Divisione intera

Il codice per il calcolo iterativo della divisione intera

```
class Divisione{
/**
Dati in ingresso: interi X,D
Dati in uscita: interi q,r
Condizione di ingresso: X >= 0, D > 0
Condizione di uscita: X = q * D + r && r < D
*/
public static void main (String [] args){
int X = 14;
int D = 3;
int q = 0;
int r = X;
while (r >= D) {  //invariante: ?
       q = q + 1;
       r = r - D;
}
System.out.println(q + "   " + r);
}
}
```

Il codice per il calcolo iterativo della divisione intera

```
class Divisione{
/**
Dati in ingresso: interi X,D
Dati in uscita: interi q,r
Condizione di ingresso: X >= 0, D > 0
Condizione di uscita: X = q * D + r && r < D
*/
public static void main (String [] args){
int X = 14;
int D = 3;;
int q = 0;
int r = X;
while (r >= D) {  //invariante: X = q * D + r
       q = q + 1;
       r = r - D;
}
System.out.println(q + "   " + r);
}
}
```

Invarianti

Dato un ciclo generico, per esempio della forma

```
while (C)
      S
```

dove `C` è una espressione booleana (la condizione del ciclo) e `S` una istruzione (il corpo del ciclo), un

invariante del ciclo

è una relazione R tra alcune delle variabili utilizzate nel ciclo (ed eventualmente altri valori) che è vera dopo un numero arbitrario $n \geqq 0$ di iterazioni del corpo S del ciclo.

Correttezza

Supponiamo che **prima** di una iterazione generica la relazione $X = q * D + r$ sia vera, e dimostriamo che è vera anche con i valori presi da q e r **dopo** questa iterazione.

Dopo l'iterazione:

$$q' = q + 1$$
$$r' = r - D$$

Allora:

$$\begin{aligned} q' * D + r' &= (q + 1) * D + (r - D) \\ &= q * D + D + r - D \\ &= q * D + r \\ &= X \end{aligned}$$

Inoltre, poiché $D > 0$ per la condizione di ingresso, ad ogni iterazione $r' < r$, quindi il ciclo termina.

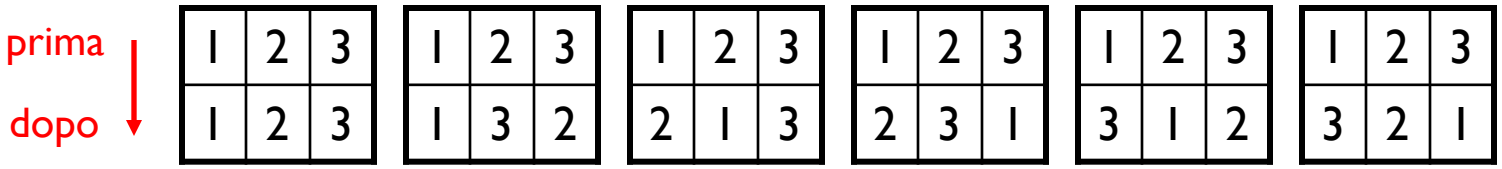
Il fattoriale

Calcolo del fattoriale

Il fattoriale è la funzione definita su N come:

$$n! = 1 * 2 * \ldots * n$$

con 0! = 1
Conta il numero delle permutazioni di un insieme di n oggetti; per esempio, per n = 3:



