ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DIPARTIMENTO DI
INFORMATICA - SCIENZA E INGEGNERIA

PROGRAMMARE IL MASSIMO COMUNE DIVISORE

COSIMO LANEVE
cosimo.laneve@unibo.it

CORSO 00819 – PROGRAMMAZIONE

ARGOMENTI

1. il massimo comune divisore
2. l'algoritmo della scuola e ottimizzazioni
3. il teorema di Euclide e l'algoritmo relativo
4. ottimizzazione dell'algoritmo di Euclide
5. cenni di complessità computazionale

CALCOLARE IL TEMPO DI ESECUZIONE

- le **istruzioni elementari** sono l'assegnamento e le guardie di comandi condizionali e iterazioni
- l'esecuzione di ogni istruzione elementare ha un **costo in tempo** (= il numero di cicli del processore per eseguirlo) che è costante per quella istruzione
- **il costo in tempo** di un comando condizionale: prendere caso pessimo

```
costo_della_guardia + max(costo_ramo_then, costo_ramo_else)
```

- il **costo in tempo** di un comando iterativo: prendere caso pessimo

```
numero_iterazioni*(costo_della_guardia+costo_del_corpo)
+ costo della guardia
```

- il **costo in tempo** di un programma è la somma dei costi delle sue istruzioni
- di solito il costo in tempo è una funzione di qualche variabile in input
  - ad esempio T(*n*), oppure T(*m*,*n*)

LA COMPLESSITÀ COMPUTAZIONALE

un programma con costo in tempo T(*n*) ha

complessità computazionale O(f(*n*))

quando esistono due costanti *h*, *k* tale che

per ogni $n > h$: $T(n) < k * f(n)$

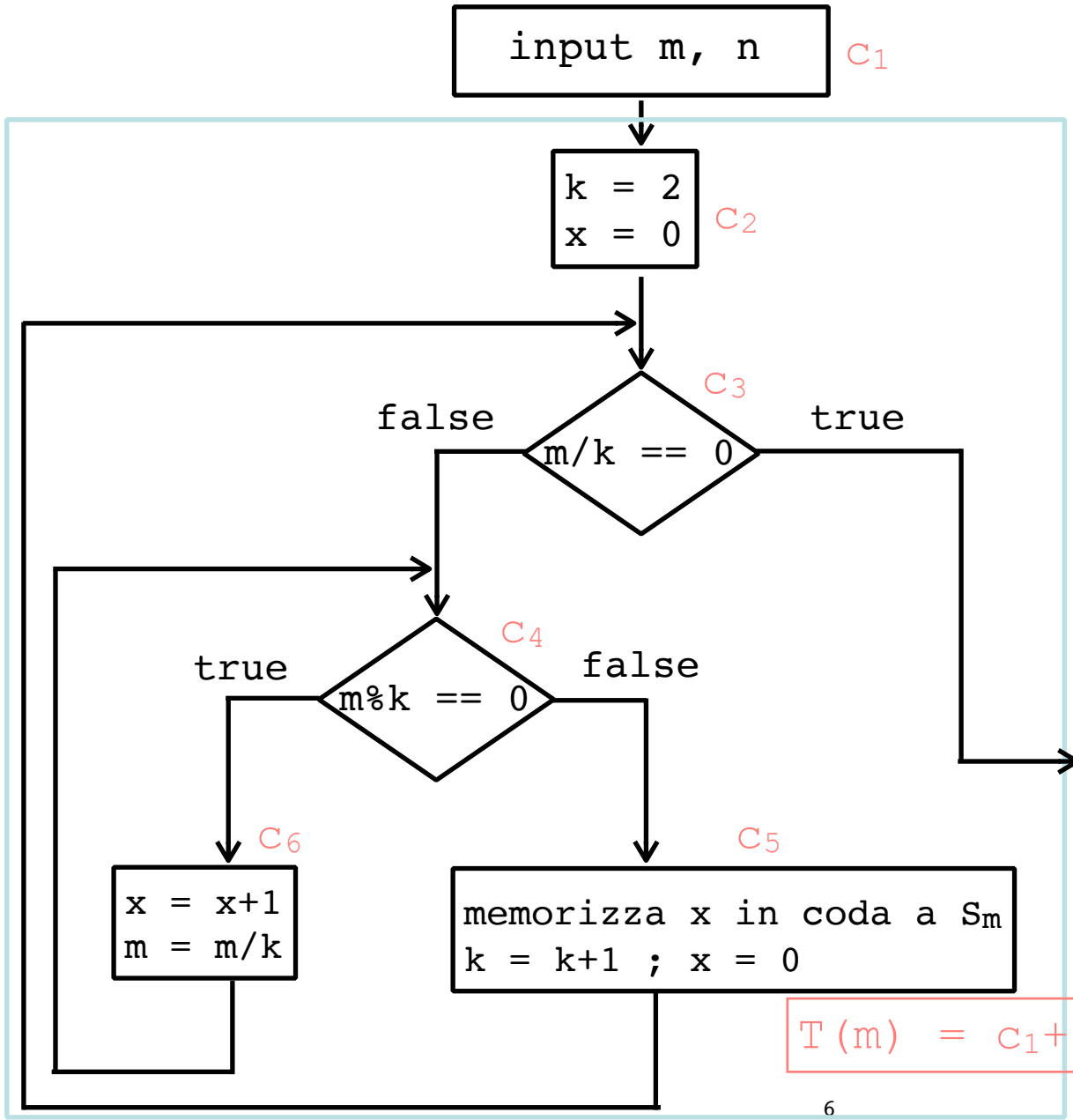
si legge O-grande di f(n)

