

# Advanced School in Artificial Intelligence

## **Constraint Processing: Il vincolo element**

**Marco Gavanelli**

**marco.gavanelli@unife.it**



*Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021*

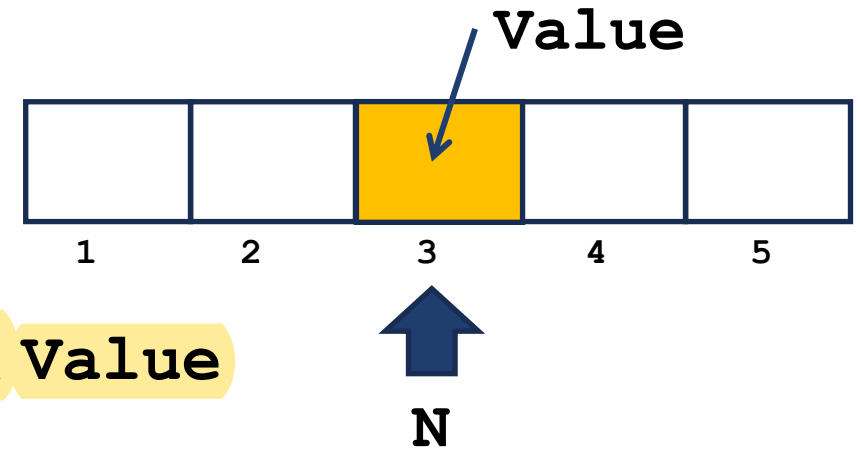



**Università  
degli Studi  
di Ferrara**

## VINCOLO element

- **element(N, [X<sub>1</sub>, ..., X<sub>m</sub>], Value)**

l'**N**-esimo elemento della lista è uguale a **Value**



- propagazione da **N** a **Value** : 
  - $N=i \rightarrow X_i = \text{Value}$
- propagazione da **Value** a **N**:
  - Se  $\text{Value} = x \rightarrow N$  ha nel dominio gli  $i$  per cui  $\text{Value} = X_i$

## Esempio

- Si hanno 5 prodotti, identificati con i numeri da 1 a 5
- Ciascun prodotto ha un prezzo:
- Prodotto 1: prezzo: 10
- Prodotto 2: prezzo: 5
- Prodotto 3: prezzo: 6
- Prodotto 4: prezzo: 8
- Prodotto 5: prezzo: 11

10	5	6	8	11
1	2	3	4	5

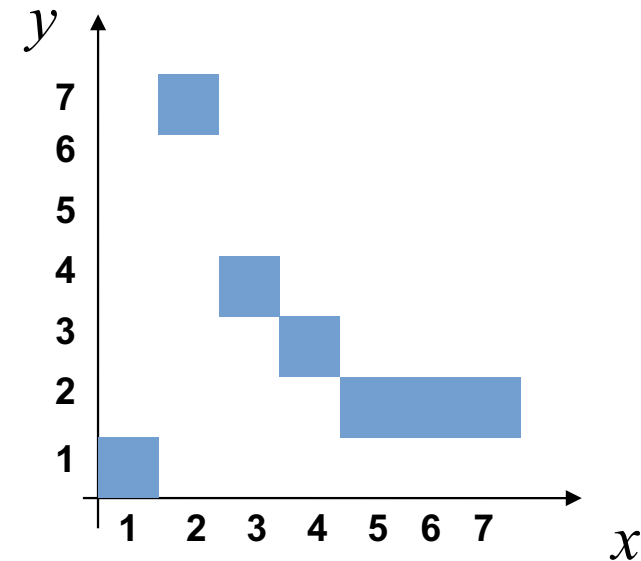
- Si desidera scegliere 2 prodotti diversi il cui prezzo totale sia inferiore a 15

```
var 1..5:X;  var int : PrezzoX;  
constraint element(X,[10,5,6,8,11],PrezzoX) ;  
var 1..5:Y;  var int : PrezzoY;  
constraint element(Y,[10,5,6,8,11],PrezzoY) ;  
constraint X != Y;  
constraint PrezzoX + PrezzoY <15;  
solve satisfy;
```

