

Università degli Studi di Trieste

Corso di ingegneria industriale

Esercitazioni di Fondamenti di Informatica

Giacomo Strangolino

mailto/chat: delleceste@gmail.com

Materiale didattico anche su:

<http://www.giacomos.it>

(<http://www.giacomos.it/didattica/units/2011/>)

Lezione 5 (11/11/2012)

Funzioni ricorsive

- Una funzione può chiamare se stessa. Una funzione si dice ricorsiva se un'istruzione al suo interno chiama la funzione stessa. La ricorsione è un processo con il quale si definisce qualcosa in termini di se stesso (definizione circolare)

Lezione 5 (11/11/2012)

Funzioni ricorsive

Un algoritmo ricorsivo per la risoluzione di un dato problema deve essere definito nel modo seguente:

- si definisce come risolvere dei problemi analoghi a quello di partenza, ma che hanno "dimensione piccola" e possono essere risolti in maniera estremamente semplice (detti casi base);
- si definisce come ottenere la soluzione del problema di partenza combinando la soluzione di uno o più problemi analoghi, ma di "dimensione inferiore".

Lezione 5 (11/11/2012)

Ad esempio, per calcolare il *fattoriale* di un numero intero:

- so calcolare facilmente il fattoriale di **1**;
- calcolo il fattoriale di **n** supponendo di saper calcolare il fattoriale di **n - 1**

Lezione 5

Funzioni ricorsive: calcolo del fattoriale

- Il calcolo del *fattoriale* di un intero n è dato dal prodotto di tutti i numeri tra uno e n
- Quando la funzione ricorsiva per il calcolo del fattoriale è chiamata con argomento pari ad 1, allora essa restituisce 1. Altrimenti, restituisce il valore di $\text{fattoriale}(n - 1) * n$
- Es: $\text{fattoriale}(2)$. La prima chiamata causa una seconda chiamata ricorsiva con argomento 1. Quest'ultima restituisce quindi 1, che poi viene moltiplicato per 2 (il valore originale della chiamata).

Lezione 5

Funzioni ricorsive: calcolo del fattoriale

- Provate a tenere traccia delle chiamate per n che vale 3 e così via...
- Un buon metodo per verificare a che punto della computazione ricorsiva ci si trova è quello di inserire delle *print* nella funzione atte a evidenziare ad esempio il risultato parziale e il livello.

Lezione 5

Funzioni ricorsive

- Riscrivere la funzione fattoriale in modo ricorsivo.
- Scrivere un'implementazione in *python* dell'algoritmo *merge sort*.

Ricorsione: numero triangolare n-mo

- Supponiamo di voler calcolare l'area di una forma triangolare di dimensione n come quella riportata sotto, nell'ipotesi che ciascun quadrato \square abbia area unitaria. Il valore dell'area corrispondente viene a volte chiamato numero triangolare n -esimo.
- Per $n=4$ il triangolo è fatto come segue:
- \square
- $\square\square$
- $\square\square\square$
- $\square\square\square\square$
- Osservando questo schema possiamo affermare che il quarto numero triangolare è 10.

Ricorsione: numero triangolare n-mo

- Se l'ampiezza del triangolo è 1, il triangolo consiste di un unico quadrato □ e la sua area è uguale a 1.
- Per trattare il caso più generale, consideriamo questa figura:

□

□□

□□□

□□□□

- Supponiamo adesso di conoscere l'area del triangolo più piccolo, formato dalle prime tre righe: il terzo numero triangolare. In questo caso possiamo calcolare facilmente l'area del triangolo grande: il quarto numero triangolare:

$$\text{risultato} = n + \text{numero_triangolare_di_}n-1;$$

Ricorsione: numero triangolare n-mo

- Come possiamo calcolare il valore di numero_triangolare_di_n-1?

Facciamolo calcolare a(una diversa istanza de)lla funzione numeroTriangolare() che stiamo per definire:

- **risultato = n + numeroTriangolare(n-1);**

```
def numeroTriangolare(n):
```

```
    if n == 1:
```

```
        return 1 # caso base o clausola di chiusura
```

```
    else:
```

```
        return n + numeroTriangolare(n - 1)
```

