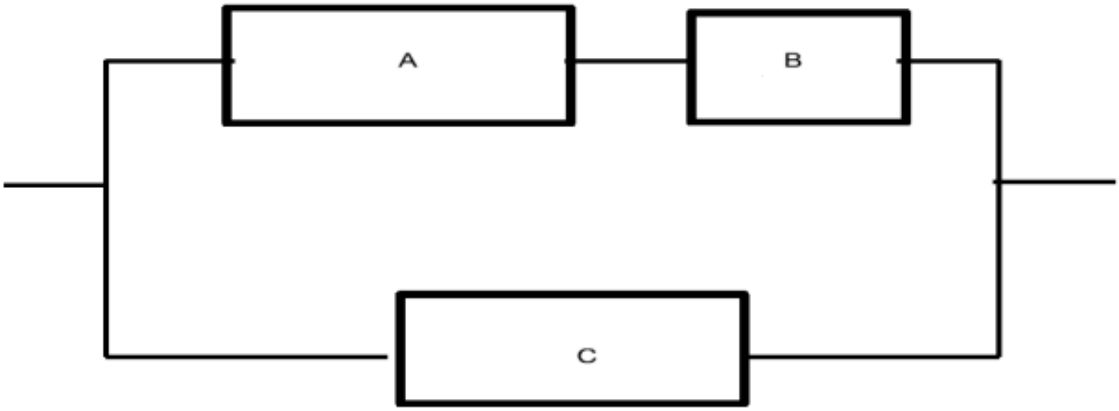
Eserciziario Metodi

1) Elementi di calcolo delle probabilità

Problema 1

Un congegno idraulico è costituito da tre valvole collegate come da figura. Le probabilità che le valvole A, B, C si guastino è rispettivamente pari a 1/3, 1/4, 1/2. Il congegno è funzionante se funziona almeno la coppia di valvole A, B o la valvola C. Si calcoli la probabilità che il congegno si guasti.



In [1]:

```
PA = 1/3
PB = 1/4
PC = 1/2
#P(G) = (P(A) || P(B)) && P(C)
PG = (PA + PB - PA * PB) * PC
print(PG)
```

```
0.24999999999999997
```

Problema 2

Una stazione radio riceve in maniera random segnali da due sorgenti, A e B. La probabilità di ricevere un segnale distorto da A è pari a 0.1 mentre la probabilità di ricevere un segnale distorto da B è pari a 0.2. Qual è la probabilità di ricevere un segnale distorto?

In [2]:

```
SA = 0.5
SB = 0.5
DSA = 0.1
DSB = 0.2
#P(D) = P(D|SA)*P(SA) + P(D|SB)*P(SB)
PD = DSA * SA + DSB * SB
print(PD)
```

```
0.15000000000000002
```

Problema 3

Una industria ha due catene di produzione, A e B, che forniscono rispettivamente il 40% e il 60% dei prodotti. Si scopre che il 25% dei pezzi prodotti dalla catena A sono difettosi contro il 7% dei pezzi prodotti nel centro B. Qual è la probabilità che un pezzo scelto a caso dall'intera produzione sia difettoso? Qual è la probabilità che un

Qual è la probabilità che un pezzo scelto a caso dall'intera produzione sia difettoso? Qual è la probabilità che un pezzo difettoso appartenga alla produzione A?

In [3]:

```
PA = 40/100
PB = 60/100
DA = 25/100
DB = 7/100
#P(D) = P(D|A)*P(A) + P(D/SB)*P(SB)
PD = DA * PA + DB * PB
print(PD)
```

```
0.14200000000000002
```

In [4]:

```
#P(A|D) = P(D|A)*P(A)/P(D)
AD = DA * PA / PD
print(AD)
```

```
0.704225352112676
```

Gioco di Monty-Hall

Un gioco televisivo prevede che un concorrente venga invitato a scegliere una fra tre porte dietro ad una delle quali si trova il premio. Dopo la scelta del concorrente il presentatore apre una delle due porte non scelte e dietro la quale non si trova il premio e invita poi il concorrente a decidere se mantenere la scelta iniziale o cambiare per l'altra porta ancora chiusa. Cambiare porta aumenta la probabilità di vittoria?

