$L = \{a^n b^m \mid n \neq m \wedge n+m > 0\}$ non è regolare.

Non si riesce a dimostrare col PUMPING LEMMA.

$K \in \mathbb{N}$ arbitrario

$w \in L \quad |w| \geq k \quad w = a^t b^s \quad t \neq s \quad t+s \geq k$

$w = xyz \quad |xy| \leq k \quad |y| \geq 1$

$xy^i z \notin L$ devo trovare un i : $xy^i z = a^h b^h$

??

PROGRAMMAZIONE in C

Successione di Fibonacci

$\{ S_0, S_1, S_2, \ldots\ldots \}$

$S_0 = \phi$

$S_1 = 1$

$S_n = S_{n-1} + S_{n-2} \qquad n \geq 2$

$\{ \phi, 1, 1, 2, 3, 5, 8, 13, \ldots\ldots \}$

$S_0 \; S_1 \; S_2 \; S_3 \; S_4 \; S_5$

Scrivere una funzione

```
int fib(int n)
```

in modo che `fib(k)` calcoli il termine S_k della successione di Fibonacci

$fib(5) = 5 \qquad fib(8) = 21$

```
int fib(int n)
{ int ris;
  if (n==0) ris = ∅;
  else if (n==1) ris = 1;
  else
  { int indice = 1
    int prec = ∅; int corr = 1;
    while (indice < n)
    { int t = prec + corr;
      prec = corr;
      corr = t;
      indice = indice + 1;
    }
    ris = corr;
  }
  return ris;
}
```

return ∅;

return 1;

return corr;

$S_{indice} \rightarrow corr$

$S_{indice-1} \rightarrow prec$

~~corr = prec + corr;~~

corr = prec + corr;

prec = corr - prec;

prec: il penultimo termine calcolato della succ. di Fibonacci

corr: l'ultimo termine calcolato della succ. di Fibonacci

corr prec + corr

n = 4

prec	corr	t	indice
∅	1		1
1	1	1	2
1	2	2	3
2	3	3	4

Essendo una iterazione DETERMINATA (per tutti i valori di indice in $[1, n)$) possiamo riscriverla così:

```
int fib (int n)
  { int ris;
    if (n==0)  ris=0;
       else
          { int prec=0; int corr=1;
            int indice;
            for (indice=1; indice<n; indice++)
               { int t=prec+corr;
                 prec=corr;
                 corr=t;
               }
            ris=corr;
          }
    return (ris);
  }
```

return (Exp)

non c'è l'incremento di indice

Dato un numero naturale n e una cifra c voglio calcolare il seguente valore di verità

"c appartiene alla rappresentazione in base 10 di n"

check (3672, 5) = false (0)
check (3672, 2) = true (1)
check (3672, 3) = true (1)

Ipotesi: la funzione verrà SEMPRE chiamata con $0 \leq c \leq 9$

int check (int n, int c)

3672 / 10 = 367
3672 % 10 = 2

367 / 10 = 36
367 % 10 = 7

36 / 10 = 3
36 % 10 = 6

3 / 10 = 0
3 % 10 = 3

```
int check (int n, int c)
{  int trovato = 0;
   if (n<10) if(n==c) trovato=1; else trovato=0;
      else
         while (n>0 && !trovato)
               if (n%10 == c)
                     trovato = 1;
               else  n = n/10;
   return(trovato);
}
```

trovato = n == c ;

assegnamento — uguaglianza

Scrivere una funzione C che dati un array a e la sua dimensione dim restituisce il seguente valore di verità

$$\left(\forall i.\ i \in [\phi, dim-1) \Rightarrow a[i] \leq a[i+1] \right)$$

$$a[dim-1] \leq a[\cancel{dim}] \quad ?$$

$$[\phi, dim) \equiv [\phi, dim-1]$$

$$[\phi, dim-1) \equiv [\phi, dim-2]$$

a 4

$$a[\phi] \leq a[1] \leq a[2] \leq a[3]$$

$i = 2$ $\quad i+1 = 3$

$i-1 = 2$ $\quad i = 3$

$$\left(\forall i.\ i \in [1, dim) \Rightarrow a[i-1] \leq a[i] \right)$$

$dim-2$ $\quad dim-1$

Verifica se l'array è ordinato in modo NON DECRESCENTE

RICERCA LINEARE INCERTA

$$\left(\forall i.\ i \in [a,b) \Rightarrow P(i) \right)$$

se $P(i)$ è sintatticamente esprimibile in C posso risolverlo CERCANDO il primo elemento in $[a,b)$ che NON soddisfa la prop. P.

se lo trovo restituisco FALSE

se non lo trovo " TRUE

```
int trovato = 0;
int i = a;
while (i < b && !trovato)
    if !P(i) trovato = 1;
      else   i = i+1;
return (!trovato);
```

$\forall i.\ i \in [a, b) \Rightarrow P(i)$

```
int trovato = 0;
int i = a;
while (i < b && !trovato)
  if (!P(i)) trovato = 1;
     else i = i+1;
return (!trovato);
```

dim-1

$!(a[i] <= a[i+1])$

$(a[i] > a[i+1])$

$x > y \equiv x - y > 0$

$x + x - y > x$

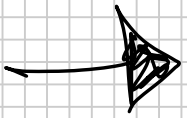
$(\forall i.\ i \in [0, dim-1)) \Rightarrow a[i] \leq a[i+1]$

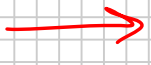
```
int ordinato (int a[], int dim)
{ int trovato = 0;
  int i = 0;
  while (i < dim-1 && !trovato)
    if (a[i] > a[i+1])
          trovato = 1;
    else i = i+1;
  return (!trovato);
}
```

Verificare che tutti gli elementi di un array a di dimensione dim siano non negativi

$$\forall i.\ i \in [0, dim) \Rightarrow a[i] \geq 0$$

```
int pos (int a[], int dim)
{ int trovato = 0;
  int i = 0;
  while (i < dim && !trovato)
    if (a[i] < 0) trovato = 1;
      else i = i+1;
  return (!trovato);
}
```





```
int pos (int a[], int dim)
{ int i = 0;
  while (a[i] >= 0 && i<dim)
      i = i+1;
  if (i == dim)
      return 1;
  else return 0;
}
```

10	0	8
0	1	2

Il C valuta && in questo modo (CONDITIONAL AND - CAND)

(A && B)

① si valuta A. Se A è false si restituisce FALSE

② solo se A è true, si valuta B e si restituisce il valore di B

$A \text{ CAND } B \not\equiv B \text{ CAND } A$

$a[i] >= 0 \text{ CAND } i < dim \not\equiv i < dim \text{ CAND } a[i] >= 0$

$A \wedge B \equiv B \wedge A$