## Funzioni

Il vincolo `element` può essere usato per definire qualsiasi dipendenza funzionale di due variabili

`element(X, [1,7,4,3,2,2,2], Y)`



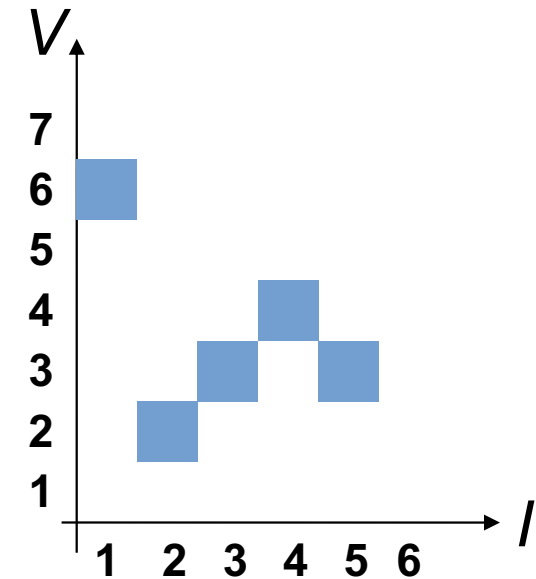


## Vincolo element: **propagazione**

**element** ( $I, L, V$ )

- Se l'elemento  $j \in D(I)$  allora l'elemento  $j$ -esimo di  $L$  (chiamiamolo  $L_j$ ) non deve essere cancellato dal dominio di  $V$
- Se l'elemento  $k \in D(V)$  allora tutti gli indici  $j$  tali che  $L_j = k$  non vanno eliminati dal dominio di  $I$
- Tutti gli altri elementi (che non rispettano le regole precedenti) vanno eliminati

```
var {1, 3, 4, 5} : I;    var {2, 3, 6} : V;  
constraint element(I, [6, 2, 3, 4, 3], V);  
Quali valori rimangono nel dominio dopo l'AC?
```



## element( $I, L, V$ ): propagazione naive

Per tutti  $j \in D(I)$

$L_j$  è nel dominio di  $V$ ?

Se no  $\rightarrow$  elimina  $j$  da  $D(I)$

Per tutti  $k \in D(V)$

Trova i valori  $j$  tali che  $L_j = k$

Se nessuno di questi è nel  $D(I)$

$\rightarrow$  elimina  $k$  da  $D(V)$

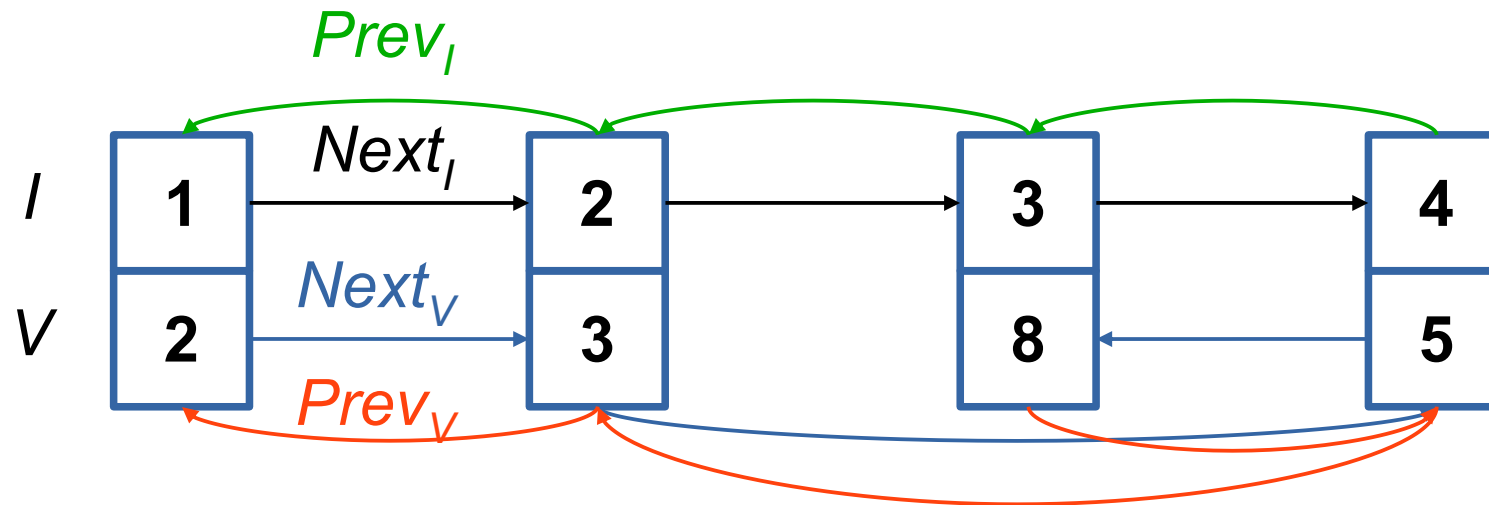
Complessità?