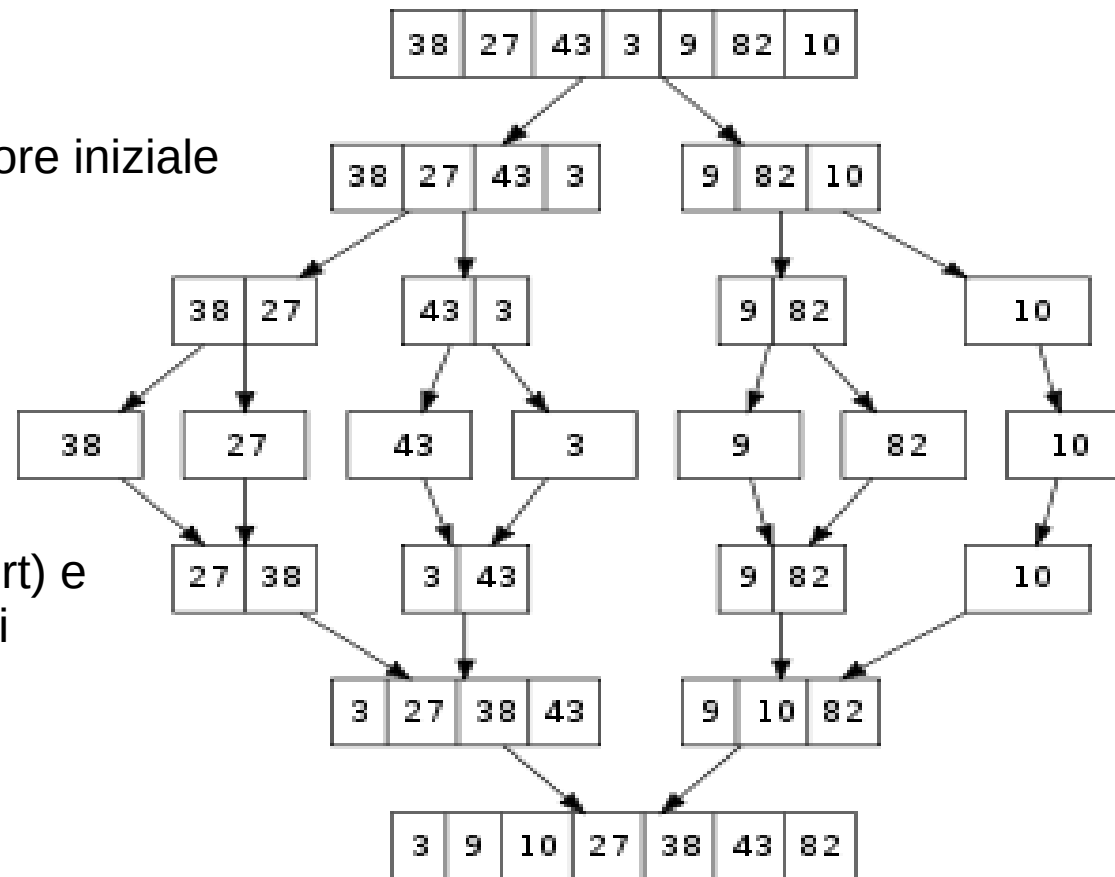
Lezione 5

Merge sort

Divido a metà il vettore iniziale

Metà della metà...

Merge... riordino (sort) e
riassemblo (merge) i
pezzi...



Lezione 5

Merge sort - prestazioni

- Tempo di esecuzione del merge dell'ordine di n
- mergesort richiama se stessa due volte ogni volta su metà della porzione di input



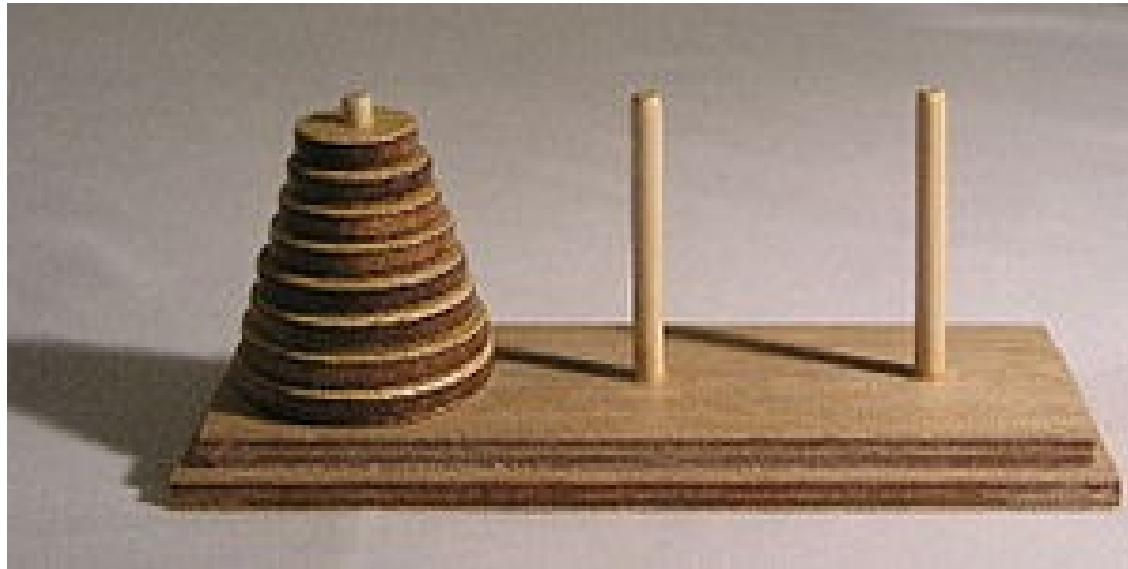
$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$



Caso peggiore $O(n \log n)$

Lezione 5

Torre di Hanoi



Torre di Hanoi

Si scriva un programma per risolvere il rompicapo della Torre di Hanoi: un numero n di dischi, di diametro via via decrescente, sono impilati sul primo dei tre pioli della Torre di Hanoi. L'obiettivo è spostare tutti i dischi sull'ultimo piolo, rispettando le seguenti regole:

- si può spostare un disco solo per volta, su un qualsiasi piolo;
- non si può impilare un disco su uno più piccolo.

