UNIVERSITÀ degli STUDI di CATANIA

DIPARTIMENTO DI MATEMATICA E INFORMATICA

Ricorsione

Alessandro Ortis
Image Processing Lab - iplab.dmi.unict.it

ortis@dmi.unict.it
www.dmi.unict.it/ortis/

IMAGE PROCESSING LABORATORY

Ricorsione

quando una funzione chiama se stessa, sia direttamente che tramite altre funzioni, essa viene detta *ricorsiva*

- es: il fattoriale è una funzione intrinsecamente ricorsiva:

$$n! = n*(n - 1)!$$

- ogni funzione ricorsiva può avere un'implementazione iterativa:

```
// implementazione ricorsiva
int fattoriale(int n)
  { if (n == 0) return 1;
    else return n * fattoriale(n - 1);
}

// implementazione iterativa
fattoriale = 1;
for (int contatore = n; contatore >= 1; contatore --)
   fattoriale *= contatore ;
```

Esempio: Prodotto di due numeri naturali

Prodotto di due numeri naturali a e b

- Soluzione iterativa

$$\text{prod}(a,b) = \underbrace{a+a+a+\ldots+a}_{\text{b volte}}$$

- Soluzione ricorsiva

$$\text{prod}(a,b) = a \qquad \text{se } b=1$$

$$\text{prod}(a,b) = a + \text{prod}(a, b-1) \qquad \text{altrimenti}$$

Ricorsione

Si può usare l'induzione matematica per convincersi che un programma ricorsivo si comporta correttamente:

- Caso base: calcola direttamente 0! = 1
- Altrimenti: assumendo che il programma calcoli *k!* per *k*<*N* (ipotesi induttiva), allora esso calcola *N!*

```
int fattoriale(int n)
  { if (n == 0) return 1;
    else return n * fattoriale(n - 1);
}
```

In pratica, il legame con l'induzione matematica ci dice che le funzioni ricorsive devono soddisfare due requisiti fondamentali:

1. Risolvere in modo esplicito il caso base
2. Ogni chiamata ricorsiva deve avere come argomenti valori più piccoli

Ricorsione

Possiamo dimostrare la correttezza della seguente funzione *puzzle*?

```
int puzzle(int N){
   if (N == 1) return 1;
   if (N % 2 == 0)
      return puzzle(N/2);
   else
      return puzzle(3*N+1);
}
```

Ricorsione

Possiamo dimostrare la correttezza della seguente funzione *puzzle*?

```
int puzzle(int N){
    if (N == 1) return 1;
    if (N % 2 == 0)
        return puzzle(N/2);
    else
        return puzzle(3*N+1);
}
```

Se N è dispari la funzione chiama se stessa sull'argomento 3N+1, mentre se N è pari la funzione chiama se stessa su N/2. Non possiamo dimostrare per induzione che questo programma termina perché non tutte le chiamate ricorsive hanno come argomento valori più piccoli di quello dato.

Ricorsione e iterazione

- Qualunque problema risolvibile ricorsivamente può essere risolto con un algoritmo iterativo;
  - per ogni funzione ricorsiva se ne può trovare un'altra che fa la stessa cosa attraverso un ciclo (senza richiamare se stessa)

- la ricorsione spesso produce soluzioni concettualmente più semplici
  - la corrispondente soluzione iterativa sarà normalmente più efficiente, sia in termini di occupazione di spazio di memoria che in termini di tempo di computazione.

Svantaggi della Ricorsione

- Spreco di tempo
  - Ogni chiamata della funzione richiede per se un tempo di esecuzione (indipendente da cosa farà la funzione)

- Spreco di memoria
  - Ad ogni chiamata bisogna memorizzare nello stack una serie di registri e parametri
    - Es. indirizzo dell’istruzione da seguire quando la funzione terminerà la sua esecuzione.
    - Argomenti della funzione
    - Variabili locali

Vantaggi della Ricorsione

- i programmi sono più chiari, più semplici, più brevi e più facili da capire delle corrispondenti versioni iterative

- il programma riflette la strategia di soluzione del problema

- spesso la soluzione trovata può poi trasformarsi in una soluzione iterativa equivalente ma più efficiente

