5. RICORSIONE

Dato un oggetto, come funzioni, insiemi, algoritmi, … in alcuni casi esso può essere ***definito in termini di sé stesso, ma di più piccole dimensioni***.

Esempi:

Consideriamo la sequenza di interi 1, 3, 9, 27, 81, …

È evidente che è la sequenza di potenze di 3, cioè

- 1, 3, 9, 27, 81, …, 3^n, …
- $3^0, 3^1, 3^2, 3^3, 3^4, \dots, 3^n, \dots$
- $b_n = 3^n$ (n--simo termine della sequenza), $n \in \mathbf{N}$

Definizione ricorsiva della sequenza

- $b_n = 3 \; 3^{n-1}$
- $b_n = 3 \, b_{n-1}$ per n≥1
  $b_0 = 1$

Consideriamo la sequenza aritmetica (progressione aritmetica)

- a, a+d, a+2d, a+3d, …, a+nd, …
  - a= −1 e d= 4 => −1, 3, 7, 11, …, −1+n4, …
- b_n = a+nd (n--simo termine della sequenza), $n \in \mathbf{N}$

Definizione ricorsiva della sequenza

- b_n = a+nd = a+ (n−1)d +d
- $b_n = b_{n-1} + d$ per n≥1
  $b_0 = a$

Consideriamo la sequenza geometrica (progressione geometrica)

- a, ar, ar^2, ar^3, …, ar^n, …
  - a= 6 e r= 1/3 => 6, 2, 2/3, 2/9, …, $6\,(1/3)^n$, …
- $b_n = a\,r^n$ (n--simo termine della sequenza), $n \in \mathbf{N}$

Definizione ricorsiva della sequenza

- $b_n = a\,r^n = a\,r^{n-1}\,r$
- $b_n = b_{n-1}\,r$ per n≥1
  $b_0 = a$

5.1 DEFINIRE FUNZINI RICORSIVE

In alcuni casi la definizione ricorsiva di un oggetto può essere molto facile da scrivere.

Esempio: Funzione fattoriale

- n! = n (n − 1)! per n≥1
  0!=1

In altri casi la definizione ricorsiva di un oggetto è l'unico modo per descriverlo.

Esempio: Numeri di Fibonacci

- $F_n = F_{n-1} + F_{n-2}$ per n≥2
  F_0=0 F_1=1

Per definire una funzione ricorsiva ***sull'insieme degli interi non negativi***.

1. (***Passo base***) Specificare il valore della funzione in 0
2. (***Passo ricorsivo***) Dare una regola per determinare il valore della funzione in n in termini del valore della funzione in interi n-1

Esempio: definire ricorsivamente la funzione f(n) = 2 n +1

f(0) = 1

f(1) = 2 • 1+ 1= 3 = 1 + 2 = f(0) + 2

f(2) = 2 • 2+ 1= 5 = 3 + 2 = f(1) + 2

f(3) = 2 • 3+ 1= 7 = 5 + 2 = f(2) + 2

…..

f(n) = 2 n +1= 2 (n −1+1) + 1 = 2 (n −1) +1 + 2 = f(n−1) + 2

Quindi

1. f(0)=1 e 2. f(n)= f(n−1) +2 per n≥1

Esempio: definire ricorsivamente la funzione che somma i primi n interi positivi f(n) = 1+2+3+…. +n per n≥1

- f(1)=1

So che f(n) = 1+2+3+…. +n−1 + n

- f(n) = (1+2+3+…. +n−1) + n
  = f(n−1) +n

Quindi

1. f(1)=1
2. f(n)= f(n−1) +n per n≥2

Esempio: definire ricorsivamente la funzione f(n) = n^2 per n≥1

- f(1) = 1

So che f(n-1) = $(n-1)^2$ Devo arrivare a f(n) =n^2

Come posso "manipolare" f(n-1) = $(n-1)^2$ per arrivare a f(n) =n^2 ?