IL MASSIMO COMUNE DIVISORE

dati due numeri, `m` ed `n`

1. il MCD è un divisore di entrambi
2. il MCD è il più grande

ALGORITMO MCD/1 – MCD DELLA SCUOLA



NOTA: per memorizzare i divisori primi si usa una sequenza di naturali: il primo elemento sarà l'esponente di 2, il secondo sarà l'esponente di 3, il terzo sarà l'esponente di 4, etc.

- **k** rappresenta l'**i**-esimo divisore primo
- **x** rappresenta l'esponente

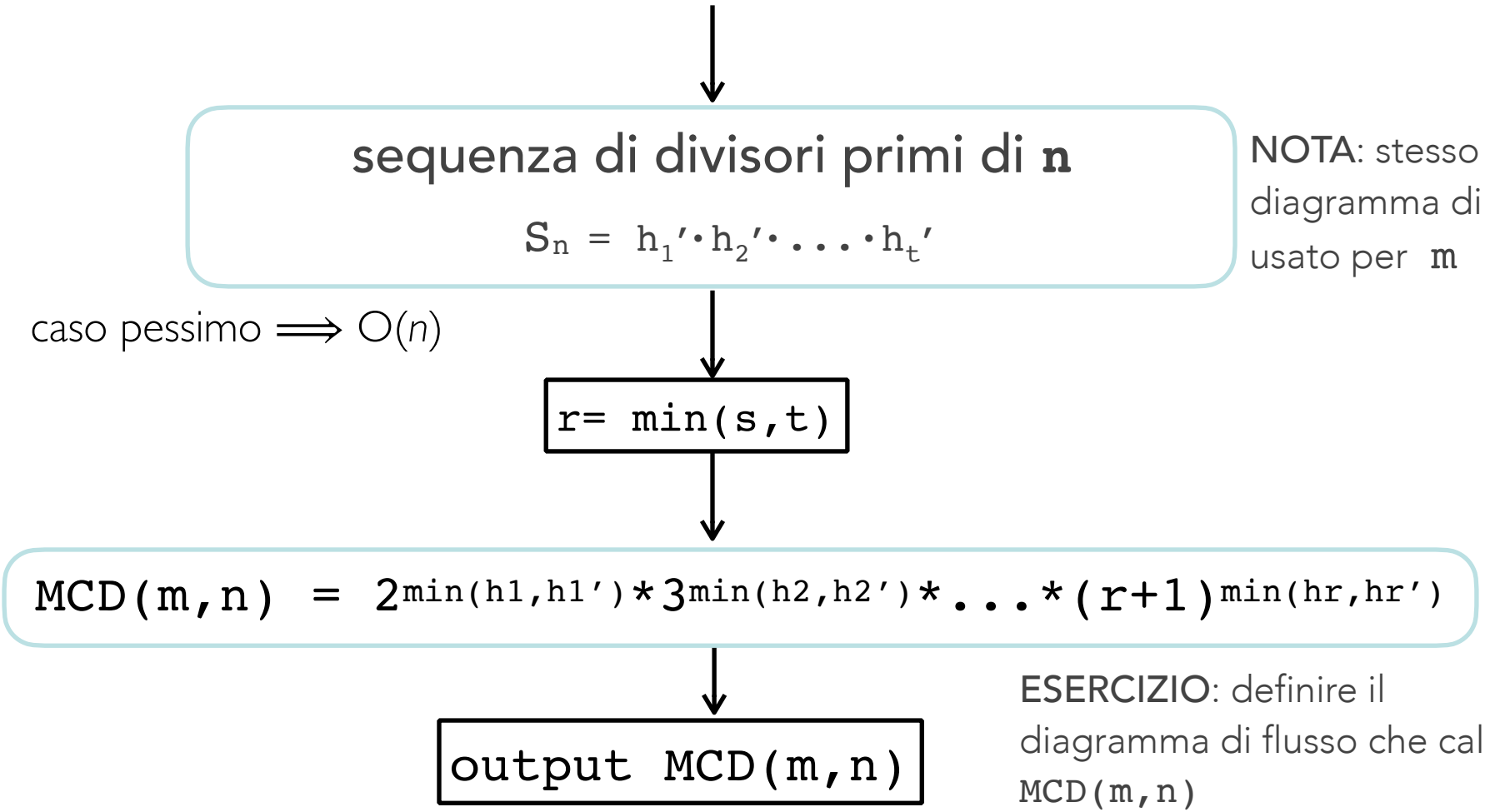
sequenza di divisori primi di n

$$S_m = h_1 \cdot h_2 \cdot \ldots \cdot h_s$$

$$T(m) = c_1 + c_2 + (c_3 + c_4 + c_5) * (m-2) + c_3$$

caso pessimo $\Longrightarrow O(m)$

ALGORITMO MCD/1 – MCD DELLA SCUOLA



NOTA: stesso diagramma di flusso usato per `m`

caso pessimo $\Longrightarrow O(n)$

ESERCIZIO: definire il diagramma di flusso che calcola `MCD(m,n)`

- il calcolo richiede `r` iterazioni (vedi codice)
- complessità computazionale $O(\min(m,n))$

