Uso della ricorsione

Prof. Rocco Zaccagnino

2022/2023

Uso della ricorsione

Un algoritmo è detto **ricorsivo** se risolve un problema riducendo esso ad una istanza dello stesso problema ma con un input più piccolo.

Esempio: $f(0)=3$

$f(n)=2 * f(n-1) +3$ per $n \geq 1$

```
procedure funz(n)
    if n=0
        then return 3
     else
        return 2 * funz(n-1) + 3
```

Uso della ricorsione

Esempio: $f(0)=3$

$f(n)=2 * f(n-1) +3$ per $n \geq 1$

```
procedure funz(n)
        if n=0
                then return 3
        else
                return 2 * funz(n-1) + 3
```

$f(3) = 2 * f(2) + 3$

$f(2) = 2 * f(1) + 3$

$f(1) = 2 * f(0) + 3$

$f(0) = 3$

Uso della ricorsione

Esempio: $f(0)=3$

$f(n)=2 * f(n-1) +3$ per $n \geq 1$

```
procedure funz(n)
        if n=0
                then return 3
        else
                return 2 * funz(n-1) + 3
```

f(0) = **3**

f(1) = 2 * 3 + 3 = **9**

f(2) = 2 * 9 + 3 = **21**

f(3) = 2 * 21 + 3 = **45**

Correttezza degli algoritmi ricorsivi

Proviamo la correttezza dell'algoritmo descritto dalla procedura **funz(n)**

Dimostriamo, cioè che

*il valore restituito dalla procedura **funz(n)** coincide con **f(n)***

Dim. Usiamo l'induzione matematica su **n**

Base: Se **n=0**, il primo passo dell'algoritmo ci dice che il valore restituito da **funz(0)** è **3**. Corretto perché **f(0)=3**.

Correttezza degli algoritmi ricorsivi

Esempio: Proviamo la correttezza dell'algoritmo descritto dalla procedura **funz(n)**

Dimostriamo, cioè che il valore restituito dalla procedura **funz(n)** coincide con **f(n)**

Dim.

Ipotesi induttiva: per un **n** intero positivo arbitrario, l'algoritmo computa correttamente **f(n),** cioè **funz(n)** restituisce **f(n)** .

Passo di induzione: Ora mostriamo che la procedura **funz(n+1)** computa correttamente anche **f(n+1).**

Correttezza degli algoritmi ricorsivi

Esempio: Proviamo la correttezza dell'algoritmo descritto dalla procedura **funz(n)**

Dimostriamo, cioè che il valore restituito dalla procedura **funz(n)** coincide con **f(n)**

Dim.

Passo di induzione:

La procedura **funz(n+1)** restituisce **2 * funz(n) + 3**

Per ipotesi induttiva **funz(n)** coincide con **f(n)**, quindi

2 * funz(n) + 3 coincide con **2 * f(n) +3 = f(n+1)**

Correttezza degli algoritmi ricorsivi

Esempio: funzione fattoriale

$0!=1$ e $n!=n * (n-1)!$ per $n \geq 1$

```
procedure fattoriale(n)
    if n=0 then return 1
        else return n * fattoriale(n-1)
```

fattoriale(3) = 3 * fattoriale(2) =

fattoriale(2) = 2 * fattoriale(1) =

fattoriale(1) = 1 * fattoriale(0)

fattoriale(0) = 1

Correttezza degli algoritmi ricorsivi

Esempio: funzione fattoriale

0!=1 e n!=n * (n−1)! per n≥1

```
procedure fattoriale(n)
    if n=0 then return 1
        else return n * fattoriale(n−1)
```

fattoriale(0) = **1**

fattoriale(1) = 1 * **1** = **1**

fattoriale(2) = 2 * **1** = **2**

fattoriale(3) = 3 * **2** = **6**

Correttezza degli algoritmi ricorsivi

Proviamo la correttezza dell'algoritmo descritto dalla procedura **fattoriale(n)**

Dimostriamo, cioè che il valore restituito dalla procedura **fattoriale(n)** coincide con **n!**

Dim. Usiamo l'induzione matematica su **n**

Base: Se **n=0**, il primo passo dell'algoritmo ci dice che il valore restituito da **fattoriale(0)** è **1**. Corretto perché **0!=1**.