- f(n-1) =$(n-1)^2$ = n^2 - 2n +1= f(n) - 2n +1

Quindi

1. f(1)=1 e 2. f(n)= f(n-1) +2n-1 per n≥2

5.2 CALCOLO DI FUNZIONI RICORSIVE

Esempio: sia f una funzione ricorsiva definita come

1. Passo base f(0)=3
2. Passo ricorsivo f(n)=2 f(n−1) +3 per n≥1

Quale è il valore di :

- f(3) = ?
- f(3) = 2 f(2) + 3 Passo ricorsivo
- f(2) = 2 f(1) + 3 Passo ricorsivo
  f(1) = 2 f(0) + 3 Passo ricorsivo
  f(0) = 3 Passo base
  f(1) = 2 • 3 + 3 = 9
  f(2) = 2• 9 + 3 = 21
  f(3) = 2• 21 + 3 = 45

Esempio: Funzione fattoriale

1. Passo base 0!=1
2. Passo ricorsivo n! = n (n − 1)! per n≥1

Calcoliamo 4!

- 4! = 4 • 3! Passo ricorsivo
  = 4 • 6 =24
- 3! = 3 • 2! Passo ricorsivo
  = 3 • 2 =6
- 2! = 2 • 1! Passo ricorsivo
  = 2 • 1 =2
- 1! = 1 • 0! Passo ricorsivo
  = 1 • 1 =1
- 0! = 1 Passo base

Esempio: 1. f(1)=1 e 2. f(n)= f(n-1) +2n-1 per n≥2

- f(4) = f(3) + 2 • 4 - 1
- f(3) = f(2) + 2 • 3 - 1
- f(2) = f(1) + 2 • 2 - 1
- f(1) = 1

f(2) = f(1) + 2 • 2 - 1 = 1 + 2 • 2 - 1 = 4

f(3) = f(2) + 2 • 3 - 1 = 4 + 2 • 3 - 1 = 9

f(4) = f(3) + 2 • 4 - 1 = 9 + 2 • 4 - 1 = 16

5.3 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 1: 1. f(0)=3 e 2. f(n)=2 f(n−1) +3 per n≥1

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

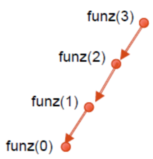
Tale procedura realizza quanto abbiamo fatto precedentemente nella slide 16

f(3) = 2 f(2) + 3

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

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

f(0) = 3



f(3) = 2 f(2) + 3

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

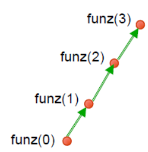
f(1) = 2 f(0) + 3

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: (correttezza = produce output corretto per ogni possibile input)

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

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

Dimostrazione:

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.

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).***

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)***

Esempio 2: funzione fattoriale

1. 0!=1 e 2. n!=n (n−1)! per n≥1

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

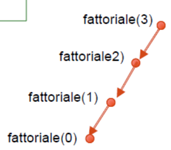
Dovendo calcolare fattoriale(3)

fattoriale(3) = 3 • fattoriale(2) =

fattoriale(2) = 2 • fattoriale(1) =

fattoriale(1) = 1 •fattoriale(0)

fattoriale(0) = 1



fattoriale(3) = 3 • fattoriale(2) =

fattoriale(2) = 2 • fattoriale(1) =

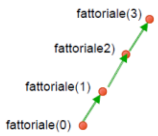
fattoriale(1) = 1 •fattoriale(0)

fattoriale(0) = 1

fattoriale(1) = 1 • 1 = 1

fattoriale(2) = 2 • 1 = 2

fattoriale(3) = 3 • 2 = 6



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

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

Dimostrazione:

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.

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)!***

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

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

Esempio 3: Numeri di Fibonacci

1. F(0)=0, F(1)=1 e 2. F(n) = F(n−1) + F(n−2) per n≥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)
```

Dovendo calcolare Fibonacci(4)

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

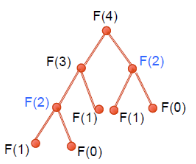
Fibonacci(3) = Fibonacci(2) + Fibonacci(1)

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

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

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

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

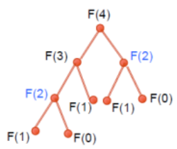


NOTA Non sempre gli algoritmi ricorsivi sono efficienti, essi però sono semplici da progettare

Nel calcolo di Fibonacci(4) con l'algoritmo ricorsivo valutiamo due volte Fibonacci(2)

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

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



Consideriamo ora una ***procedura iterativa*** per il calcolo dei ***numeri di Fibonacci***:

1. F(0)=0, F(1)=1

2. F(n) = F(n-1) + F(n-2) per n≥3