complessità computazionale totale nel caso pessimo:
$O(m + n + \min(m,n))$

ALGORITMO MCD/1 – MCD DELLA SCUOLA/COMMENTI

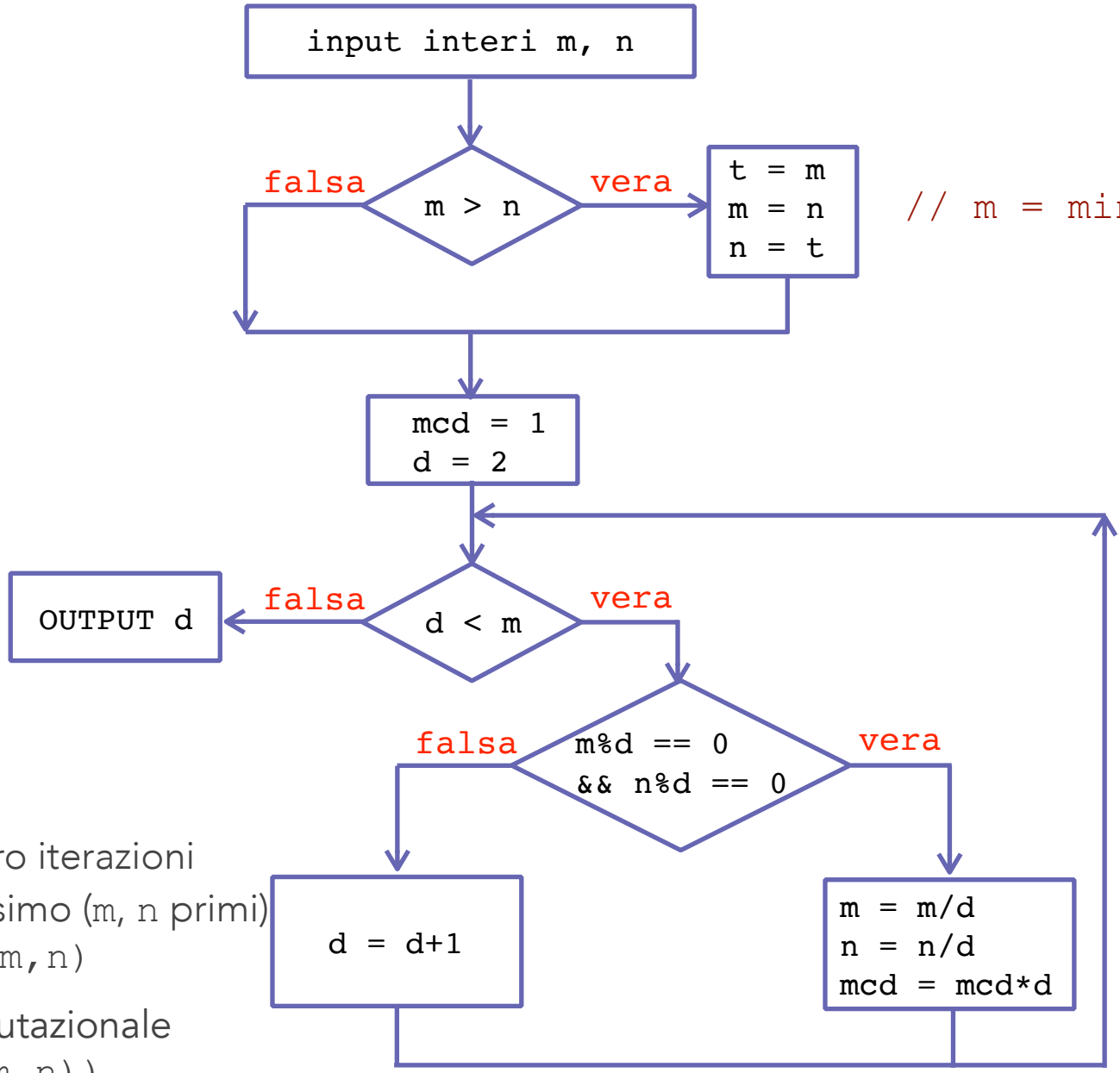
- nel calcolo dei divisori di un numero, gli unici esponenti non nulli corrisponderanno a divisori primi (non è scritto ma deriva dall'algoritmo)
- la complessità computazionale dell'algoritmo nel caso pessimo (`m` ed `n` sono **primi**) è $O(m + n + \texttt{min}(m,n))$
- l'implementazione in `C++` dell'algoritmo ha un errore…

l'algoritmo si può migliorare iterando sino a **`sqrt(m)`**

TEOREMA: un numero `m` **non è primo** se e solo se è divisibile per i numeri compresi tra `2` e **`sqrt(m)`** (estremi inclusi)

