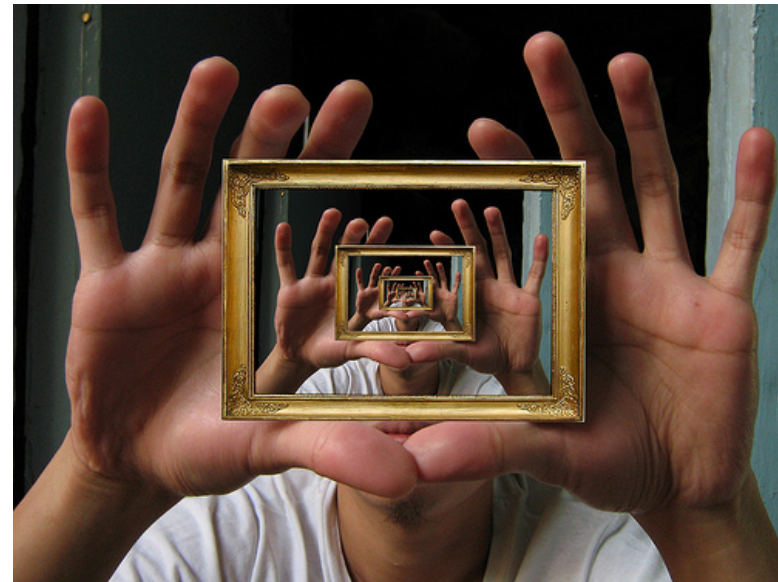




La ricorsione



*Iterare è
umano, usare
la ricorsione
divino.*



(Anonimo)



Introduzione

- Problema: Si consideri il problema di dover versare la somma di 16,51€ con il minor numero possibile di pezzi, tra banconote e monete.



Introduzione

- Soluzione: Il problema può essere suddiviso in due sottoproblemi: versare 10,00€ e versare 6,51€. Il primo sottoproblema può essere subito risolto con una banconota da 10,00€, mentre il secondo sottoproblema è molto simile al problema di partenza, ad eccezione del fatto che la somma da versare è più piccola.

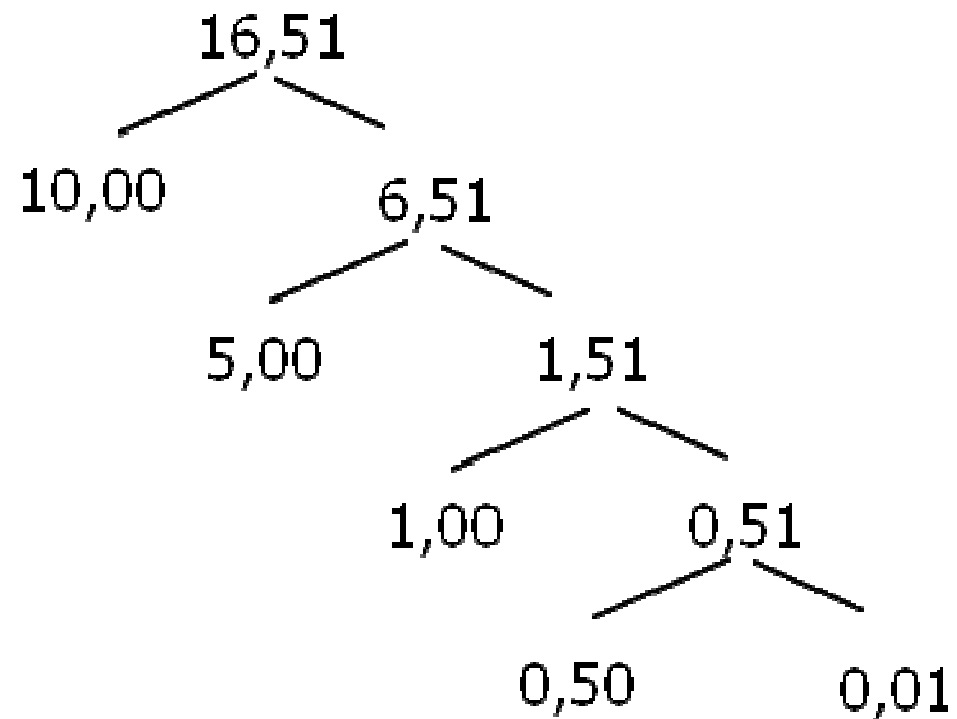


Introduzione

- Generalizzando il problema per una somma qualsiasi, la relativa soluzione consiste dei 2 passi seguenti:
 1. Cerca la banconota o la moneta il cui valore è il più prossimo, ma non superiore, alla somma da versare;
 2. Versa tale valore e risolvi, nello stesso modo, il problema di versar la parte restante della somma.



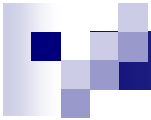
Introduzione





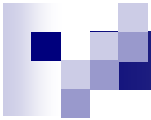
Introduzione

- La ricorsività è semplicemente il processo per risolvere un problema riducendolo in sottoproblemi che sono:
 - Identici in struttura al problema originale
 - Un po' più semplice da risolvere



Basi matematiche della ricorsione: il principio di induzione.

- Il termine ricorsione indica la possibilità in una definizione di far ricorso alla definizione stessa (definizione ricorsiva).
- Definizione ricorsiva (o per induzione) dei numeri naturali N (Peano):
 - 0 è un numero naturale
 - se n è un numero naturale, allora $n+1$ è un numero naturale



Basi matematiche della ricorsione: il principio di induzione.

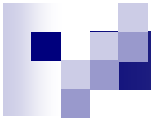
- Consideriamo un enunciato o proposizione che contenga come parametro un numero naturale n e la indichiamo genericamente con $P(n)$.
- Ci chiediamo se $P(n)$ sia vera o falsa.
- Eulero (1707-1783, matematico e fisico svizzero) ha proposto lo studio dei valori del trinomio $n^2 + n + 41$.

