

La ricorsione

Andrea Marin

Università Ca' Foscari Venezia
Laurea in Informatica
Corso di Programmazione

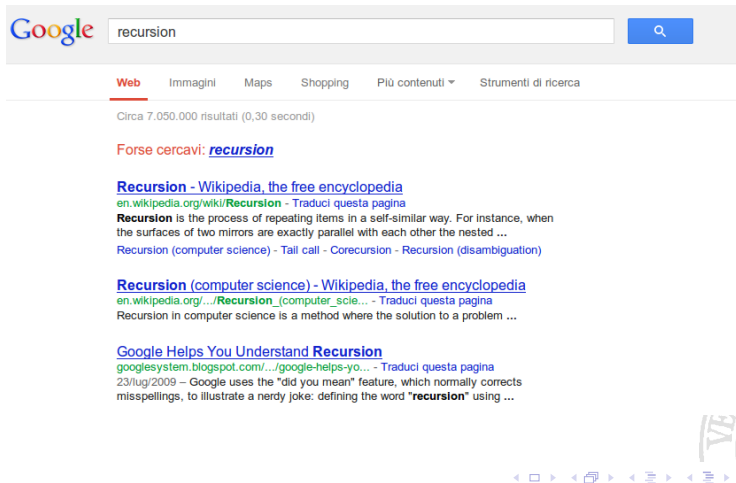
a.a. 2012/2013

Section 1

La ricorsione: idee generali



Cosa dice Google?



The screenshot shows a Google search interface. At the top, the Google logo is on the left, a search bar containing the word "recursion" is in the center, and a blue search button with a magnifying glass icon is on the right. Below the search bar, there are tabs for "Web", "Immagini", "Maps", "Shopping", "Più contenuti", and "Strumenti di ricerca". The "Web" tab is selected and underlined. Below the tabs, it says "Circa 7.050.000 risultati (0,30 secondi)".

Forse cercavi: [recursion](#)

[Recursion - Wikipedia, the free encyclopedia](#)
[en.wikipedia.org/wiki/Recursion](#) - Traduci questa pagina
Recursion is the process of repeating items in a self-similar way. For instance, when the surfaces of two mirrors are exactly parallel with each other the nested ...
[Recursion \(computer science\)](#) - [Tail call](#) - [Corecursion](#) - [Recursion \(disambiguation\)](#)

[Recursion \(computer science\) - Wikipedia, the free encyclopedia](#)
[en.wikipedia.org/.../Recursion_\(computer_scle...](#) - Traduci questa pagina
Recursion in computer science is a method where the solution to a problem ...

[Google Helps You Understand Recursion](#)
[googlesystem.blogspot.com/.../google-helps-yo...](#) - Traduci questa pagina
23/lug/2009 – Google uses the "did you mean" feature, which normally corrects misspellings, to illustrate a nerdy joke: defining the word "**recursion**" using ...

At the bottom right of the page, there is a faint watermark of the University of Ca' Foscari Venezia logo, which features a circular emblem with a lion's head and the text "UNIVERSITÄT VENEZIA" and "CA' FOSCARI VENEZIA".

Navigation icons: back, forward, home, search, and other standard browser controls.

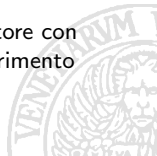
Intuizione

- ▶ Un algoritmo è ricorsivo quando la sua soluzione può dipendere dalla soluzione di un problema più semplice per la cui soluzione si può applicare lo stesso algoritmo
- ▶ Una funzione ricorsiva in C è una funzione che richiama (direttamente o indirettamente) se stessa
- ▶ La ricorsione termina al raggiungimento di un caso base, ossia di un input per il quale non è necessario effettuare la chiamata ricorsiva per il calcolo del corrispondente output



Quando è applicabile la soluzione ricorsiva?

- ▶ Quando la soluzione del problema sarebbe *semplice* se si conoscesse la soluzione allo stesso problema formulato per un input più semplice
- ▶ **Esempio:** decidere se un vettore di N elementi è ordinato
 - ▶ Supponiamo di saper decidere se un vettore di $N - 1$ elementi è ordinato in senso crescente
 - ▶ Posso quindi decidere se gli ultimi $N - 1$ elementi del vettore sono ordinati
 - ▶ Se non lo sono, allora il vettore non è ordinato
 - ▶ Se lo sono, allora confronto il primo elemento del vettore con il secondo, se è minore allora il vettore è ordinato, altrimenti non lo è



Raggiungimento del caso base

- ▶ Per essere sicuri che la ricorsione termini dobbiamo essere certi che per qualsiasi input si raggiungerà, a forza di semplificare il problema, il caso base
- ▶ Il caso base arresta la ricorsione perchè non effettua alcuna chiamata ricorsiva
- ▶ Nell'esempio precedente i casi base possono essere:
 - ▶ Array vuoto che è ordinato
 - ▶ Array con un elemento che è ordinato