Suggerimento: per spostare k dischi dal piolo 1 al piolo 2, bisogna prima spostare i $k-1$ dischi superiori dal piolo 1 al piolo 3.

Questo esercizio può essere risolto in modo “semplice” utilizzando la ricorsione.

Gioco online

http://www.softschools.com/games/logic_games/tower_of_hanoi/

<http://www.math.it/torrih/torri.htm>

Lezione 5 (homework)

Si scriva un programma che legga da tastiera un testo e stampi (in verticale) l'istogramma che rappresenta il numero di volte in cui ciascuna lettera è comparsa nel testo.

- Suggerimento: salvare asterischi in una matrice con numero di righe pari al massimo alla lunghezza stringa e numero di colonne pari al numero possibile di caratteri ASCII standard. Successivamente stampare la matrice all'incontrario (dall'alto verso il basso). Ad esempio ogni volta che trovo una 'p' salvo un '*' alla riga r pari alla somma delle 'p' trovate finora nella colonna `ord('p')`

Lezione 5

Homework (istover)

```
giacomo@woody:~/universita/trieste/python
```

File Modifica Visualizza Cerca Terminale Schede Aiuto

giacomo@woody:~/universita/trieste/python x istover2.py (~/universita/trieste/python) - VIM x

```
pytgiacomo@woody ~/universita/trieste/python $ python3
Python 3.2.2 (default, Sep 21 2011, 10:20:57)
[GCC 4.5.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import istover2
>>> istover2.istover("pippo e pluto Eran0 2 tlp1 lnt3r3554nTi")

                                     *
                                     *
      *                               * *   *
    * * *                             *   *** * *
*****          *                   *   *   *   *   *   *   *
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
>>>
```

Si noti che lo spazio viene trascurato perché ha codice ASCII $32 < \text{ord}('0')$ (v. tabella slide seguente)

Lezione 5

Tabella ASCII:

000	(nul)	016	► (dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	☉ (soh)	017	◄ (dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	☼ (stx)	018	↕ (dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥ (etx)	019	≡ (dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦ (eot)	020	⌘ (dc4)	036	\$	052	4	068	D	084	T	100	d	116	t
005	♣ (enq)	021	§ (nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠ (ack)	022	— (syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	• (bel)	023	↕ (etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	■ (bs)	024	↑ (can)	040	(056	8	072	H	088	X	104	h	120	x
009	(tab)	025	↓ (em)	041)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂ (vt)	027	← (esc)	043	+	059	;	075	K	091	[107	k	123	{
012	♀ (np)	028	L (fs)	044	,	060	<	076	L	092	\	108	l	124	
013	(cr)	029	↔ (gs)	045	-	061	=	077	M	093]	109	m	125	}
014	♪ (so)	030	▲ (rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	✱ (si)	031	▼ (us)	047	/	063	?	079	O	095	_	111	o	127	△

Esercizio tratto da una provetta

Tracciare il diagramma di flusso che rappresenti la seguente pseudo-codifica :

inizio

leggi Numero

assegna a Resto il valore 0

assegna a Divisore il valore 1

|--ripeti

| assegna a Vartemp il valore di Numero

| assegna a Divisore il valore Divisore+1

| fino a quando Vartemp è maggiore di zero ripeti

| assegna a Vartemp il valore (Vartemp meno Divisore)

| fine ciclo

| se Vartemp uguale a zero allora

| assegna a Resto il valore 1

| fine condizione

|--fino a quando Resto è uguale a zero e Divisore è minore di Numero-1

se Resto = 1 allora

scrivi "Numero non è numero primo, è divisibile per Divisore"

altrimenti

scrivi "Numero è primo"

fine condizione

stop

Esercizio tratto da una provetta (II)

Tracciare il diagramma di flusso che rappresenta la seguente pseudo-codifica :

```
inizio
leggi Numero
assegna a Indicatore il valore 0
leggi Alfa
assegna a Indice il valore 1
fino a quando Indice è minore di Numero
    leggi Beta
    se Beta è minore o uguale a Alfa allora
        assegna a Indicatore il valore 1
    fine condizione
    assegna a Alfa il valore di Beta
    Incrementa di 1 in valore di Indice
fine ciclo
se Indicatore è zero allora
    scrivi "Gli elementi sono in ordine crescente"
altrimenti
    scrivi "Gli elementi NON sono in ordine crescente"
fine condizione
stop
```

Lezione 5

Homework

- Si scriva una versione ricorsiva della successione di Fibonacci:

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

- Si riscriva l'implementazione dell'algoritmo per il calcolo del MCD in modo ricorsivo;
- Si riscriva la funzione che implementa l'algoritmo *binary search* in forma ricorsiva.