Supponiamo che  
 $D(I)$ ,  $D(V)$  e  $L$   
abbiano  $n$  elementi



## Element: propagazione

- Es: `element(I, [2, 3, 8, 5], V)`



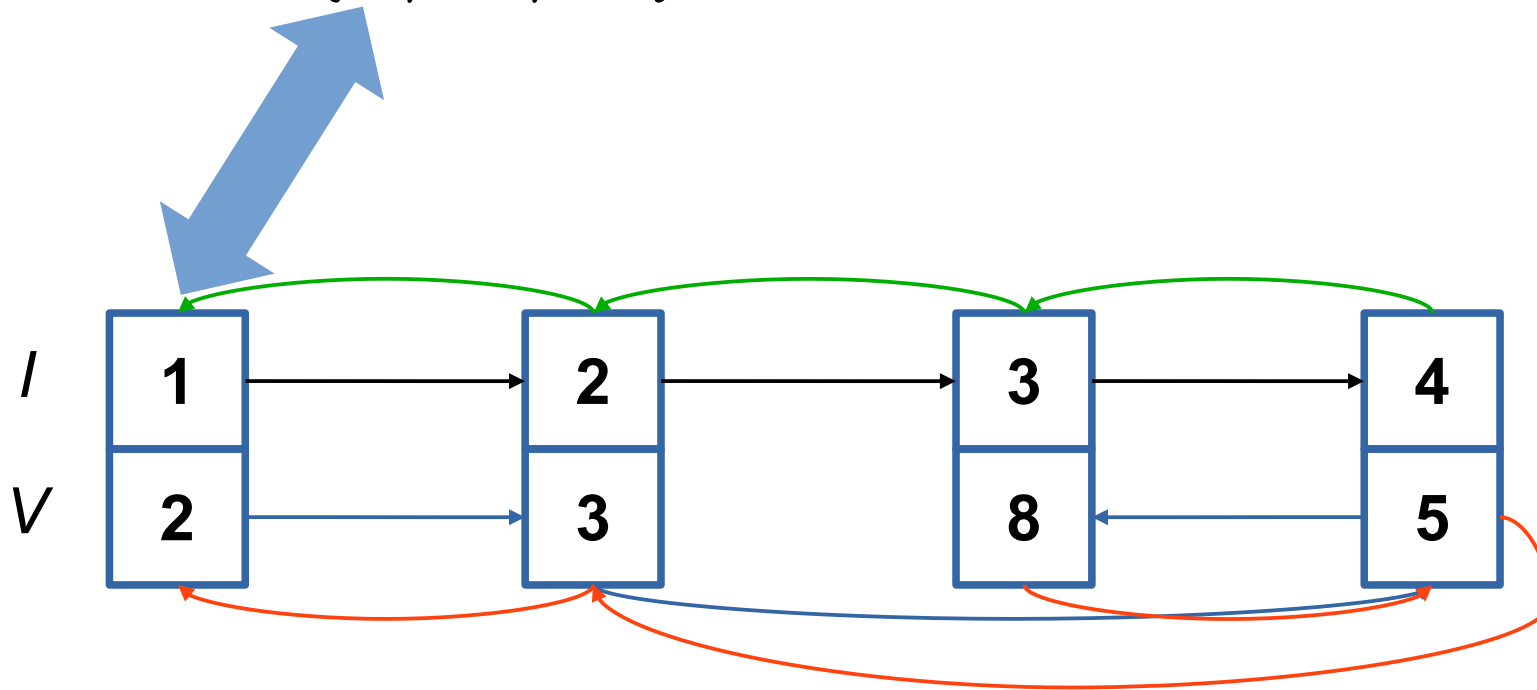
## Element: propagazione

- Scorri contemporaneamente il dominio di  $I$  e la lista  $Next_I$ 
  - Se c'è un valore nella lista e non nel dominio, elimina il valore dalla lista
- Lo stesso con  $D(V)$  e la lista  $Next_V$