Basi matematiche della ricorsione: il principio di induzione.

- $P(n) = n^2 + n + 41$ è un numero primo?

- $P(40) = 40^2 + 40 + 41 =$
 $40(40+1) + 41 =$
 $40 \cdot 41 + 41 =$
 $41(40 + 1) = 41^2$

n	$n^2 + n + 41$
0	41 primo
1	43 primo
2	47 primo
3	... primo
.....
39	... primo
40	41^2 non primo



Basi matematiche della ricorsione: il principio di induzione.

- E' ingenuo ritenere che una certa $P(n)$ sia vera per tutti gli n se è vera solo per alcuni.
- Data una $P(n)$ basta un contro-esempio per concludere che è falsa.
- Per dimostrare che $P(n)$ è vera non bastano 10, 100, 1000... verifiche.



Principio di induzione

- Il principio di induzione (matematica) è una tecnica per dimostrare che una famiglia di proposizioni $P(n)$ è vera $\forall n \in \mathbb{N}$.
- $P(n) = 1 + 2 + 3 + \dots + n = n(n+1)/2$



Principio di induzione

Principio di induzione:

Sia $P(n)$ un enunciato parametrizzato rispetto ad un numero naturale n .

Dalla verità dei seguenti due enunciati:

- Base di induzione: $P(0)$ è vero;
- Passo induttivo: detto n un generico numero naturale, se $P(n)$ è vero (ipotesi di induzione) allora anche $P(n+1)$ è vero;

si deduce che $P(n)$ è vero $\forall n \in \mathbb{N}$.



Esempio di dimostrazione

- Dimostrare che la somma dei numeri naturali da 0 ad n , vale $n(n+1)/2$ (formula di Gauss (1777-1855, matematico, astronomo e fisico tedesco)).

$P(0)$ è vera, in quanto $0 = 0(0+1)/2$

Suppongo che $P(n)$ sia vera (ipotesi di induzione) e cioè che:

$$P(n) = 0+1 +2 + 3 + \dots + n = n(n+1)/2$$

$P(n+1) = 0 + 1 + 2 + 3 + \dots + n + (n+1) = P(n) + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{(n(n+1) + 2(n+1))}{2} = \frac{(n+1)(n+2)}{2}$ che è proprio ciò che volevamo dimostrare.

Quindi $P(n)$, per il principio di induzione, è vera $\forall n \in \mathbb{N}$.

$$\sum_{k=0}^n 2^k = 2^{n+1} - 1$$

Esercizio

- Dimostra per induzione.
- Giustifica tutti i passi della dimostrazione.

$$\sum_{k=0}^n 2^k = 2^{n+1} - 1$$



Definizioni ricorsive

■ Definizione ricorsiva (o per induzione) di fattoriale di un numero naturale n .

□ $0! = 1$ (chiusura della ricorsione/base di induzione)

□ $n! = n (n - 1)!$ (passo ricorsivo/passio induttivo)

Definizione iterativa di fattoriale: il fattoriale di n è il prodotto dei numeri naturali compresi tra 1 ed n (estremi inclusi). ($0!=1$)



Definizioni ricorsive

- Giustificazione della posizione $0! = 1$
 $n! = 1 * 2 * 3 * \dots * n$ per $n \geq 1$

$$(n+1)! = (n+1)n!$$

Per $n = 0$, sostituendo si ha:

$$1! = 1 * 0! \Rightarrow 0! = 1/1 = 1$$

$$n! := \prod_{k=1}^n k = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$$



Definizioni ricorsive

- Calcolare 3! usando la definizione (ricorsiva).

$$\begin{aligned} 3! &= 3 * 2! = 3 * 2 * 1! = 3 * 2 * 1 * 0! = \\ &= 3 * 2 * 1 * 1 = 6 \end{aligned}$$



Definizioni ricorsive

- L'operazione di prodotto fra due numeri naturali può essere formalmente definita come segue:

- $\text{prod}(n, 0) = 0$ (base di induzione)

- $\text{prod}(n, m+1) = \text{prod}(n, m) + n$ (passo induttivo)

Es.: $\text{prod}(3, 2) = \text{prod}(3, 1+1) = \text{prod}(3, 1) + 3 =$
 $= \text{prod}(3, 0+1) + 3 = \text{prod}(3, 0) + 3 + 3 = 0 + 3 + 3 = 6$



Definizioni ricorsive

- Definizione ricorsiva della potenza di un numero $a \neq 0$.

$$\square a^n = 1 \quad \text{se } n = 0$$

$$\square a^n = a * a^{n-1} \quad \text{se } n > 0$$

$$\begin{aligned} \text{Es.: } 6^4 &= 6 * 6^3 = 6 * 6 * 6^2 = 6 * 6 * 6 * 6^1 = \\ &= 6 * 6 * 6 * 6 * 6^0 = 6 * 6 * 6 * 6 * 1 = \\ &= 1296 \end{aligned}$$



Definizioni ricorsive

■ Giustificazione che $a^0 = 1$

$$a^n = a * a * \dots * a \quad (n \text{ fattori})$$

$$a^{n+1} = a * a^n$$

$$\text{per } n = 1, a^2 = a * a^1 \Rightarrow a^1 = a * a / a = a$$

$$\text{per } n = 0, a^1 = a * a^0 \Rightarrow a^0 = a / a = 1$$



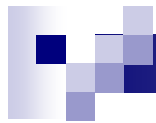
Esercizio

- Per ciascuna successione di numeri naturali si individui una formula o regola esplicita che permetta di trovare qualsiasi termine della successione:
 - ☐ A) 8, 12, 16, 20, 24, ...
 - ☐ B) 4, 7, 10, 13, 16, ...