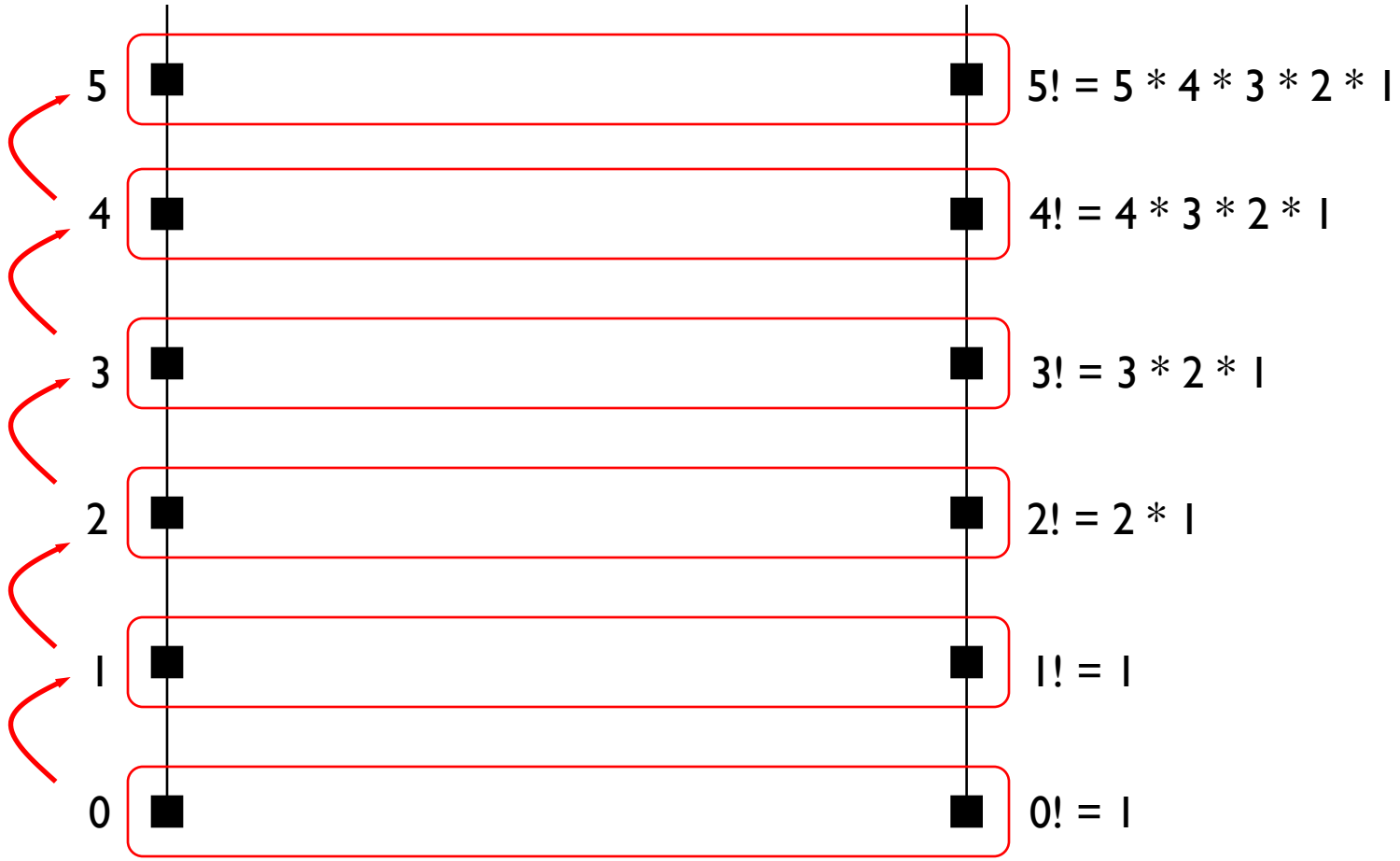
Una funzione f : **N** $\rightarrow$ **N** è descritta da un insieme di coppie ordinate della forma (x,y), dove y = f(x).

Questo insieme si chiama il **grafo** di f.