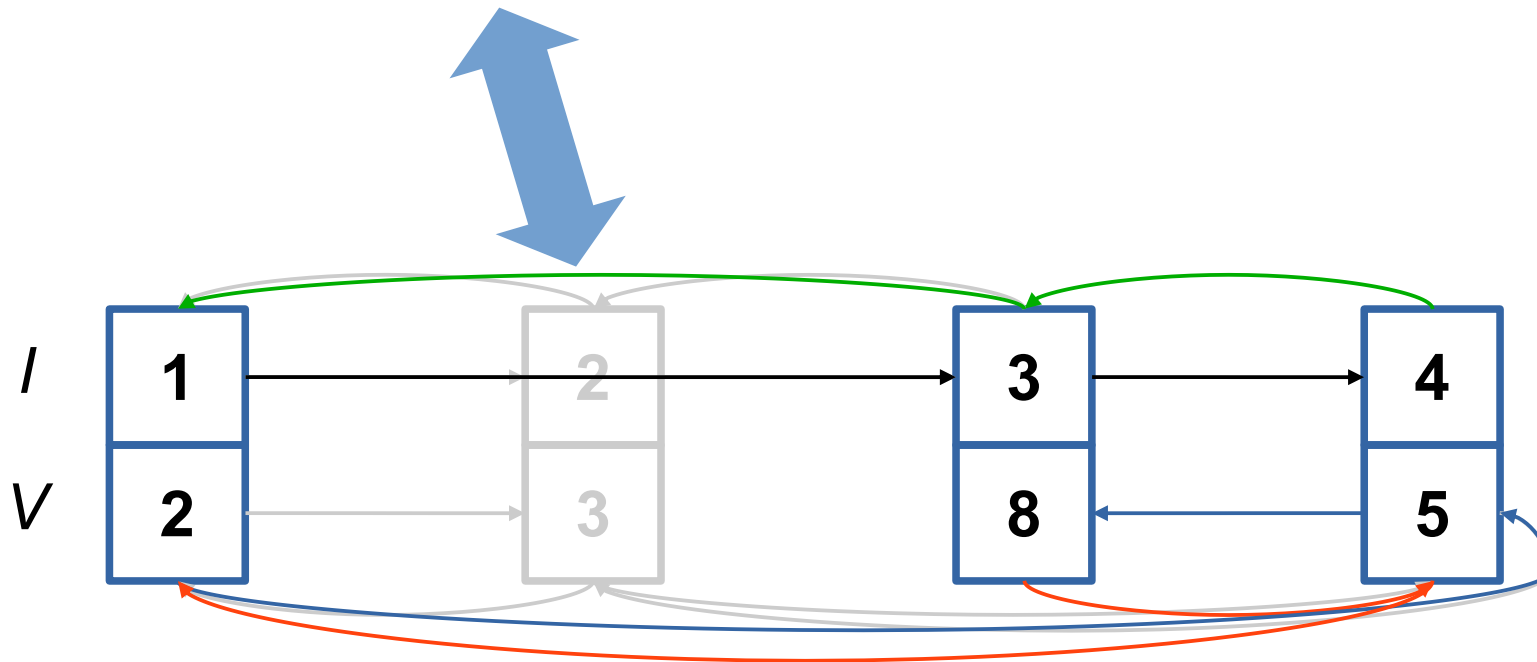
## Element: propagazione

- Es: `element(I, [2, 3, 8, 5], V)`  
`var {1, 3, 4} : I;`



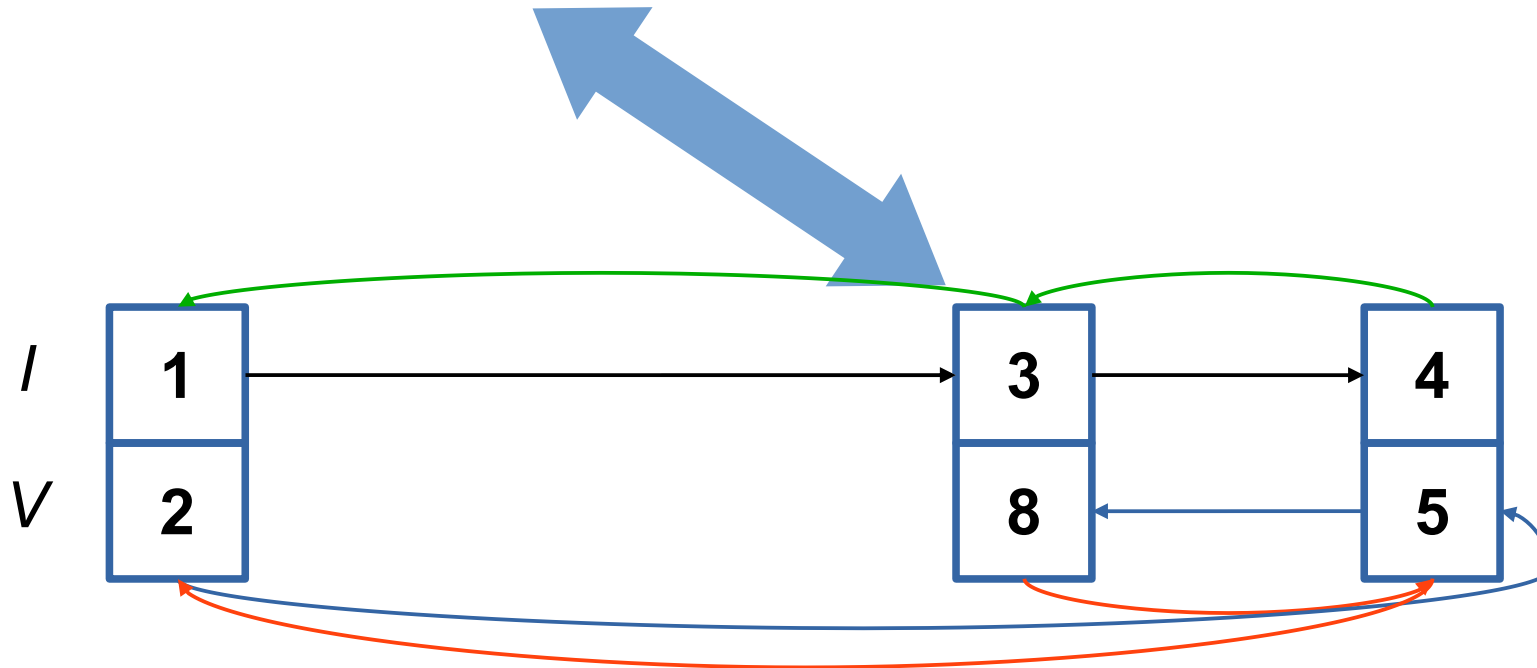
## Element: propagazione

- Es: `element(I, [2, 3, 8, 5], V)`  
`var {1, 3, 4} : I;`



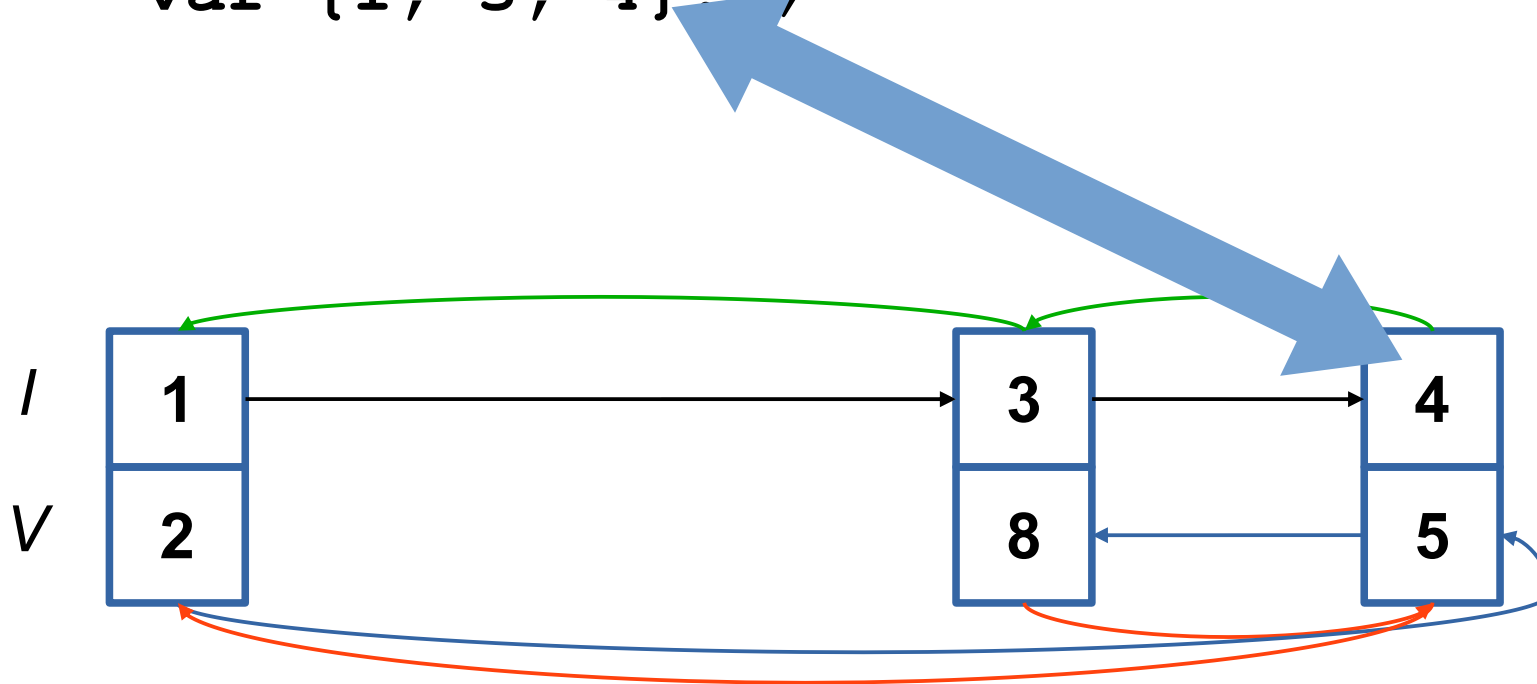
## Element: propagazione

- Es: `element(I, [2, 3, 8, 5], V)`  
`var {1, 3, 4} : I;`



## Element: propagazione

- Es:     `element(I, [2, 3, 8, 5], V)`  
      `var {1, 3, 4} : T;`

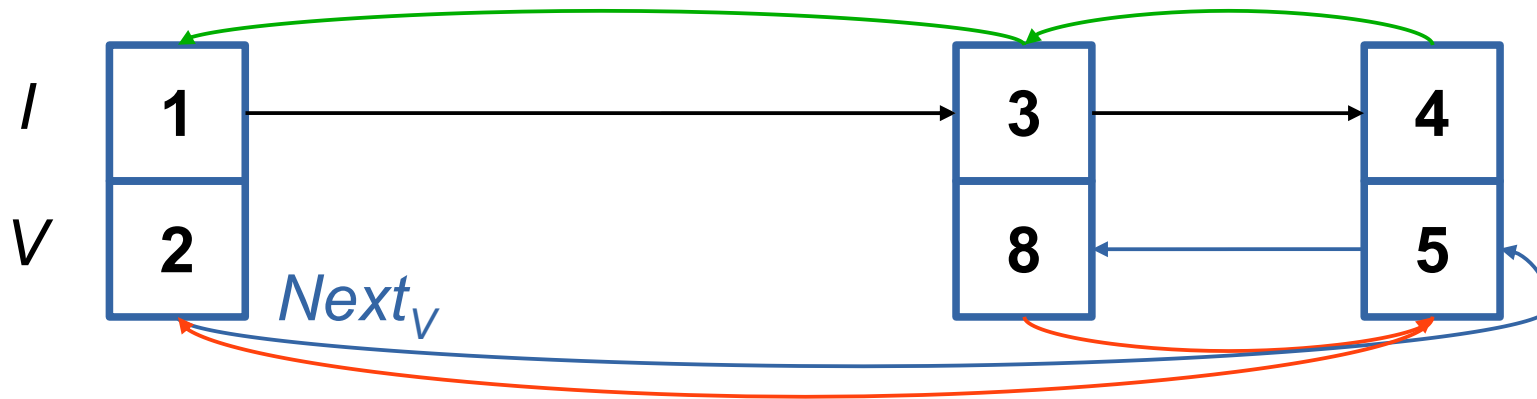


## Element: propagazione

- Es: `element(I, [2, 3, 8, 5], V)`