Soluzione

- A) 8, 12, 16, 20, 24, ..., $4n + 4$, ...
- B 4, 7, 10, 13, 16, ..., $3n + 1$, ...



Esercizio 2

- Prova a scrivere ora una regola ricorsiva per entrambe le successioni.



Soluzione

$$\begin{cases} P_1 = 8 \\ P_n = P_{n-1} + 4, n \geq 2 \end{cases}$$

$$\begin{cases} S_1 = 4 \\ S_n = S_{n-1} + 3, n \geq 2 \end{cases}$$



Esercizio

- Dimostrare che un insieme di n elementi ha 2^n sottoinsiemi.
- Suggerimento: Si dimostri che tale asserzione è vera per $n=0$. La si supponga vera per $n-1$ e la si dimostri per n .



La programmazione ricorsiva

- Una funzione è detta **ricorsiva** quando direttamente (ricorsione diretta) o indirettamente (ricorsione mutua) chiama se stessa.
- Ricorsione dal latino re-currere (ricorrere, fare ripetutamente la stessa azione).
- Il concetto di ricorsione viene usato nel contesto di
 - Algoritmi
 - Strutture dati (liste, alberi, ...)



Prima funzione ricorsiva: fattoriale

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



Versione iterativa del fattoriale

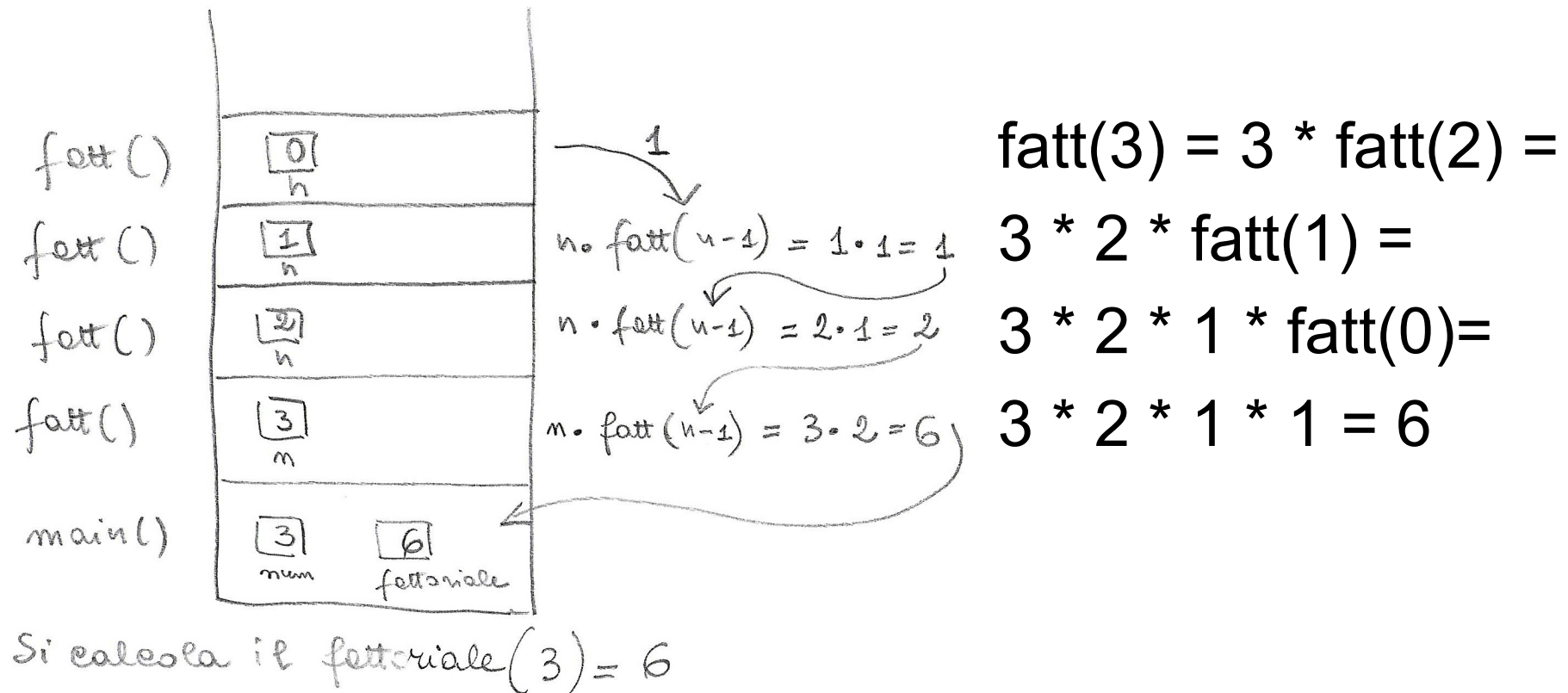
```
unsigned long fatt (unsigned int n){  
    unsigned long ris=1;  
    while (n > 1){  
        ris = ris * n;  
        n --;  
    }  
    return ris;  
}
```



Analisi funzione ricorsiva fattoriale

- Calcoliamo $\text{fatt}(3)$, indicando passo passo le operazioni che vengono eseguite dal sistema e che rimangono nascoste al programmatore.

Analisi funzione ricorsiva fattoriale





Analisi funzione ricorsiva fattoriale

- Nel “ramo allora” troviamo la condizione di chiusura delle chiamate successive.
- Nel “ramo altrimenti” troviamo la definizione vera e propria che ha la forma di una nuova chiamata di se stessa $\text{fatt}(n-1)$.
- L'algoritmo decrementa il valore di n fino a 0, conservando in un apposito spazio di memoria, in pila uno sull'altro, i valori successivi del parametro n fino al valore 0. In questa fase di chiamate ricorsive non viene eseguito alcun calcolo; non appena $n = 0$ vengono eseguiti i calcoli a ritroso secondo i valori di n prelevati dalla pila, in ordine inverso rispetto a quello dell'immissione.