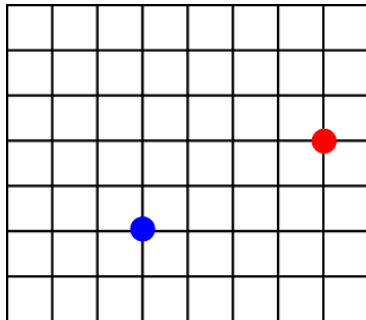


Section 2

Primo esempio: Le strade di Manhattan



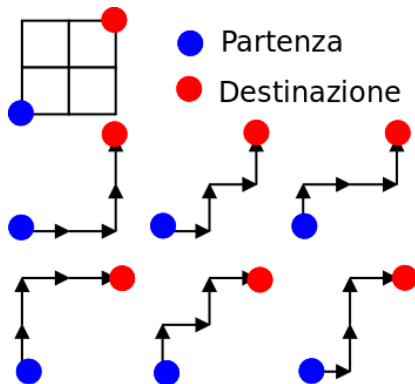
Problema



● Partenza
● Destinazione

- Dato un punto di partenza e uno di destinazione, quanti sono i percorsi di lunghezza minima fra la partenza e la destinazione?

Quanti sono i percorsi minimi in questo caso?

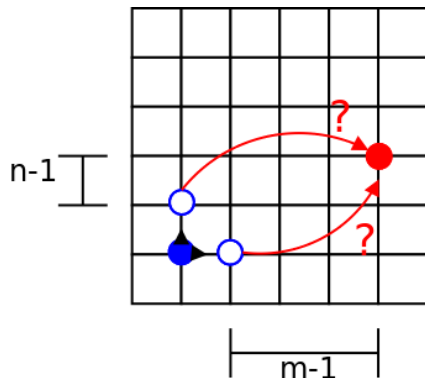


Osservazioni

- ▶ Il numero di percorsi dipende solamente dalla distanza orizzontale e verticale tra la partenza e la destinazione e non dalla loro posizione assoluta nella griglia
- ▶ Chiamiamo m la distanza orizzontale e n quella verticale
- ▶ Vogliamo definire la funzione $\text{manhattan}(m,n)$ che restituisce il numero di percorsi minimi tra una partenza e una destinazione che distano m e n caselle
 - ▶ Secondo quanto detto prima $\text{manhattan}(2,2) \rightarrow 6$



Impostazione ricorsiva



- Partenza
- Destinazione



Casi base

- ▶ Se $m = 0$ OR $n = 0$ allora c'è un solo percorso minimo, cioè quello dritto (verticale o orizzontale)



Codifica c

```
int manhattan(int n, int m) {  
    if (n==0 || m==0) /* caso base */  
        return 1;  
    else {  
        int percorsi;  
        percorsi = manhattan(n-1, m);  
        percorsi = percorsi + manhattan(n, m-1);  
        return percorsi;  
    }  
}
```



Section 3

Biglie rosse e blue

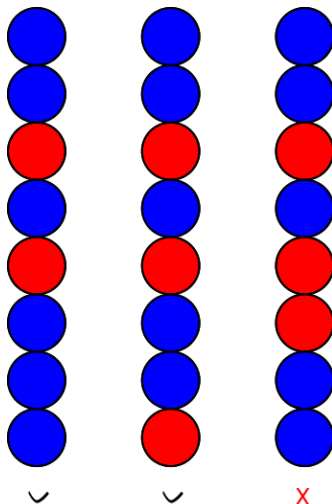


Biglie rosse e blue

- ▶ Supponiamo di avere una scatola con un numero illimitato di biglie blue e rosse e un cilindro di vetro
- ▶ Nel cilindro le biglie cadono una sopra l'altra
- ▶ Qualsiasi disposizione di biglie è valida tranne quelle che hanno due biglie rosse consecutive



Esempi con 8 biglie

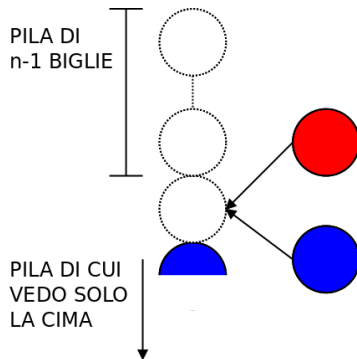
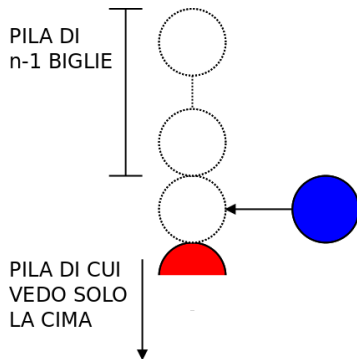