`var {1, 3, 4} : I;`

A questo punto, seguendo la lista  $Next_V$  si hanno in ordine i valori accettabili per  $V$



## Element: propagazione

	Algoritmo	Complessità
Ciclo eseguito $O(n)$ volte	<ul style="list-style-type: none"><li>• Scorri contemporaneamente <math>D(I)</math> e la lista <math>Next_I</math><ul style="list-style-type: none"><li>• Se c'è un valore nella lista e non nel dominio, elimina il valore dalla lista</li></ul></li></ul>	$n \times$ $1 +$
	<ul style="list-style-type: none"><li>• Scorri contemporaneamente <math>D(V)</math> e <math>Next_V</math><ul style="list-style-type: none"><li>• Se c'è un valore nella lista e non nel dominio, elimina il valore dalla lista</li></ul></li></ul>	$n \times$ $1 +$
$O(1)$ con liste doppiamente concatenate	<ul style="list-style-type: none"><li>• Infine, scorrendo <math>Next_I</math> si ha il dominio di <math>I</math> (in ordine)</li></ul>	$n +$
	<ul style="list-style-type: none"><li>• Scorrendo <math>Next_V</math> si ha il dominio di <math>V</math> (in ordine)</li></ul>	$n =$ $O(n)$



## Esercizio: element

- Sia dato il seguente problema di soddisfacimento di vincoli:

`var 2..5:I;`

`var 1..5:V;`

`constraint element(I, [1, 4, 2, 3, 6], V);`

- Si mostri la propagazione effettuata dall'Arc-Consistency.
- Cosa succede se si aggiunge il vincolo  
 $I > V$ ?

## Indicizzazione di array

- In MiniZinc il vincolo **element** viene sistematicamente utilizzato per indicizzare gli array
- ```
var 1..5 : i;  
var 1..10: x;  
array [1..5] of int: c = [6,5,3,1,2];
```
- ```
constraint c[i]=x;
```
- viene tradotto in
- ```
constraint element(i,c,x);
```
- in genere non c'è quindi bisogno di utilizzare esplicitamente il vincolo element; in MiniZinc si può direttamente indicizzare un array con una variabile con dominio.