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                                    1
help(numeriacaso)                                                   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 minimo>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

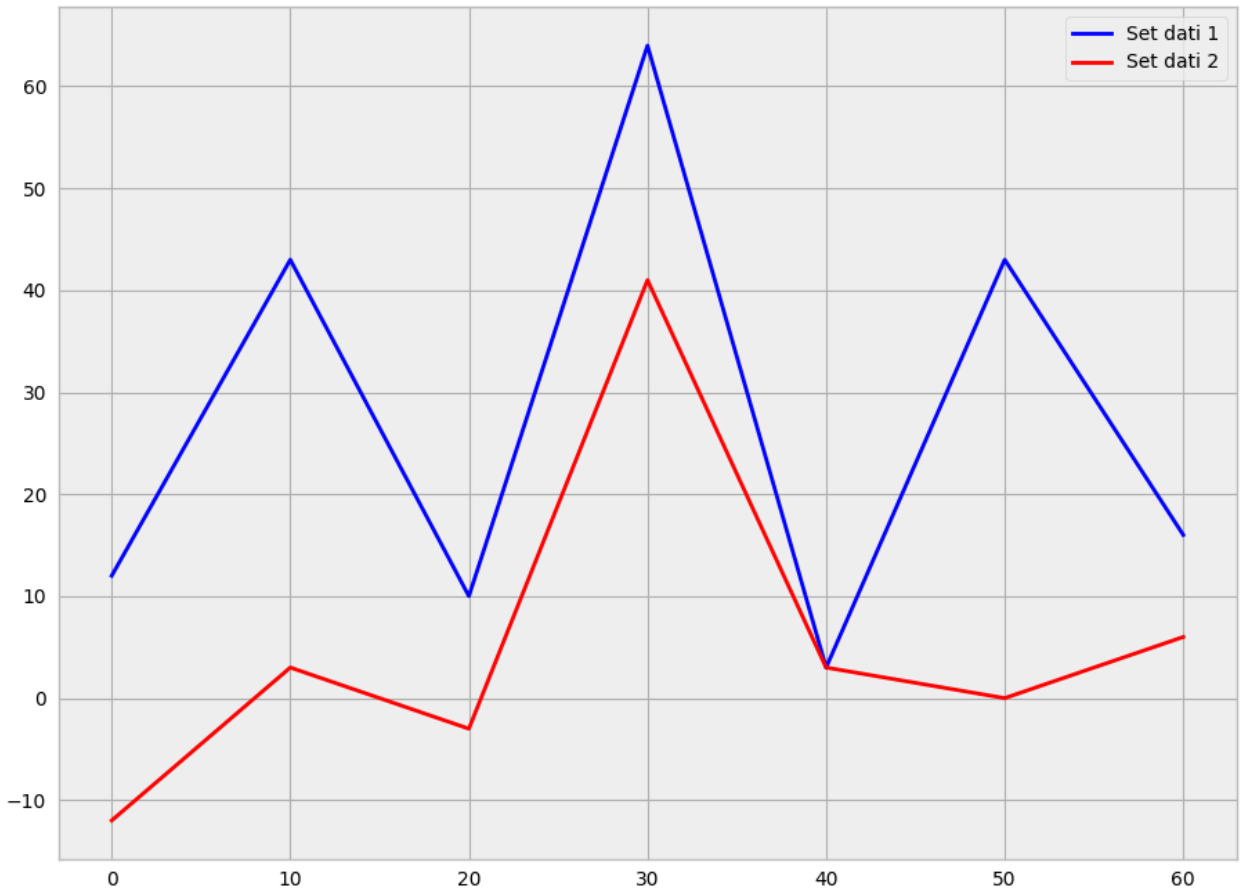
```
from time import process_time
from lab06 import fib,ifib

start = process_time()
for i in range(100):
    fib(25)
end = process_time()
print("Induttivo: {}".format((end-start)/100))

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

