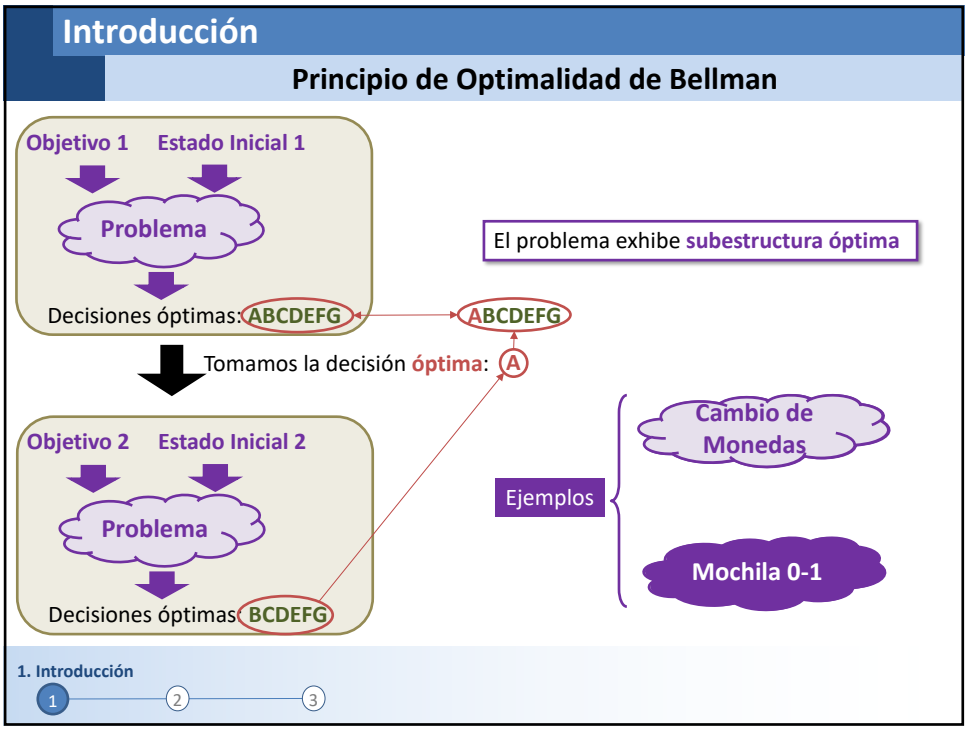
15


Introducción
Principio de Optimalidad de Bellman: Problema de Monedas
Dado un sistema monetario
1
5
10
20
50
Y una cantidad:
62 cent
Cambio de Monedas
50
10
1
1
Tomamos una decisión correcta
10
Dado un sistema monetario
1
5
10
20
50
Y una cantidad:
52 cent
Cambio de Monedas

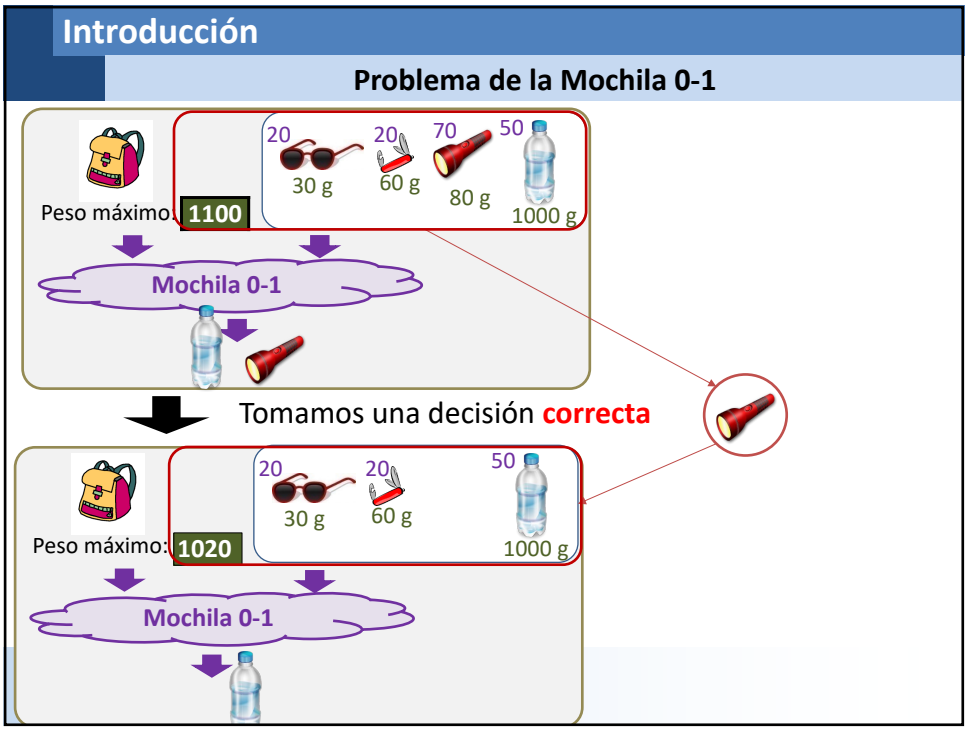
16


Introducción
Principio de Optimalidad de Bellman: Problema de Monedas
Dado un sistema monetario
1
5
10
20
50
Y una cantidad:
62 cent
Cambio de Monedas
50
10
1
1
Tomamos una decisión correcta
10
Dado un sistema monetario
1
5
10
20
50
Y una cantidad:
52 cent
Cambio de Monedas
50
1
1

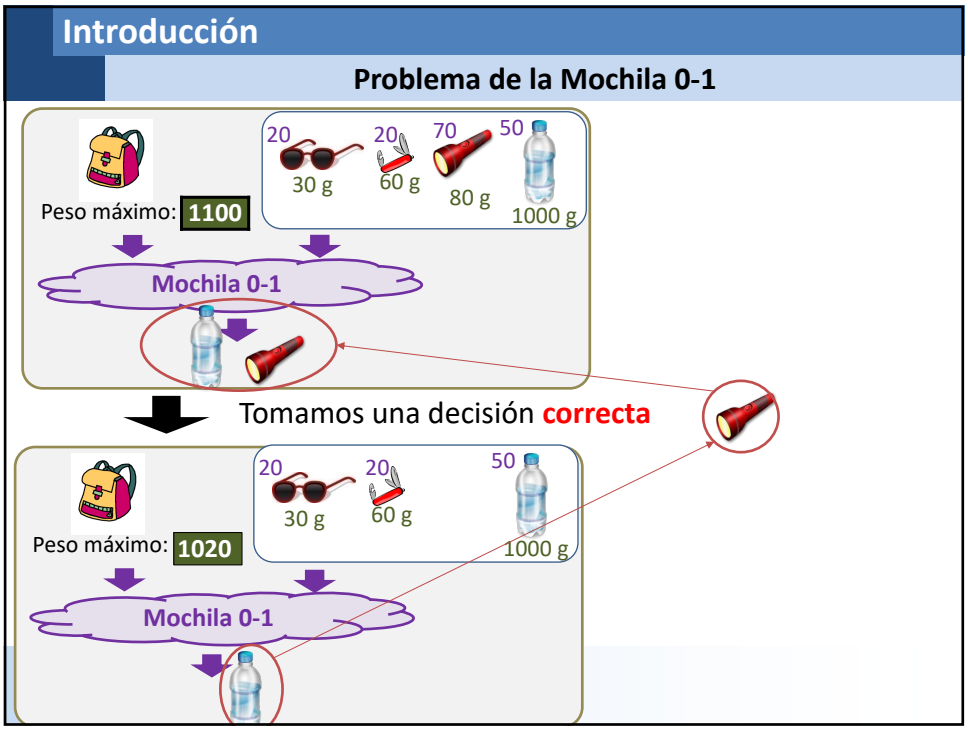
17


Introducción
Principio de Optimalidad de Bellman
Objetivo 1
Estado Inicial 1
Problema
Decisiones óptimas: ABCDEFG
ABCDEFG
El problema exhibe subestructura óptima
Tomamos la decisión óptima: A
Objetivo 2
Estado Inicial 2
Problema
Decisiones óptimas: BCDEFG
Ejemplos
Cambio de Monedas
Mochila 0-1
1. Introducción
1
2
3

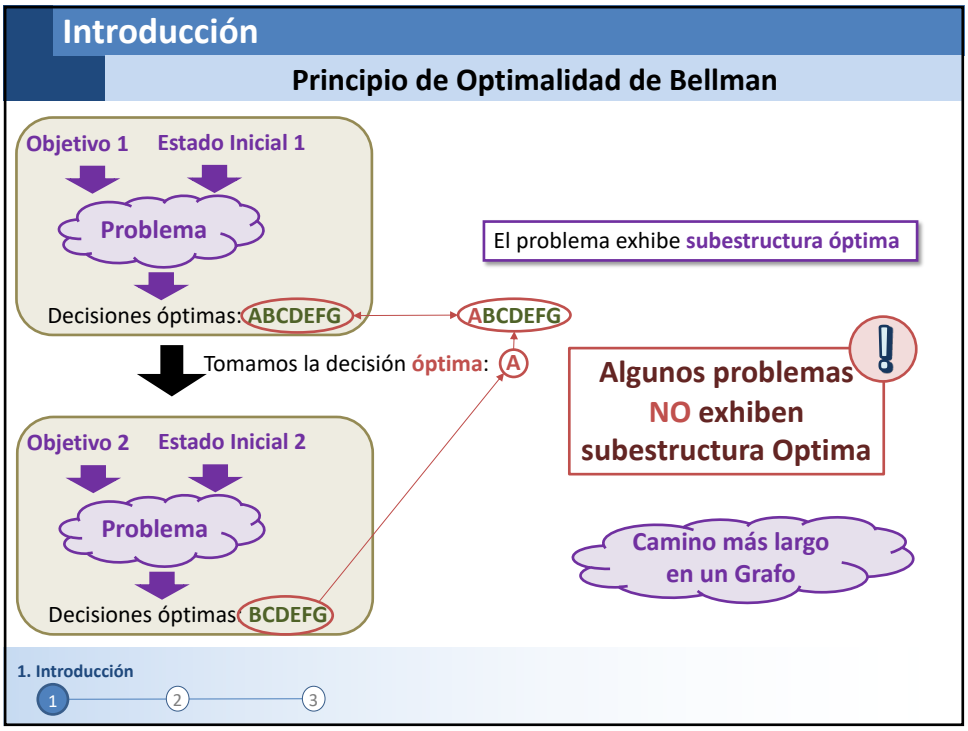
18


Introducción
Problema de la Mochila 0-1
Peso máximo: 1100
20
30 g
20
60 g
70
80 g
50
1000 g
Mochila 0-1
Tomamos una decisión correcta
Peso máximo: 1020
20
30 g
20
60 g
50
1000 g
Mochila 0-1

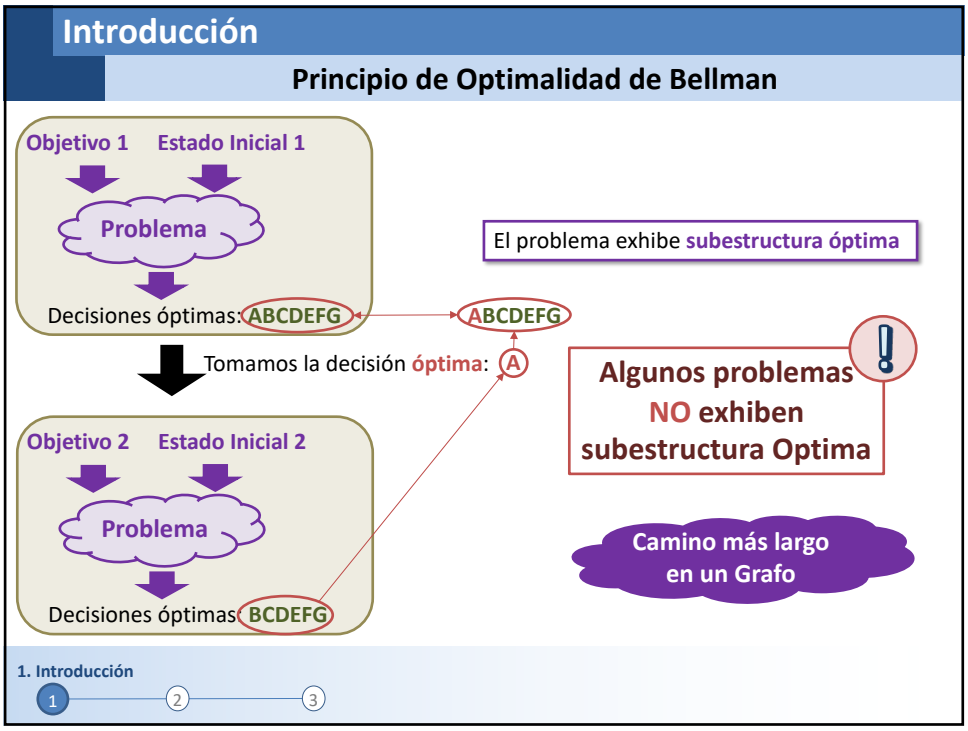
19


Introducción
Problema de la Mochila 0-1
Peso máximo: 1100
20
30 g
20
60 g
70
80 g
50
1000 g
Mochila 0-1
Tomamos una decisión correcta
Peso máximo: 1020
20
30 g
20
60 g
50
1000 g
Mochila 0-1

20


Introducción
Principio de Optimalidad de Bellman
Objetivo 1
Estado Inicial 1
Problema
Decisiones óptimas: ABCDEFG
ABCDEFG
El problema exhibe subestructura óptima
Tomamos la decisión óptima: A
Algunos problemas NO exhiben subestructura Optima
Objetivo 2
Estado Inicial 2
Problema
Decisiones óptimas: BCDEFG
Camino más largo en un Grafo
1. Introducción
1
2
3

21


Introducción
Principio de Optimalidad de Bellman
Objetivo 1
Estado Inicial 1
Problema
Decisiones óptimas: ABCDEFG
ABCDEFG
El problema exhibe subestructura óptima
Tomamos la decisión óptima: A
Algunos problemas NO exhiben subestructura Optima
Objetivo 2
Estado Inicial 2
Problema
Decisiones óptimas: BCDEFG
Camino más largo en un Grafo
1. Introducción
1
2
3

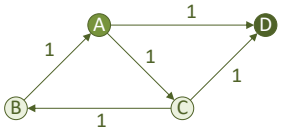
22

Introducción

Camino más largo en un Grafo

Problema

Dado un grafo y dos vértices, encontrar el camino sin ciclos más largo entre esos dos vértices.



Camino más largo "sin ciclos" entre **A** y **D**

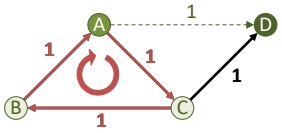
23

Introducción

Camino más largo en un Grafo

Problema

Dado un grafo y dos vértices, encontrar el camino sin ciclos más largo entre esos dos vértices.



Ciclo

Camino más largo "sin ciclos" entre **A** y **D**

¿ ACB ACB ACB .... ACB ?

Pasamos varias veces por un mismo vértice

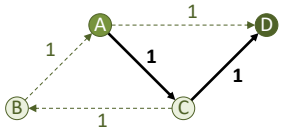
24

Introducción

Camino más largo en un Grafo

Problema

Dado un grafo y dos vértices, encontrar el camino sin ciclos más largo entre esos dos vértices.



Camino más largo "sin ciclos" entre **A** y **D**

A C D

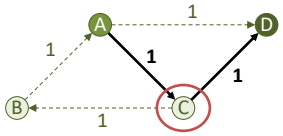
25

Introducción

Camino más largo en un Grafo

Problema

Dado un grafo y dos vértices, encontrar el camino sin ciclos más largo entre esos dos vértices.



Forma parte de la solución

Camino más largo "sin ciclos" entre **A** y **D**

A C D

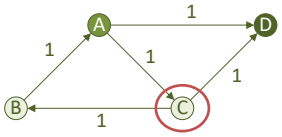
26

Introducción

Camino más largo en un Grafo

Problema

Dado un grafo y dos vértices, encontrar el camino sin ciclos más largo entre esos dos vértices.



Forma parte de la solución

Camino más largo "sin ciclos" entre **A** y **D**

A C D

¿Es **C D** el camino más largo "sin ciclos" entre **C** y **D** ?

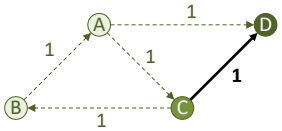
27

Introducción

Camino más largo en un Grafo

Problema

Dado un grafo y dos vértices, encontrar el camino sin ciclos más largo entre esos dos vértices.



Camino más largo "sin ciclos" entre **A** y **D**

A C D

¿Es **C D** el camino más largo "sin ciclos" entre **C** y **D** ?

C D ➡ 1

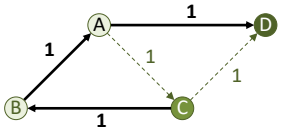
28

Introducción

Camino más largo en un Grafo

Problema

Dado un grafo y dos vértices, encontrar el camino sin ciclos más largo entre esos dos vértices.