Osservazioni

- E' importante notare come i valori di n vengono modificati ad ogni chiamata, mentre la pila (STACK) consente di non perdere i valori precedenti.
- Nei linguaggi che consentono la ricorsione questa gestione è un fatto interno al sistema della quale il programmatore non si deve curare.
- La traccia indicata consente una verifica sulla correttezza delle funzioni/procedure ricorsive adottate ed è un aiuto al programmatore all'atto della verifica di un pgm.



Perché si usa uno STACK?

- La convenienza nell'uso dello STACK è che le procedure terminano in ordine inverso a quello di attivazione e che quindi i record stessi seguono la medesima sorte.



Funzione ricorsiva per il calcolo del MCD

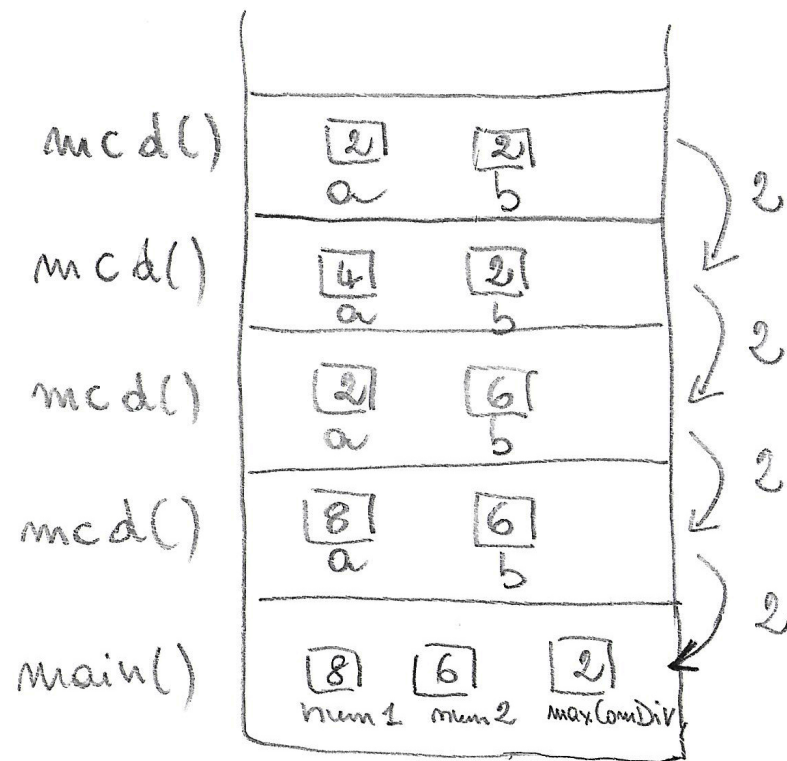
- Scrivere una funzione C che calcoli il MCD fra due interi a e b con il metodo ricorsivo della sottrazione.
- $\text{MCD}(a,b) = a$ se $a = b$
- $\text{MCD}(a,b) = \text{MCD}(a-b,b)$ se $a > b$
- $\text{MCD}(a,b) = \text{MCD}(b-a,a)$ se $a < b$



Funzione ricorsiva per il calcolo del MCD

```
int mcd (int a, int b){  
    if(a == b)  
        return a;  
    else if (a > b)  
        return mcd(a-b,b);  
    else  
        return mcd(b-a,a);  
}
```

Funzione ricorsiva per il calcolo del MCD: STACK



Si vuole calcolare $\text{mcd}(8,6) = 2$

- In questo caso, raggiunta la condizione di chiusura $a = b$ non esistono calcoli a ritroso, poiché le chiamate ricorsive non prevedono operazioni, quindi l'ultimo valore in cima alla pila, oltre a rappresentare la condizione di chiusura, trasmette anche il valore del MCD cercato.



Calcolo del MCD secondo l'alg. di Euclide (300 a.C., matematico greco)

- $\text{MCD}(a,b) = a$ per $b = 0$
- $\text{MCD}(a,b) = \text{MCD}(b,a\%b)$ per $a > 0$ e $b > 0$

```
int mcd (int a, int b) {  
    return (b) ? mcd(b,a%b):a;    //operatore ternario  
}
```

Equivalente a:

```
if(b)  
    return mcd(b,a%b);  
else  
    return a;
```



Versione iterativa dell'alg. di Euclide

```
int mcd (int a, int b) {  
    if(b==0)  
        return 0;  
    while(a%b) {  
        int r = a%b;  
        a = b;  
        b = r;  
    }  
    return b;  
}
```



Esercizio

- Calcolare il MCD fra 4 interi, scrivere una sola istruzione.



Soluzione

```
int risultato;
```

```
risultato = mcd(mcd(a,b),mcd(c,d));
```



Osservazione

- E' buona norma ricordare le regole applicate nella formulazione degli alg. ricorsivi, in quanto condizioni necessarie non sufficienti per ottenere la correttezza, precisamente:
 1. Deve essere sempre presente la condizione di chiusura della ricorsione.
 2. Ogni successiva chiamata deve riferirsi a valori minori di quello di partenza o ad un sottoinsieme dell'insieme originale di elementi.



Lo “spirito” del metodo ricorsivo

- Esiste un CASO BASE, rappresentante un sotto-problema facilmente risolubile.
- Per es., nel caso del fattoriale, se $n=0$ so calcolare il fattoriale di n , vale 1.
- Negli altri casi ci si deve ricondurre al caso base.



