

Grafici di tempi di esecuzione

Informatica@SEFA 2017/2018 - Laboratorio 6

Massimo Lauria <massimo.lauria@uniroma1.it>
<http://massimolauria.net/courses/infosefa2017/>

Lunedì, 13 Novembre 2017

Grafici dei tempi di esecuzione

Vogliamo fare un plot comparativo del tempo di esecuzione di vari algoritmi.

- ricerca sequenziale vs ricerca binaria
- insertion sort vs bubble sort

Ingredienti necessari

1. implementazioni degli algoritmi
2. dati di test
3. misurare i tempi
4. fare il plot

1. Implementazione degli algoritmi

Datevi da fare!

2. Dati di test: numeriacaso

```
from infosefa import numeriacaso  
help(numeriacaso)
```

1

2

Help on function numeriacaso in module infosefa:

```
numeriacaso(N, minimo, massimo, ordinati=False)  
    Produce una lista di numeri generati a caso.
```

Produce una lista di N elementi, ognuno dei quali preso a caso (con uguale probabilità) tra tutti i numeri interi compresi tra 'minimo' e 'massimo', estremi inclusi.

Se $N < 0$ o $\text{minimo} > \text{massimo}$ la funzione solleva un `ValueError`.

Se 'ordinati' è vero la lista restituita è ordinata.

3. Misurare i tempi di esecuzione

Se volete misurare tempi di esecuzione:

- dati omogenei
- dovete ripetere diverse volte e fare una media
- se i tempi sono piccoli ripetete **molte** volte
- non contate il tempo necessario a generare i dati

3. Esempio - Fibonacci

```
def fib(n):                                1
    if n <= 2:                             2
        return 1                          3
    else:                                  4
        return fib(n-1)+fib(n-2)          5
                                           6

def ifib(n):                              7
    cur,prev=1,1                          8
    if n <= 2:                             9
        return 1                         10
    for i in range(3,n+1):                11
        cur,prev = cur+prev,cur          12
    return cur                            13
```

3. Esempio (II) - Fibonacci

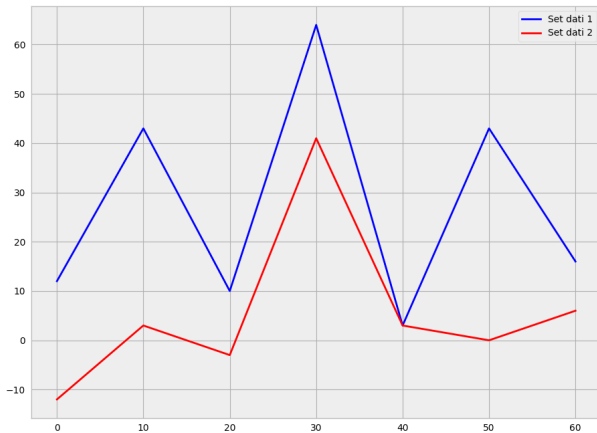
```
from time import process_time      1
from lab06 import fib, ifib        2

                                    3
start = process_time()              4
for i in range(100):                5
    fib(25)                          6
end = process_time()                7
print("Induttivo: {}".format((end-start)/100)) 8
                                    9

start = process_time()             10
for i in range(1000):              11
    ifib(25)                       12
end = process_time()               13
print("Iterativo: {}".format((end-start)/1000)) 14
```

```
Induttivo: 0.022297819999999996
Iterativo: 1.6549999999995179e-06
```


4. Grafici



4. Grafici (codice sorgente)

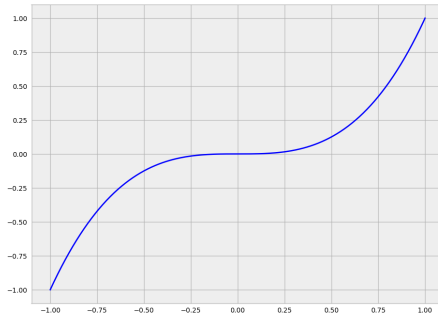
```
from matplotlib.pyplot import plot,savefig,legend      1
                                                        2
x = [0,10,20,30,40,50,60]                             3
dati1 = [12,43,10,64,3,43,16]                         4
dati2 = [-12,3,-3,41,3,0,6]                           5
                                                        6
plot(x,dati1,label='Set dati 1')                      7
plot(x,dati2,label='Set dati 2')                     8
legend(loc='best')                                    9
savefig("assets/lab06_esempio1.png")                 10
```

- ▶ plot inserisce un grafico nella figura
- ▶ potete mettere più grafici nella stessa figura
- ▶ savefig salva la figura in un file a vostra scelta

plot(x,y)

```
from matplotlib.pyplot import plot,savefig  
x = [ i/100 for i in range(-100,101)]  
y = [val**3 for val in x]  
plot(x,y)  
savefig("assets/lab06_esempio2.png")
```

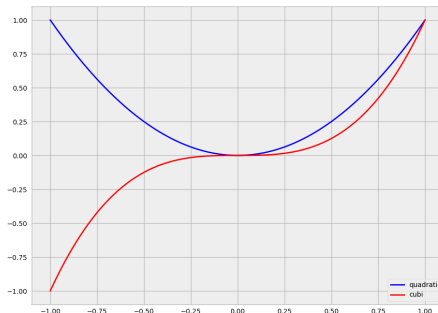
1
2
3
4
5



Plot di più funzioni

```
from matplotlib.pyplot import plot,savefig,legend
x = [ i/100 for i in range(-100,101)]
plot(x,[val**2 for val in x],label='quadrati')
plot(x,[val**3 for val in x],label='cubi')
legend(loc='best')
savefig("assets/lab06_esempio3.png")
```

1
2
3
4
5
6



Un esempio di plot di runtime