Esempio: sommare gli elementi di una lista

- Soluzione ricorsiva

somma(v,i) = v[0]   se i=0

v[i] + somma(v,i-1)   altrimenti

Se v ha dimensione N, la prima chiamata sarà somma(v, N-1)

Esempio: sommare gli elementi di una lista

somma(v,i) = v[0]   se i=0

v[i] + somma(v,i-1)   altrimenti

Esempio: elevamento a potenza

- Soluzione ricorsiva: Pot(b,n)

  Pot(b,0) = 1

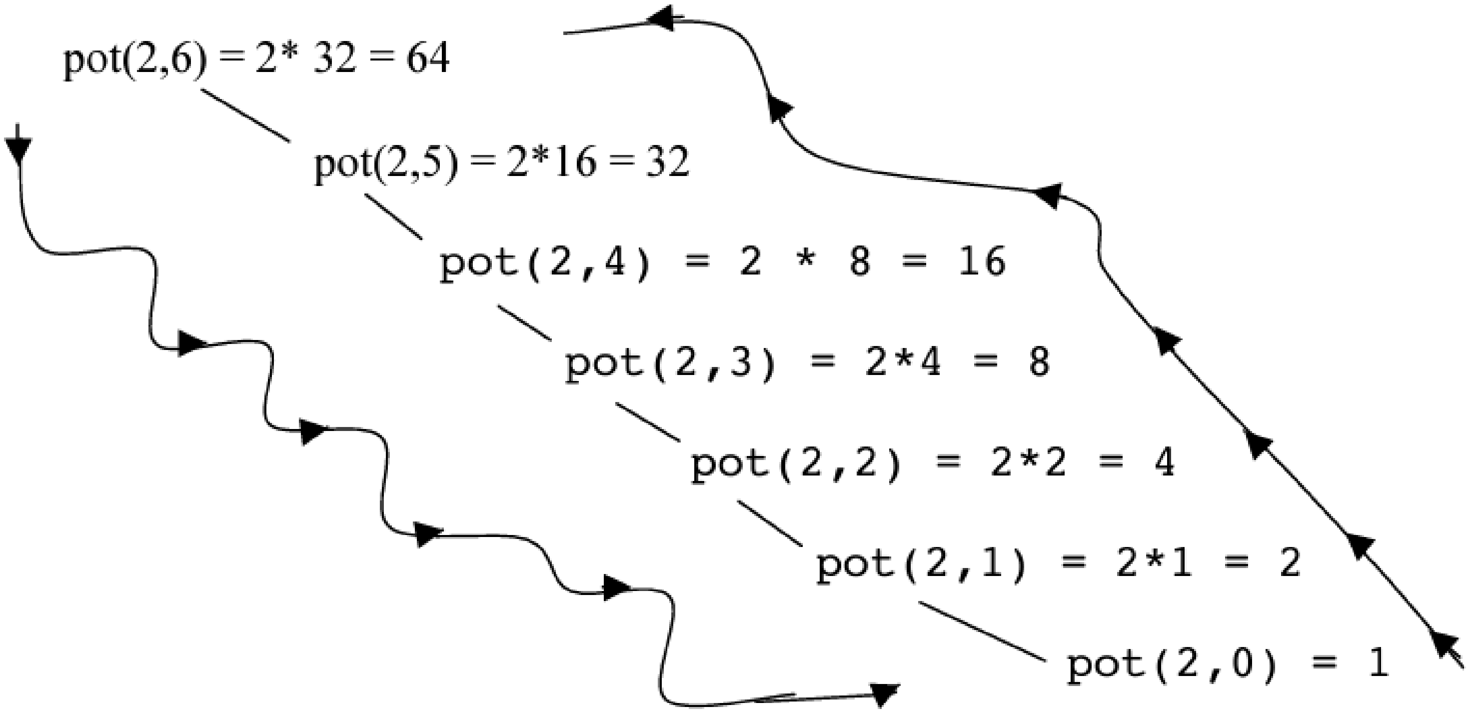
  Pot(b,n) = b * Pot(b,n-1))

Esempio: elevamento a potenza

Quanto costa calcolare ricorsivamente la potenza n-sima di un numero b?