Camino más largo "sin ciclos" entre **A** y **D**

A C D

¿Es **C D** el camino más largo "sin ciclos" entre **C** y **D** ? **NO**

C D ➡ 1
C B A D ➡ 3

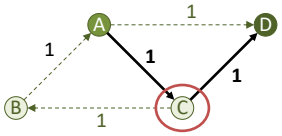
29

Introducción

Camino más largo en un Grafo

Problema

Dado un grafo y dos vértices, encontrar el camino sin ciclos más largo entre esos dos vértices.



Forma parte de la solución

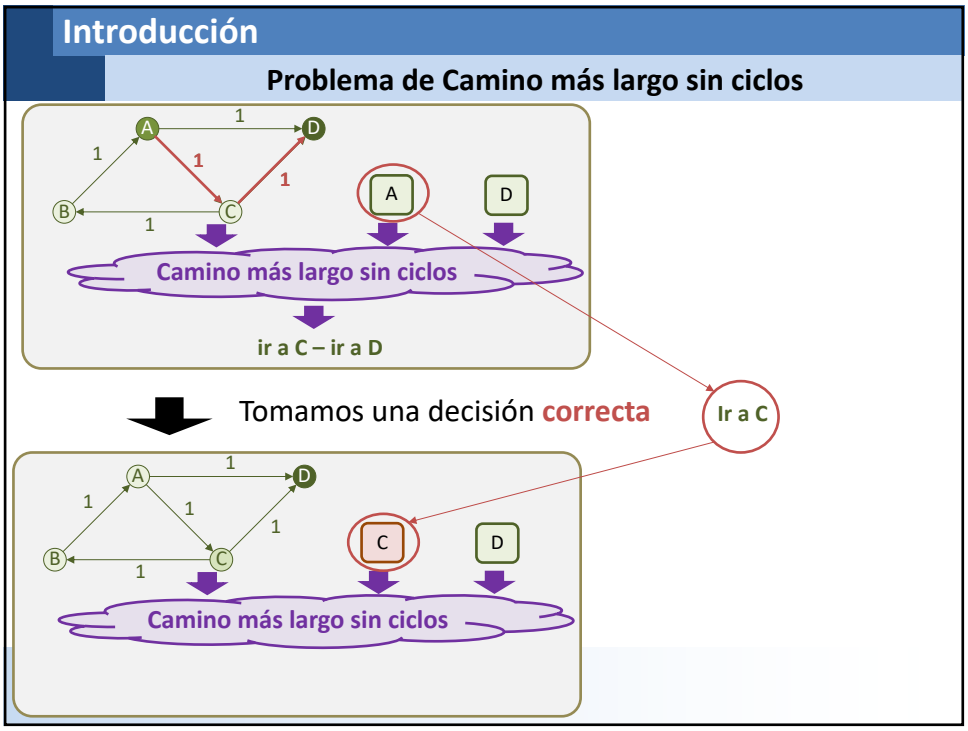
Camino más largo "sin ciclos" entre **A** y **D**

A C D

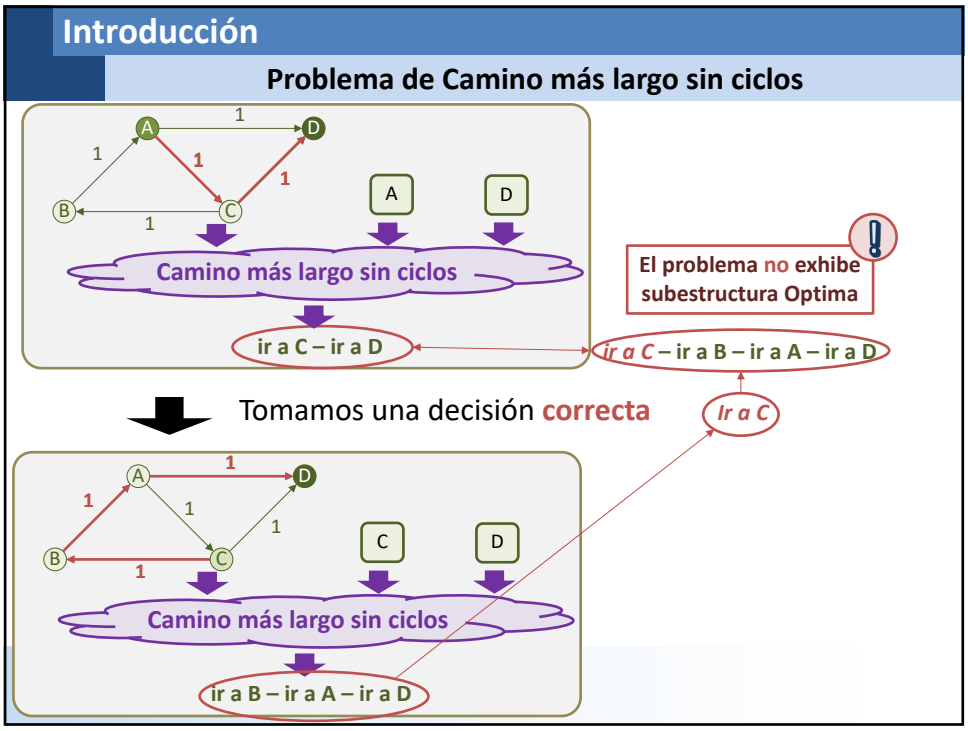
¿Es **C D** el camino más largo "sin ciclos" entre **C** y **D** ? **NO**

El problema **NO** tiene subestructura óptima

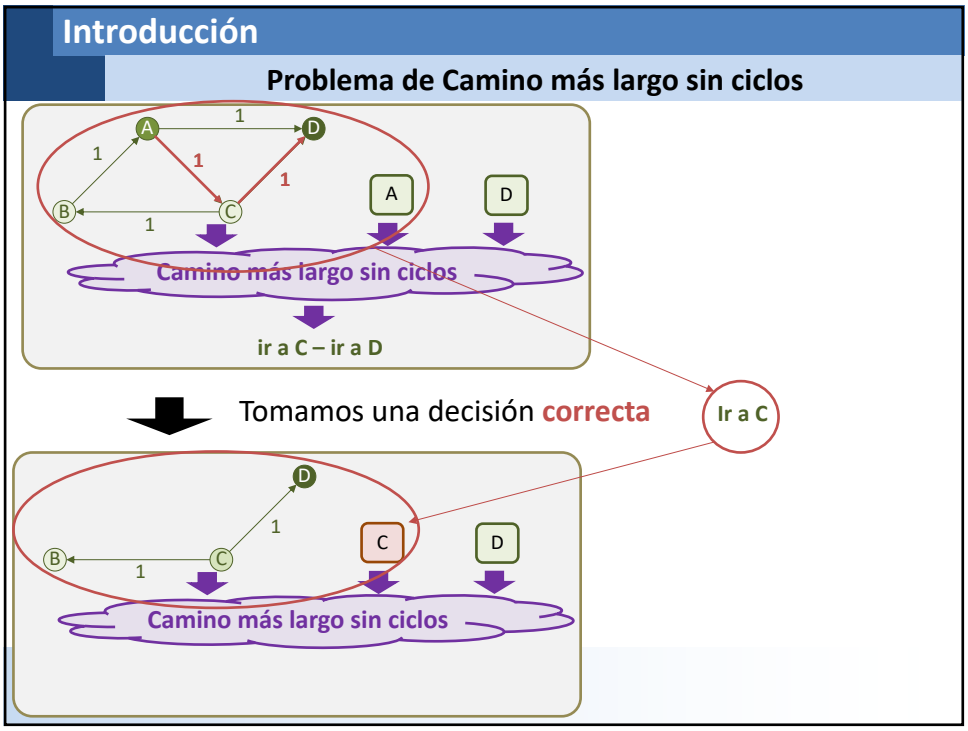
30


Introducción
Problema de Camino más largo sin ciclos
A
D
Camino más largo sin ciclos
ir a C – ir a D
Tomamos una decisión correcta
Ir a C
C
D
Camino más largo sin ciclos

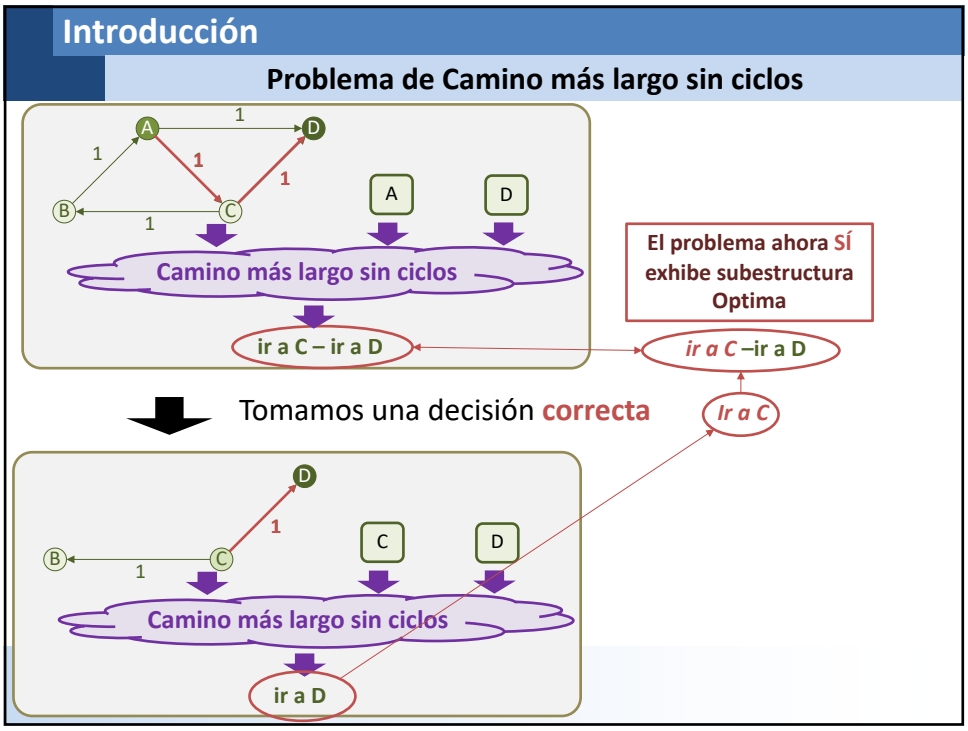
31


Introducción
Problema de Camino más largo sin ciclos
A
D
Camino más largo sin ciclos
ir a C – ir a D
El problema no exhibe subestructura Optima
ir a C – ir a B – ir a A – ir a D
Tomamos una decisión correcta
Ir a C
C
D
Camino más largo sin ciclos
ir a B – ir a A – ir a D

32


Introducción
Problema de Camino más largo sin ciclos
A
D
B
C
1
Camino más largo sin ciclos
ir a C – ir a D
Tomamos una decisión correcta
Ir a C
Camino más largo sin ciclos

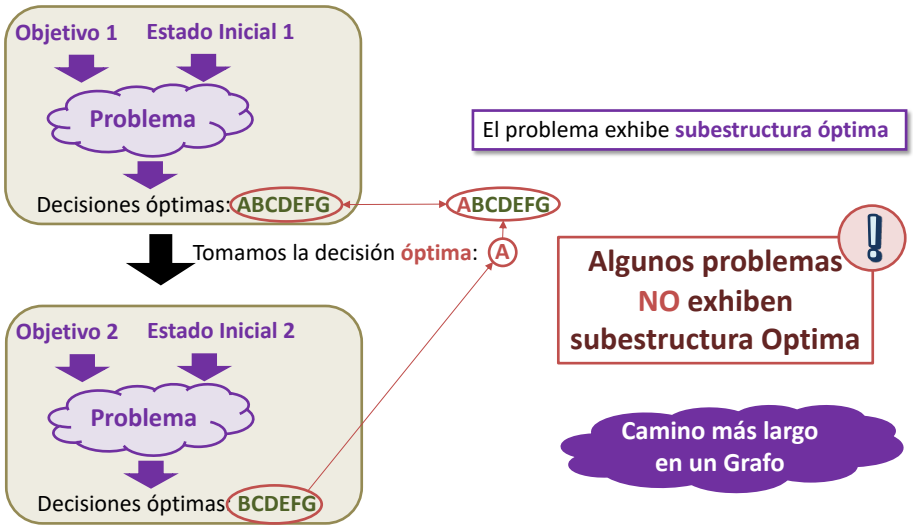
33


Introducción
Problema de Camino más largo sin ciclos
A
D
B
C
1
Camino más largo sin ciclos
ir a C – ir a D
El problema ahora SÍ exhibe subestructura Optima
ir a C –ir a D
Tomamos una decisión correcta
Ir a C
Camino más largo sin ciclos
ir a D

34

Introducción

Principio de Optimalidad de Bellman



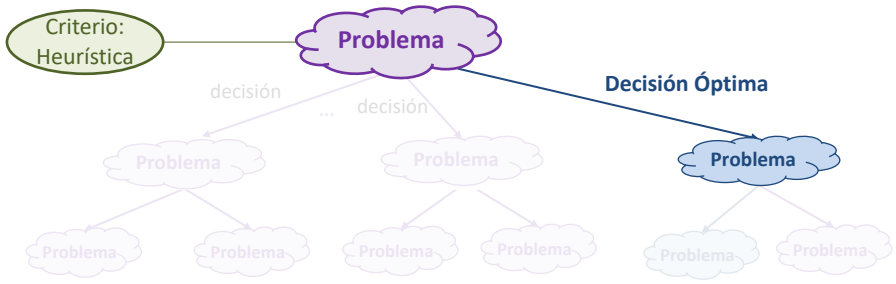
1. Introducción

1 2 3

35

Introducción

Árbol de Decisiones



Módulo 4. Algoritmo Voraz

Tenemos un **criterio sencillo** para elegir la decisión óptima

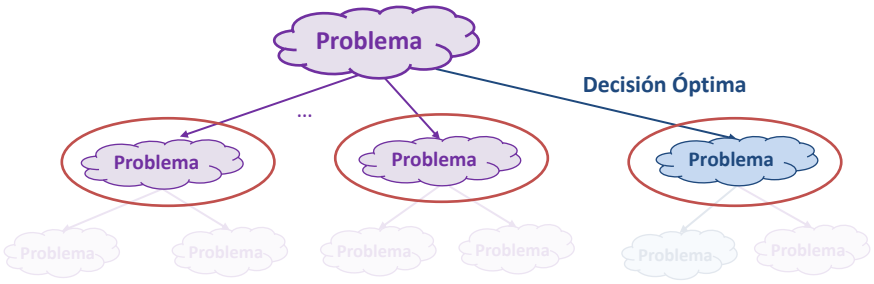
1. Introducción

1 2 3

36

Introducción

Árbol de Decisiones



Módulo 3. Backtracking

Exploramos la solución de cada uno de los problemas y elegimos la mejor

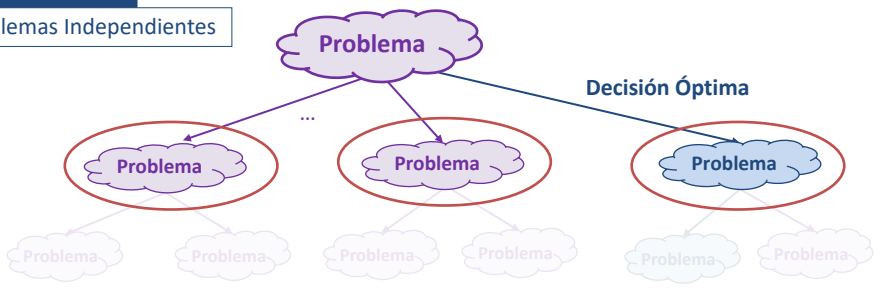
37

Introducción

Árbol de Decisiones

Divide y Vencerás

Problemas Independientes



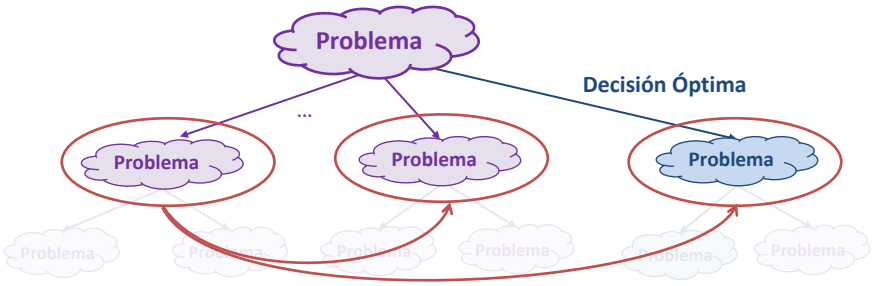
Módulo 2. Divide y Vencerás

Se parece a la técnica Divide y Vencerás: Resolvemos **subproblemas** y **elegimos** la que mejor satisfaga nuestro objetivo