Correttezza degli algoritmi ricorsivi

Proviamo la correttezza dell'algoritmo descritto dalla procedura **fattoriale(n)**

Dimostriamo, cioè che il valore restituito dalla procedura **fattoriale(n)** coincide con **n!**

Dim.

Ipotesi induttiva: per un **n** intero positivo arbitrario, l'algoritmo computa correttamente **n!,** cioè **fattoriale(n)** restituisce **n!**

Passo di induzione: Ora mostriamo che la procedura **fattoriale(n+1)** computa correttamente anche **(n+1)!**

Correttezza degli algoritmi ricorsivi

Proviamo la correttezza dell'algoritmo descritto dalla procedura **fattoriale(n)**

Dimostriamo, cioè che il valore restituito dalla procedura **fattoriale(n)** coincide con **n!**

Dim.

Passo di induzione:

La procedura **fattoriale(n+1)** restituisce **(n+1) * fattoriale(n)**

Per ipotesi induttiva **(n+1) * fattoriale(n)** coincide con **(n+1) * n! = (n+1)!**

Correttezza degli algoritmi ricorsivi

Esempio: Numeri di Fibonacci

$F(0)=0,\ F(1)=1$ e $F(n)=F(n-1)+F(n-2)$ per $n \geq 2$

```
procedure Fibonacci(n)
if n=0 then
      return 0
else  if n=1 then
           return 1
       else
           return  Fibonacci(n−1)+Fibonacci(n−2)
```

Correttezza degli algoritmi ricorsivi

Esempio: Numeri di Fibonacci

$F(0)=0,\ F(1)=1 \quad e \quad F(n)=F(n-1) + F(n-2) \quad per\ n \geq 2$

Fibonacci(4) = Fibonacci(3) + Fibonacci(2)

Fibonacci(3) = Fibonacci(2) + Fibonacci(1)

Fibonacci(2) = Fibonacci(1) + Fibonacci(0)

Fibonacci(1) = 1 Fibonacci(0) = 0

Fibonacci(2) = Fibonacci(1) + Fibonacci(0)

Fibonacci(1) = 1 Fibonacci(0) = 0

Uso di definizioni ricorsive

Un **insieme** può essere definito

- *Elencando i suoi elementi:*
  - **{a, b, c}** ha elementi **a,b,c**
- Specificando le proprietà caratteristiche di suoi elementi
  - **A = {w | w ha la proprietà P}**

Un altro modo per descrivere insiemi è attraverso una definizioni ricorsiva

Un insieme **A** è ***definito ricorsivamente*** nel modo seguente:

Passo base: Si definiscono uno o più oggetti elementari

Passo ricorsivo: definisce la regola che permette di costruire oggetti più complessi in termini di quelli già definiti dell'insieme.

Uso di definizioni ricorsive

Esempio: Sia A un sottoinsieme di interi definito ricorsivamente come segue:

Passo base: $1 \in A$
Passo ricorsivo: se $x \in A$ allora $x + 2 \in A$

Quali sono gli elementi di **A**?

- ***Passo base:*** $1 \in A$
- Applico il ***Passo ricorsivo***: $x=1 \in A$ allora $x + 2 = 1 + 2 = 3 \in A$
- Applico il ***Passo ricorsivo***: $x=3 \in A$ allora $x + 2 = 3 + 2 = 5 \in A$
- $1, 3, 5, 7, 9 \ldots. \in A$

Proveremo utilizzando l'induzione strutturale che A è l'insieme degli interi dispari positivi

Uso di definizioni ricorsive

Le definizioni ricorsive possono essere usate per descrivere **insiemi di stringhe**

Un **alfabeto** e un insieme finito di elementi (chiamati **lettere** o **simboli**)