Esempio: elevamento a potenza

Quanto costa calcolare ricorsivamente la potenza n-sima di un numero b?



Si calcolano n prodotti e si occupa uno spazio di memoria proporzionale a n, perché si deve considerare lo spazio per le chiamate in sospeso.

Esempio: elevamento a potenza

La versione iterativa equivalente invece comporta l'esecuzione di n prodotti ma in spazio costante:

```
int potIter(int b, int n){
   int ris = 1;
   for (;0<n;n--)
      ris = ris*b;
   return ris;
}
```

Esempio: Serie di Fibonacci

`0,1,1,2,3,5,8,13,21,44,…`

Soluzione ricorsiva

```
fibonacci(0)=0
fibonacci(1)=1
fibonacci(n)=fibonacci(n-1)+fibonacci(n-2)
```

in modo ancora più compatto

```
fibonacci(n)=n                                  if n=0,1
fibonacci(n)=fibonacci(n-1)+fibonacci(n-2)  per n>=2
```

Esercizio

Individuare la sequenza di chiamate ricorsive dell'esecuzione della seguente chiamata di funzione, come visto nell'esempio dell'elevamento a potenza

```
fibonacci(5)
```

Serie di Fibonacci: soluzione iterativa

0,1,1,2,3,5,8,13,21,44,…

```
int fibonacci(int n)
{  int primo=0, secondo=1, temp;

   if (n<=1) return n;
   for (i=2; i<=n; i++)
   {  temp=primo;
      primo=secondo;
      secondo=primo + temp;
   }
   return secondo;
```

Ricorsione indiretta

La funzione chiama se stessa non direttamente ma tramite una concatenazione di chiamate con altre funzioni.

Esempio

```
void A(int c)
{  if (c> 5) B(c);
   cout << c << " ";}

void B(int c)
{  A(--c);  }

int main()
{
   A(25);
}
```

Esercizio: Massimo Comune Divisore (MCD)

Il MCD tra m ed n con m>n è lo stesso del MCD tra n ed il resto della divisione di m per n.

MCD(m,n)

$$
\texttt{MCD(m,n)} = \begin{cases} \texttt{n} & \textbf{se n<=m e n è divisore di m} \\ \texttt{MCD(n,m)} & \textbf{se m < n} \\ \texttt{MCD(n, resto di m diviso n)} & \textbf{altrimenti} \end{cases}
$$

Esercizio: Calcolare la somma dei primi N numeri interi

Versione ricorsiva

$$\texttt{Sommatoria(N)} = \begin{cases} \texttt{1} & \texttt{se N=1} \\ \texttt{N+Sommatoria(N-1)} & \texttt{altrimenti} \end{cases}$$

Esercizio: Calcolare la somma dei quadrati dei primi N numeri interi

Versione ricorsiva

$$\texttt{Sommatoria(N)} = \begin{cases} \texttt{1} & \texttt{se N=1} \\ \texttt{N}^2\texttt{+Sommatoria(N-1)} & \texttt{altrimenti} \end{cases}$$

Esercizio: definire una funzione ricorsiva per determinare il massimo in un array di N elementi

Esercizio: definire una funzione ricorsiva per determinare il massimo in un array di N elementi