Osservazione - Esempio

```
int mistero (int x) {  
    if (x == 0)  
        return 1;  
    else  
        return mistero(x - 2);  
}
```

- In essa sono rispettate le due condizioni sopra citate e tuttavia, se x è dispari o se x è negativo il calcolo non termina o termina con una segnalazione di errore “stack overflow error”



Numeri di Fibonacci

(1170-1250, matematico italiano(PISA))

- La potenza di una definizione ricorsiva è legata alla possibilità di definire un insieme di oggetti infinito mediante una descrizione finita.

- Numeri di Fibonacci

$$f_0 = 1, f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2} \quad \text{per } n \geq 2$$

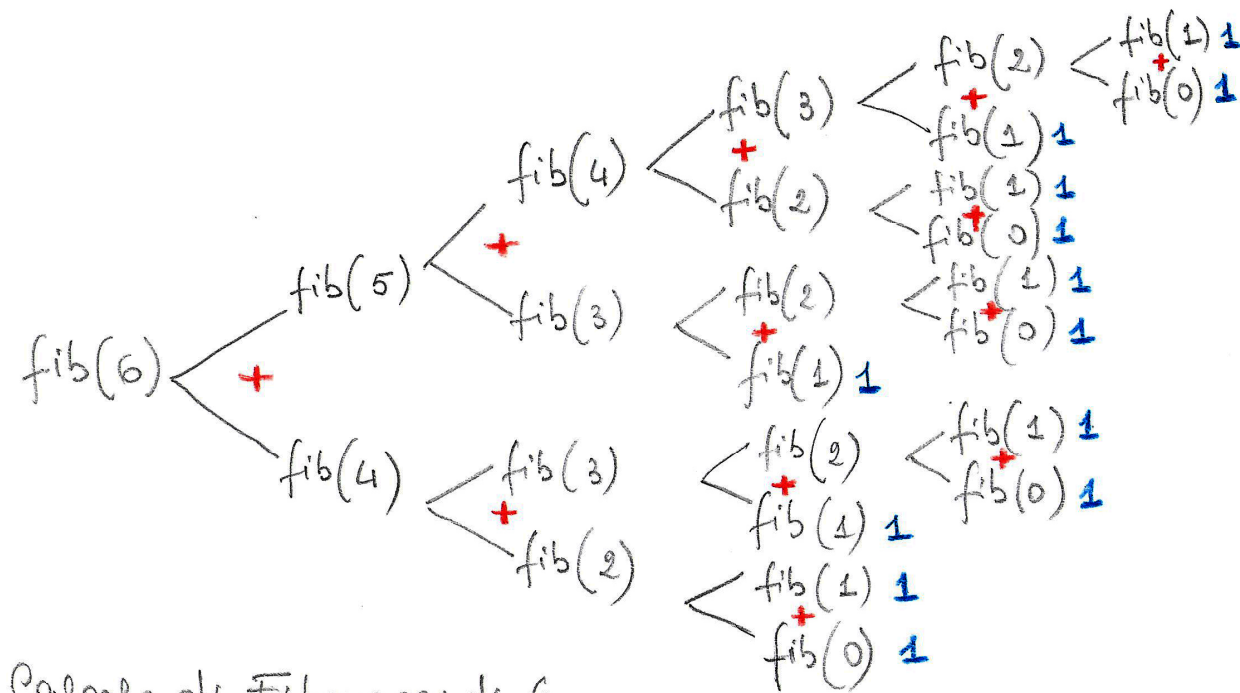


Numeri di Fibonacci – versione ricorsiva

```
int fibonacci (int n) {  
    if(n == 0 || n == 1)  
        return 1;  
    else  
        return fibonacci(n-1) + fibonacci(n-2);  
}
```

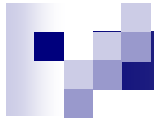
Numeri di Fibonacci

- Disegnare l'albero delle chiamate ricorsive per il calcolo di Fibonacci di 6.



Calcolo di Fibonacci di 6

F_0	F_1	F_2	F_3	F_4	F_5	F_6
1	1	2	3	5	8	<u>13</u>



Osservazione

- Inefficienza: Le funzioni ricorsive alcune volte ripetono dei calcoli già eseguiti in precedenza (v. albero delle chiamate).



Fibonacci – versione iterativa

```
int fibonacci (int num) {  
    if (num==0) return 1;  
    else if (num==1) return 1;  
    else  
    {  
        int fib, prec1,prec2;  
        fib=prec1=1;  
        for(int i=2; i<num; i++) {  
            prec2 = prec1;  
            prec1 = fib;  
            fib = prec1+prec2;  
        }  
    }  
    return fib;  
}
```



Esercizio

- Scrivere una funzione C++ che determini se una parola è palindroma.
- Una parola è palindroma se i caratteri più a sx e più a dx coincidono e se la parte tra loro compresa è palindroma.
- Es.: radar, oro, anna, ...