In [5]:

```
PA1 = PA2 = PA3 = 1/3
```

Senza ledere la generalità supponiamo che il concorrente abbia scelto la porta 3. A questo punto il conduttore aprirà la porta 1 o la porta 2. Senza ledere la generalità supponiamo che apra la porta 1.

In [6]:

```
PC1 = 1/2
C1A2 = 1
#P(A2|C1) = P(C1|A2)*P(A2)/P(C1)
A2C1 = C1A2 * PA2 / PC1
print(A2C1)
```

```
0.6666666666666666
```

Di conseguenza conviene cambiare.

Paradosso del compleanno

Qual è la probabilità che tra n ≤ 365 persone scelte a caso almeno due festeggino il compleanno lo stesso giorno?

In [7]:

```
import math

#P(A) = 1 - P(A^c) = 1 - D(365, n)/365**n
n = 23
D = (math.factorial(365)) / (math.factorial((365-n)))
PA = 1 - D/((365)**n)
print(PA)
```

```
0.5072972343239854
```

Bastano 23 persone per far in modo che la probabilità che almeno due festeggino il compleanno lo stesso giorno

Bastano 23 persone per far in modo che la probabilità che almeno due festeggino il compleanno lo stesso giorno sia maggiore del 50%.

Esercizio 1

In una rete idrica vi sono quattro valvole. Ciascuna valvola ha probabilità 1/4 di guastarsi in un anno. Quando una valvola si guasta impedisce il flusso di acqua. Le valvole funzionano indipendentemente l'una dall'altra. Due valvole sono collegate in serie e queste due sono collegate a loro volta in parallelo con le altre due. Qual è la probabilità che la rete smetta di fornire acqua?

In [8]:

```
PA = PB = PC = PD = 1/4
#P(G) = (P(A) || P(B)) && P(C) && P(D)
PG = (PA + PB - PA * PB) * PC * PD
print(PG)
```

```
0.02734375
```

Esercizio 2

Per determinare una password di 3 caratteri si hanno a disposizione le lettere A, B, C, D. Provando a caso con tali lettere, qual è la probabilità di indovinare la password in un singolo tentativo? Qual è tale probabilità se le lettere devono essere distinte? Qual è tale probabilità se le lettere devono essere distinte e non conta l'ordine?

In [9]:

```
P1 = 1/(4**3)
P2 = 1/(math.factorial(4))
P3 = 1/(math.factorial(4)/(math.factorial(3)*math.factorial(4-3)))
print("Probabilità di indovinare la password in un singolo tentativo: ",P1)
print("Probabilità di indovinare la password in un singolo tentativo se le lettere devono
essere distinte: ",P2)
print("Probabilità di indovinare la password in un singolo tentativo se le lettere devono
essere distinte e non conta l'ordine: ",P3)
```

```
Probabilità di indovinare la password in un singolo tentativo:  0.015625
Probabilità di indovinare la password in un singolo tentativo se le lettere devono essere
distinte:  0.041666666666666664
Probabilità di indovinare la password in un singolo tentativo se le lettere devono essere
distinte e non conta l'ordine:  0.25
```

Esercizio 3

Per determinare una password di 3 caratteri si hanno a disposizione le lettere A, B, C, D, E. Provando a caso con tali lettere, qual è la probabilità di indovinare la password in un singolo tentativo? Qual è tale probabilità se si sa che una delle lettere che figurano nella password è la A? Qual è tale probabilità se si sa che nella password figurano sia la A che la B?