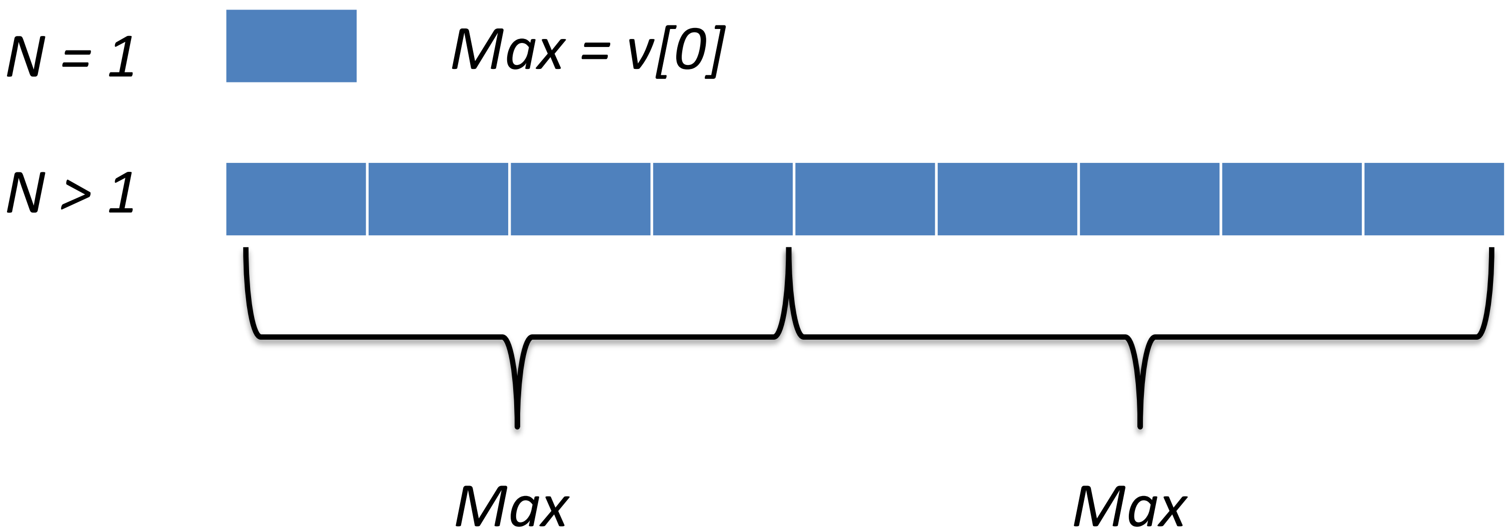
N = 1 *Max = v[0]*

Esercizio: definire una funzione ricorsiva per determinare il massimo in un array di N elementi

N = 1  *Max = v[0]*

N > 1 

Esercizio: definire una funzione ricorsiva per determinare il massimo in un array di N elementi



Esercizio: definire una funzione ricorsiva per determinare il massimo in un array di N elementi

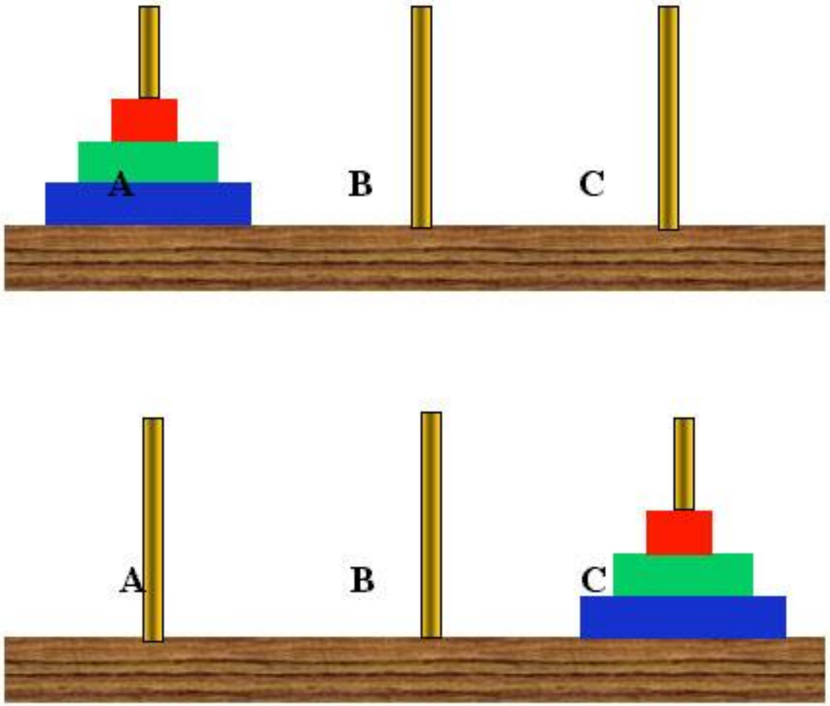
Dimostrazione induttiva:

- Caso base: se N = 1 allora max=v[0]
- Altrimenti: se N>1 dividi l’array in due sottoarray di dimensioni inferiori ad N, trova il max tra i due sottoarray e restituisci il più grande dei due valori.