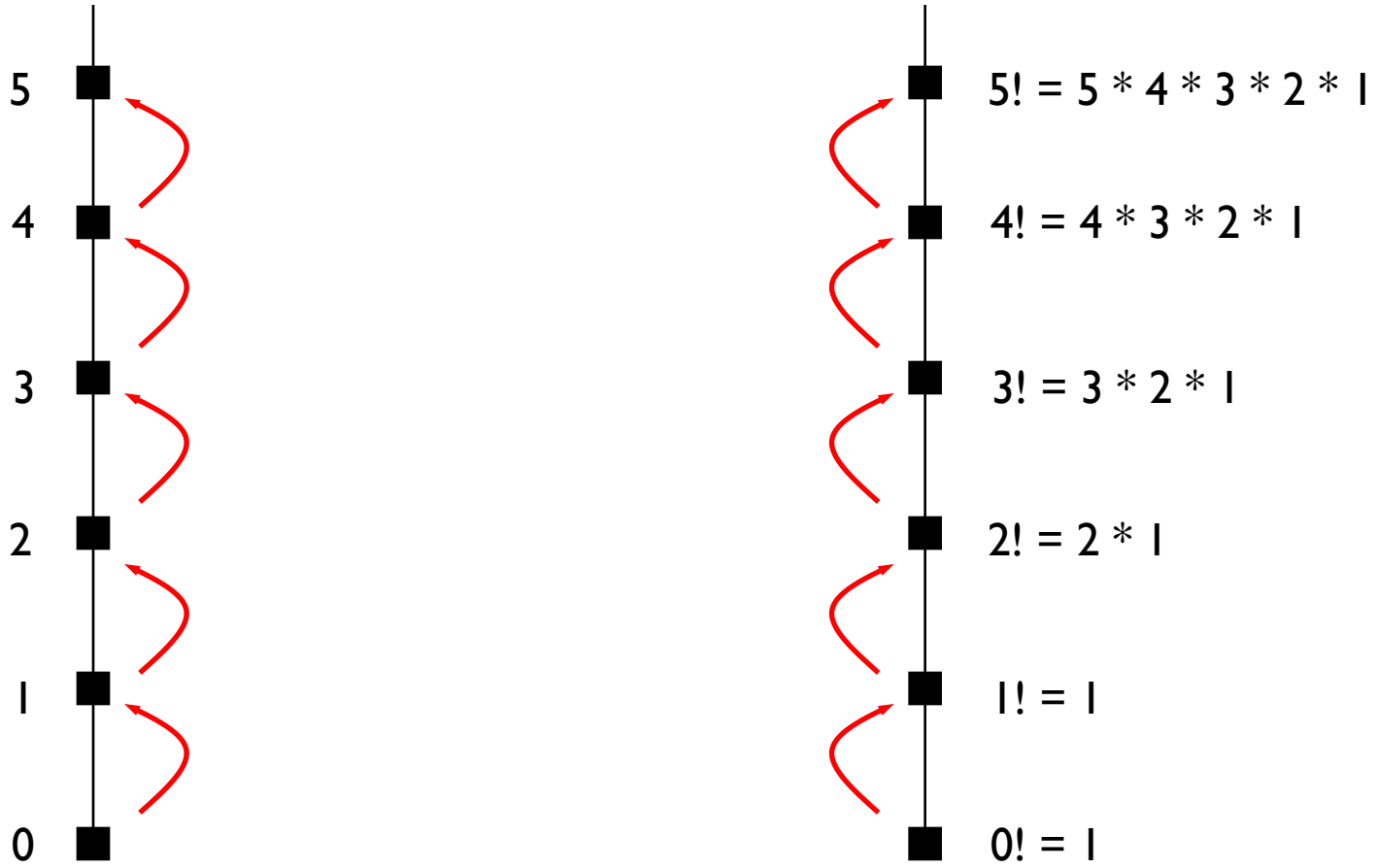
Per implementare il fattoriale, si descrive mediante un ciclo while l'esplorazione del suo grafo attraverso transizioni della forma