In [10]:

```
P1 = 1/(5**3)
P2 = 1/(5**3 - 4**3)
P3 = 1/(5**3 - 4**3 - 4**3 + 3**3)
print("Probabilità di indovinare la password in un singolo tentativo: ",P1)
print("Probabilità di indovinare la password in un singolo tentativo sapendo che nella pa
ssword c'è almeno un A: ",P2)
print("Probabilità di indovinare la password in un singolo tentativo sapendo che nella pa
ssword ci sono almeno una A e una B: ",P3)
```

```
Probabilità di indovinare la password in un singolo tentativo:  0.008
Probabilità di indovinare la password in un singolo tentativo sapendo che nella password
c'è almeno un A:  0.01639344262295082
Probabilità di indovinare la password in un singolo tentativo sapendo che nella password
ci sono almeno una A e una B:  0.041666666666666664
```

2) Variabili aleatorie discrete

Problema 1

**Una ditta produce lampadine, di cui il 5% risulta difettoso, e le vende in confezioni da quattro.
1) Qual è la probabilità che in una confezione ci sia una sola lampadina difettosa? 2) Qual è la probabilità che in una confezione ci siano al più due pezzi difettosi? 3) Se ogni scatola contiene 40 pezzi, quanti pezzi difettosi conterebbe in media?**

In [11]:

```
from scipy.stats import binom

p = 5/100
n = 4
P1 = binom.pmf(1, 4, p)
print("Probabilità che in una confezione ci sia una sola lampadina difettosa: ",P1)
```

```
Probabilità che in una confezione ci sia una sola lampadina difettosa:  0.171475000000000
13
```

In [12]:

```
P2 = binom.pmf(0, 4, p) + binom.pmf(1, 4, p) + binom.pmf(2, 4, p)
print("Probabilità che in una confezione ci siano al più due pezzi difettosi: ",P2)
```

```
Probabilità che in una confezione ci siano al più due pezzi difettosi:  0.999518750000000
5
```

In [13]:

```
n = 40
media = n * p
print("Pezzi difettosi contenuti in media in una scatola da 40 pezzi: ",media)
```

```
Pezzi difettosi contenuti in media in una scatola da 40 pezzi:  2.0
```

Problema 2

**La memoria di un computer è composta da 30 hard disk ognuno dei quali contiene 100 file. Un programma dovrà accedere a 28 di questi file (tutti diversi).
1) Qual è la probabilità che non ci siano file provenienti dall'hard disk n. 1? 2) Qual è la probabilità del punto 1 nel caso in cui i file possono ripetersi?**

In [14]:

```
from scipy.special import binom

b = 100
r = (30-1)*100
n = 28

P1 = binom(b, 0)*binom(r, n)/binom(30*b, n)
print("Probabilità che non ci siano file provenienti dall'hard disk n. 1: ", P1)
```

```
Probabilità che non ci siano file provenienti dall'hard disk n. 1:  0.38534529585883975
```

In [15]:

```
from scipy.stats import binom

P2 = binom.pmf(0, n, 1/30)
print("Probabilità che non ci siano file provenienti dall'hard disk n. 1 con ripetizione:
", P2)
```

```
Probabilità che non ci siano file provenienti dall'hard disk n. 1 con ripetizione:  0.387
0337242752073
```

Problema 3

Lancio di un dado non truccato.
1) Qual è la probabilità che esca 6 per la prima volta esattamente al terzo lancio? 2) Sapendo che nei primi 3 lanci non si è avuto alcun 6, qual è la probabilità che esca 6 per la prima volta al quinto tentativo?

In [16]:

```
p = 1/6

P1 = p*((1-p)**2)
print("Probabilità di ottenere un 6 per la prima volta al terzo lancio: ", P1)
```

```
Probabilità di ottenere un 6 per la prima volta al terzo lancio:  0.11574074074074076
```

In [17]:

```
#P(T=5 | T>3) = P(T=2)
P2 = p*((1-p))
print("Probabilità di ottenere un 6 al quinto tentativo sapendo che non si è ottenuto un
6 nei primi 3 lanci: ",P2)
```

```
Probabilità di ottenere un 6 al quinto tentativo sapendo che non si è ottenuto un 6 nei p
rimi 3 lanci:  0.1388888888888889
```

Problema 4

In un libro di 500 pagine sono distribuiti a caso 300 erori di stampa. Qual è la probabilità che una data pagina contenga almeno due errori?