38

Introducción

Árbol de Decisiones



Programación Dinámica

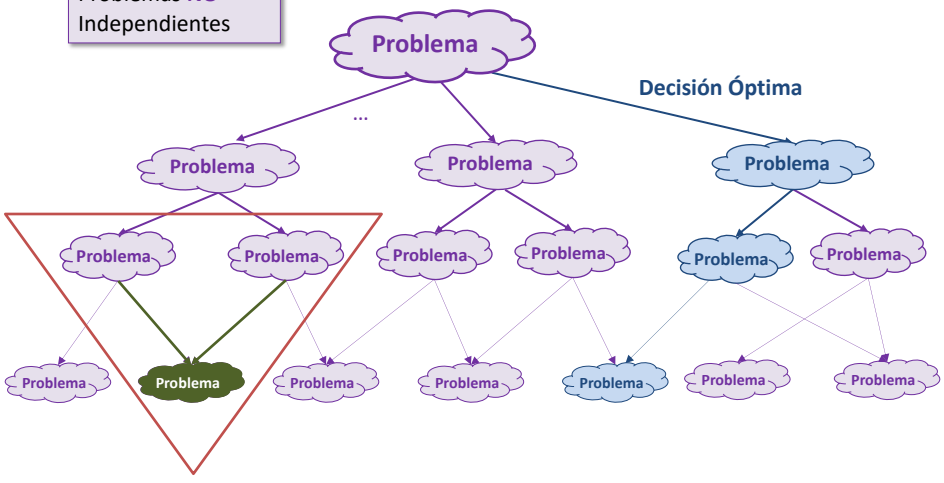
Problemas Dependientes: Resolver un problema **ayuda** a resolver otro

39

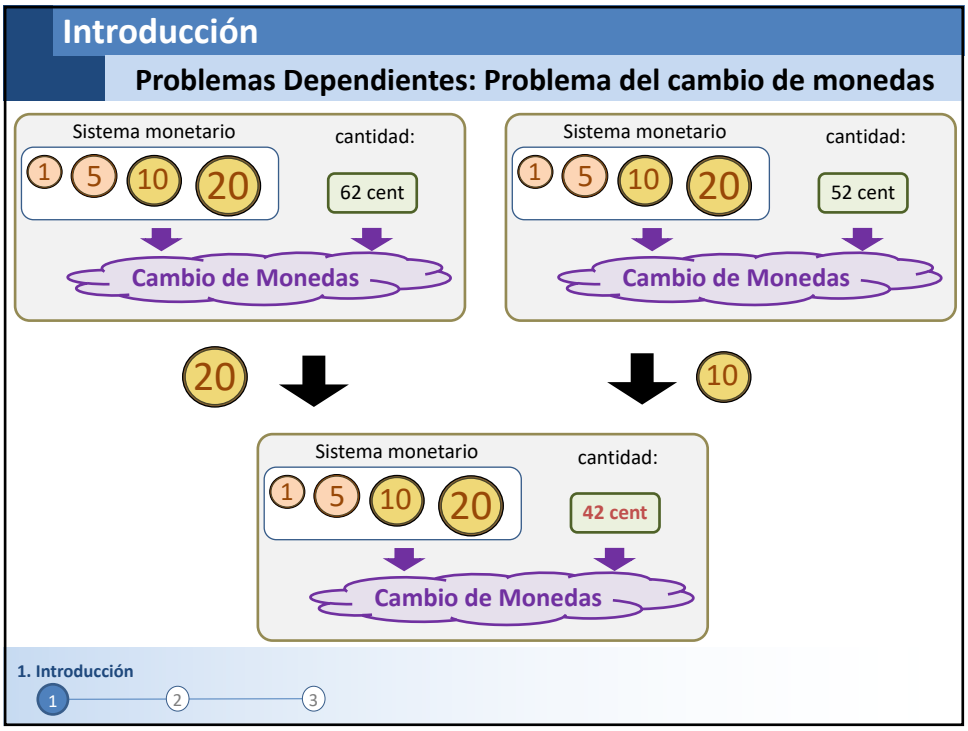
Introducción

Problemas dependientes

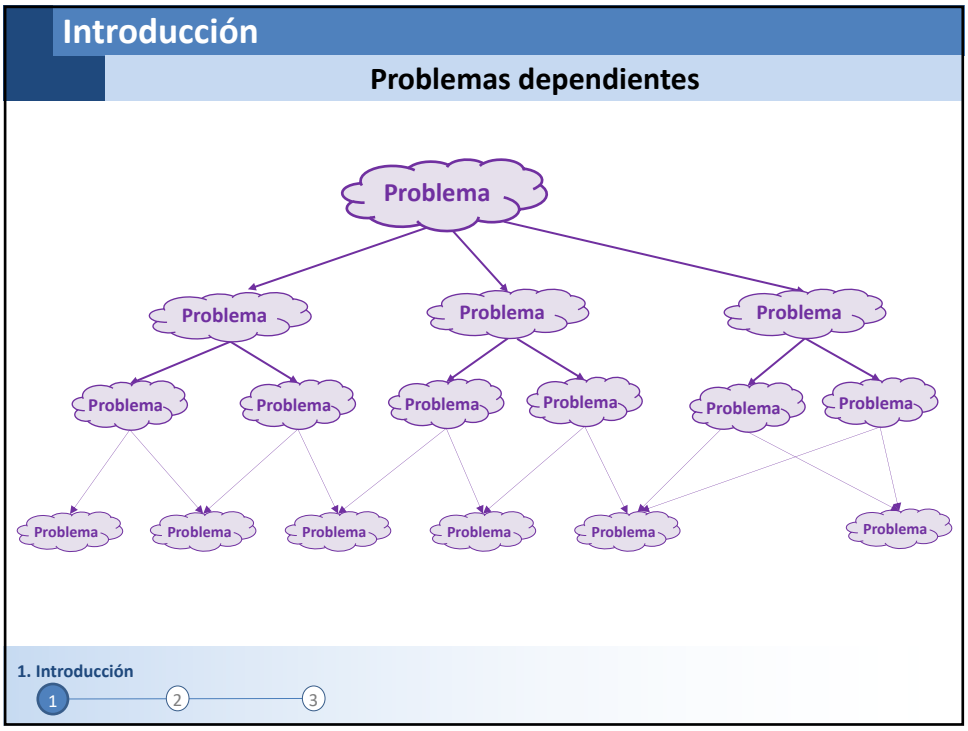
Problemas **NO** Independientes



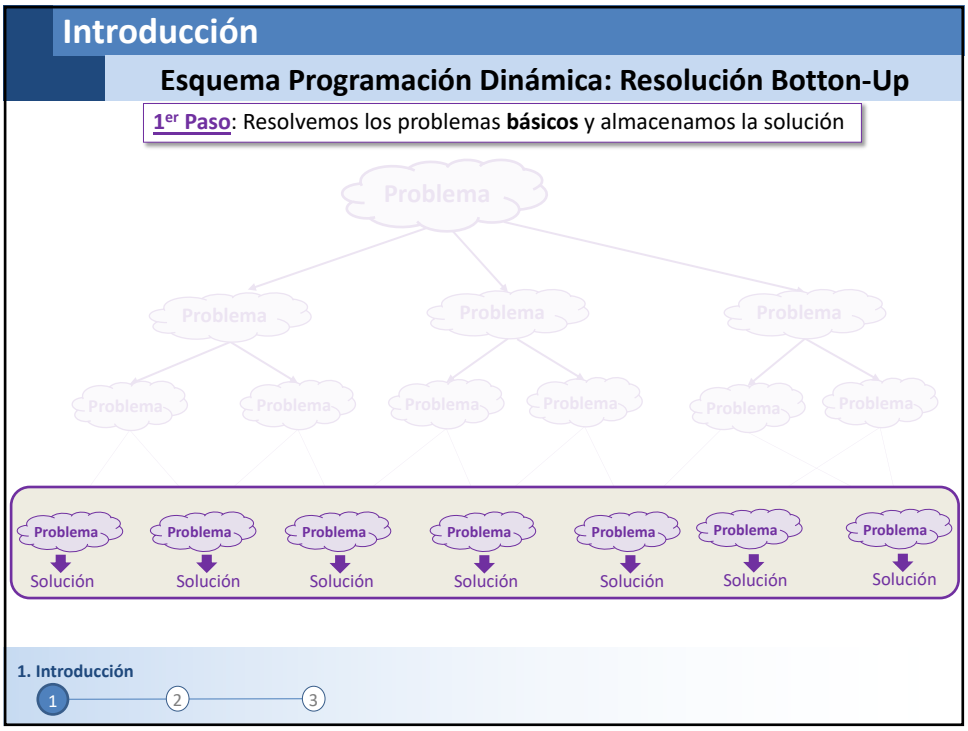
40


Introducción
Problemas Dependientes: Problema del cambio de monedas
Sistema monetario
1
5
10
20
cantidad:
62 cent
Cambio de Monedas
Sistema monetario
1
5
10
20
cantidad:
52 cent
Cambio de Monedas
20
10
Sistema monetario
1
5
10
20
cantidad:
42 cent
Cambio de Monedas
1. Introducción
1
2
3

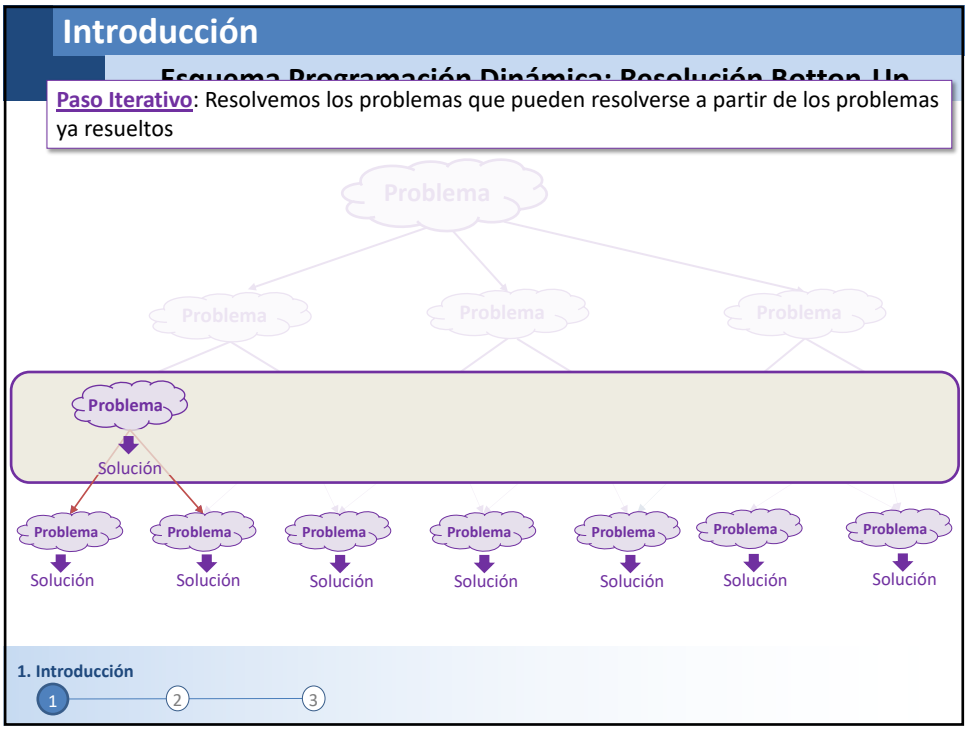
41


Introducción
Problemas dependientes
Problema
Problema
Problema
Problema
Problema
Problema
Problema
Problema
Problema
Problema
Problema
Problema
Problema
Problema
Problema
1. Introducción
1
2
3

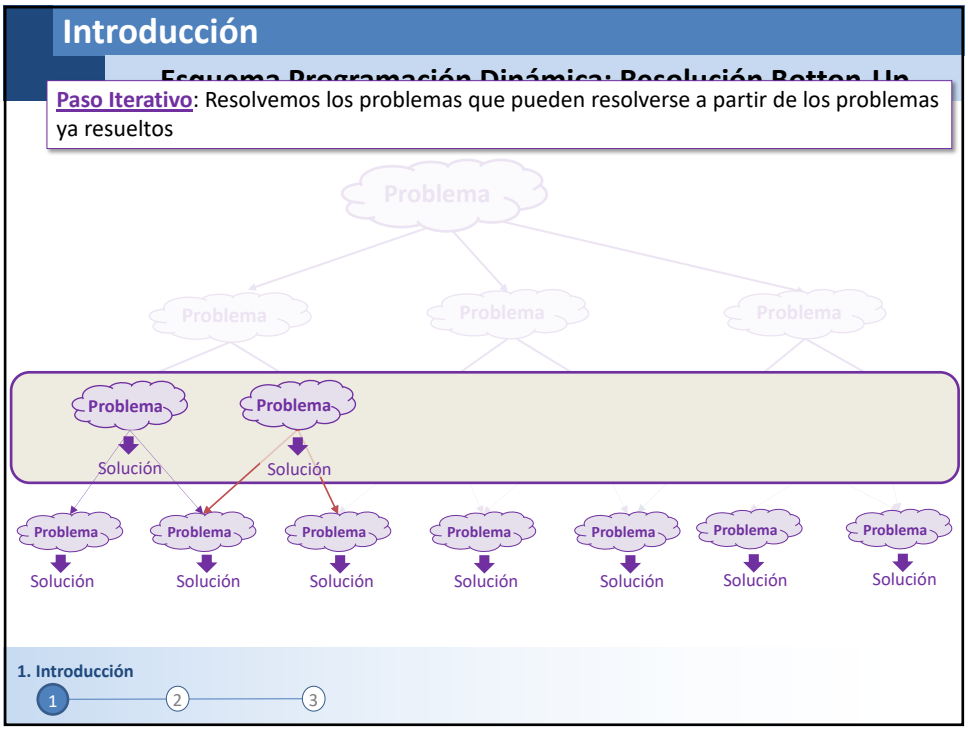
42


Introducción
Esquema Programación Dinámica: Resolución Botton-Up
1er Paso: Resolvemos los problemas básicos y almacenamos la solución
Problema
Solución
1. Introducción
1
2
3

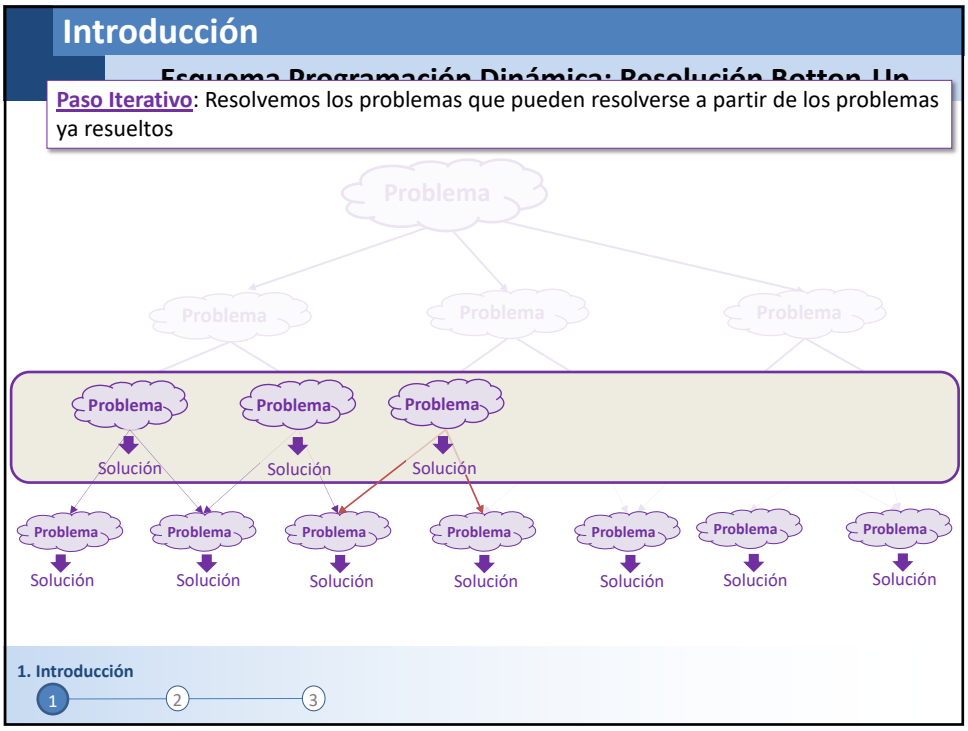
43


Introducción
Paso Iterativo: Resolvemos los problemas que pueden resolverse a partir de los problemas ya resueltos
Problema
Solución
1. Introducción
1
2
3