$$(x,y) \rightarrow (x + 1 , (x + 1) * y)$$

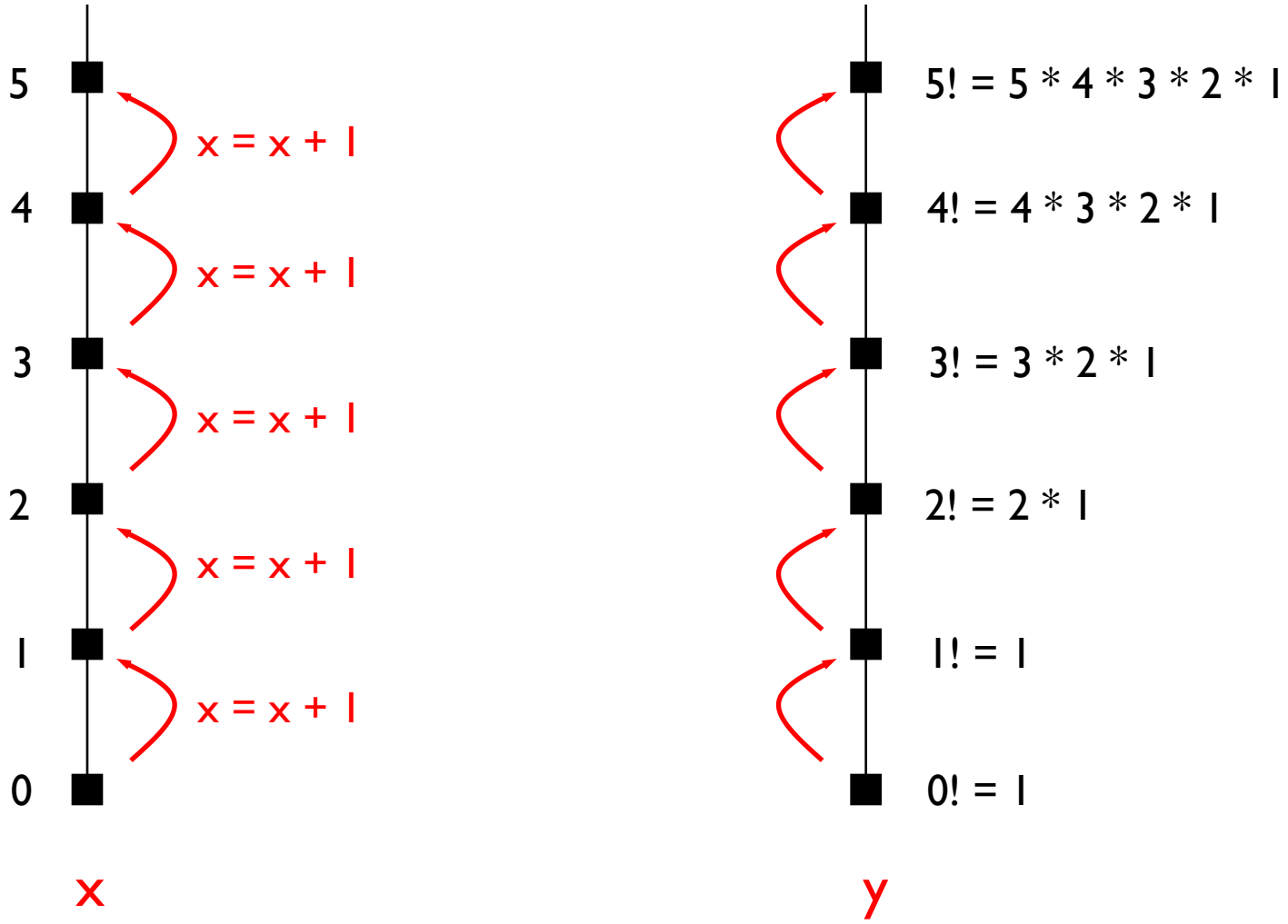
Calcolo del fattoriale



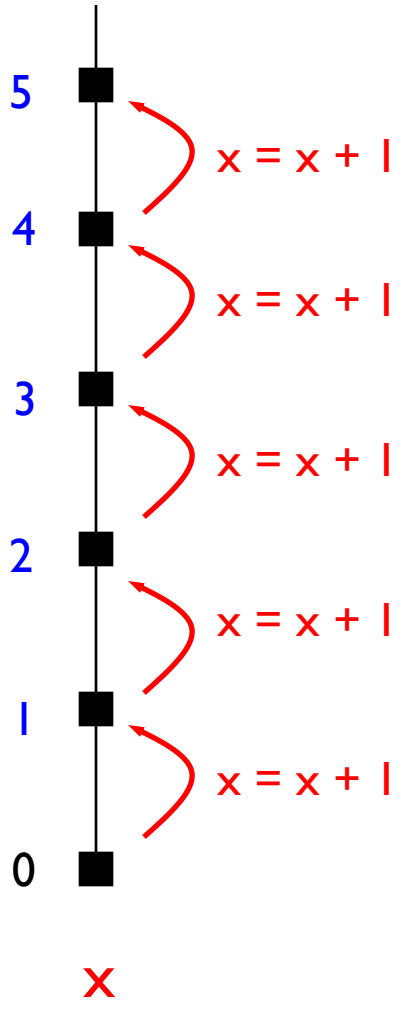
Calcolo del fattoriale

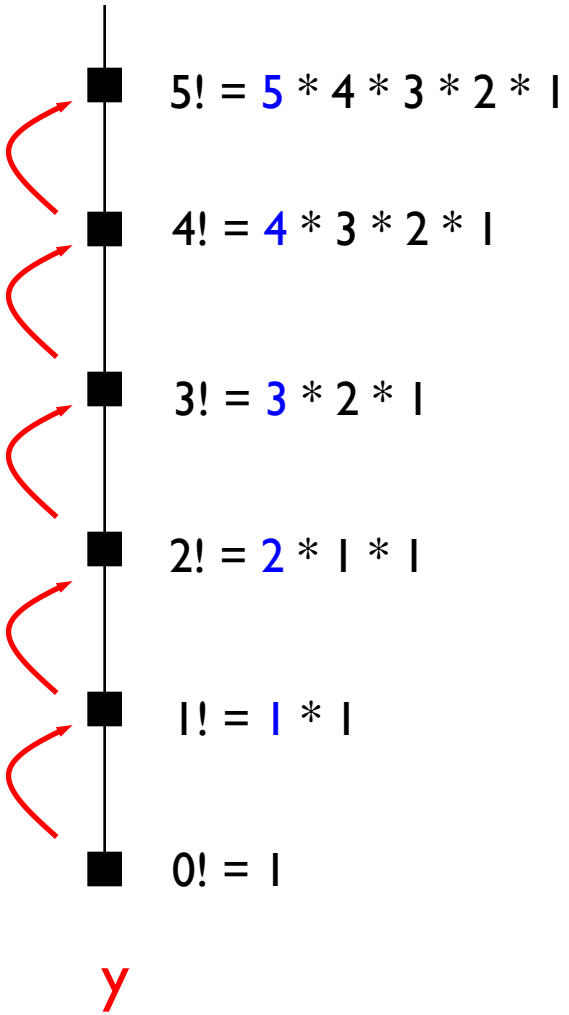


Calcolo del fattoriale

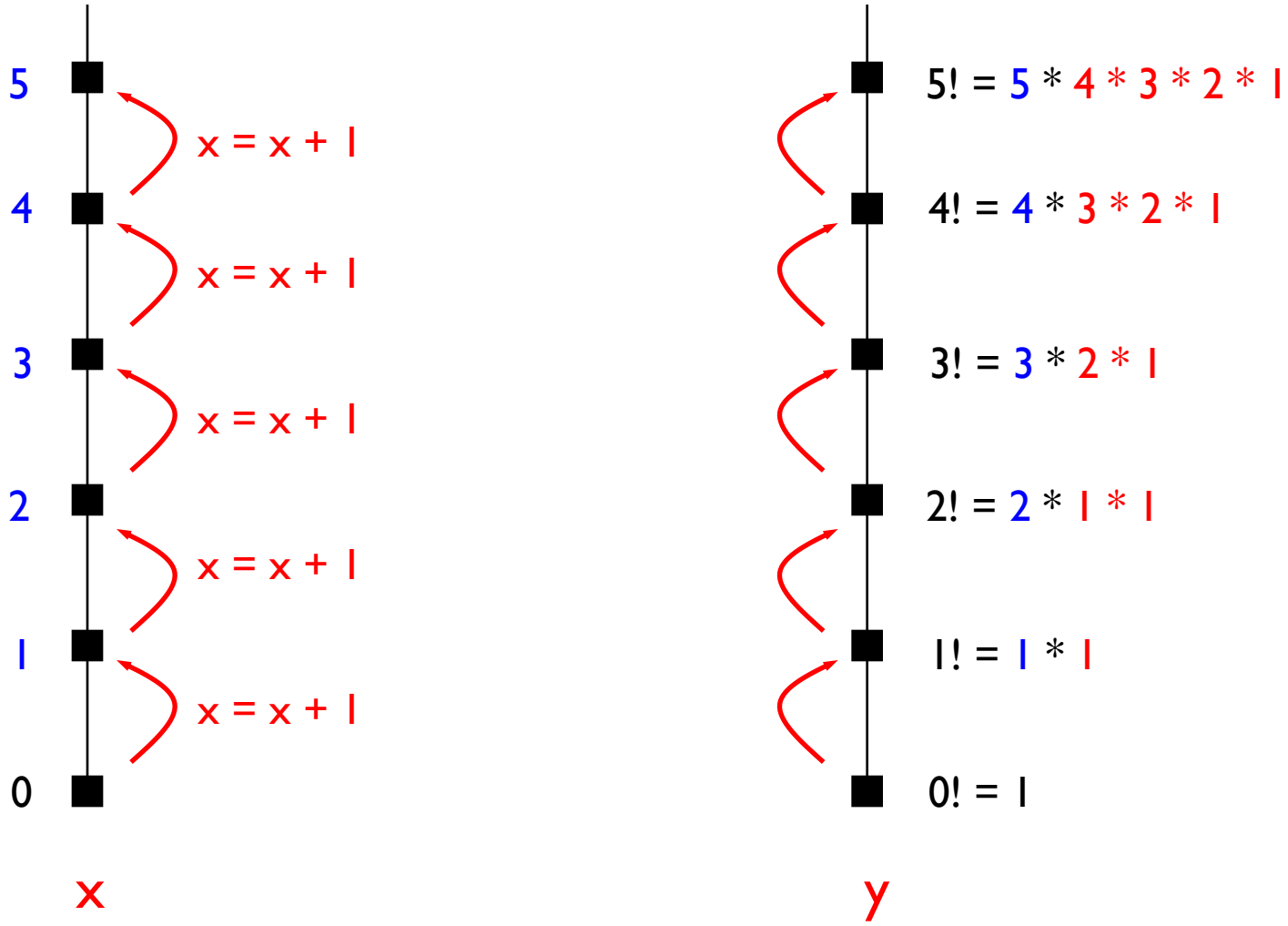


Calcolo del fattoriale

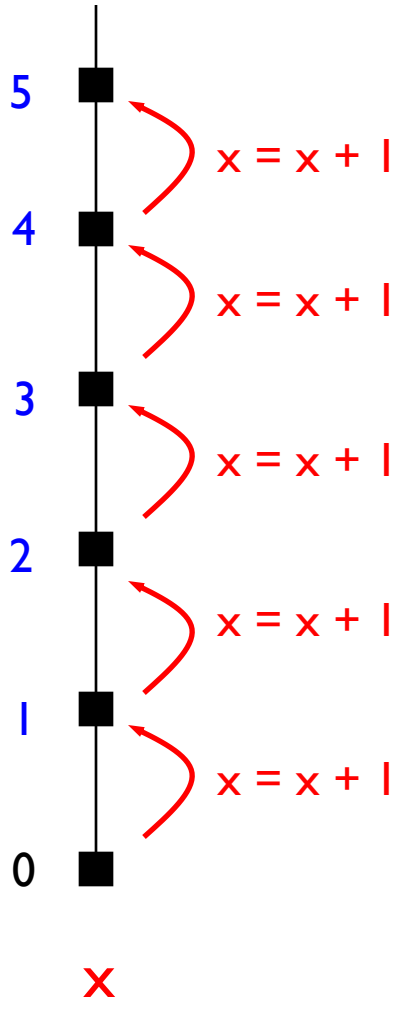


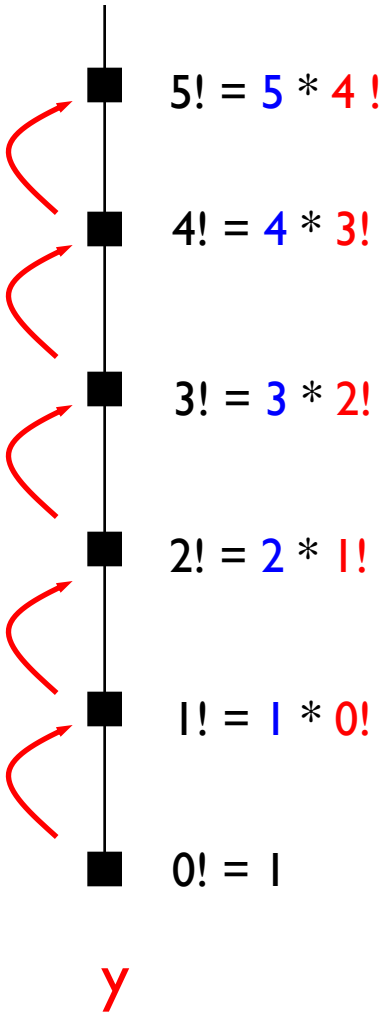


Calcolo del fattoriale

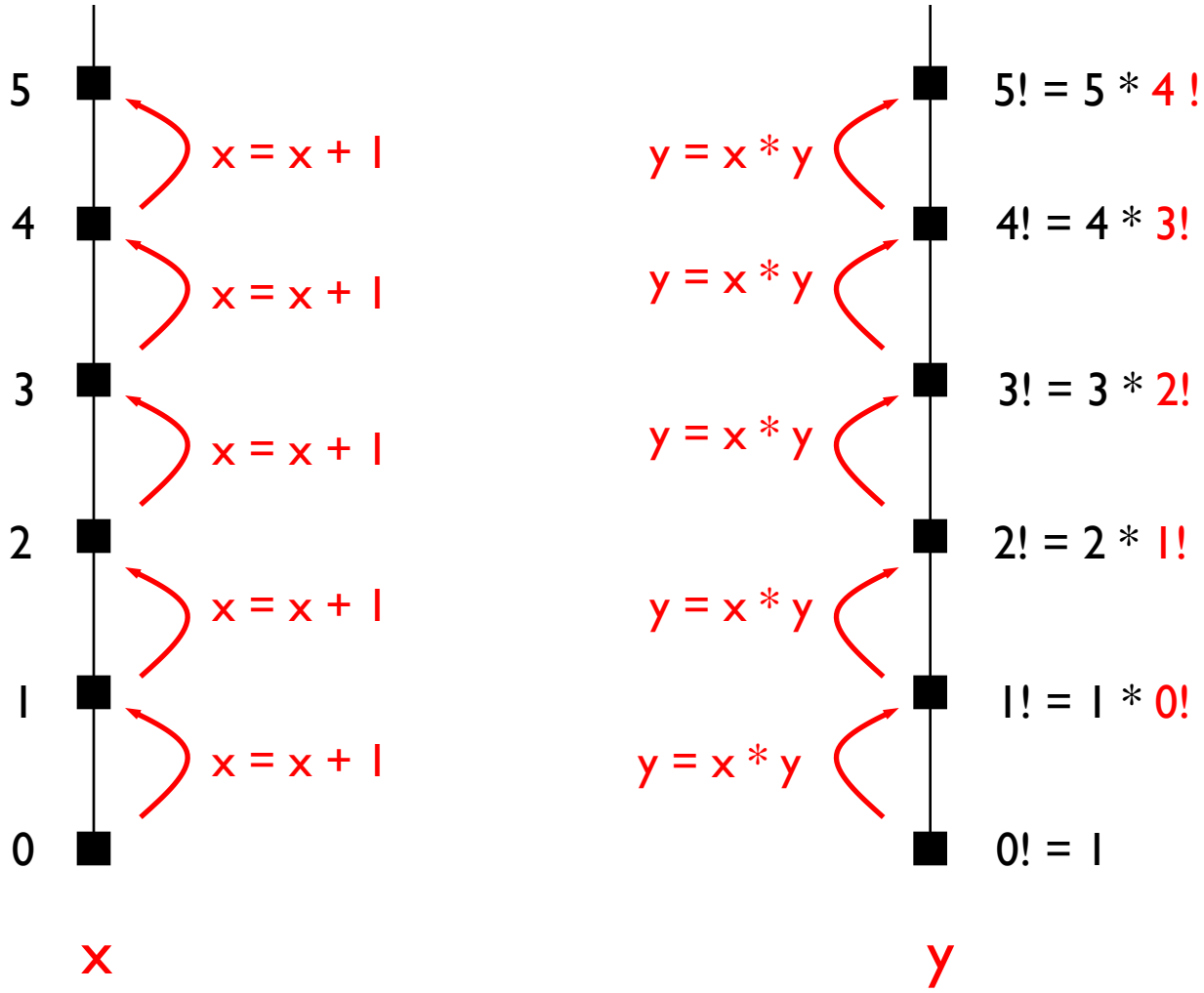