Metodologia top-down

- La metodologia top-down è descrivibile in termini ricorsivi:


Metodologia Top-Down applicata ad un problema P

SE P è elementare (“sufficientemente” semplice)

ALLORA risolvo direttamente

ALTRIMENTI

1. Individua in P un certo numero di sottoproblemi
2. Risolvi con la **Metodologia Top-Down** ciascun sottoproblema
3. Costruisci la soluzione finale sfruttando le sotto-soluzioni



Strutture dati che si autoreferenziano o strutture ricorsive

```
struct persona{  
    char nome[19];  
    char cognome [25];  
    char nascita[9];  
    char professione[15];  
    struct persona * link;  
};
```



Ricerca binaria ricorsiva

```
#define MAX_QUADRI 20
```

```
struct quadro {  
    char nome[25];           //chiave primaria  
    char autore[30];  
    double prezzo;  
    char data[9];  
};
```

```
int insert(struct quadro galleria[]);  
int ricBin(const struct quadro galleria[],const char dipinto[],int inizio,int fine);
```

```
struct quadro pinacoteca[MAX_QUADRI];
```



Ricerca binaria ricorsiva

```
int inizio=0;
int fine=numQuadri-1;
char dipinto[25];
printf("Quale quadro vuoi cercare?\n");
gets(dipinto);
int posiz=ricBin(pinacoteca,dipinto,inizio,fine);
if(posiz!=-1){
    printf("%s %f\n",pinacoteca[posiz].autore,pinacoteca[posiz].prezzo,pinacoteca[posiz].data);
} else {
    printf("Il quadro richiesto non e' presente\n");
}
```



Ricerca binaria ricorsiva

```
int ricBin(const struct quadro galleria[],const char dipinto[],int inizio,int fine) {  
    if(inizio>fine)  
        return -1;  
    else{  
        int centro=(inizio+fine)/2;  
        if(!strcmp(galleria[centro].nome,dipinto))  
            return centro;  
        else if(strcmp(galleria[centro].nome,dipinto) > 0)  
            return ricBin(galleria,dipinto,inizio,centro-1);  
        else  
            return ricBin(galleria,dipinto,centro+1,fine);  
    }  
}
```



Ricerca binaria iterativa

```
struct piatto{
    char nome[25];           //chiave primaria
    int prezzo;
};

int ricercaBinaria (const struct piatto menu[], int numPiatti, const char  nome[]);

struct piatto menu[MAX_PIATTI];

char nomePiatto[25];

printf("Quale piatto vuoi cercare ?\n");
gets(nomePiatto);

int posiz=ricercaBinaria(menu,numPiatti,nomePiatto);

if(posiz!=-1)
    printf("Elemento presente\n");
else
    printf("Elemento assente\n");
```




Ricerca binaria iterativa

```
int ricercaBinaria(const struct piatto menu[], int numPiatti, const char nomePiatto[]) {
    int inizio=0;
    int fine=numPiatti -1;
    int pos=-1;
    do {
        int i=(inizio+fine)/2;
        if (!(strcmp(nome,menu[i].nomePiatto)))
            return i;
        else if(strcmp(nome,menu[i].nomePiatto) > 0)
            inizio=i+1;
        else
            fine=i-1;
    }
    while(inizio<=fine);
    return pos;
}
```



Struttura (generale) della tecnica ricorsiva

- Praticamente tutte le funzioni ricorsive hanno la stessa struttura di base:

```
if (test per verificare se siamo nel caso base){  
    //chiusura della ricorsione  
    Metodo diretto, cioè non ricorsivo, per ottenere la soluzione  
} else {  
    //passo ricorsivo  
    Metodo per ottenere la soluzione che coinvolge chiamate alla funzione  
    stessa ma su istanze del problema più piccole  
}
```

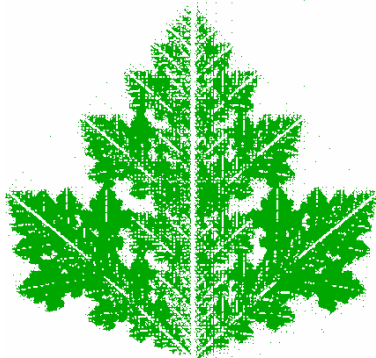


Natura Matematica Informatica (Ricorsione)

.....

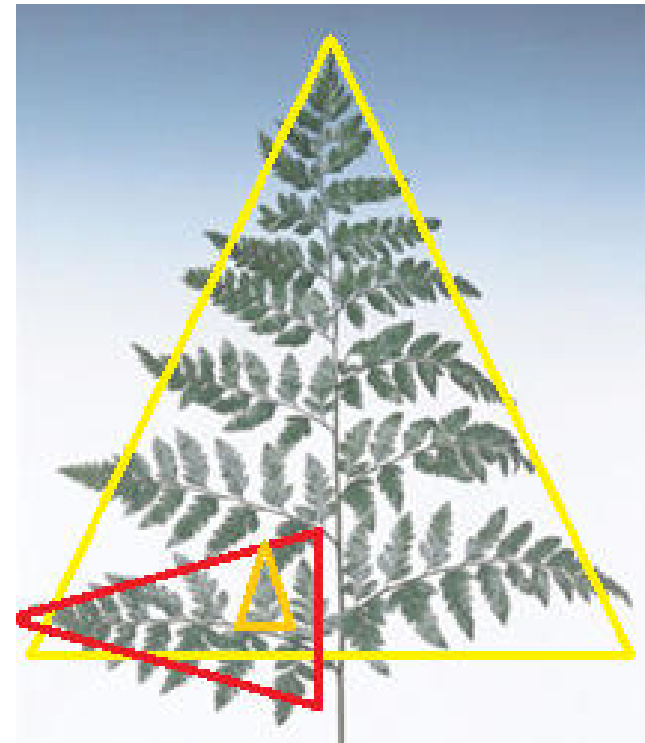


- Cristalli, foglie, fiocchi di neve, rami degli alberi, conchiglie, felci, ...
- Le cose elencate (e molte altre ancora in natura) hanno qualcosa in comune: **il loro sistema di accrescimento** che si basa su alcune regole numeriche e geometriche.