$$O(m^{1/2} + n^{1/2} + \texttt{min}(m^{1/2}, n^{1/2}))$$

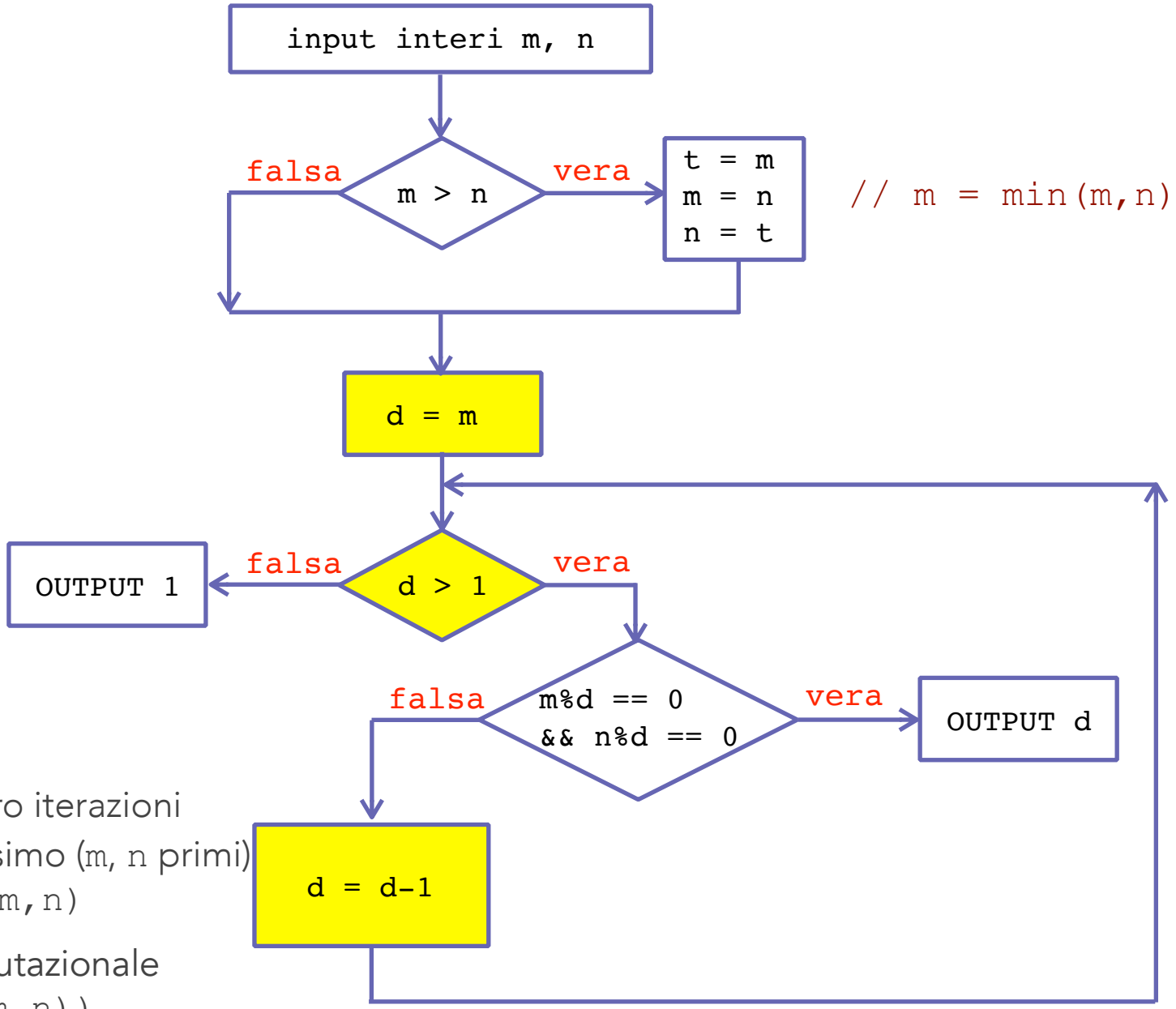
ALGORITMO MCD/2: UN METODO PIÙ DIRETTO PER IL MCD



commento: numero iterazioni eseguite caso pessimo (m, n primi) $\leq \min(m,n)$

complessità computazionale $O(\min(m,n))$

ALGORITMO MCD/2BIS: UN METODO PIÙ DIRETTO PER IL MCD



commento: numero iterazioni eseguite caso pessimo (`m`, `n` primi)

$\leq$ `min(m,n)`

complessità computazionale

`O(min(m,n))`

ALGORITMO MCD/3: IL TEOREMA DI EUCLIDE

Teorema di Euclide: se `m>n` allora `MCD(m,n) = MCD(m-n,n)`

prova: innanzitutto dimostriamo che `MCD(m,n)` è anche un divisore di `m-n`. Per definizione di `MCD`, esistono `h` e `k` tali che

```
m     = MCD(m,n)*h
n     = MCD(m,n)*k
```

Quindi

```
m-n   = MCD(m,n)*(h-k)
```

Ora dimostriamo che `MCD(m,n)` è anche il massimo divisore tra `m-n` e `n`: per assurdo assumiamo che `MCD(m,n)` non sia il massimo, perciò esiste `d>1` tale che