Calcolo del fattoriale





Calcolo del fattoriale



Il codice per il calcolo iterativo del fattoriale

```
class Fattoriale {
/**
Dati in ingresso: interi i >= 0
Dati in uscita: interi y >= 0
Condizione di ingresso: true
Condizione di uscita: y = i!
*/
public static void main (String [] args){
int i = 10;
int x = 0;
int y = 1;
while (x < i) {  //invariante: ?
       x = x + 1;
       y = x * y;
}
System.out.println("Il fattoriale di " + i + " è: " + y);
}
}
```

Il codice per il calcolo iterativo del fattoriale

```
class Fattoriale {
/**
Dati in ingresso: interi i >= 0
Dati in uscita: interi y >= 0
Condizione di ingresso: true
Condizione di uscita: y = i!
*/
public static void main (String [] args){
int i = 10;
int x = 0;
int y = 1;
while (x < i) {   //invariante: y = x!
       x = x + 1;
       y = x * y;
}
System.out.println("Il fattoriale di " + i + " è: " + y);
}
}
```

$0! \cdot 1 = 1!$

$n! \cdot n+1 \overset{x\,def}{=} n+1!$

Correttezza

Supponiamo che **prima** di una iterazione generica la relazione y = x! sia vera, e dimostriamo che è vera anche con i valori presi da x e y **dopo** questa iterazione.

Dopo l'iterazione:

$$x' = x + 1$$
$$y' = x' * y$$

Allora:

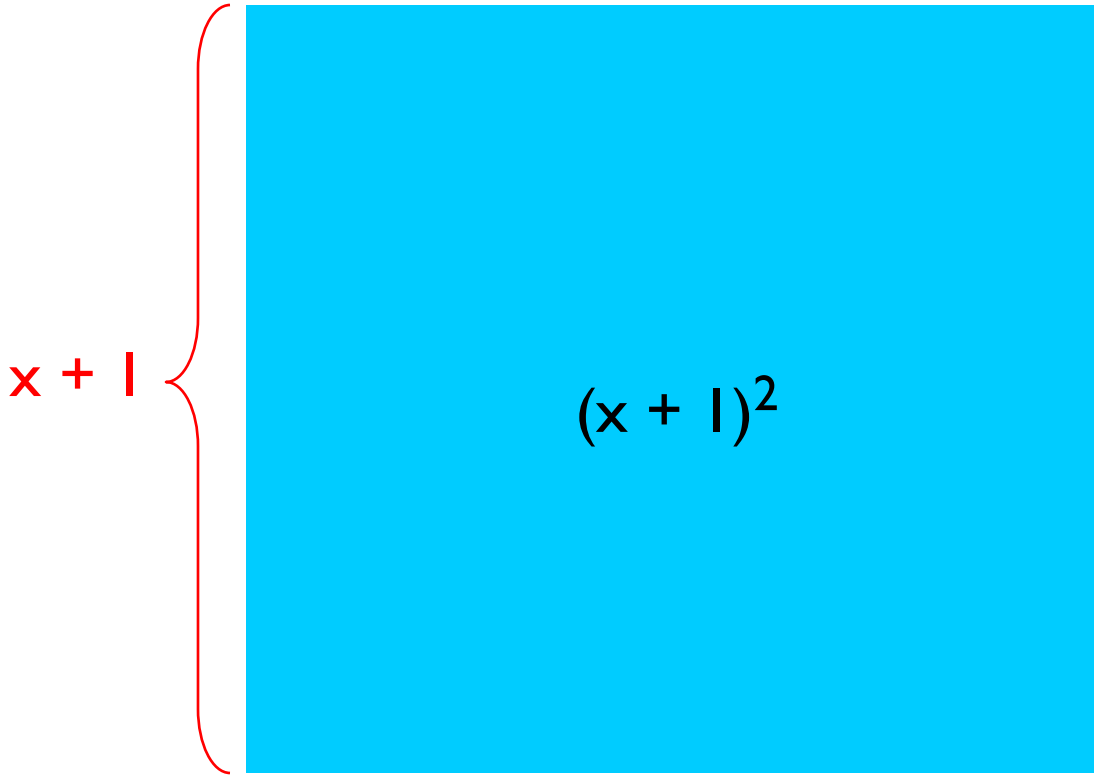
$$y' = x' * y$$
$$= (x + 1) * y$$
$$= (x + 1) * x!$$
$$= (x + 1)!$$

All'uscita dal ciclo, x = i, quindi y = x! = i!