In [18]:

```
from scipy.stats import poisson
n = 300
p = 1/500

P = 1 - poisson.pmf(0, n*p) - poisson.pmf(1, n*p)
print("Probabilità che una data pagina contenga almeno due errori: ", P)
```

```
Probabilità che una data pagina contenga almeno due errori:  0.1219013822495576
```

Problema 5

Calcolare la probabilità che, lanciando un dado non truccato quattro volte, esca tre volte 6 e una volta 2.

In [19]:

```
from scipy.stats import multinomial
import numpy as np
k = np.array([0, 1, 0, 0, 0, 3])
n = 4
p = np.array([1/6, 1/6, 1/6, 1/6, 1/6, 1/6])

P = multinomial.pmf(k, n, p)
print(P)
```

```
0.0030864197530864265
```

Esercizi calcolo delle probabilità

Esercizio 1

Da un'urna contenente 4 palline bianche e 3 nere si eseguono due estrazioni con rimpiazzo (cioè la pallina estratta viene subito rimessa nell'urna)

estratta viene subito rimessa nell'urna).
A) Calcolare la probabilità che le due palline estratte siano del medesimo colore.
B) Calcolare la probabilità che almeno una delle due palline estratte sia nera.

In [20]:

```
from scipy.stats import binom

p = 4/7
n = 2
k = 2
PA = binom.pmf(k, n, p) + binom.pmf(k, n, 1-p)
print("Probabilità che le due palline estratte siano del medesimo colore: ", PA)
```

```
Probabilità che le due palline estratte siano del medesimo colore:  0.5102040816326531
```

In [21]:

```
PB = binom.pmf(1, n, 1-p) + binom.pmf(k, n, 1-p)
print("Probabilità che almeno una delle due palline estratte sia nera: ", PB)
```

```
Probabilità che almeno una delle due palline estratte sia nera:  0.6734693877551021
```

Esercizio 2

I componenti prodotto da una certa ditta possono presentare due tipo di difetti, con percentuali del 3% e del 7% rispettivamente. I due tipo di difettosità si possono produrre in momenti diversi della produzione per cui si può assumere che le presenze dell'uno o dell'altro siano indipendenti tra loro.
A) Qual è la probabilità che un componente presenti entrambi i difetti?
B) Qual è la probabilità che un componente sia difettoso (cioè che presenti almeno uno dei due difetti)?
C) Qual è la probabilità che il componente presenti il difetto 1, sapendo che esso è difettoso?
D) Qual è la probabilità che esso presenti uno solo dei due difetti sapendo che esso è difettoso?

In [22]:

```
pA = 3/100
pB = 7/100

PA = pA * pB
print("Probabilità che un compenente presenti entrambi i difetti: ", PA)
```

```
Probabilità che un compenente presenti entrambi i difetti:  0.0021000000000000003
```

In [23]:

```
PB = pA + pB - PA
print("Probabilità che un componenete sia difettoso: ", PB)
```

```
Probabilità che un componenete sia difettoso:  0.0979
```

In [24]:

```
#P(A|PB) = P(PB|A)*P(A)/P(PB)
PC = 1 * pA / PB
print("Probabilità che il componente presenti il difetto 1 sapendo che è difettoso: ", PC
)
```

```
Probabilità che il componente presenti il difetto 1 sapendo che è difettoso:  0.306435137
89581206
```

In [25]:

```
PE = PB - PA #probabilità che presenti solo uno dei due difetti
#P(E|PB) = P(PB|E)*P(E)/P(PB)
PD = 1 * PE/PB
print("Probabilità che il componente presenti uno solo dei due difetti sapendo che è dife
ttoso: ", PD)
```

```
Probabilità che il componente presenti uno solo dei due difetti sapendo che è difettoso:
0.9785495403472031
```

0.9785495403472931

Esercizio 3

Un dado viene lanciato 3 volte.
A) Qual è la probabilità di ottenere 6 almeno una volta?
B) Quante volte deve essere lanciato il dado perché la probabilità di ottenere 6 almeno una volta sia maggiore o uguale al 90%?