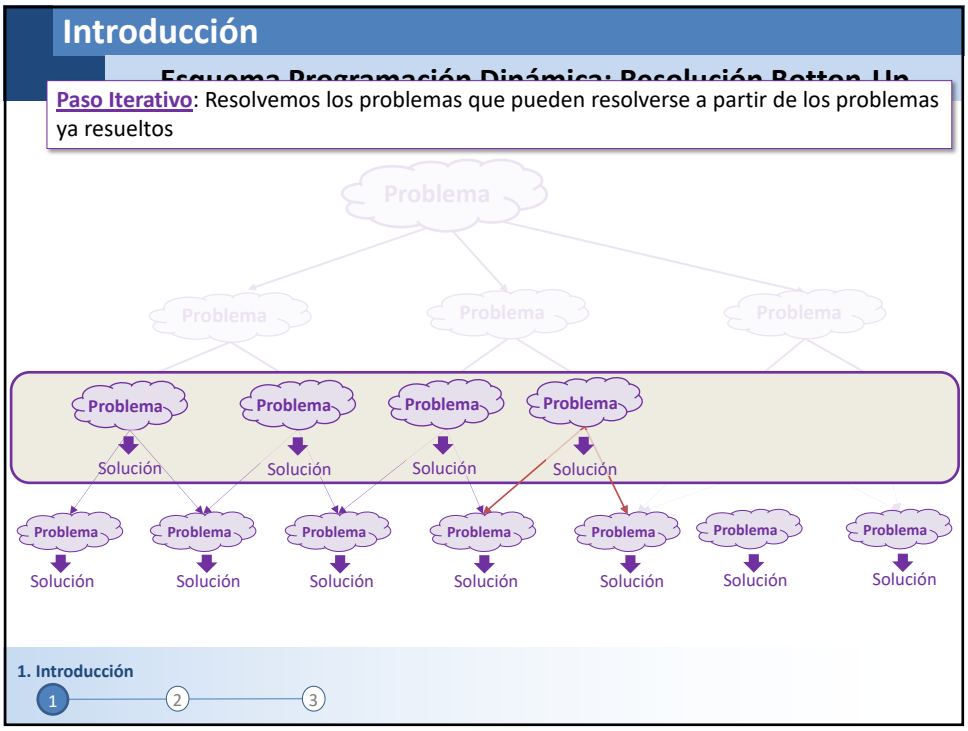
44


Introducción
Esquema Programación Dinámica: Resolución Botton-Up
Paso Iterativo: Resolvemos los problemas que pueden resolverse a partir de los problemas ya resueltos
Problema
Solución
1. Introducción

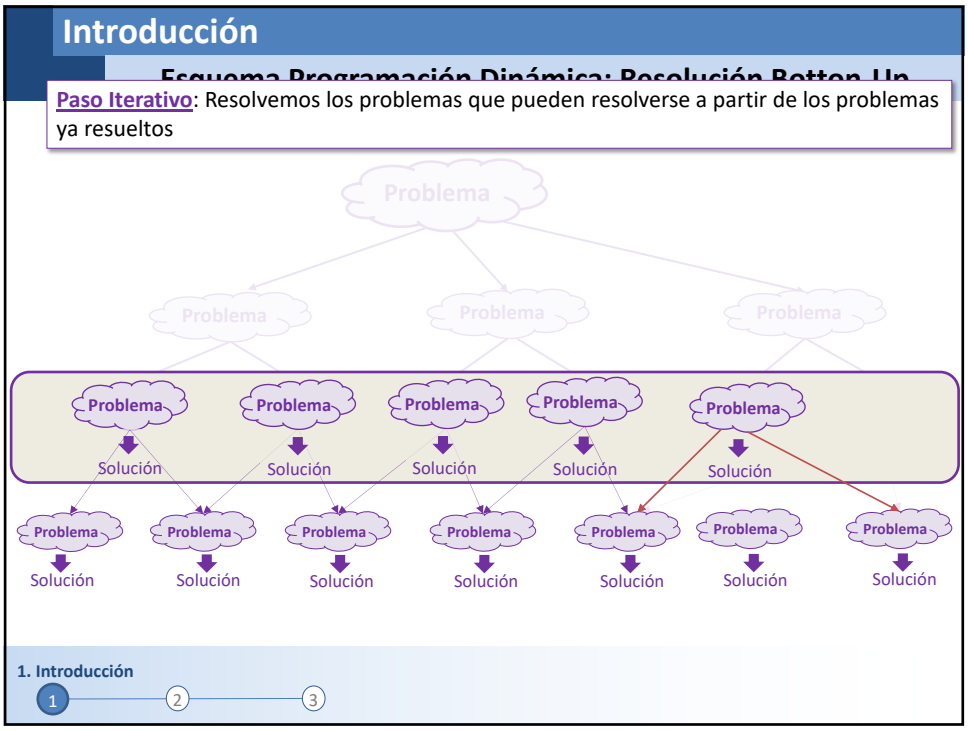
45


Introducción
Esquema Programación Dinámica: Resolución Botton-Up
Paso Iterativo: Resolvemos los problemas que pueden resolverse a partir de los problemas ya resueltos
Problema
Solución
1. Introducción

46


Introducción
Esquema Programación Dinámica: Resolución Botton-Up
Paso Iterativo: Resolvemos los problemas que pueden resolverse a partir de los problemas ya resueltos
Problema
Solución
1. Introducción
1
2
3

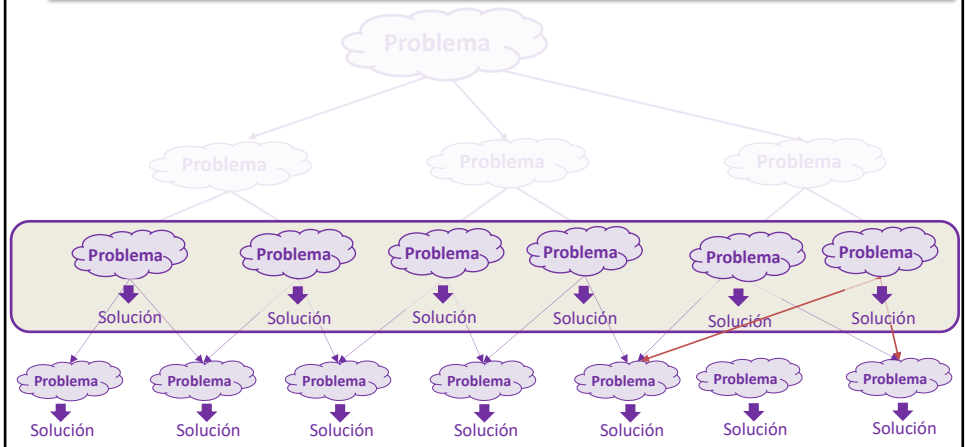
47


Introducción
Esquema Programación Dinámica: Resolución Botton-Up
Paso Iterativo: Resolvemos los problemas que pueden resolverse a partir de los problemas ya resueltos
Problema
Solución
1. Introducción
1
2
3

48

Introducción

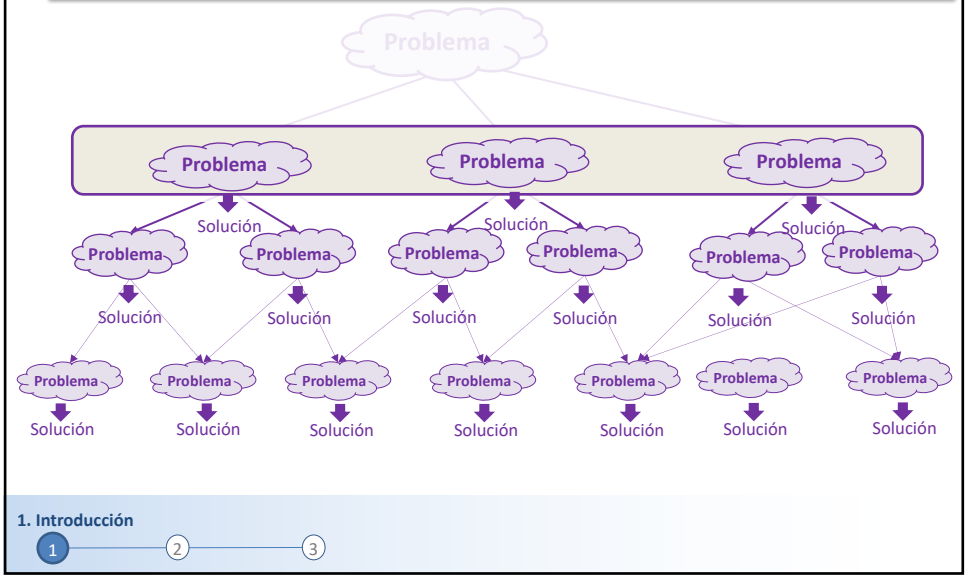
Paso Iterativo: Resolvemos los problemas que pueden resolverse a partir de los problemas ya resueltos



49

Introducción

Paso Iterativo: Resolvemos los problemas que pueden resolverse a partir de los problemas ya resueltos