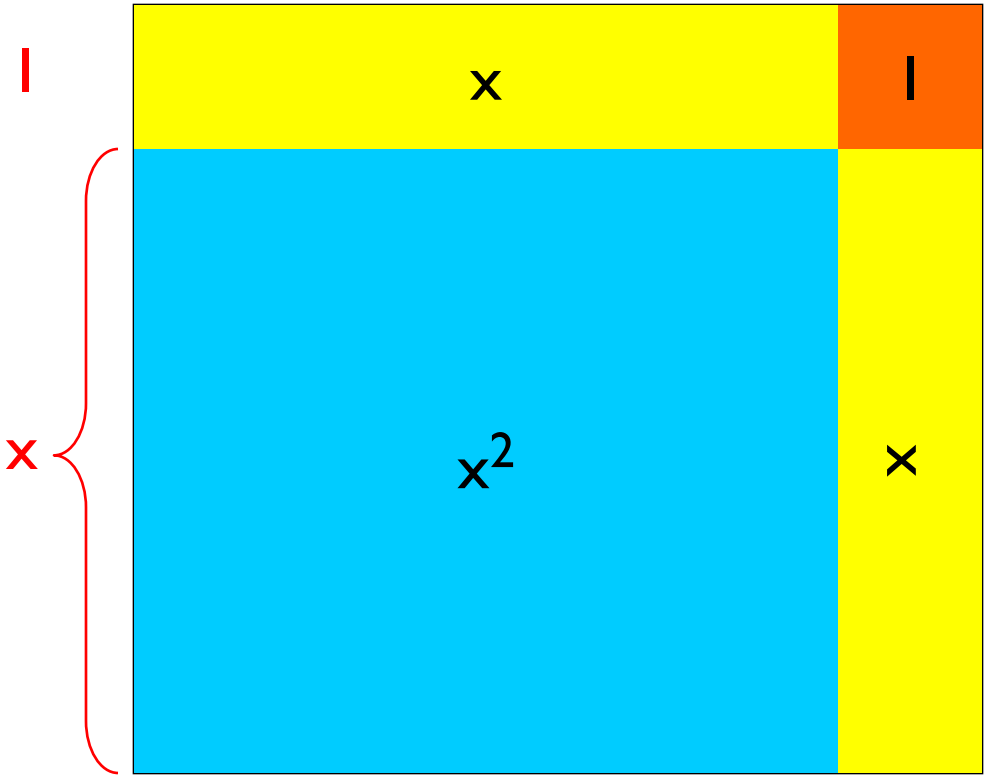
Il ciclo termina perché la quantità i - x decresce strettamente ad ogni iterazione, perciò il programma è totalmente corretto.

Il quadrato

Calcolo del quadrato



Calcolo del quadrato



$$(x + 1)^2 = x^2 + 2x + 1$$

Il codice per il calcolo iterativo del quadrato

```
class Quadrato{
/**
Dati in ingresso: interi x >= 0
Dati in uscita: interi q >= 0
Condizione di ingresso: true
Condizione di uscita: q = x*x
*/
public static void main (String[] args){
int x = 7;
int n = 0;
int q = 0;
while (n < x){   //invariante: ?
       q = q + 2 * n + 1;
       n = n + 1;
}
System.out.println("Il quadrato di " + x + " è: " + q);
}
}
```

Il codice per il calcolo iterativo del quadrato

```
class Quadrato{
/**
Dati in ingresso: interi x >= 0
Dati in uscita: interi q >= 0
Condizione di ingresso: true
Condizione di uscita: q = x*x
*/
public static void main (String[] args){
int x = 7;
int n = 0;
int q = 0;
while (n < x){   //invariante: q = n*n
       q = q + 2 * n + 1;
       n = n + 1;
}
System.out.println("Il quadrato di " + x + " è: " + q);
}
}
```

Correttezza

Assumiamo che q = n*n prima della n-esima iterazione, dimostriamo che l'equazione
è vera dopo la n-esima iterazione. Durante la n-esima iterazione:

$$n' = n + 1$$
$$q' = q + 2n + 1$$

quindi alla fine della n-esima iterazione abbiamo

$$\begin{aligned} q' &= q + 2n + 1 \\ &= (n+1)(n+1) \text{ (per il disegno di prima)} \\ &= n' * n' \end{aligned}$$

Quando si esce dal ciclo si ha n = x, che stabilisce la condizione di uscita.

Terminazione: la quantità x - n decresce strettamente ad ogni iterazione.

Dimostrazioni?

Principio di induzione!