- **Ex:** L'alfabeto delle *lettere romane minuscole* è
  - ✓ **Σ= {a,b,…,z}**
- **Ex:** L'alfabeto delle *cifre arabe* è
  - ✓ **Σ= {0,1,…,9}**
- **Ex:** L'alfabeto *binario* è
  - ✓ **Σ= {0,1}**

Uso di definizioni ricorsive

Le definizioni ricorsive possono essere usate per descrivere **insiemi di stringhe**

L'insieme di stringhe Σ^* sull'alfabeto Σ è definito ricorsivamente nel modo seguente
Passo base: la stringa vuota $\lambda \in \Sigma^*$
Passo ricorsivo: Se $w \in \Sigma^*$ e $x \in \Sigma$ allora $wx \in \Sigma^*$

La **lunghezza di una parola** in Σ^* sull'alfabeto Σ è definito ricorsivamente nel modo seguente
Passo base: $l(\lambda) = 0$
Passo ricorsivo: Se $w \in \Sigma^*$ e $x \in \Sigma$ allora $l(wx) = l(w) + 1$

Uso di definizioni ricorsive

Le definizioni ricorsive possono essere usate per descrivere **parole palindrome**

Una stringa è **palindroma** se letta da sinistra a destra o da destra a sinistra da luogo alla stessa sequenza

Esempi: *alla, otto, ingegni, Anna, ottetto*.

Nota: Per semplicità diamo la definizione per un piccolo alfabeto **Σ={a,b}**

L'insieme delle **parole palindrome** sull'alfabeto **Σ = {a,b}** è definito ricorsivamente nel modo seguente:
Passo base: **a,b,λ** sono parole palindrome
Passo ricorsivo: Se **w** è una parola palindroma allora anche **awa** e **bwb** sono parole palindrome

Uso di definizioni ricorsive

Le definizioni ricorsive possono essere usate per descrivere espressioni aritmetiche

Una espressione aritmetica è definita ricorsivamente nel modo seguente

Passo base: i numeri (interi o reali) e le variabili sono espressioni aritmetiche
Passo ricorsivo: se **E1** ed **E2** sono espressioni aritmetiche allora
(E1 + E2), (E1 – E2), (E1 × E2), (E1\ E2)
sono **espressioni aritmetiche**.

Se **E** e un'**espressione aritmetica** allora **(–E)** e un'**espressione aritmetica**.

Uso di definizioni ricorsive

Le definizioni ricorsive possono essere usate per descrivere **strutture dati**

Un **albero radicato** è un albero contenente un vertice particolare detto radice

Un albero radicato può essere descritto ricorsivamente nel modo seguente

Base: un singolo vertice **r** è un **albero radicato**
Passo ricorsivo: Supponiamo che $\mathbf{T_1}$, $\mathbf{T_2}$, …., $\mathbf{T_n}$ sono **alberi radicati disgiunti** con radici $\mathbf{r_1}$, $\mathbf{r_2}$, …. , $\mathbf{r_n}$. Allora, il **grafo** formato dalla radice **r**, che non è in nessuno degli alberi radicati $\mathbf{T_1}$, $\mathbf{T_2}$, …., $\mathbf{T_n}$, ottenuto connettendo con un arco **r** a ciascun $\mathbf{r_1}$, $\mathbf{r_2}$, …. , $\mathbf{r_n}$ è anch'esso un albero radicato

Uso di definizioni ricorsive

Un **albero binario pieno** è un albero radicato dove ciascun vertice ha **o** oppure **due** figli; se tali figli esistono, essi sono chiamati **figlio destro** e **figlio sinistro**

Un albero binario pieno è descritto ricorsivamente nel modo seguente

Base: un singolo vertice **r** è un **albero binario pieno**
Passo ricorsivo: Se $\mathbf{T_1}$ e $\mathbf{T_2}$ sono **alberi binari pieni**, allora l'albero **T** formato connettendo la radice **r** con un arco alla radice del sottoalbero sinistro $\mathbf{T_1}$ e con un altro arco la radice del sottoalbero destro $\mathbf{T_2}$ è un **albero binario pieno**