Torri di Hanoi

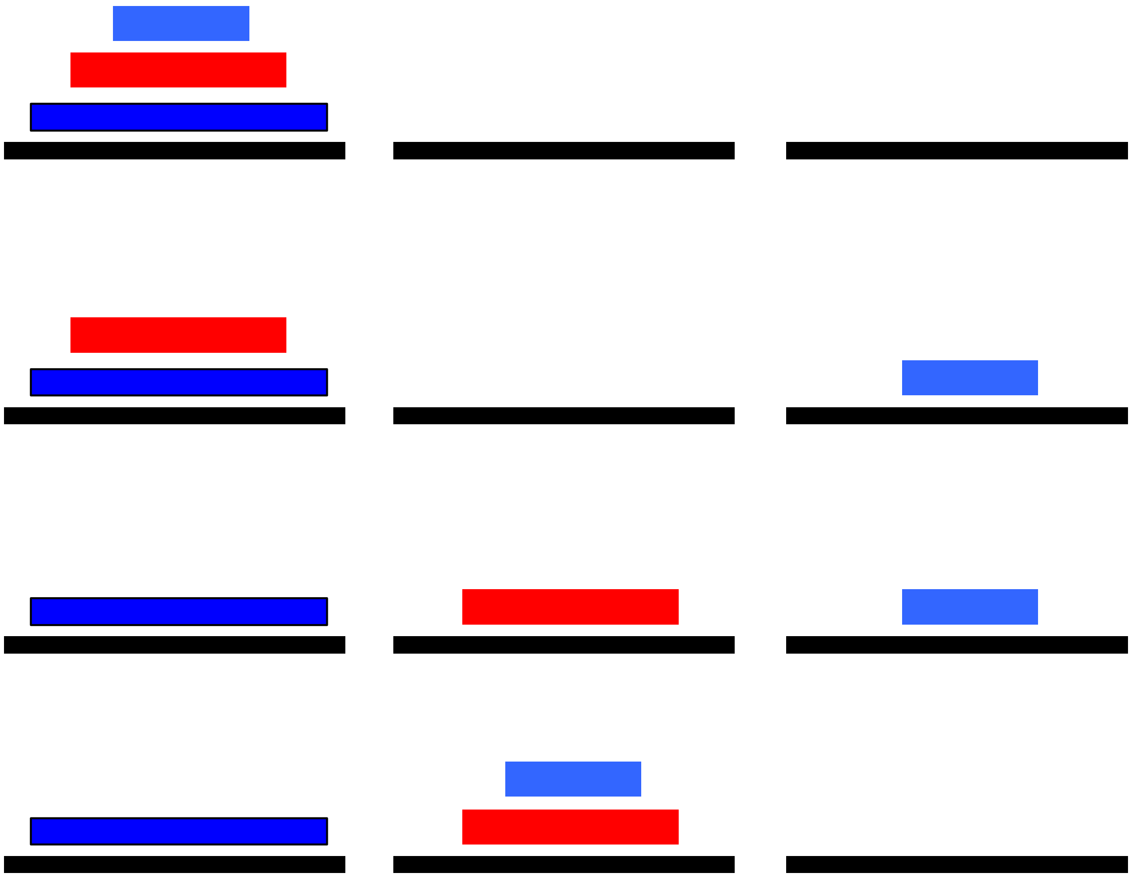
Abbiamo 3 pioli ed N dischi che possono essere inseriti nei pioli. I dischi hanno diversa dimensione e sono inizialmente sistemati in un piolo, dal più grande posto alla base (disco N) al più piccolo in cima (disco 1). L’obiettivo è quello di spostare la pila di dischi sul piolo più a destra rispettando le due regole seguenti:

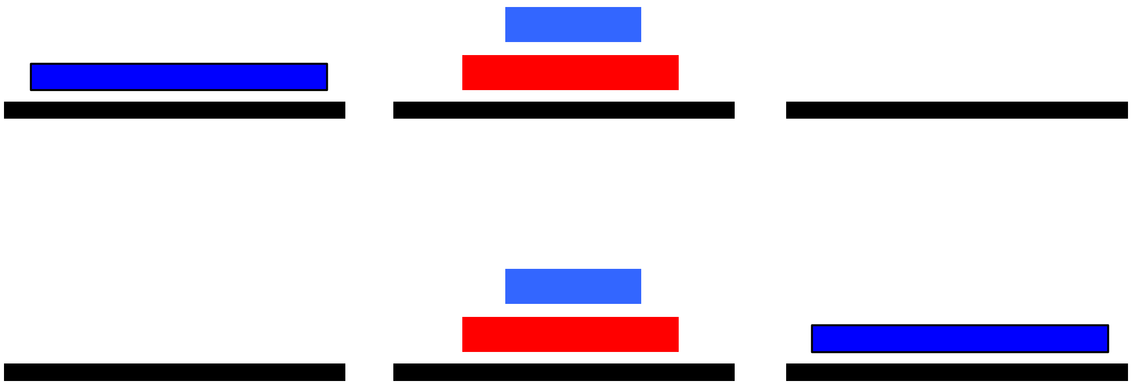
1. Può essere spostato solo un disco per volta
2. Non è possibile mettere un disco più grande sopra uno più piccolo



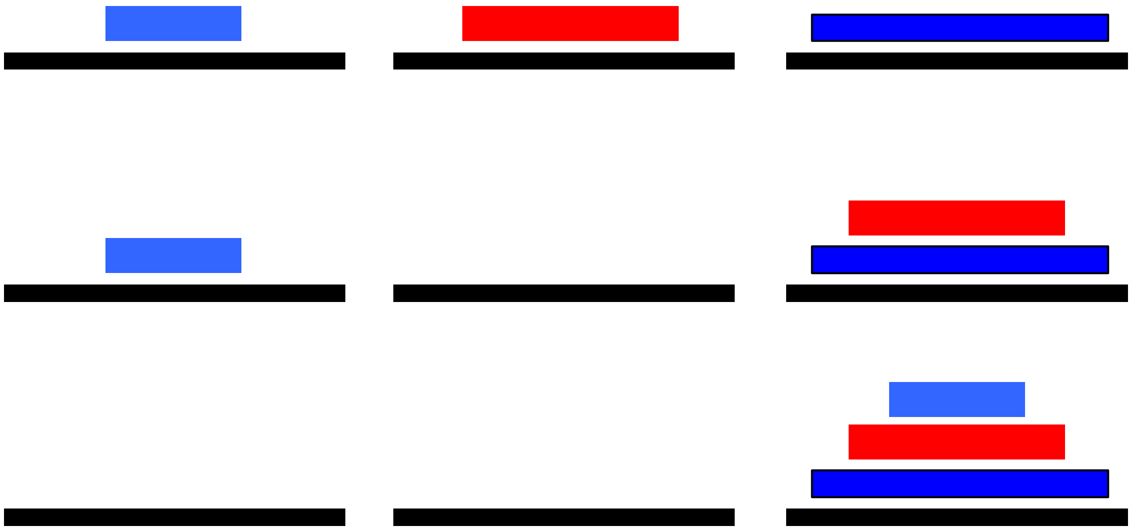
Configurazione iniziale e finale della *Torre di Hanoi* a tre dischi.

Esempio: n=3





Situazione molto simile a quella iniziale!



Torri di Hanoi idee di base

- Se n=1 spostare (l’unico) disco sul piatto finale
- Se n>1
  - Spostare n-1 dischi dal piatto iniziale al piatto centrale
  - Spostare il disco più grande dal piatto iniziale al piatto finale
  - Spostare gli n-1 dischi rimanenti dal piatto centrale al piatto finale
    - Usando il piatto iniziale come piatto ausiliario

Esercizi

Implementare i seguenti algoritmi in maniera ricorsiva:

1. Sommare gli elementi di una lista
2. Trovare il massimo in una lista
3. Generare il n-esimo numero della serie di Fibonacci e stampare la sequenza fino ad n
4. Calcolare il MCD tra due interi dati in input
5. calcolare il numero di occorrenze di un carattere c in una stringa s
6. Dato il numero di dischi *n* risolvere il problema delle torri di Hanoi elencando le mosse da effettuare