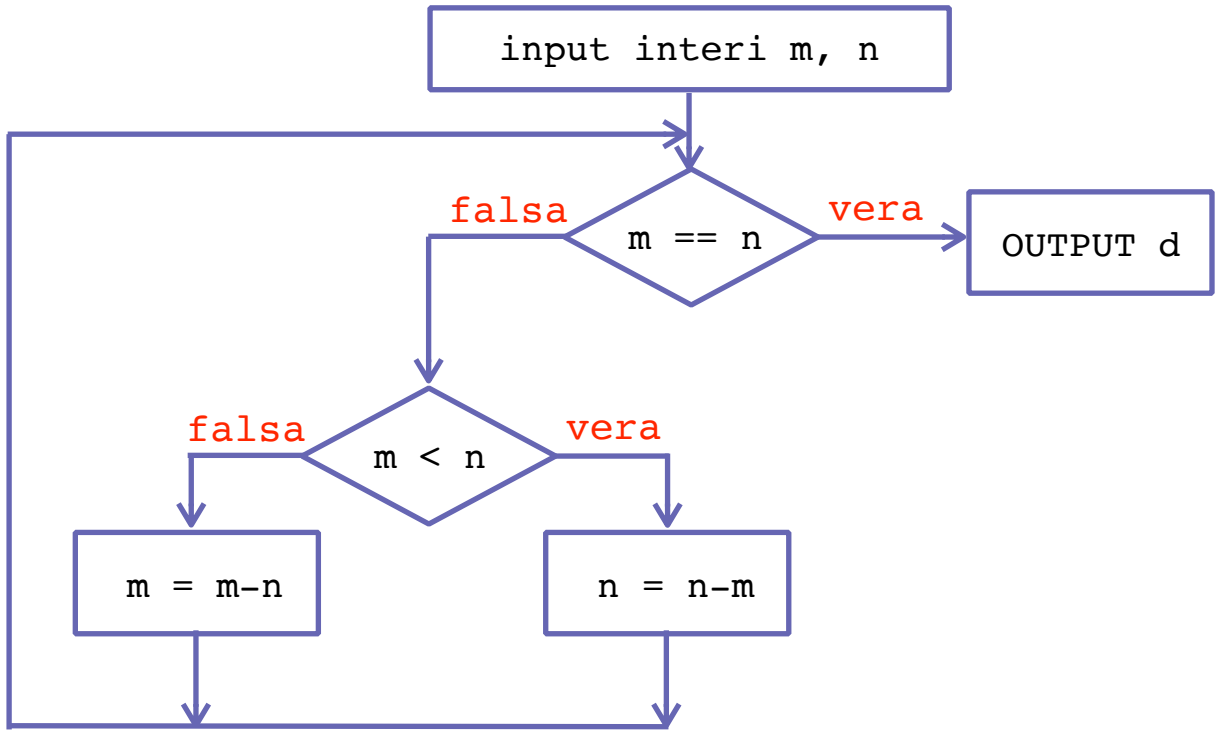
`k=d*k'` e `h-k = d*k"`

Perciò il `MCD` di `m-n` e `n` è `MCD(m,n)*d`. Ma ora si osservi che

```
n       = MCD(m,n)*d*k'
m-n     = MCD(m,n)*d*k"
(m-n)+n = MCD(m,n)*d*(k"+k')
```

assurdo perchè `MCD(m,n)` non è il massimo divisore **!**

ALGORITMO MCD/3: L'ALGORITMO DI EUCLIDE



commento: numero iterazioni eseguite nel caso pessimo (quando uno dei due valori è 1)

$$\leq \max(m,n)$$

complessità computazionale: $O(\max(m,n))$

ALGORITMO MCD/4: L'ALGORITMO DI EUCLIDE OTTIMIZZATO

osservazione: quando uno dei due numeri è molto più grande dell'altro, allora l'algoritmo di Euclide esegue numerosi cicli per sottrarre la quantità più piccola dal numero più grande

esempio: `m = 1000, n = 2`

1° iterazione:	`m = 1000,`	`n = 2`
2° iterazione:	`m = 998,`	`n = 2`
3° iterazione:	`m = 996,`	`n = 2`
	...	
499° iterazione:	`m = 2,`	`n = 2`

soluzione più efficiente: utilizzare il **resto** invece della **differenza**!

se `m >> n` anziché fare tante volte `m-n`, `(m-n)-n`, `((m-n)-n)-n`, ...

conviene fare `m%n`

esempio: input `m = 1000, n = 2`

1° iterazione: `m = 1000%2, n = 2`

ALGORITMO MCD/4: L'ALGORITMO DI EUCLIDE OTTIMIZZATO

```
1. se m>n allora r = n altrimenti r = m; m = n ; n =
   r ;

2. se  (r != 0) allora
    a)  r = m % n ;
    b)  m = n ;
    c)  n = r ;
    d)  ritorna a 2

3. stampa m
```

commenti: numero di iterazioni eseguite (caso pessimo) $\approx \log_2(\max(m,n))$

perchè $m \% n \leq m/2$