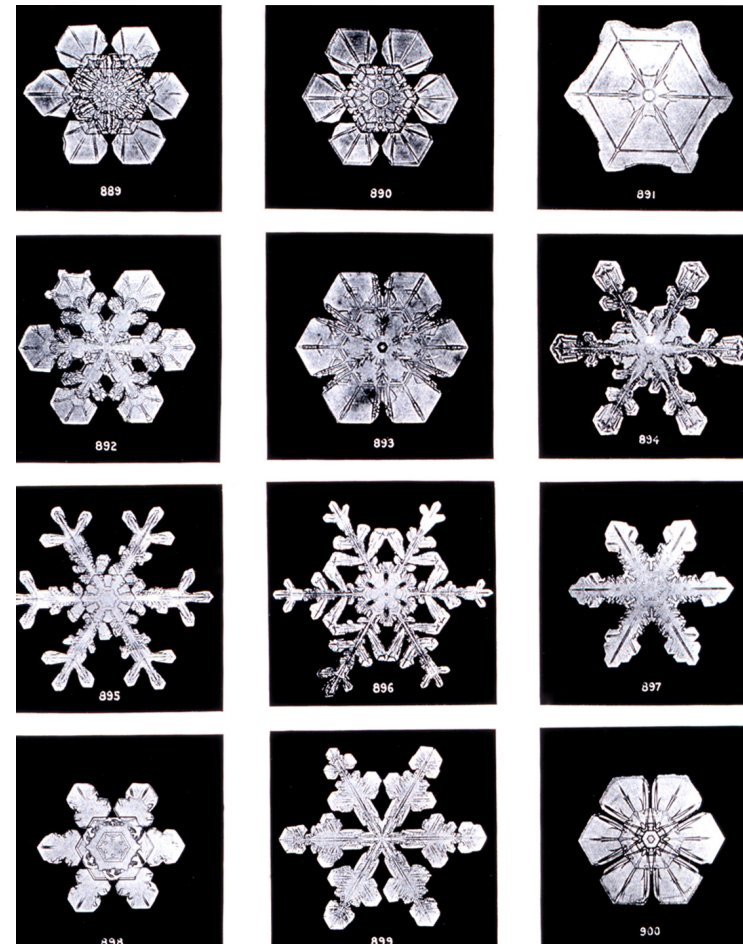


- La felce che vedete qui, ad esempio, è costituita da molte piccole parti che hanno la stessa forma della felce intera.



TUTTO HA ORIGINE NELL'ESAGONO

- I fiocchi di neve, per fare un altro esempio, se osservati al microscopio, mostrano moltissime forme diverse, ma tutte riconducibili ad un poligono di partenza: l'esagono regolare.



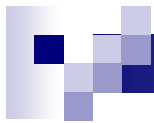


- Molti studiosi hanno accertato che alla base di questi fenomeni naturali ci sono regole matematiche "**ricorsive**" (cioè che si ripetono con regolarità e con ordine). E hanno creato (all'inizio manualmente e poi con l'aiuto del computer) disegni e forme ricorsive che si chiamano "**frattali**".



FRATTALI: geometria della natura

La geometria frattale è una recente branca della matematica; parte dall'osservazione che alcune forme presenti in natura (coste, rami di un albero, fiocchi di neve, ecc...) sono ben lontane dalle figure regolari della geometria euclidea, e quindi si propone di usare enti geometrici non convenzionali per “leggere” e “descrivere” proprio le forme di irregolarità presenti in natura.



Da Galileo Galilei ...

... a Benoit Mandelbrot

Galileo Galilei, che è universalmente considerato il padre del metodo scientifico, sintetizzava magistralmente il suo pensiero:

“Il libro della natura è scritto in lingua matematica ed i suoi caratteri sono triangoli, cerchi ed altre figure geometriche, senza i quali mezzi è impossibile intenderne umanamente parola; senza questi è un aggirarsi vanamente per un oscuro labirinto.” (Opere)



Da Galileo Galilei a Benoit Mandelbrot

A più di 3 secoli di distanza Benoit Mandelbrot:

“La geometria euclidea è incapace di descrivere la natura nella sua complessità, in quanto si limita a descrivere tutto ciò che è regolare. Tutti gli oggetti che hanno una forma perfettamente sferica, oppure ... mentre osservando la natura vediamo che le montagne non sono dei coni, le nuvole non sono delle sfere, le coste non sono dei cerchi, ma sono oggetti geometricamente molto complessi.”

(da Les objects fractals 1975”)

Benoit Mandelbrot, nato a Varsavia nel 1924, vive a New York



- Nascono i frattali, modelli atti ad imprigionare in formule matematiche quelle forme della natura come fiori, alberi, fulmini, fiocchi di neve, cristalli, che fin'ora non erano state considerate riproducibili con regole matematiche.
- La geometria frattale (dal latino frangere cioè spezzare) è lo studio di forme ripetitive di base che ci consentono di trovare le regole per generare alcune strutture presenti in natura.



- In questo modo Mandelbrot introduce la geometria frattale, che nasce come un nuovo linguaggio di descrizione delle forme complesse della natura; ma mentre gli elementi della geometria (linee, cerchi, triangoli,...) si possono visualizzare facilmente, quelli del nuovo linguaggio non si prestano all'osservazione diretta; essi sono algoritmi, processi che possono essere trasformati in forme e strutture solo con l'aiuto di un computer. E' proprio ciò che oggi avviene nelle produzioni cinematografiche, nelle quali interi paesaggi vengono ricostruiti al calcolatore, come se fossero reali, utilizzando costruzioni iterative o meglio ricorsive.



Cos'è un frattale

- "Figura geometrica o oggetto naturale con una parte della sua forma o struttura che si ripete a scala differente, con forma estremamente irregolare interrotta e frammentata a qualsiasi scala e con elementi distinti di molte dimensioni differenti".

Benoit Mandelbrot (les objets fractales, 1975)

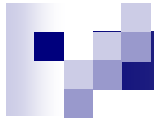


- I frattali sono figure geometriche caratterizzate dal ripetersi sino all'infinito di uno stesso motivo su scala sempre più ridotta.
- Questa è la definizione più intuitiva che si possa dare di figure che in natura si presentano con una frequenza impressionante ma che non hanno ancora una definizione matematica precisa.



