Programmazione

Giovanni Da San Martino

Dipartimento of Matematica, Università degli Studi di Padova
giovanni.dasanmartino@unipd.it
A.A. 2021-2022

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Previously on Programmazione


```
int Fatt(int n) {

    if(n==0)
        return 1;
    return n*Fatt(n-1);

}

printf("%d", Fatt(3));
```

- Fatt(3) = 3*Fatt(2)
- Fatt(2) = 2*Fatt(1)
- Fatt(1) = 1*Fatt(0)
- Fatt(0) = 1

quindi

- Fatt(3) = 3*Fatt(2)
- Fatt(2) = 2*Fatt(1)
- Fatt(1) = 1*1

Funzioni Ricorsive


```
int Fatt(int n) {

    if(n==0)
        return 1;
    return n*Fatt(n-1);

}

printf("%d", Fatt(3));
```

- Fatt(3) = 3*Fatt(2)
- Fatt(2) = 2*Fatt(1)
- Fatt(1) = 1*Fatt(0)
- Fatt(0) = 1

quindi

- Fatt(3) = 3*Fatt(2)
- Fatt(2) = 2*1

Funzioni Ricorsive


```
int Fatt(int n) {

    if(n==0)
        return 1;
    return n*Fatt(n-1);

}

printf("%d", Fatt(3));
```

- Fatt(3) = 3*Fatt(2)
- Fatt(2) = 2*Fatt(1)
- Fatt(1) = 1*Fatt(0)
- Fatt(0) = 1

quindi

- Fatt(3) = 3*2

Ricorsione: Quando Utilizzarla

- Spesso la ricorsione permette di risolvere un problema anche molto complesso con poche linee di codice
- La ricorsione richiede tempo per la gestione dello stack
- Può consumare meno memoria rispetto alla soluzione iterativa equivalente perché in genere si utilizzano meno variabili nel codice

Ricorsione: Quando Utilizzarla

- Spesso la ricorsione permette di risolvere un problema anche molto complesso con poche linee di codice
- La ricorsione richiede tempo per la gestione dello stack
- ~~Può consumare meno memoria rispetto alla soluzione iterativa equivalente perché in genere si utilizzano meno variabili nel codice~~
- consuma molta memoria (alloca un nuovo stack frame a ogni chiamata, definendo una nuova ulteriore istanza delle variabili locali e dei parametri ogni volta)

Funzioni Ricorsive

- Si modella un problema per casi
  - dal più semplice (caso base)
  - al più complesso (caso ricorsivo), nel quale si esprime il problema in termini di un caso più semplice
- Si esprime la “complessità” mappandola sui numeri naturali
  - maggiore il numero n, maggiore la complessità
  - i casi basi sono collegati ai valori minori minimi che n può assumere nel nostro problema (casi base)
- Fattoriale(0) = 1; Fattoriale(n) = n*Fattoriale(n-1);
- Nel caso ricorsivo si esprime il problema per il caso n in funzione di un valore minore di n; si continua fino a raggiungere i casi base

Funzioni Ricorsive

- Fattoriale(0) = 1; Fattoriale(n) = n*Fattoriale(n-1);
- Nel caso ricorsivo si esprime il problema per il caso n in funzione di un valore minore di n; si continua fino a raggiungere i casi base

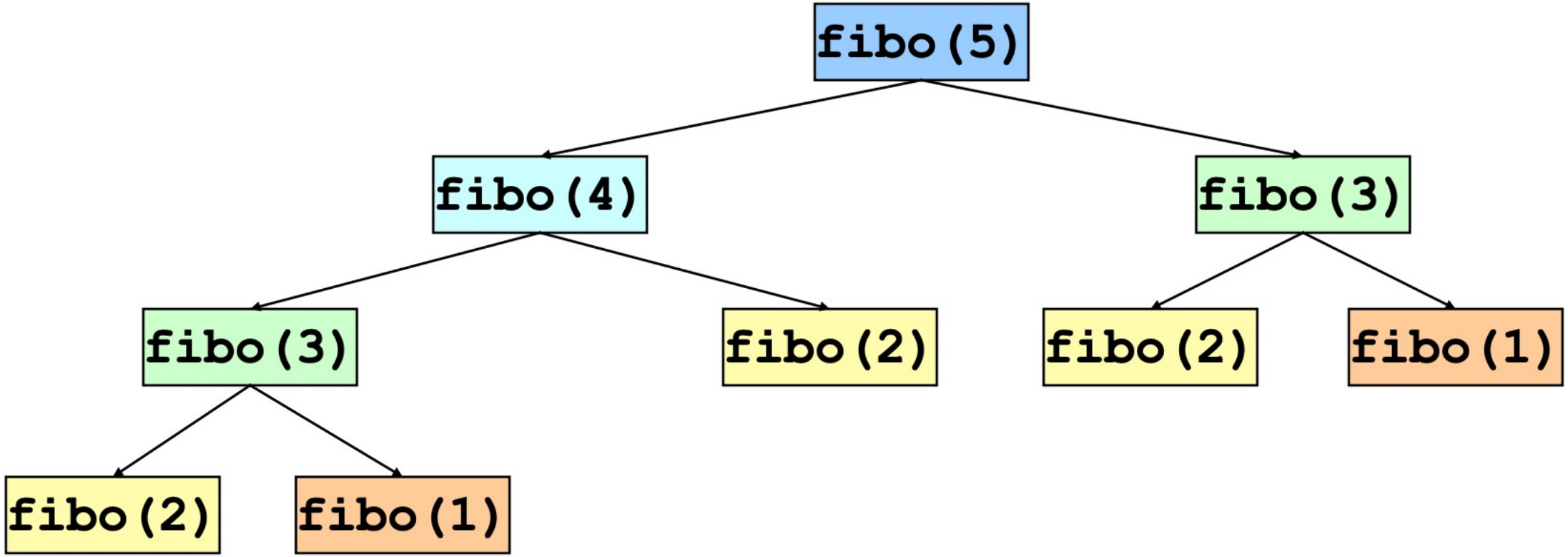
1. Identificare la variabile n da cui dipende la complessità del problema
2. Riconoscere tutti i casi base (se ne perdiamo uno, il programma potrebbe non terminare mai, usare Ctrl-c per interrompere l'esecuzione da terminale)
3. Esprimere il problema per il caso n in funzione di un caso n1<n

Ricorsione: Esempio


```
int fibonacci(int n) {
    //PRE: n>=0
    if (n<=1)
       return n;
    else
      return fibonacci(n-1) + fibonacci(n-2);
}
```

Ricorsione: Quando Non Utilizzarla

- La nostra funzione fibo ripete gli stessi calcoli più volte -> inefficiente



Esercizi

- Cosa stampa il seguente codice?

```
int x;
int y=2;
int *p, *q = &y;
int **qq = &p;
**qq = 3;
printf("%d\n", **qq);
```

Esercizi

- Cosa stampa il seguente codice?

```
int y=2;
int **qq = &&y;
printf("%d\n", **qq);
```