In [26]:

```
p = 1/6
n = 3
PA = binom.pmf(1, n, p) + binom.pmf(2, n, p) + binom.pmf(n, n, p)
print("Proabilità di ottenere un 6 almeno una volta in 3 lanci: ", PA)
```

```
Proabilità di ottenere un 6 almeno una volta in 3 lanci:  0.4212962962962964
```

In [27]:

```
import numpy as np
#Calcolo la probabilità di non ottenere un 6 in nessuno degli n lanci e la pongo <= 0.1
#(1-p)**n <= 0.1   passo ai logaritmi
#ln((1-p)**n) <= ln(1/10)
#n <= ln(1/10)/ln(5/6)
n = np.round((np.log(1/10))/(np.log(5/6)))
print("Numero di lanci necessari per far si che la probabilità di ottenere 6 almeno una v
olta sia >=90%: ", n)
```

```
Numero di lanci necessari per far si che la probabilità di ottenere 6 almeno una volta si
a >=90%:  13.0
```

Esercizio 4

Un giocatore di poker riceve all'inizio del gioco cinque carte da un normale mazzo di 52.
A) Qual è la probabilità che riceva almeno 2 assi?
B) Qual è la probabilità che riceva cinque carte dello stesso seme?
C) Qual è la probabilità che riceva un poker servito?

In [28]:

```
from scipy.stats import hypergeom

k = 2
b = 4
r = 48
n = 5

PA = 0
for i in range(4):
    PA += hypergeom.pmf(k+i, b+r, b, n)
print("Probabilità di ricevere almeno 2 assi: ", PA)
```

```
Probabilità di ricevere almeno 2 assi:  0.04168436605411396
```

In [29]:

```
k = 5
b = 13
r = 39
n = 5
PB = 4 * hypergeom.pmf(k, b+r, b, n)
print("Probabilità di ricevere cinque carte dello stesso seme: ", PB)
```

```
Probabilità di ricevere cinque carte dello stesso seme:  0.0019807923169267707
```

In [30]:

```
k = 4
b = 4
r = 48
n = 5
PC = 13*hypergeom.pmf(k, b+r, b, n)
print("Probabilità di ricevere un poker servito: ", PC)
```

```
Probabilità di ricevere un poker servito:  0.00024009603841536613
```

Esercizio 5

Si stima che il 30% degli adulti negli Stati Uniti siano obesi, che il 3% siano diabetici e che il 2% siano sia obesi che diabetici. Determina la probabilità che un individuo scelto casualmente
A) sia diabetico se è obeso;
B) sia obeso se è diabetico.

In [31]:

```
pObeso = 30/100
pDiabetico = 3/100
pObesoDiabetico = 2/100

#P(Diabetico/Obeso)=P(Obeso && Diabetico)/P(Obeso)
PA = pObesoDiabetico/pObeso
print("Probabilità che un individuo sia diabetico sapendo che è obeso: ", PA)
```

```
Probabilità che un individuo sia diabetico sapendo che è obeso:  0.06666666666666667
```

In [32]:

```
#P(Obeso/Diabetico)=P(Obeso && Diabetico)/P(Diabetico)
PB = pObesoDiabetico/pDiabetico
print("Probabilità che un individuo sia obeso sapendo che è diabetico: ", PB)
```

```
Probabilità che un individuo sia obeso sapendo che è diabetico:  0.6666666666666667
```

Esercizio 6

Su un tavolo ci sono 2 monete. Quando vengono lanciate, una moneta dà testa con probabilità 0.5 mentre l'altra dà testa con probabilità 0.6. Una moneta viene scelta a caso e lanciata.
A) Qual è la probabilità che esca testa?
B) Se esce croce, qual è la probabilità che fosse la moneta equilibrata?