Ricorsione

- I procedimenti iterativi o meglio ricorsivi, sono lo strumento base per lo studio delle immagini frattali.



Ricorsione

- Una legge ricorsiva è qualcosa di magico: quando in un fenomeno si coglie la ricorsione tutto si semplifica e si illumina.
- Quante sono le strette di mano fra n persone? Non lo so, ma se n persone si sono già salutate tutte con S_n strette ed arriva il signor $n+1$, questi deve salutare tutti gli altri: $S_{n+1} = S_n + n$, poiché $S_2 = 1$ il gioco è fatto.



Ricorsione

- Qualcuno può obiettare che la legge ricorsiva è più debole di quella funzionale: bastava, nell'esempio precedente, usare le combinazioni classe 2 su n elementi per evitare di dover calcolare tutti gli stadi fino ad n , ma aver colto il “germe” per passare da n ad $n+1$, mi fa dimenticare la complessità del problema e mi fa risolvere situazioni dove non è possibile trovare l'equivalente legge funzionale.

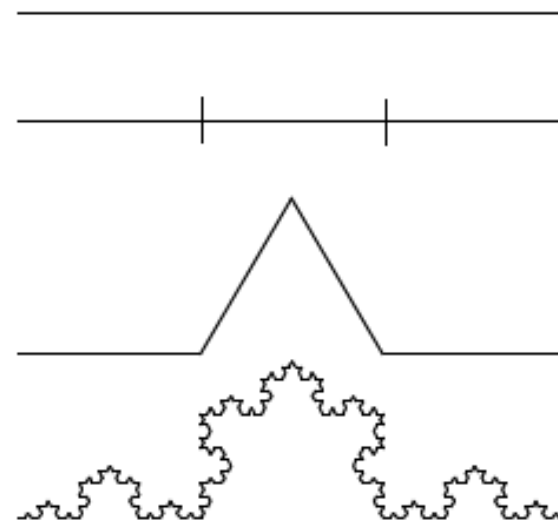


Analisi della curva di Von Koch: il merletto matematico e il fiocco di neve

- Hogle von Kock (1870-1924) studiando il problema delle aree, trovò un esempio di curva chiusa, continua ma non differenziabile, con perimetro infinito che limita un'area finita.

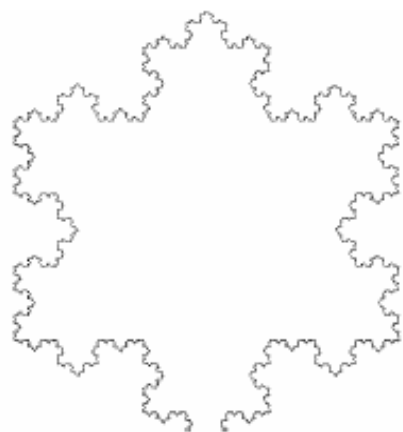
Analisi della curva di Von Koch: il merletto matematico e il fiocco di neve

- Si parte da un segmento di lunghezza l
- Si divide il segmento in tre parti uguali
- Si sostituisce la parte centrale con i due lati di un triangolo equilatero
- Si itera il procedimento per ogni nuovo segmento



Analisi della curva di Von Koch: il merletto matematico e il fiocco di neve

Koch aveva trovato una curva con tutte le caratteristiche di un frattale anche se non vide mai l'attrattore finale della sua curva elaborato



da un computer, potè solo immaginare cosa si sarebbe ottenuto iterando con infiniti passi.

Dalla curva frattale al fiocco di neve il passo è breve, basta cos-

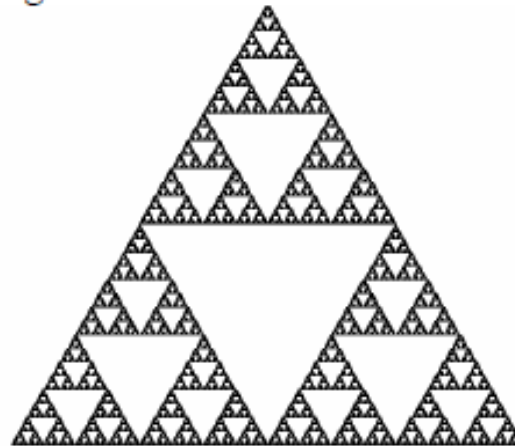
truire la curva sui 3 lati di un triangolo equilatero.

Comunque qualsiasi rappresentazione della curva, anche al computer, sarà fatta con un numero finito di iterazioni della funzione ricorsiva che la genera, mai avremo l'opportunità di vedere l'attrattore finale con infinite iterazioni. Tuttavia con lo sviluppo continuo ed esponenziale delle capacità di calcolo, e con l'uso del computer, si possono creare figure che hanno la stessa valenza matematica per la rappresentazione del frattale vero e proprio (quello che ha, cioè, significato matematico e che gode delle suddette proprietà) della valenza di un segno su un foglio per la rappresentazione della retta: un segmento disegnato su un foglio non è una retta è solo una sua rappresentazione, così come la figura disegnata con un programma iterativo che fa n passi non è un frattale ma solo una buona approssimazione man mano che n tende all'infinito.

Analisi del triangolo di Sierpinski



1. Si parte da un triangolo equilatero
2. Si sottrae al triangolo iniziale il triangolo avente come vertici i punti medi di ogni lato
3. Si itera il procedimento per ogni nuovo triangolo ottenuto



Esempio di frattale realizzato in Pascal.

Paesaggio: montagne innevate, una cattedrale in fondo ad un viale alberato.