Impostazione della soluzione ricorsiva

- ▶ Dobbiamo calcolare il numero di pile ammissibili con n biglie
- ▶ Riformuliamo il problema in questo modo: quante sono le pile ammissibili alte n che si possono costruire sapendo che l'ultima biglia inserita è di colore blue o rosso?
- ▶ Inizialmente la pila viene costruita su una biglia di colore speciale 'V'



Visione grafica della ricorsione



Codifica della soluzione

```
int biglie(int numero, char cima) {  
    if (numero>0) {  
        if (cima == 'V' || cima == 'B') /*posso inserire blu o rossa*/  
            return biglie(numero-1, 'B') + biglie(numero-1, 'R');  
        else /*cima rossa*/  
            return biglie(numero-1, 'B');  
    }  
    else  
        return 1;  
}
```

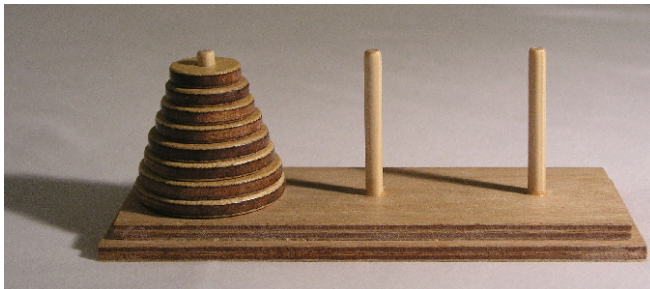


Section 4

La torre di Hanoi

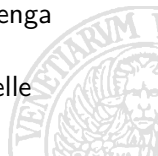


Dischi e steli



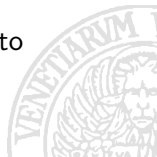
- ▶ Spostare tutti i dischi dal primo al terzo stelo muovendo un disco alla volta ed evitando che un disco più pesante venga mai a trovarsi sopra un disco più leggero
- ▶ Problema: stampare su standard output la sequenza delle mosse che risolvono il problema

Immagine tratta da: http://it.wikipedia.org/wiki/File:Tower_of_Hanoi.jpeg



Impostazione ricorsiva

- ▶ Dobbiamo spostare n dischi dallo stelo i allo stelo j usando lo stelo k come supporto
- ▶ Supponiamo di saper spostare $n - 1$ dischi da uno stelo ad un altro sapendo usare uno stelo come supporto, come facciamo a spostarne n ?
- ▶ Spostiamo $n - 1$ dischi da i a k usando j come supporto
- ▶ Spostiamo l' n -mo disco da i a j
- ▶ Spostiamo $n - 1$ dischi da k a j usando i come supporto



Codifica C

```
void hanoi(int dischi, int da, int a, int sup) {  
    if (dischi > 0) {  
        hanoi(dischi-1, da, sup, a);  
        printf("%d->%d\n", da, a);  
        hanoi(dischi-1, sup, a, da);  
    }  
}
```



Soluzione di 3 dischi

- ▶ $1 \rightarrow 3$
- ▶ $1 \rightarrow 2$
- ▶ $3 \rightarrow 2$
- ▶ $1 \rightarrow 3$
- ▶ $2 \rightarrow 1$
- ▶ $2 \rightarrow 3$
- ▶ $1 \rightarrow 3$



Section 5

Array e ricorsione



Decidere se una porzione di array contiene un particolare valore

- ▶ Delimitiamo la porzione con due indici `inf` e `sup`, estremi compresi
- ▶ Impostazione ricorsiva:
 - ▶ Se l'elemento in posizione `inf` è quello che cerchiamo allora la risposta è `true`
 - ▶ Se la porzione è vuota (`inf > sup`) allora la risposta è `false`
 - ▶ Altrimenti la risposta è decisa dalla presenza dell'elemento nella porzione di array che va da `inf+1` a `sup`
 - ▶ Questa porzione è più piccola di quella considerata precedentemente



Codifica in C con indici

```
int is_present_ric(int vet[], int inf, int sup, int elem) {  
    if (inf > sup)  
        return 0;  
    else  
        if (vet[inf] == elem)  
            return 1;  
        else  
            return is_present_ric(vet, inf+1, sup, elem);  
}
```



Codifica in C con puntatori

```
int is_present_ric(int *inf, int *sup, int elem) {  
    if (inf > sup)  
        return 0;  
    else  
        if (*inf == elem)  
            return 1;  
        else  
            return is_present_ric(inf+1, sup, elem);  
}
```



Esercizi

- ▶ Scrivere una funzione ricorsiva che decida se una porzione di array di interi è ordinata
- ▶ Scrivere una funzione ricorsiva che calcoli la somma di una porzione di array di interi
- ▶ Scrivere una funzione ricorsiva che conti quanti numeri pari ci sono in un array di interi
- ▶ Scrivere una funzione ricorsiva che stabilisca se due array sono uguali. Usare la notazione a puntatori