```
Procedure Fibonacci-iterativa(n)
if n=0 then return 0
      else if n=1 then return 1
                 else x=0, y=1
                      for i=2 to n do
                            z= x+y
                            x=y
                            y=z
return y
```

x e y conterranno sempre
x = F(n-2) y=F(n-1)

F(n) = z = x+y = F(n-2) + F(n-1)

Per prepararsi alla prossima iterazione (calcolare F(n+1)), dobbiamo aggiornare x e y:
x dovrà contenere F(n-1) e
y dovrà contenere F(n)

Ciascun numero di Fibonacci viene calcolato esattamente ***una volta***.

5.4 USO DI DEFINIZIONI RICORSIVE

Le definizioni ricorsive possono essere usate nelle ***dimostrazioni***:

Esempio:

Sia F(n) n-simo termine della sequenza dei numeri di Fibonacci.

Provare che per $n \geq 3$, $F(n) > \alpha^{n-2}$ dove $\alpha = (1+\sqrt{5})/2$

Dimostrazione:

Usiamo l'induzione forte:

Base: $F(3) = F(2) + F(1) = 1+1 = 2 > (1+\sqrt{5})/2 = \alpha$

$F(4) = F(3) + F(2) = 2+1 = 3 > [(1+\sqrt{5})/2]^2 = \alpha^2$

Ipotesi induttiva: Assumiamo che per $3 \leq j \leq n$ con $n \geq 3$

si ha $F(j) > \alpha^{j-2}$

Passo di induzione: Consideriamo ora F(n+1) e la sua definizione, poi applichiamo l'ipotesi induttiva

$F(n+1) = F(n-1)+F(n) > \alpha^{n-3} + \alpha^{n-2} = \alpha^{n-3}(1+\alpha) = \alpha^{n-3}\alpha^2 = \alpha^{n-1}$

Ricordate: 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 proprieta P}***

Un altro modo per descrivere ***insiemi*** è attraverso una definizione 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.

Nota: Gli elementi dell'insieme sono definiti esclusivamente dalle regole date.

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?

$1 \in A$ ***Passo base***

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

Le definizioni ricorsive possono essere usate per descrivere ***insiemi di stringhe***.
Un ***alfabeto*** e un insieme finito di elementi (chiamati lettere o simboli):

L'alfabeto delle ***lettere romane minuscole*** è $\Sigma = \{a,b,\ldots,z\}$.
L'alfabeto delle ***cifre arabe*** è $\Sigma = \{0,1,\ldots,9\}$.
L'alfabeto ***binario*** è $\Sigma = \{0,1\}$.

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 $wx \in \Sigma^*$

Esempio:

Sia $\Sigma = \{0,1\}$. Σ^* è l'insieme di tutte le stringhe binarie.

Infatti, $\lambda \in \Sigma^*$ applicando il ***Passo base***

0 e 1 $\in \Sigma^*$ applicando la prima volta il ***Passo ricorsivo***

00, 01, 10, 11 $\in \Sigma^*$ con la seconda applicazione

Le definizioni ricorsive possono essere usate per ottenere ***nuove definizioni ricorsive:***

La lunghezza ***l(w)*** di una parola ***w*** è definita come il numero di caratteri di cui ***w*** è costituita.
$\Sigma = \{a,b,\ldots,z\}$, zaino $\in \Sigma^*$ l(zaino) = 5

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(***w***x) = l(***w***) + 1

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

Una stringa è ***palindroma*** se letta da sinistra a destra o viceversa, è la stessa.
alla, otto, ingegni, Anna, ottetto.

L'insieme delle parole palindrome sull'alfabeto $\Sigma = \{a, b\}$ 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

Esempio:

abba è una parola palindroma:

- λ è una parola palindroma ***Passo base***
- $b\lambda b$ =bb è una parola palindroma ***Passo ricorsivo***
- a bb a è una parola palindroma ***Passo ricorsivo***