```
Induttivo: 0.02335108000000003
Iterativo: 1.6449999999998966e-06
```

4. Grafici



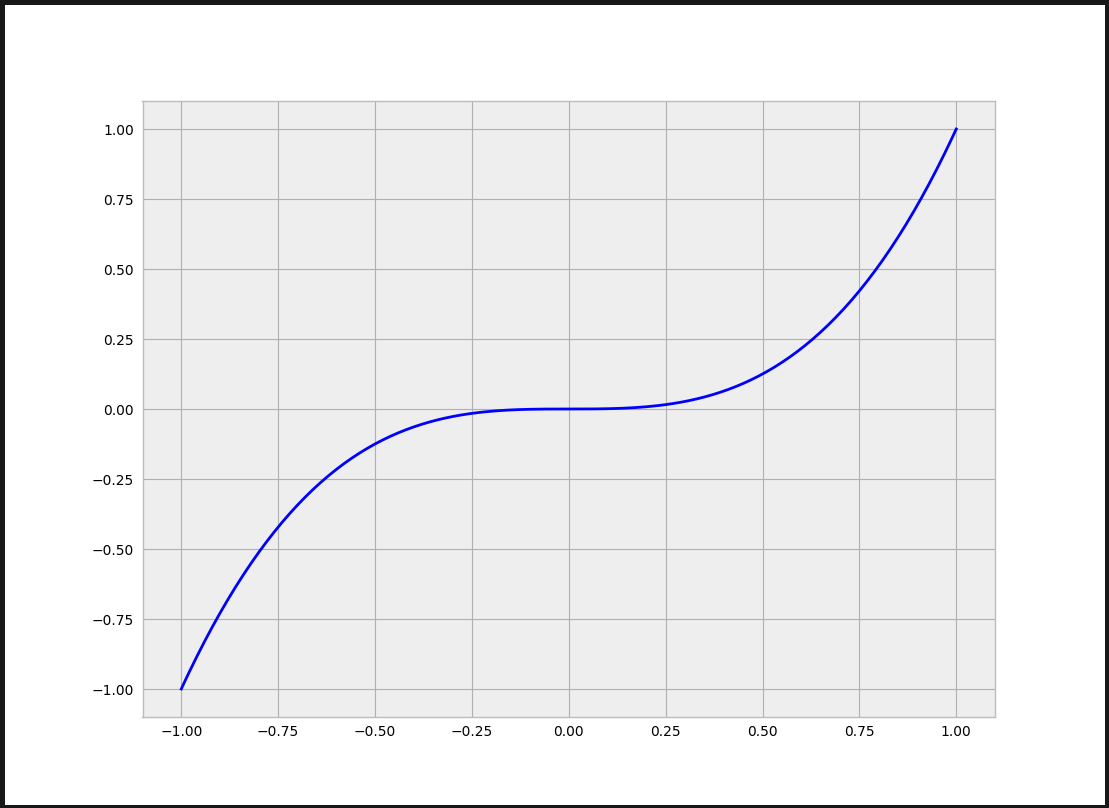
4. Grafici (codice sorgente)

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

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

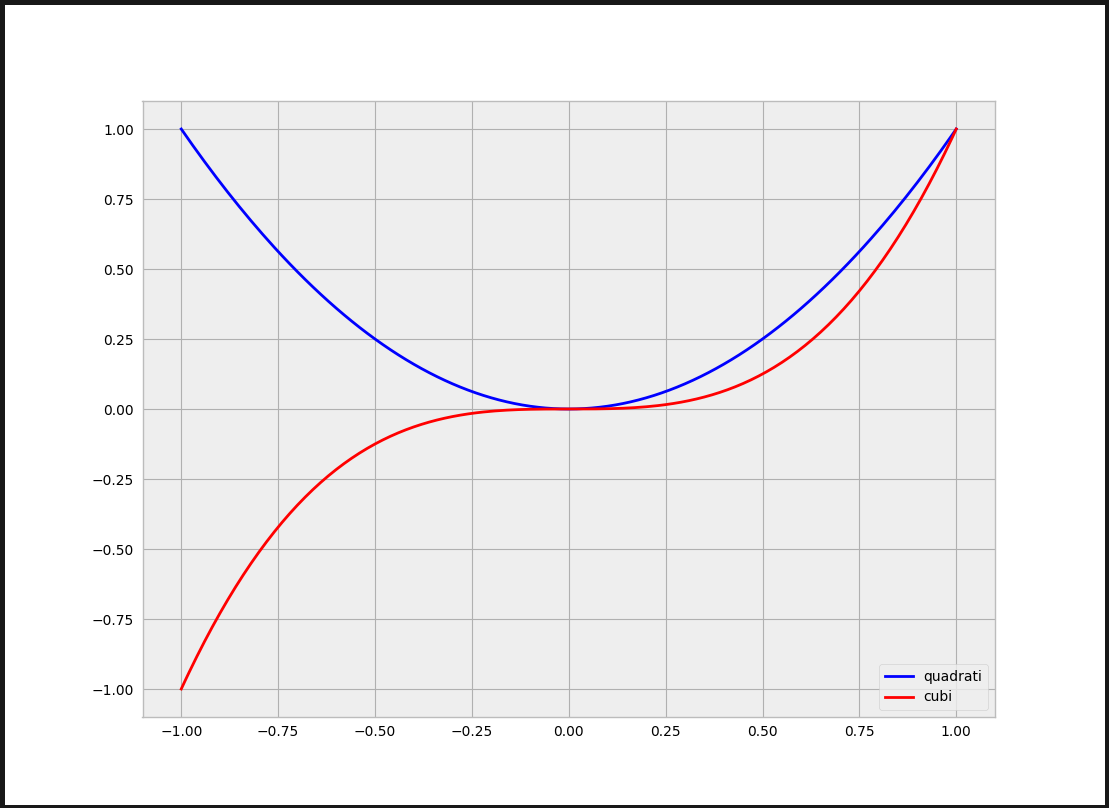
plot(x,y)

```
from matplotlib.pyplot import plot,savefig                    1
x = [ i/100 for i in range(-100,101)]                         2
y = [val**3 for val in x]                                     3
plot(x,y)                                                     4
savefig("assets/lab06_esempio2.png")                          5
```



Plot di più funzioni

```
from matplotlib.pyplot import plot,savefig,legend                    1
x = [ i/100 for i in range(-100,101)]                                2
plot(x,[val**2 for val in x],label='quadrati')                       3
plot(x,[val**3 for val in x],label='cubi')                           4
legend(loc='best')                                                   5
savefig("assets/lab06_esempio3.png")                                 6
```



Un esempio di plot di runtime