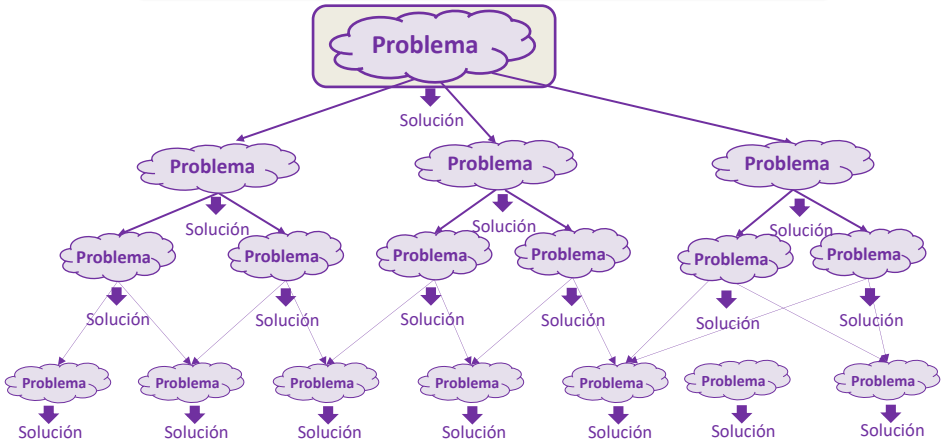


50

Introducción

Esquema Programación Dinámica: Resolución Botton-Up

Condición Terminación: Podemos resolver el problema original



1. Introducción

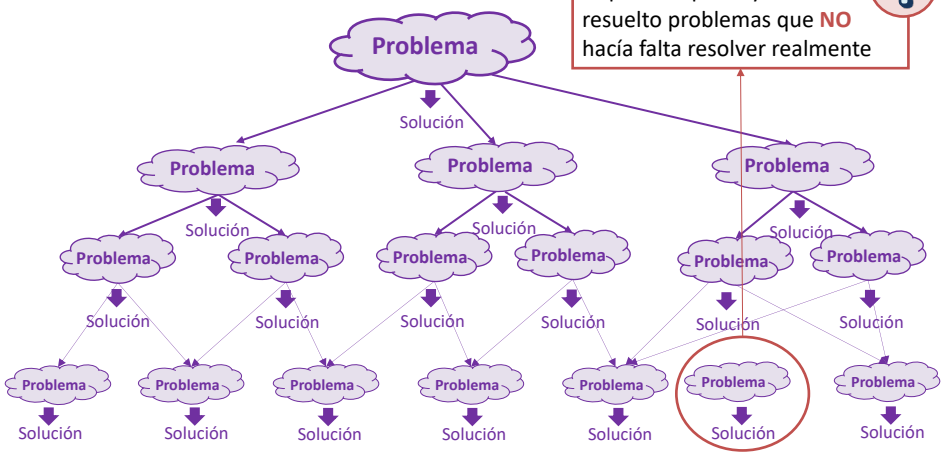
1 2 3

51

Introducción

Esquema Programación Dinámica: Resolución Botton-Up

Es posible que hayamos resuelto problemas que NO hacía falta resolver realmente



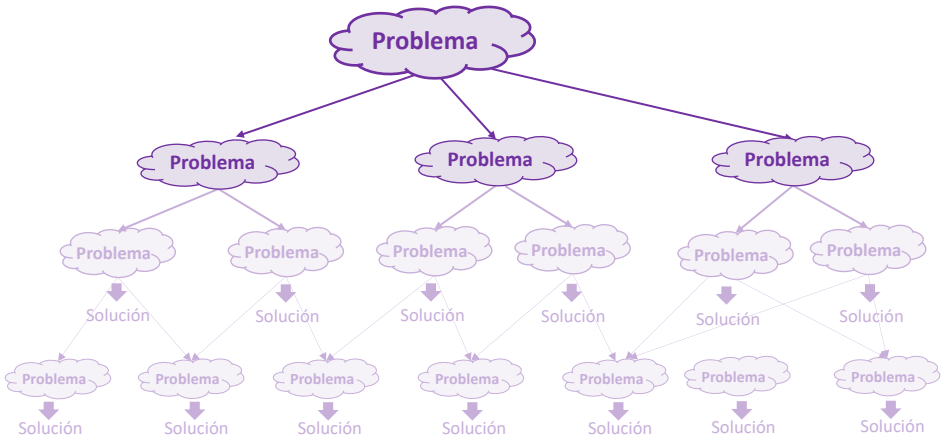
1. Introducción

1 2 3

52

Introducción

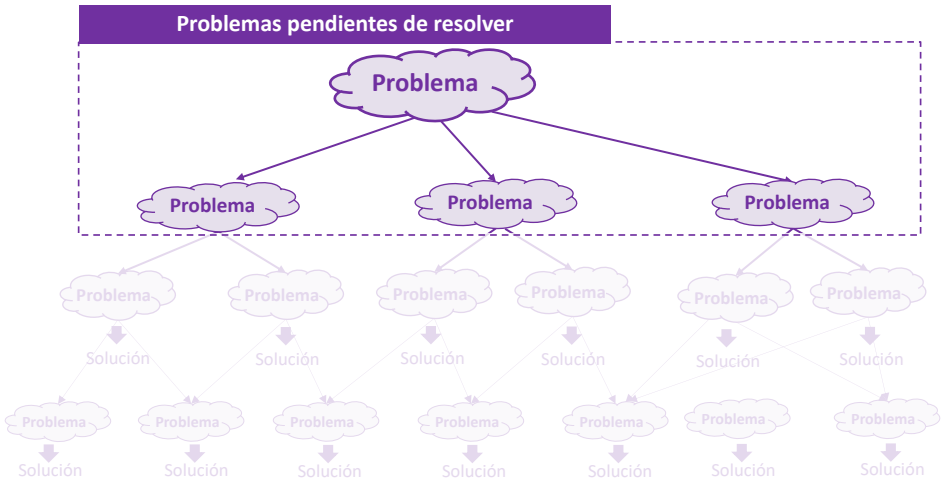
Optimización de Memoria



53

Introducción

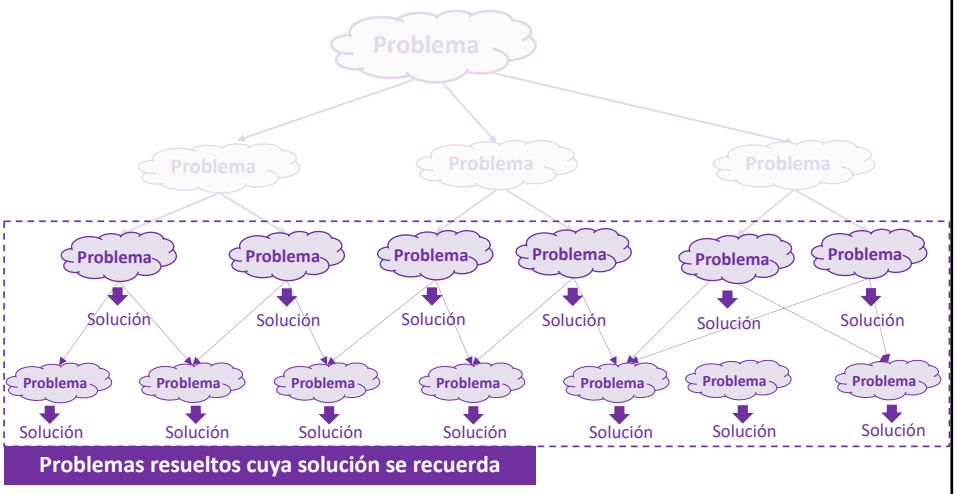
Optimización de Memoria



54

Introducción

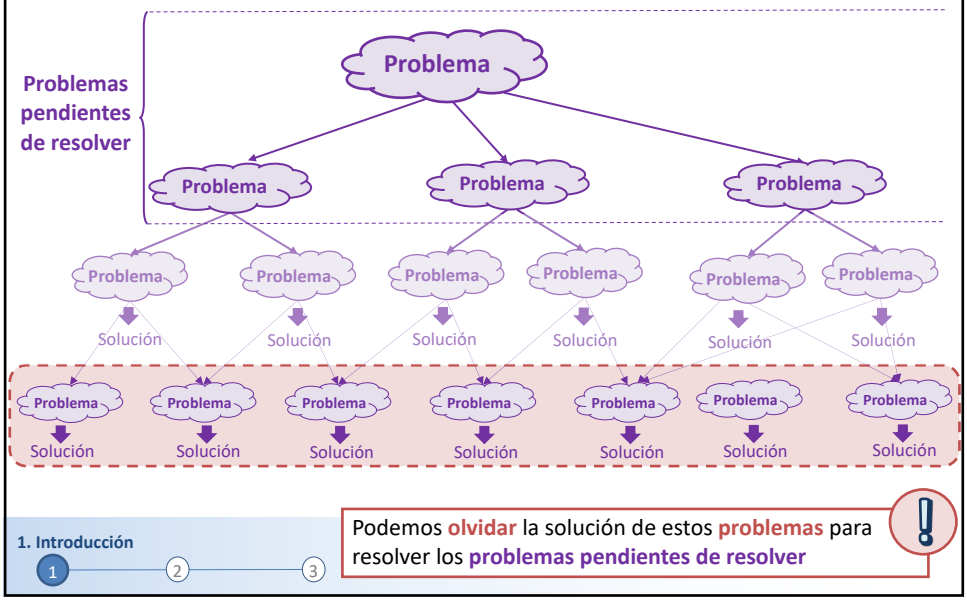
Optimización de Memoria



55

Introducción

Optimización de Memoria

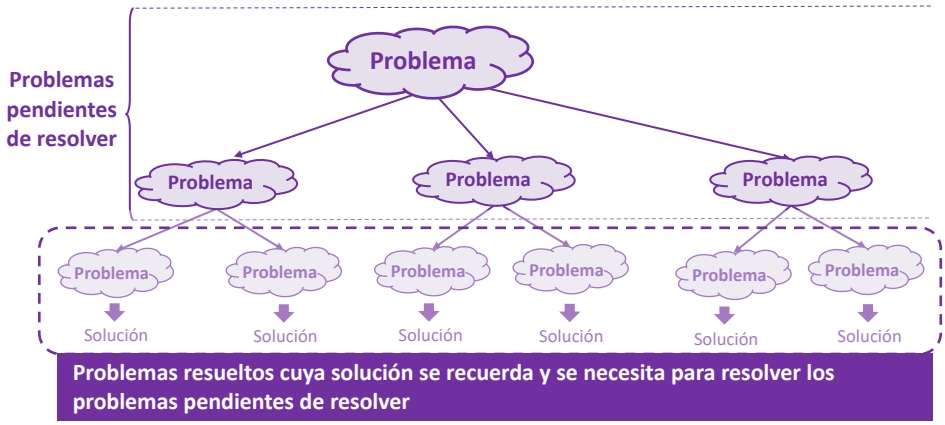


Podemos **olvidar** la solución de estos **problemas** para resolver los **problemas pendientes de resolver**

56

Introducción

Optimización de Memoria

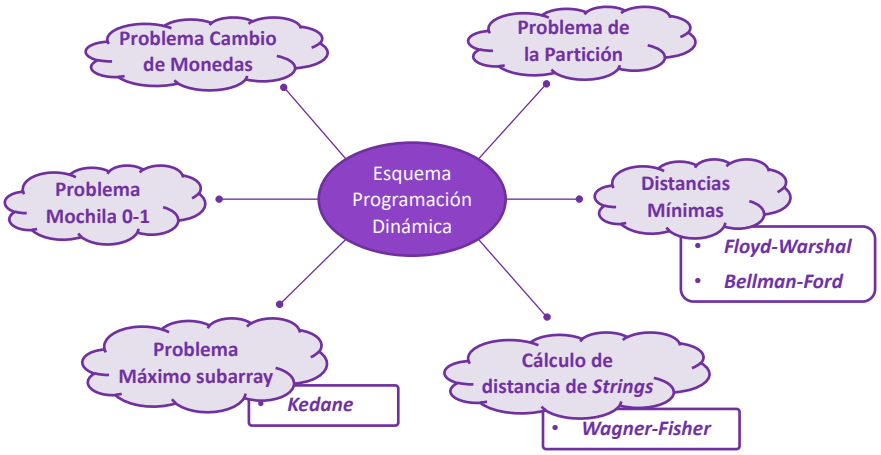


Problemas resueltos cuya solución se recuerda y se necesita para resolver los problemas pendientes de resolver

57

Introducción

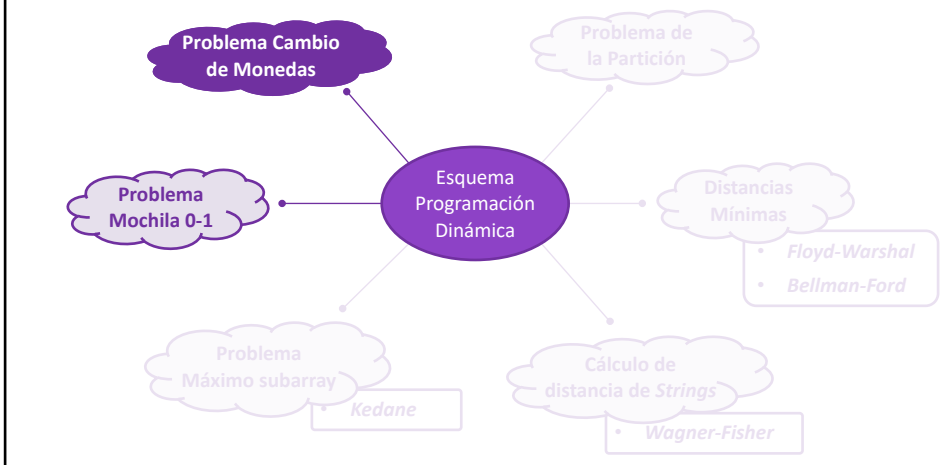
Algunos problemas planteados



58

Introducción

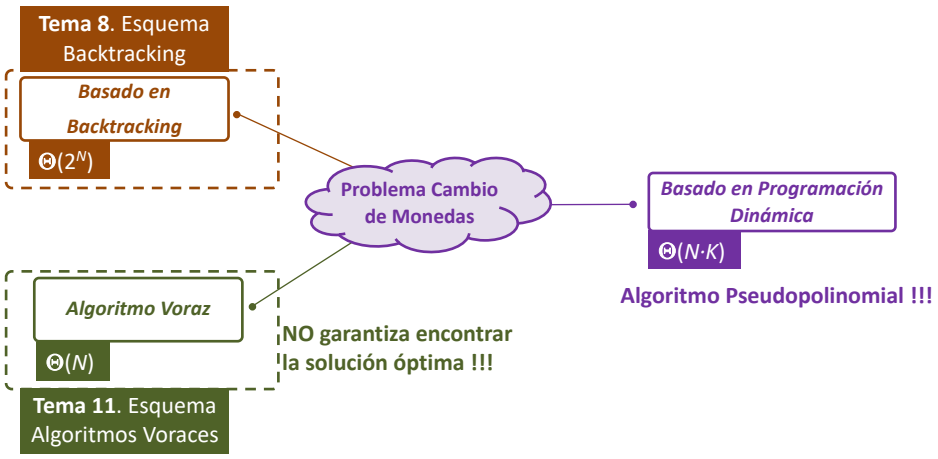
Algunos problemas planteados



59

Problema de Cambio de Monedas

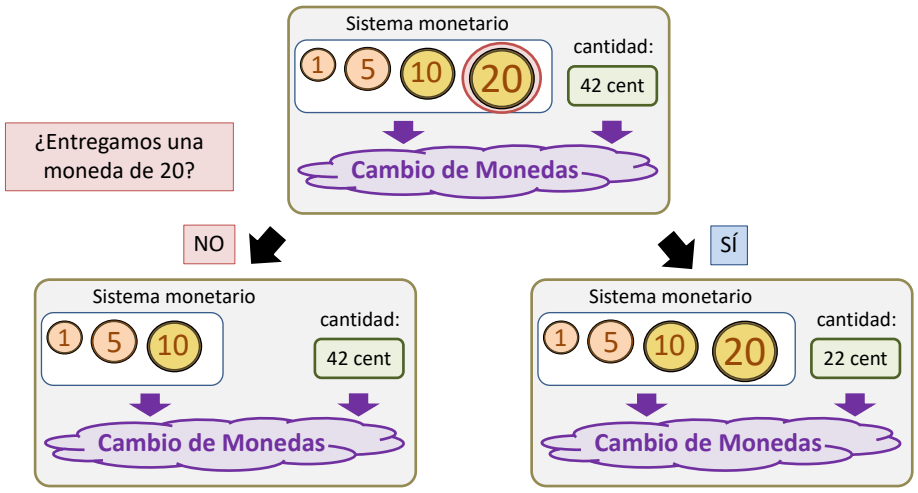
Algoritmos de Resolución



60

Problema de Cambio de Monedas

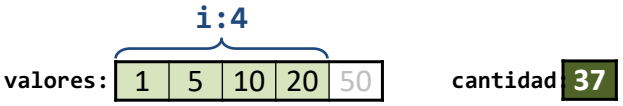
Caso recursivo



61

Problema de Cambio de Monedas

Relación de recurrencia

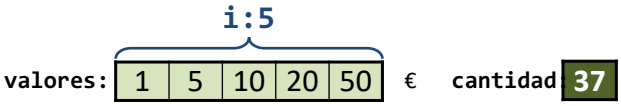


numMonedas(i,c): Número de monedas que se entregan como mínimo usando los tipos de monedas `1,…,i` para entregar la cantidad `c`

62

Problema de Cambio de Monedas

Relación de recurrencia



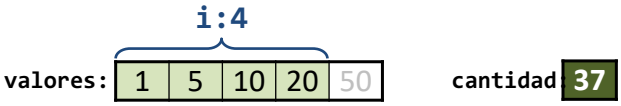
numMonedas(5,37): Número de monedas que se entregan como mínimo usando los tipos de monedas 1,…,5 para entregar la cantidad 37

Problema del Cambio de Monedas!!

63

Problema de Cambio de Monedas

Relación de recurrencia

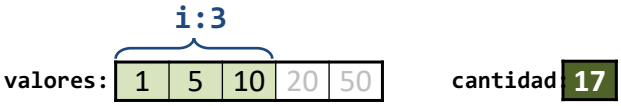


numMonedas(4,37): Número de monedas que se entregan como mínimo usando los tipos de monedas 1,2,3,4 para entregar la cantidad 37

64

Problema de Cambio de Monedas

Relación de recurrencia



numMonedas(3,17): Número de monedas que se entregan como mínimo usando los tipos de monedas 1,2,3 para entregar la cantidad 17

65

Problema de Cambio de Monedas

Relación de recurrencia

numMonedas(2,7): Número de monedas que se entregan como mínimo usando los tipos de monedas 1,2 para entregar la cantidad 7

66

Problema de Cambio de Monedas

Relación de recurrencia

numMonedas(1,7): Número de monedas que se entregan como mínimo usando los tipos de monedas **1** para entregar la cantidad **7**

67

Problema de Cambio de Monedas

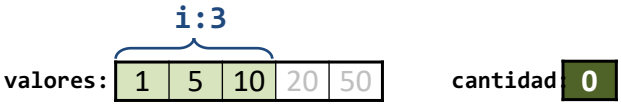
Relación de recurrencia

numMonedas(0,7): Número de monedas que se entregan como mínimo sin usar monedas para entregar la cantidad **7**

68

Problema de Cambio de Monedas

Relación de recurrencia



numMonedas(i,c): Número de monedas que se entregan como mínimo usando las tipos de monedas 1,…,i para entregar la cantidad c

Caso Base:

Para c>0 **numMonedas(0,c)=∞**

numMonedas(i,0)=0

69

Problema de Cambio de Monedas

Relación de recurrencia

numMonedas(i,c): Número de monedas que se entregan como mínimo usando las tipos de monedas 1,…,i para entregar la cantidad c

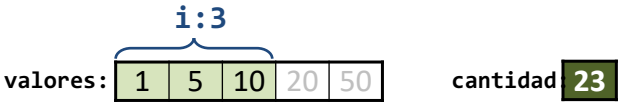
Caso Recursivo:

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

70

Problema de Cambio de Monedas

Relación de recurrencia



numMonedas(i,c): Número de monedas que se entregan como mínimo usando las tipos de monedas 1,…,i para entregar la cantidad c

Caso Recursivo:

No se entrega una moneda *i*

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

2. Problema de Cambio de Monedas

1 2 3

71

Problema de Cambio de Monedas

Relación de recurrencia

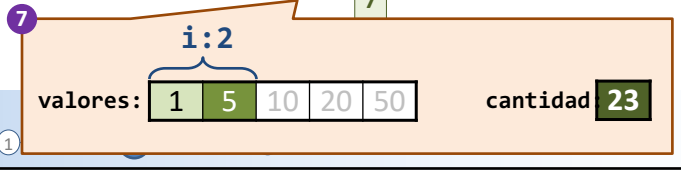
numMonedas(i,c): Número de monedas que se entregan como mínimo usando las tipos de monedas 1,…,i para entregar la cantidad c

Caso Recursivo:

No se entrega una moneda *i*

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

7

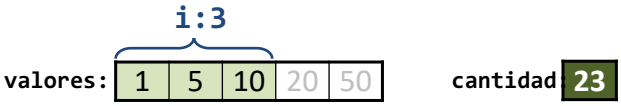


1

72

Problema de Cambio de Monedas

Relación de recurrencia



numMonedas(i,c): Número de monedas que se entregan como mínimo usando las tipos de monedas 1,…,i para entregar la cantidad c

Caso Recursivo:

Sí se entrega una moneda *i*

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

7

73

Problema de Cambio de Monedas

Relación de recurrencia

numMonedas(i,c): Número de monedas que se entregan como mínimo usando las tipos de monedas 1,…,i para entregar la cantidad c

Caso Recursivo:

Sí se entrega una moneda *i*

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

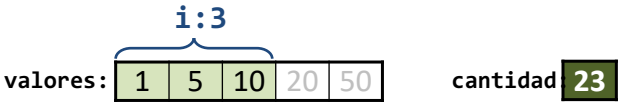
7 1+4

4

74

Problema de Cambio de Monedas

Relación de recurrencia



numMonedas(i,c): Número de monedas que se entregan como mínimo usando las tipos de monedas 1,…,i para entregar la cantidad c

Caso Recursivo:

Sí se entrega una moneda *i*

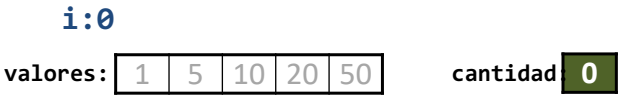
$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

5 7 1+4

75

Problema de Cambio de Monedas

Relación de recurrencia



Para c>0 **numMonedas(0,c)=∞**

numMonedas(i,0)=0

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

76

Problema de Cambio de Monedas

Relación de recurrencia

i:3

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 0

Para c>0 numMonedas(0,c)=∞

numMonedas(i,0)=0

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

77

Problema de Cambio de Monedas

Relación de recurrencia

i:0

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 4

Para c>0 numMonedas(0,c)=∞

numMonedas(i,0)=0

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

78

Problema de Cambio de Monedas

Relación de recurrencia

i:1

valores: | 1 | 5 | 10 | 20 | 50 |

cantidad: 1

$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i,c-valores[i])\}$$

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

79

Problema de Cambio de Monedas

Relación de recurrencia

i:1

valores: | 1 | 5 | 10 | 20 | 50 |

cantidad: 10

$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i,c-valores[i])\}$$

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

80

Problema de Cambio de Monedas

Relación de recurrencia

i:2

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 1

$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i, c - valores[i])\}$$

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

81

Problema de Cambio de Monedas

Relación de recurrencia

i:2

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 5

$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i, c - valores[i])\}$$

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

82

Problema de Cambio de Monedas

Relación de recurrencia

i:2

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 6

$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i, c - valores[i])\}$$

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

83

Problema de Cambio de Monedas

Relación de recurrencia

i:2

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 9

$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i, c - valores[i])\}$$

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

84

Problema de Cambio de Monedas

Relación de recurrencia

i:2

valores: 1 | 5 | 10 | 20 | 50 cantidad: 10

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

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

85

Problema de Cambio de Monedas

Relación de recurrencia

i:3

valores: 1 | 5 | 10 | 20 | 50 cantidad: 10

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

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

86

Problema de Cambio de Monedas

Relación de recurrencia

i:4

valores: | 1 | 5 | 10 | 20 | 50 |

cantidad: 10

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

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

87

Problema de Cambio de Monedas

Relación de recurrencia

i:5

valores: | 1 | 5 | 10 | 20 | 50 |

cantidad: 10

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

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

88

Problema de Cambio de Monedas

Relación de recurrencia

i:5

valores:	1	5	10	20	50

cantidad: 10

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

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

89

Problema de Cambio de Monedas

Relación de recurrencia

Actividad 14.1. Implementa el algoritmo que determina el número de monedas que tenemos que entregar

```
int numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[valores.length+1][cantidad+1];
  for (int c=0;c<=cantidad; c++)
       cant[0][c] = Integer.MAX_VALUE;
  for (int i=0;i<=valores.length; i++)
       cant[i][0] = 0;
  for (int i=1;i<=valores.length; i++)
    for (int c=1;c<=cantidad; c++)
       if (c<valores[i-1])
          cant[i][c]=cant[i-1][c];
       else
          cant[i][c]=Math.min(cant[i-1][c],1+cant[i][c-valores[i-1]]);
  return cant[valores.length][cantidad];
}
```