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 E_1 ed E_2 sono espressioni aritmetiche allora $(E_1 + E_2)$, $(E_1 - E_2)$, $(E_1 \times E_2)$, $(E_1 \backslash E_2)$ sono espressioni aritmetiche.

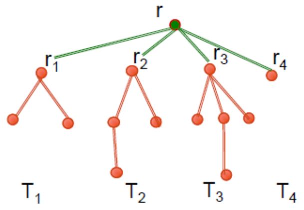
Se E è un'espressione aritmetica allora $(-E)$ è un'espressione aritmetica.

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

Un ***albero radicato*** può essere descritto ricorsivamente nel modo seguente:

Base: un singolo vertice r è un albero radicato

Passo ricorsivo: Supponiamo che $T_1, T_2, \ldots, T_n$ sono alberi radicati disgiunti con <u>radici</u> $r_1, r_2, \ldots, r_n$.
Allora, il grafo formato dalla <u>radice</u> r, che non è in nessuno degli alberi radicati $T_1, T_2, \ldots, T_n$, ottenuto connettendo con un arco r a ciascun $r_1, r_2, \ldots, r_n$ è anch'esso un albero radicato.



Più formalmente, un albero radicato T=(V, E) è definito:

Base: un singolo vertice r è un albero radicato, $T = (\{r\}, \emptyset)$

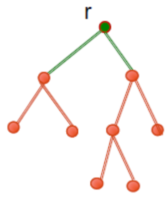
Passo Ricorsivo: Supponiamo che $T_1= (V_1,E_1)$, $T_2= (V_2,E_2)$, …, $T_n= (V_n,E_n)$ sono alberi radicati disgiunti, cioè $V_i \cap V_j =$ per $1 \leq i \neq j \leq n$ con <u>radici</u> $r_1, r_2, \ldots, r_n$, dove $r_1 \in V_1$, $r_2 \in V_2$, …, $r_n \in V_n$.
Allora, il grafo $T=(V,E)$ con $V=\{r\} \cup V_1 \cup V_2 \cup \ldots \cup V_n$ ed $E = \{(r,r_1),(r,r_2), \ldots, (r,r_n)\} \cup E_1 \cup E_2 \cup \ldots \cup E_n$
formato dalla radice r che non è in nessuno degli alberi radicati $T_1, T_2, \ldots, T_n$, dove $r \in V$ e $r \notin V_1 \cup V_2 \cup \ldots \cup V_n$ ottenuto connettendo con un arco r a ciascun $r_1, r_2, \ldots, r_n$, dove $(r,r_1) \in E$, $(r,r_2) \in E$, …, $(r,r_n) \in E$ è anch'esso un albero radicato.

Un ***albero binario pieno*** è un albero radicato dove ciascun vertice ha 0 o 2 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 T_1 e T_2 sono alberi binari pieni, allora l'albero T formato connettendo la radice r con un arco alla radice del sottoalbero sinistro T_1 e con un altro arco la radice del sottoalbero destro T_2 è un albero binario pieno.

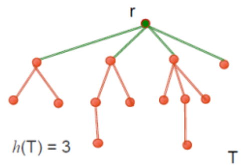


Le definizioni ricorsive possono essere usate per ***ottenere nuove definizioni ricorsive***.

L'altezza h(T) di un albero radicato T è definita ricorsivamente come:

Passo base: $h(T) = 0$ se T consiste della sola radice r

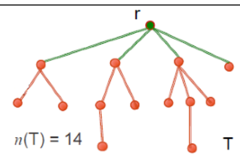
Passo ricorsivo: $h(T) = 1 + \max\{h(T_1), \ldots, h(T_n)\}$ se $T_1, \ldots, T_n$ sono i sottoalberi di T



Il **numero di vertici n(T)** di un albero radicato T è definita ricorsivamente come:

Passo base: $n(T) = 1$ se T consiste della sola radice r

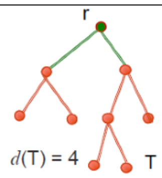
Passo ricorsivo: $n(T) = 1 + n(T_1) + \ldots + n(T_n)$ se $T_1, \ldots, T_n$ sono i sottoalberi di T



Il **numero di vertici con due figli d(T)** di un albero binario pieno T è definito ricorsivamente come:

Passo base: $d(T) = 0$ se T consiste della sola radice

Passo ricorsivo: $d(T) = 1 + d(T_1) + d(T_2)$ se T_1 e T_2 sono i sottoalberi di T



Il **numero di foglie f(T)** di un albero binario pieno T è definito ricorsivamente come:

Passo base: $f(T) = 1$ se T consiste della sola radice

Passo ricorsivo: $f(T) = f(T1) + f(T2)$ se T1 e T2 sono i sottoalberi di T