90

Problema de Cambio de Monedas

Relación de recurrencia

Actividad 14.1. Implementa el algoritmo que determina el número de monedas que tenemos que entregar

```
int numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[valores.length+1][cantidad+1];
  for (int c=0;c<=cantidad; c++)
       cant[0][c] = Integer.MAX_VALUE;
  for (int i=0;i<=valores.length; i++)
       cant[i][0] = 0;
  for (int i=1;i<=valores.length; i++)
    for (int c=1;c<=cantidad; c++)
        if (c<valores[i-1])
           cant[i][c]=cant[i-1][c];
        else
           cant[i][c]=Math.min(cant[i-1][c],1+cant[i][c-valores[i-1]]);
  return cant[valores.length][cantidad];
}
```

Caso Base

Para c>0 **numMonedas(0,c)=∞**

numMonedas(i,0)=0

91

Problema de Cambio de Monedas

Relación de recurrencia

Actividad 14.1. Implementa el algoritmo que determina el número de monedas que tenemos que entregar

```
int numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[valores.length+1][cantidad+1];
  for (int i=1;i<=valores.length; i++)
    for (int c=1;c<=cantidad; c++)
        if (c<valores[i-1])
           cant[i][c]=cant[i-1][c];
        else
           cant[i][c]=Math.min(cant[i-1][c],1+cant[i][c-valores[i-1]]);
  return cant[valores.length][cantidad];
}
```

Caso Recursivo

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i,c-valores[i])\}$$

92

Problema de Cambio de Monedas

Relación de recurrencia

Actividad 14.2. Calcula la complejidad del algoritmo en tiempo de acuerdo con la cantidad, *K*, y el número de monedas, *N*.

```
int numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[valores.length+1][cantidad+1];
  for (int c=0;c<=cantidad; c++)
       cant[0][c] = Integer.MAX_VALUE;
  for (int i=0;i<=valores.length; i++)
       cant[i][0] = 0;
  for (int i=1;i<=valores.length; i++)
    for (int c=1;c<=cantidad; c++)
       if (c<valores[i-1])
          cant[i][c]=cant[i-1][c];
       else
          cant[i][c]=Math.min(cant[i-1][c],1+cant[i][c-valores[i-1]]);
  return cant[valores.length][cantidad];
}
```

$\Theta(N \cdot K)$

93

Problema de Cambio de Monedas

Relación de recurrencia

Actividad 14.3. Calcula la complejidad del algoritmo en tiempo de acuerdo con el número de bits *k* para representar la cantidad, y el número de monedas, *N*.

```
int numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[valores.length+1][cantidad+1];
  for (int c=0;c<=cantidad; c++)
       cant[0][c] = Integer.MAX_VALUE;
  for (int i=0;i<=valores.length; i++)
       cant[i][0] = 0;
  for (int i=1;i<=valores.length; i++)
    for (int c=1;c<=cantidad; c++)
       if (c<valores[i-1])
          cant[i][c]=cant[i-1][c];
       else
          cant[i][c]=Math.min(cant[i-1][c],1+cant[i][c-valores[i-1]]);
  return cant[valores.length][cantidad];
}
```

$\Theta(N \cdot K)$

$\Theta(N \cdot 2^k)$

Algoritmo pseudopolinomial

94

Problema de Cambio de Monedas

Relación de recurrencia

Actividad 14.4. Calcula la complejidad del algoritmo en memoria de acuerdo con la cantidad *K* y el número de monedas, *N*.

```
int numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[valores.length+1][cantidad+1];
  for (int c=0;c<=cantidad; c++)
       cant[0][c] = Integer.MAX_VALUE;
  for (int i=0;i<=valores.length; i++)
       cant[i][0] = 0;
  for (int i=1;i<=valores.length; i++)
    for (int c=1;c<=cantidad; c++)
       if (c<valores[i-1])
          cant[i][c]=cant[i-1][c];
       else
          cant[i][c]=Math.min(cant[i-1][c],1+cant[i][c-valores[i-1]]);
  return cant[valores.length][cantidad];
}
```

$\Theta(N \cdot K)$

95

Problema de Cambio de Monedas

Relación de recurrencia

i:3

valores: | 1 | 5 | 10 | 20 | 50 |

cantidad: 10

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i,c - valores[i])\}$$

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

No necesitamos ya estos valores

Objetivo: Calcular este valor

96

Problema de Cambio de Monedas

Relación de recurrencia

i:1

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 1

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

		0	1	2	3	4	5	6	7	8	9	10
i:0	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
i:1	1	0	1									

97

Problema de Cambio de Monedas

Relación de recurrencia

i:1

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 10

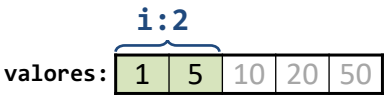
$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

		0	1	2	3	4	5	6	7	8	9	10
i:0	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
i:1	1	0	1	2	3	4	5	6	7	8	9	10

98

Problema de Cambio de Monedas

Relación de recurrencia



cantidad: 1

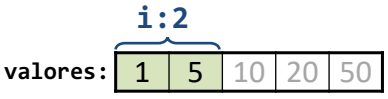
$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i,c-valores[i])\}$$

		0	1	2	3	4	5	6	7	8	9	10
i:2	0	0	1	∞	∞	∞	∞	∞	∞	∞	∞	∞
i:1	1	0	1	2	3	4	5	6	7	8	9	10

99

Problema de Cambio de Monedas

Relación de recurrencia



cantidad: 5

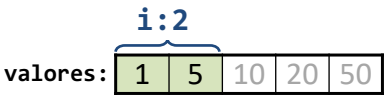
$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i,c-valores[i])\}$$

		0	1	2	3	4	5	6	7	8	9	10
i:2	0	0	1	2	3	4	1	∞	∞	∞	∞	∞
i:1	1	0	1	2	3	4	5	6	7	8	9	10

100

Problema de Cambio de Monedas

Relación de recurrencia



cantidad: **10**

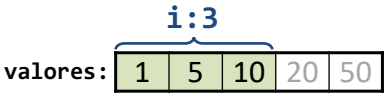
$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i,c - valores[i])\}$$

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

101

Problema de Cambio de Monedas

Relación de recurrencia



cantidad: **4**

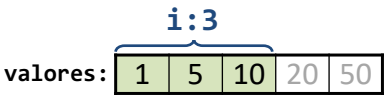
$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i,c - valores[i])\}$$

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

102

Problema de Cambio de Monedas

Relación de recurrencia



cantidad: **10**

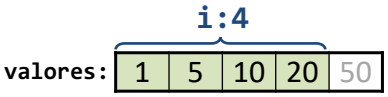
$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i, c - valores[i])\}$$

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

103

Problema de Cambio de Monedas

Relación de recurrencia



cantidad: **10**

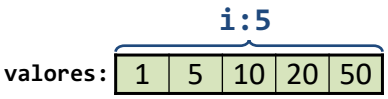
$$numMonedas(i,c) = min\{numMonedas(i-1,c),\ 1 + numMonedas(i, c - valores[i])\}$$

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

104

Problema de Cambio de Monedas

Relación de recurrencia



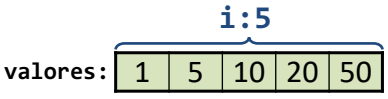
$$numMonedas(i,c) = min\{numMonedas(i-1,c)\ 1 + numMonedas(i,c-valores[i])\}$$

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

105

Problema de Cambio de Monedas

Relación de recurrencia



$$numMonedas(i,c) = min\{numMonedas(i-1,c)\ 1 + numMonedas(i,c-valores[i])\}$$

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

106

Problema de Cambio de Monedas

Relación de recurrencia

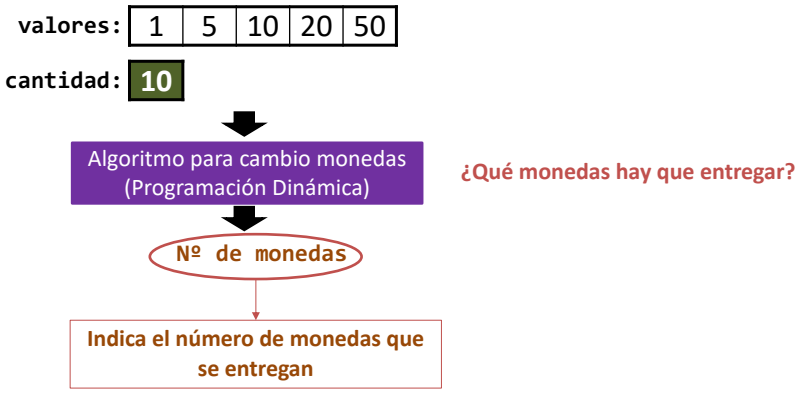
Actividad 14.5. Optimiza en memoria el algoritmo anterior que calcula el número de monedas

```
int numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[2][cantidad+1];
  for (int c=0;c<=cantidad; c++)
       cant[0][c] = Integer.MAX_VALUE;
  for (int i=0;i<=1; i++)
       cant[i][0] = 0;
  for (int i=1;i<=valores.length; i++) {
    for (int c=1;c<=cantidad; c++)
      if (c<valores[i-1])
        cant[i%2][c]=cant[(i-1)%2][c];
      else
        cant[i%2][c]=Math.min(cant[(i-1)%2][c],1+cant[i%2][c-valores[i-1]]);
   }
  return cant[valores.length % 2][cantidad];
}
```


107

Problema de Cambio de Monedas

Conociendo las monedas que se entregan



108

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: 10

Algoritmo para cambio monedas (Programación Dinámica)

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

109

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores: | **1** | 5 | 10 | 20 | 50 | cantidad: 10

Algoritmo para cambio monedas (Programación Dinámica)

Se entrega una moneda de 1 unidad

	0	1	2	3	4	5	6	7	8	9	10
0	F	F	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	**T**	T	T	T	T
2	F	F	F	F	F	T	T	T	T	T	T
3	F	F	F	F	F	F	F	F	F	F	T
4	F	F	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F	F	F

110

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores: | 1 | **5** | 10 | 20 | 50 | cantidad: **10**

No se entrega la moneda de 5 unidades

Algoritmo para cambio monedas (Programación Dinámica)

	0	1	2	3	4	5	6	7	8	9	10
0	F	F	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T	T	T
2	F	F	**F**	F	F	T	T	T	T	T	T
3	F	F	F	F	F	F	F	F	F	F	T
4	F	F	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F	F	F

111

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

i:5

valores: | 1 | 5 | 10 | 20 | 50 | cantidad: **10**

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

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

112

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

i : 2

valores: | 1 | 5 | 10 | 20 | 50 |

cantidad: 6

Se entrega una moneda de 5

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

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

113

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

i : 2

valores: | 1 | 5 | 10 | 20 | 50 |

cantidad: 6

No se entrega una moneda de 5

$$numMonedas(i,c) = min\{numMonedas(i-1,c), 1 + numMonedas(i, c - valores[i])\}$$

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

114

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:					

cantidad: 8

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

115

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:					0

cantidad: 8

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

116

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:				0	0

cantidad: 8

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

117

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:			0	0	0

cantidad: 8

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

118

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:		1	0	0	0

cantidad: 3

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

119

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:		1	0	0	0

cantidad: 3

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

120

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:	1	1	0	0	0

cantidad: 2

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

121

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:	2	1	0	0	0

cantidad: 1

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

122

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:	3	1	0	0	0

cantidad: **0**

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

123

Problema de Cambio de Monedas

Conociendo las monedas que se entregan

valores:	1	5	10	20	50
entrega:	3	1	0	0	0

cantidad: **8**

$$numMonedas(i, c) = min\{numMonedas(i-1, c), 1 + numMonedas(i, c - valores[i])\}$$

	0	1	2	3	4	5	6	7	8
0	F	F	F	F	F	F	F	F	F
1	F	T	T	T	T	T	T	T	T
2	F	F	F	F	F	T	T	T	T
3	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	F

124

Problema de Cambio de Monedas

Obteniendo las monedas que se entregan

Actividad 14.6. **Modifica el algoritmo para que se determine qué monedas se han de entregar.**

125

Problema de Cambio de Monedas

Obteniendo las monedas que se entregan

```
int[] numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[2][cantidad+1];
  boolean[][] aux = new boolean[valores.length+1][cantidad+1];
  for (int c=0;c<=cantidad; c++)
       {cant[0][c] = Integer.MAX_VALUE; aux[0][c]= false;}
  cant[0][0] = 0; cant[1][0] = 0;
  for (int i=0;i<=valores.length; i++)
       aux[i][0] = false;
  for (int i=1; i<=valores.length; i++) {
    for (int c=1; c<=cantidad; c++)
      if (c<valores[i-1]) {
         cant[i%2][c]=cant[(i-1)%2][c];
         aux[i][c]=false;}
      else {
         cant[i%2][c]=Math.min(cant[(i-1)%2][c],1+cant[i%2][c-valores[i-1]]);
         aux[i][c]=(cant[i%2][c]==1+cant[i%2][c-valores[i-1]]);}
   }
  return entrega(aux, valores);
}
```


126

Problema de Cambio de Monedas

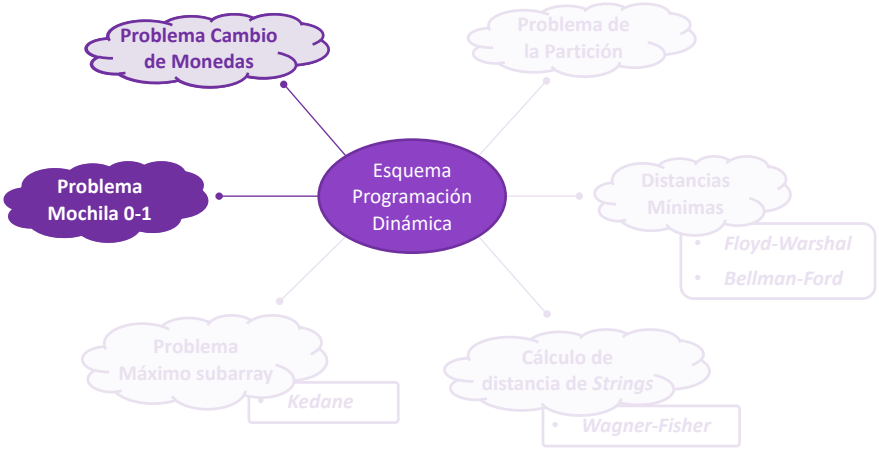
Obteniendo las monedas que se entregan

```
int[] numMonedas(int[] valores, int cantidad){
  int[][] cant = new int[2][cantidad+1];
  boolean[][] aux = new boolean[valores.length+1][cantidad+1];
  for (int c=0;c<=cantidad; c++)
    int[] entrega (boolean[][] aux, int[] valores){
      int i=aux.length-1;     int c=aux[0].length-1;
      int[] decision = new int[aux.length-1];
      decision[i-1]=0;
      while (c>0 && i>0) {
         if (aux[i][c]==true)  {
            c=c-valores[i-1]; decision[i-1]++; }
         else {
            i--; decision[i-1]=0;}
      }
      return decision;
    }
  return entrega(aux, valores);
}
```


127

Problema de Cambio de Monedas

Algunos problemas planteados



128

Problema de la Mochila 0-1

Algunos problemas planteados

Enunciado

Dado un conjunto de productos.

Cada producto tiene un peso:

24	4	1	7	15	6	3	5

Cada producto tiene un valor:

20	12	21	37	10	46	23	35

Dada una mochila donde se pueden introducir los productos

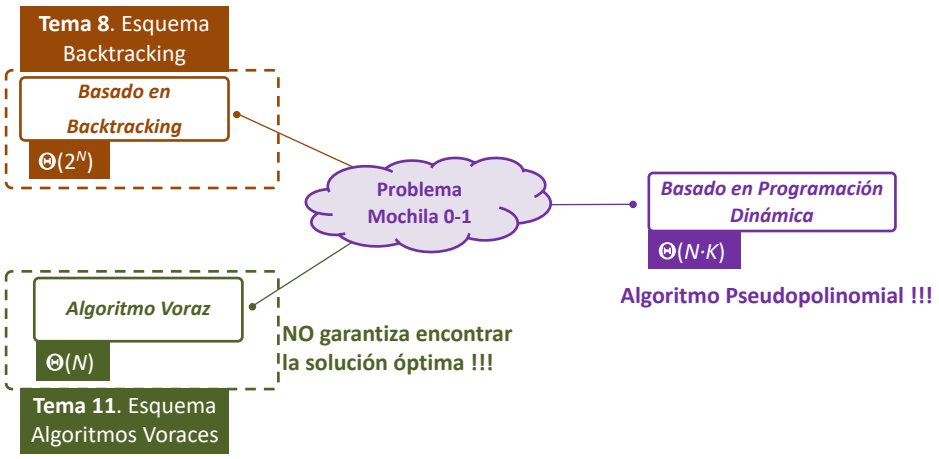
El peso de los productos no puede superar una cierta cantidad: **26** Kg.

Calcular el valor máximo que se puede llevar en la mochila

129

Problema de la Mochila 0-1

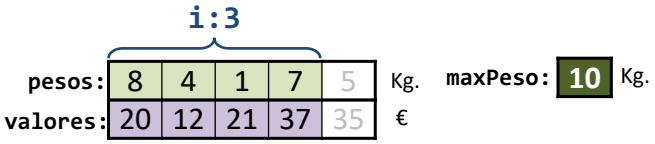
Algoritmos de Resolución



130

Problema de la Mochila 0-1

Relación de recurrencia

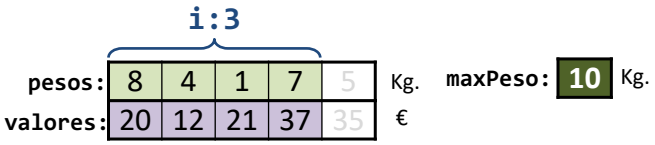


maxValor(i,p): Máximo valor que se puede introducir en la mochila con los objetos **0,1,…,i** sin excederse en el peso **p**

131

Problema de la Mochila 0-1

Relación de recurrencia

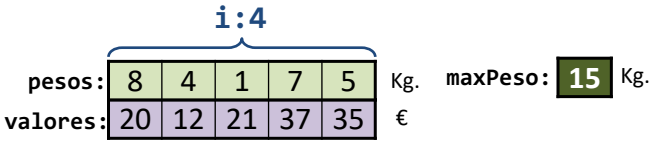


maxValor(3,10): Máximo valor que se puede introducir en la mochila con los objetos **0,1,2,3** sin excederse en el peso **10**

132

Problema de la Mochila 0-1

Relación de recurrencia



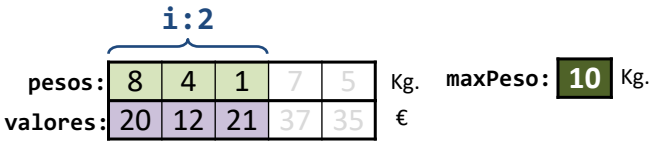
maxValor(4,15): Máximo valor que se puede introducir en la mochila con los objetos **0,1,…,4** sin excederse en el peso **15**

Problema de la Mochila 0-1 !!

133

Problema de la Mochila 0-1

Relación de recurrencia

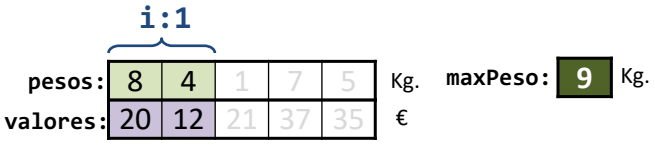


maxValor(2,10): Máximo valor que se puede introducir en la mochila con los objetos **0,1,2** sin excederse en el peso **10**

134

Problema de la Mochila 0-1

Relación de recurrencia

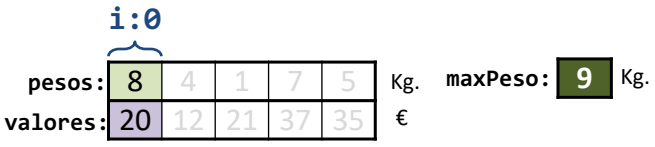


maxValor(1,9): Máximo valor que se puede introducir en la mochila con los objetos **0,1** sin excederse en el peso **9**

135

Problema de la Mochila 0-1

Relación de recurrencia. Caso Base



maxValor(0,9): Máximo valor que se puede introducir en la mochila únicamente con el objeto **0** sin excederse en el peso **9**

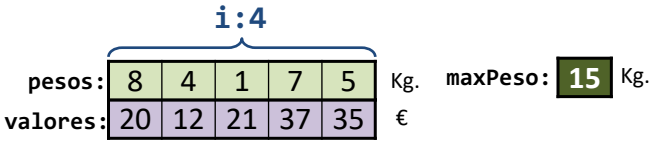
Caso Base:

$$maxValor(0,p) = \begin{cases} valores[0] & si\ pesos[0] \leq p \\ 0 & en\ otro\ caso \end{cases}$$

136

Problema de la Mochila 0-1

Relación de recurrencia. Caso Recursivo



maxValor(4,15): Máximo valor que se puede introducir en la mochila con los objetos 0,1,…,4 sin excederse en el peso 15

Caso Recursivo:

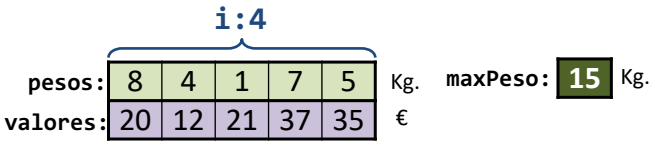
No se introduce en la mochila el producto *i*

$$maxValor(i,p) = \begin{cases} maxValor(i-1,p) & \text{si } pesos[i]>p \\ \max\{maxValor(i-1,p),\ valores[i]+maxValor(i-1,p-pesos[i])\} & \text{en otro caso} \end{cases}$$

137

Problema de la Mochila 0-1

Relación de recurrencia. Caso Recursivo



maxValor(4,15): Máximo valor que se puede introducir en la mochila con los objetos 0,1,…,4 sin excederse en el peso 15

Caso Recursivo:

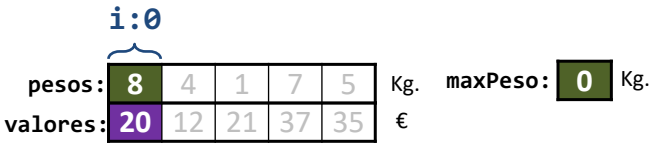
Se introduce en la mochila el producto *i*

$$maxValor(i,p) = \begin{cases} maxValor(i-1,p) & \text{si } pesos[i]>p \\ \max\{maxValor(i-1,p),\ valores[i]+maxValor(i-1,p-pesos[i])\} & \text{en otro caso} \end{cases}$$

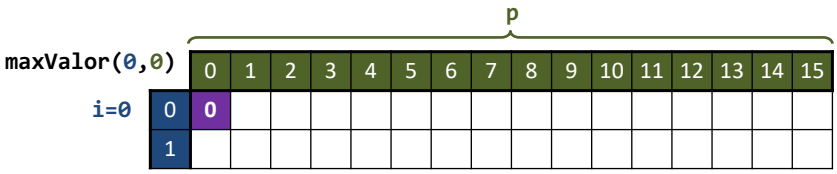
138

Problema de la Mochila 0-1

Relación de recurrencia



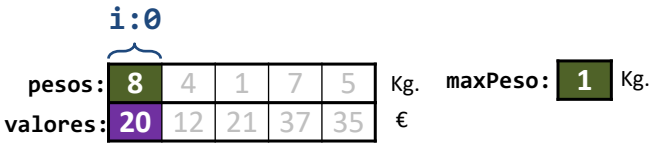
$$maxValor(0,p) = \begin{cases} valores[0] & si\ pesos[0] \le p \\ 0 & en\ otro\ caso \end{cases}$$



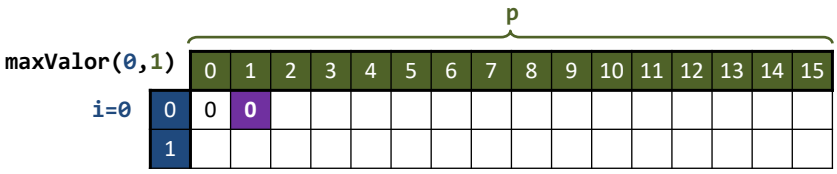
139

Problema de la Mochila 0-1

Relación de recurrencia



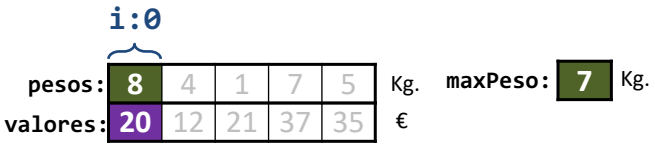
$$maxValor(0,p) = \begin{cases} valores[0] & si\ pesos[0] \le p \\ 0 & en\ otro\ caso \end{cases}$$



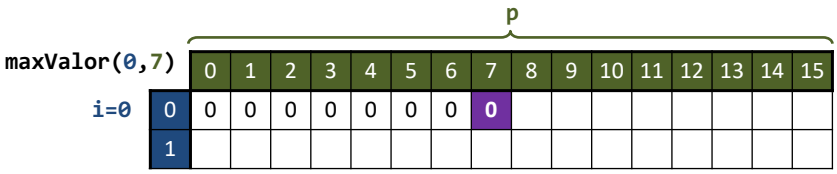
140

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(0,p) = \begin{cases} valores[0] & si\ pesos[0] \le p \\ 0 & en\ otro\ caso \end{cases}$$



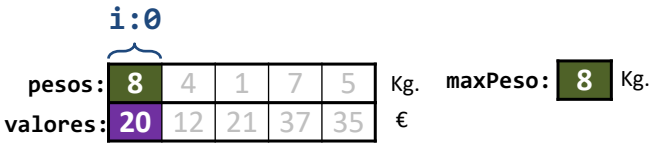
3. Problema de la Mochila 0-1

1 2 3

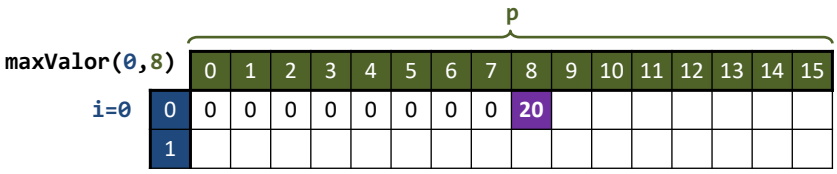
141

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(0,p) = \begin{cases} valores[0] & si\ pesos[0] \le p \\ 0 & en\ otro\ caso \end{cases}$$



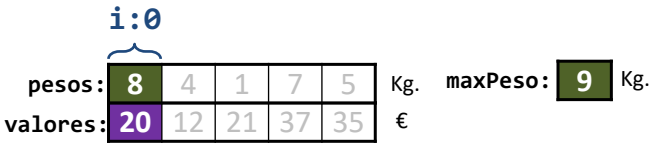
3. Problema de la Mochila 0-1

1 2 3

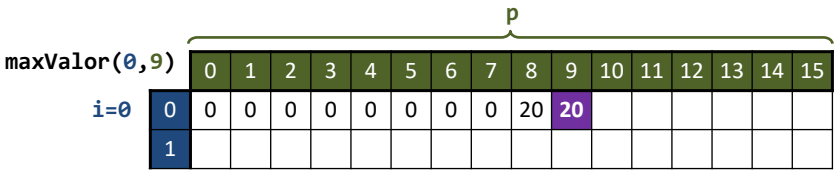
142

Problema de la Mochila 0-1

Relación de recurrencia



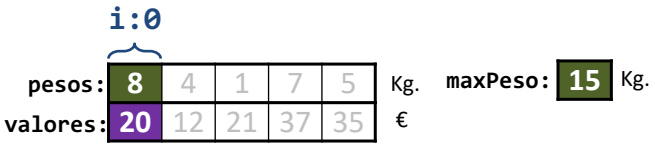
$$maxValor(0,p) = \begin{cases} valores[0] & si\ pesos[0] \le p \\ 0 & en\ otro\ caso \end{cases}$$



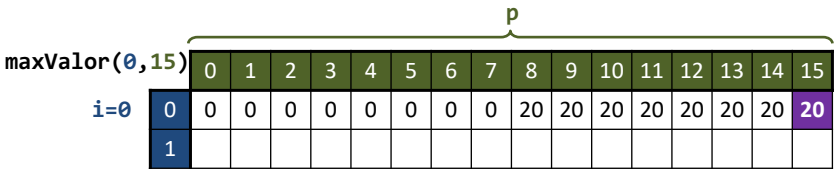
143

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(0,p) = \begin{cases} valores[0] & si\ pesos[0] \le p \\ 0 & en\ otro\ caso \end{cases}$$



144

Problema de la Mochila 0-1

Relación de recurrencia

i:1

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 0 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(1,0)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=0	0	0	0	0	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0															

3. Problema de la Mochila 0-1

1 2 3

145

Problema de la Mochila 0-1

Relación de recurrencia

i:1

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 1 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(1,1)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=0	0	0	0	0	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0														

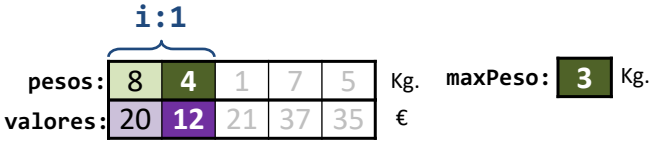
3. Problema de la Mochila 0-1

1 2 3

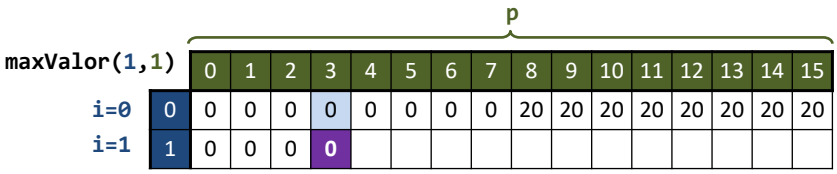
146

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$



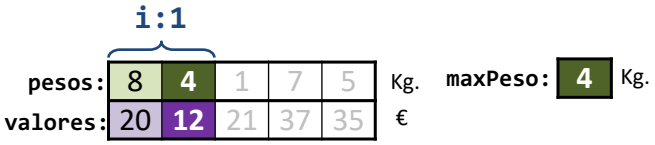
3. Problema de la Mochila 0-1

1 2 3

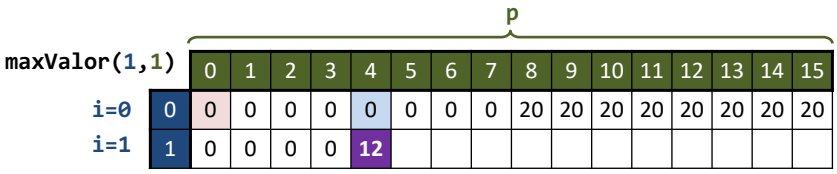
147

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$



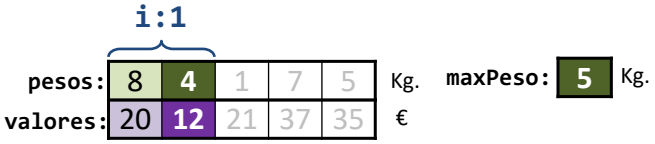
3. Problema de la Mochila 0-1

1 2 3

148

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

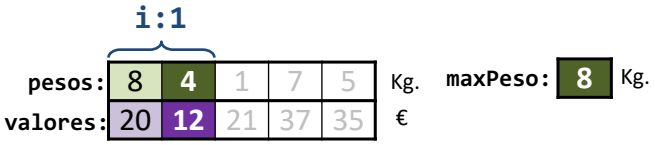
maxValor(1,5)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=0	0	0	0	0	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12										

149

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

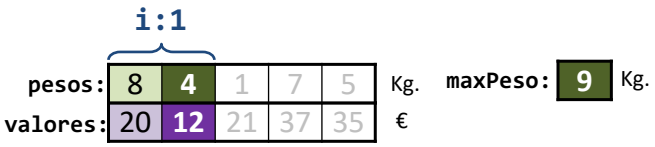
maxValor(1,8)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=0	0	0	0	0	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20							

150

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

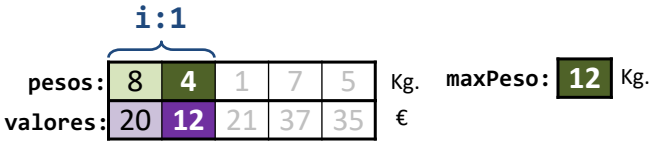
maxValor(1,9)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=0	0	0	0	0	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20						

151

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

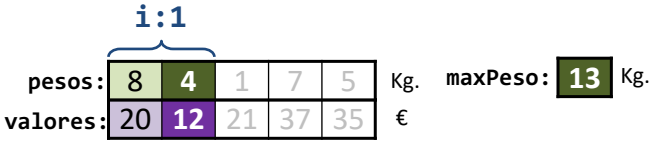
maxValor(1,12)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=0	0	0	0	0	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32			

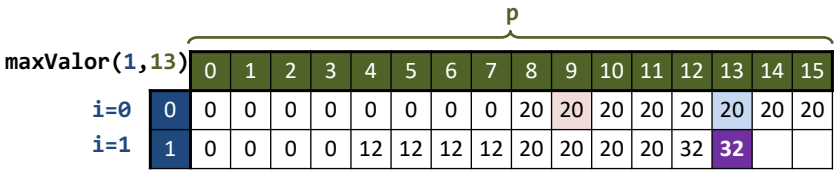
152

Problema de la Mochila 0-1

Relación de recurrencia



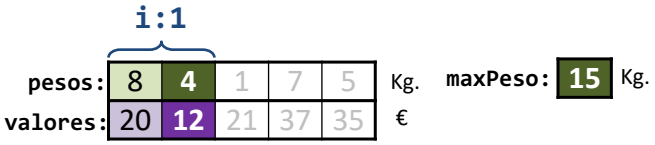
$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$



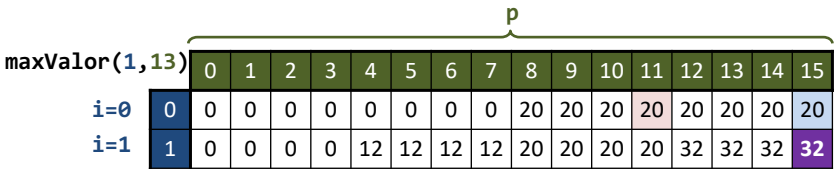
153

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$



154

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 0 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,0)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	0	0	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

155

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 1 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,1)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	0	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

156

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.	maxPeso: 2 Kg.
valores:	20	12	21	37	35	€	

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(2,2)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	21	**21**	0	0	0	0	0	20	20	20	20	20	20	20	20
i=1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

157

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.	maxPeso: 3 Kg.
valores:	20	12	21	37	35	€	

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(2,3)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	21	21	**21**	0	0	0	0	20	20	20	20	20	20	20	20
i=1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

158

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 4 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,4)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	0	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

159

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 5 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,5)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	0	0	20	20	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

160

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.	maxPeso: 7 Kg.
valores:	20	12	21	37	35	€	

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(2,7)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	21	21	21	21	33	33	**33**	20	20	20	20	20	20	20	20
i=1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

161

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.	maxPeso: 8 Kg.
valores:	20	12	21	37	35	€	

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(2,8)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	21	21	21	21	33	33	33	**33**	20	20	20	20	20	20	20
i=1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

162

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 9 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,9)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	20	20	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

163

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 11 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,11)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	20	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

164

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 12 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,12)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	20	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

165

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 13 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,13)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	20	20
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

166

Problema de la Mochila 0-1

Relación de recurrencia

i:2

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 15 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(2,15)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=1	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

167

Problema de la Mochila 0-1

Relación de recurrencia

i:3

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 0 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

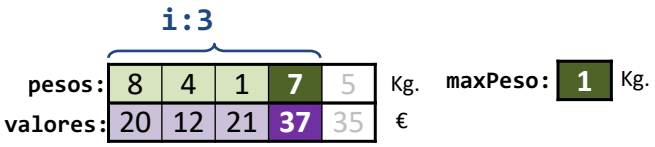
maxValor(3,0)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	0	0	0	12	12	12	12	20	20	20	20	32	32	32	32

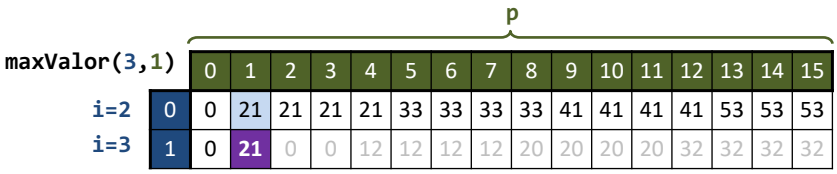
168

Problema de la Mochila 0-1

Relación de recurrencia



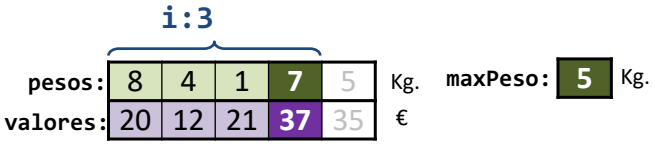
$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$



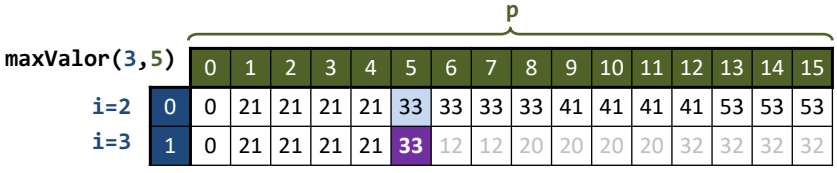
169

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$



170

Problema de la Mochila 0-1

Relación de recurrencia

i:3

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 7 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(3,7)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	20	20	20	20	32	32	32	32

171

Problema de la Mochila 0-1

Relación de recurrencia

i:3

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 8 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

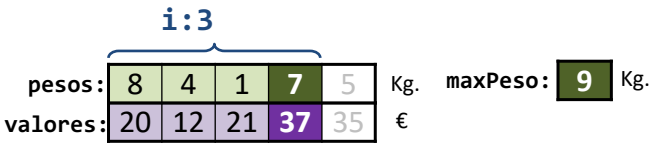
p

maxValor(3,8)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	20	20	20	32	32	32	32

172

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

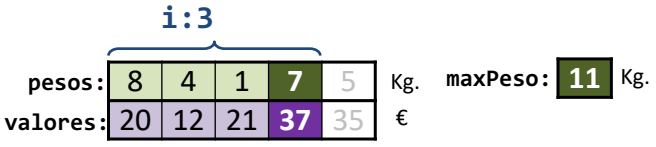
maxValor(3,9)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	20	20	32	32	32	32

173

Problema de la Mochila 0-1

Relación de recurrencia



$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(3,11)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	32	32	32	32

174

Problema de la Mochila 0-1

Relación de recurrencia

i:3

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 12 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(3,12)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	32	32	32

175

Problema de la Mochila 0-1

Relación de recurrencia

i:3

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 13 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(3,13)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	32	32	32

176

Problema de la Mochila 0-1

Relación de recurrencia

i:3

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 15 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(3,15)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=2	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

177

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 0 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,0)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

178

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 1 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(4,1)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

179

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 4 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

maxValor(4,4)

	p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	33	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

180

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 5 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,5)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	33	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

181

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 6 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,6)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	33	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

182

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 7 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,7)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	56	33	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

183

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 8 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,8)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	56	58	41	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

184

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 9 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,9)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	56	58	58	41	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

185

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 10 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,10)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	56	58	58	68	41	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

186

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 11 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,11)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	56	58	58	68	68	41	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

187

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 12 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,12)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	56	58	58	68	68	72	53	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

188

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 13 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,13)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	56	58	58	68	68	72	**93**	53	53
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

189

Problema de la Mochila 0-1

Relación de recurrencia

i:4

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

maxPeso: 15 Kg.

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

p

maxValor(4,14)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=4	0	0	21	21	21	21	35	56	56	58	58	68	68	72	93	93	**93**
i=3	1	0	21	21	21	21	33	33	37	58	58	58	58	70	70	70	70

190

Problema de la Mochila 0-1

Relación de recurrencia

Actividad 14.7. Implementa el algoritmo que determina el valor máximo que puede llevar la mochila

```
int mochila01(int[] pesos, int[] valores, int maxPeso){
  int[][] maxValor = new int[2][maxPeso+1];
  for (int p=0;p<=maxPeso; p++)
     if (pesos[0]<=p) maxValor[0][p]=valores[0]; else maxValor[0][p]=0;
  int iaux = 0;
  for (int i=1;i<valores.length; i++) {
    int iauxAnt= (i-1) % 2;
    iaux = i % 2;
    for (int p=0;p<=maxPeso; p++) {
       if (pesos[i]>p)
          maxValor[iaux][p] = maxValor[iauxAnt][p];
       else
          maxValor[iaux][p] = Math.max(maxValor[iauxAnt][p],
                                       valores[i]+maxValor[iauxAnt][p-pesos[i]]);
    }
  }
  return maxValor[(valores.length-1)%2][maxPeso];
}
```

191

Problema de la Mochila 0-1

Relación de recurrencia

Actividad 14.7. Implementa el algoritmo que determina el valor máximo que puede llevar la mochila

```
int mochila01(int[] pesos, int[] valores, int maxPeso){
  int[][] maxValor = new int[2][maxPeso+1];
  for (int p=0;p<=maxPeso; p++)
     if (pesos[0]<=p) maxValor[0][p]=valores[0]; else maxValor[0][p]=0;
  int iaux = 0;
  for (int i=1;i<va                    i++) {
    int iauxAnt= (i
    iaux = i % 2;
    for (int p=0;p<
       if (pesos[i]
          maxValor[iaux][p] = maxValor[iauxAnt][p];
       else
          maxValor[iaux][p] = Math.max(maxValor[iauxAnt][p],
                                       valores[i]+maxValor[iauxAnt][p-pesos[i]]);
    }
  }
  return maxValor[(valores.length-1)%2][maxPeso];
}
```

Caso Base:

$$maxValor(0,p) = \begin{cases} valores[0] & si\ pesos[0] \leq p \\ 0 & en\ otro\ caso \end{cases}$$

192

Problema de la Mochila 0-1

Relación de recurrencia

Actividad 14.7. Implementa el algoritmo que determina el valor máximo que puede llevar la mochila

```
int mochila01(int[] pesos, int[] valores, int maxPeso){
  int[][] maxValor = new int[2][maxPeso+1];
  for (int p=0;p<=maxPeso; p++)
     if (pesos[0]<=p) maxValor[0][p]=valores[0]; else maxValor[0][p]=0;
```

Caso Recursivo:

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

```
      if (pesos[i]>p)
         maxValor[iaux][p] = maxValor[iauxAnt][p];
      else
         maxValor[iaux][p] = Math.max(maxValor[iauxAnt][p],
                                      valores[i]+maxValor[iauxAnt][p-pesos[i]]);
    }
  }
  return maxValor[(valores.length-1)%2][maxPeso];
}
```


Problema de la Mochila 0-1

Relación de recurrencia

Actividad 14.8. Calcula la complejidad del algoritmo en tiempo según el número de bienes, *N*, y el valor del máximo peso, *K* (maxPeso).

$\Theta(N \cdot K)$

```
int mochila01(int[] pesos, int[] valores, int maxPeso){
  int[][] maxValor = new int[2][maxPeso+1];
  for (int p=0;p<=maxPeso; p++)
     if (pesos[0]<=p) maxValor[0][p]=valores[0]; else maxValor[0][p]=0;
  int iaux = 0;
  for (int i=1;i<valores.length; i++) {
    int iauxAnt= (i-1) % 2;
    iaux = i % 2;
    for (int p=0;p<=maxPeso; p++) {
       if (pesos[i]>p)
          maxValor[iaux][p] = maxValor[iauxAnt][p];
       else
          maxValor[iaux][p] = Math.max(maxValor[iauxAnt][p],
                                       valores[i]+maxValor[iauxAnt][p-pesos[i]]);
    }
  }
  return maxValor[(valores.length-1)%2][maxPeso];
}
```

Problema de la Mochila 0-1

Relación de recurrencia

Actividad 14.9. **Calcula la complejidad del algoritmo en tiempo según el número de bits, *k*, necesarios para almacenar el valor del máximo peso, *K* (maxPeso).**


```
int mochila01(int[] pesos, int[] valores, int maxPeso){
  int[][] maxValor = new int[2][maxPeso+1];
  for (int p=0;p<=maxPeso; p++)
     if (pesos[0]<=p) maxValor[0][p]=valores[0]; else maxValor[0][p]=0;
  int iaux = 0;
  for (int i=1;i<valores.length; i++) {
    int iauxAnt= (i-1) % 2;
    iaux = i % 2;
    for (int p=0;p<=maxPeso; p++) {
       if (pesos[i]>p)
          maxValor[iaux][p] = maxValor[iauxAnt][p];
       else
          maxValor[iaux][p] = Math.max(maxValor[iauxAnt][p],
                                       valores[i]+maxValor[iauxAnt][p-pesos[i]]);
    }
  }
  return maxValor[(valores.length-1)%2][maxPeso];
}
```

$\Theta(N \cdot K)$

$\Theta(N \cdot 2^k)$

Algoritmo pseudopolinomial

195

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

maxPeso: 15 Kg.

pesos:	8	4	1	7	5	Kg.
valores:	20	12	21	37	35	€

Algoritmo para Mochila 0-1 (Programación Dinámica)

Valor máximo

Indica el valor máximo que se puede alcanzar

¿Qué objetos se introducen en la mochila?

196

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5
valores:	20	12	21	37	35

maxPeso: **15**

Algoritmo para Mochila 0-1 (Programación Dinámica)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

197

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	**1**	7	5
valores:	20	12	**21**	37	35

maxPeso: **10**

Algoritmo para Mochila 0-1 (Programación Dinámica)

Sí se mete en la mochila el objeto **#2** que pesa **1** y vale **21**.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

198

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5	maxPeso: 11
valores:	20	12	21	37	35	

Algoritmo para Mochila 0-1 (Programación Dinámica)

No se mete en la mochila el objeto **#1** que pesa **4** y vale **12**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

199

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5	maxPeso: 15
valores:	20	12	21	37	35	

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

200

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5	maxPeso: 15
valores:	20	12	21	37	35	
mochila:						

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

3. Problema de la Mochila 0-1

1 — 2 — 3

201

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5	maxPeso: 10
valores:	20	12	21	37	35	
mochila:					1	

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

3. Problema de la Mochila 0-1

1 — 2 — 3

202

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5
valores:	20	12	21	37	35
mochila:				1	1

maxPeso: 3

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

203

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5
valores:	20	12	21	37	35
mochila:			1	1	1

maxPeso: 2

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

204

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5	maxPeso: 2
valores:	20	12	21	37	35	
mochila:		0	1	1	1	

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

205

Problema de la Mochila 0-1

Conociendo los objetos que se introducen

pesos:	8	4	1	7	5	maxPeso: 2
valores:	20	12	21	37	35	
mochila:	0	0	1	1	1	

$$maxValor(i, p) = \begin{cases} maxValor(i-1, p) & \text{si } pesos[i] > p \\ \max\{maxValor(i-1, p),\ valores[i] + maxValor(i-1, p - pesos[i])\} & \text{en otro caso} \end{cases}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
1	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
2	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
3	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
4	F	F	F	F	F	T	T	T	F	F	T	T	T	T	T	T

206

Problema de la Mochila 0-1

Obteniendo los objetos que se entregan

Actividad 14.10. **Modifica el algoritmo para que se determine qué objetos se introducen en la mochila.**

207

Problema de la Mochila 0-1

Obteniendo los objetos que se entregan

```
boolean[] mochila01(int[] pesos, int[] valores, int maxPeso){
  int[][] maxValor = new int[2][maxPeso+1];
  boolean[][] aux = new boolean[valores.length][maxPeso+1];
  for (int p=0;p<=maxPeso; p++)
     if (pesos[0]<=p) {
         maxValor[0][p]=valores[0]; aux[0][p] = true;}
     else {
         maxValor[0][p]=0; aux[0][p]= false;}
  int iaux = 0;
  for (int i=1;i<valores.length; i++) {
    int iauxAnt= (i-1) % 2;
    iaux = i % 2;
    for (int p=0;p<=maxPeso; p++) {
       if (pesos[i]>p)
          maxValor[iaux][p] = maxValor[iauxAnt][p];
       else
          maxValor[iaux][p] = Math.max(maxValor[iauxAnt][p],
                                       valores[i]+maxValor[iauxAnt][p-pesos[i]]);
       aux[i][p] = !(maxValor[iaux][p] == maxValor[iauxAnt][p]);
    }
  }
  return objetosIntroducidos(aux, pesos);
}
```

208

Problema de la Mochila 0-1

Obteniendo los objetos que se entregan

```
boolean[] mochila01(int[] pesos, int[] valores, int maxPeso){
  int[][] maxValor = new int[2][maxPeso+1];
  boolean[][] aux = new boolean[valores.length][maxPeso+1];
  for (int p=0;p<=maxPeso; p++)
     if (pesos[0]<=p) {
        maxValor[0][p]=valores[0]; aux[0][p] = true;}
     else {
        maxValor[0][p]=0; aux[0][p]= false;}
  int iaux = 0;
  for (int i=1;i<valores.length; i++) {
     int iauxAnt= (i-1) % 2;
```

```
boolean[] objetosIntroducidos (boolean[][] aux, int[] pesos){
  int p=aux[0].length-1;
  boolean[] decision = new boolean[pesos.length];
  for (int i=pesos.length-1; i>=0; i--) {
     decision[i] = aux[i][p];
     if (aux[i][p]==true)  p = p - pesos[i];
  }
  return decision;
}
```

```
  return objetosIntroducidos(aux, pesos);
}
```

1 2 3

209