In [33]:

```
p1 = 0.5 #probabilità che esca testa della moneta 1
p2 = 0.6 #probabilità che esca testa della moneta 2
pm = 0.5 #probabilità di scegliere casualmente una delle due monete

PA = pm*p1 + pm*p2 #teorema delle probabilità totali
print("Probabilità che esca testa: ", PA)
```

```
Probabilità che esca testa:  0.55
```

In [34]:

```
#P(M1|Croce) = P(Croce|M1)*P(M1)/P(Croce)
PB = (1-p1)*pm/(1-PA)
print("Probabilità che la moneta fosse equilibrata sapendo che è uscito croce: ", PB)
```

```
Probabilità che la moneta fosse equilibrata sapendo che è uscito croce:  0.555555555555556
```

Esercizi varibaili aleatorie discrete

Esercizio 1

Due centralini, tra di loro indipendenti, ricevono nell'unità di tempo un numero di telefonate X e Y aventi legge di Poisson di parametri rispettivamente λ e μ.
A) Qual è la probabilità che nell'unità di tempo i due centralini ricevano insieme non più di tre telefonate, supponendo λ = 2 e μ = 4?
B) Calcolare la legge condizionale di X dato X+Y=n. Si tratta di una densità nota? Quanto vale la media di questa legge condizionale?
C) Supponendo λ = 2 e μ = 4 e sapendo che nell'unità di tempo i due centralini hanno ricevuto complessivamente 8 telefonate, qual è la probabilità che il primo ne abbia ricevute 3?

In [35]:

```
from scipy.stats import poisson
#X segue una Poisson con parametro 2
#Y segue una Poisson con parametro 4
#allora segue che X+Y segue una Poisson con parametro 2+4
lam = 2
mu = 4
PA = 0
for i in range(4):
    PA += poisson.pmf(i, lam+mu)
print("Probabilità che nell'unità di tempo i due centralini ricevano non più di tre telef
onate: ", PA)
```

```
Probabilità che nell'unità di tempo i due centralini ricevano non più di tre telefonate:
0.15120388277664787
```

In [36]:

```
#P(X=k | X+Y=n) = P(X=k, X+Y=n)/P(X+Y=n) = P(X=k, Y=n-k)/P(X+Y=n)

#X e Y sono indipendenti, dunque possiamo scriverla:
#P(X=k) * P(Y=n-k) / P(X+Y=n)

#sostituiamo con la formula di Poisson e raccogliamo per ottenere una distribuzione binom
iale. Alla fine avremo:
#z = (X=k | X+Y=n) segue una distribuzione binomiale B(n, lam/(lam+mu))

#La media di questa ditribuzione equivale a:
#media = n * lam/(lam+mu)
```

In [37]:

```
from scipy.stats import binom

n = 8
t1 = 3
PC = binom.pmf(3, 8, lam/(lam+mu))
print("Probabilità che il primo centralino abbia ricevuto tre chiamate, sapendo che i due
centralini hanno ricevuto complessivamente 8 chiamate: ", PC)
```

```
Probabilità che il primo centralino abbia ricevuto tre chiamate, sapendo che i due centra
lini hanno ricevuto complessivamente 8 chiamate:  0.2731290961743638
```

Esercizio 2

Da una rilevazione risulta che il numero di incidenti stradali che avvengono ad un determinato incrocio in un mese segue una distribuzione di Poisson con valor medio 1.5.
A) Qual è la probabilità che in un mese non ci siano incidenti?
B) Qual è la probabilità che in un mese ci siano più di due incidenti?

In [38]:

```
from scipy.stats import poisson

lam = 1.5
PA = poisson.pmf(0, lam)
print("Probabilità che in un mese non ci siano incidenti: ", PA)
```

```
Probabilità che in un mese non ci siano incidenti:  0.22313016014842982
```

In [39]:

```
P = 0
for i in range(3):
    P += poisson.pmf(i, lam)
PB = 1 - P
print("Probabilità che in un mese ci siano più di due incidenti: ", PB)
```

```
Probabilità che in un mese ci siano più di due incidenti:  0.19115316946194194
```

Esercizio 3

La probabilità di contrarre una malattia rara è dello 0.03%. Qual è la probabilità che in una città dove vivono 20000 persone vi siano meno di 4 persone che contraggono la malattia? Costruire il grafico della densità e della funzione di ripartizione della distribuzione in esame.

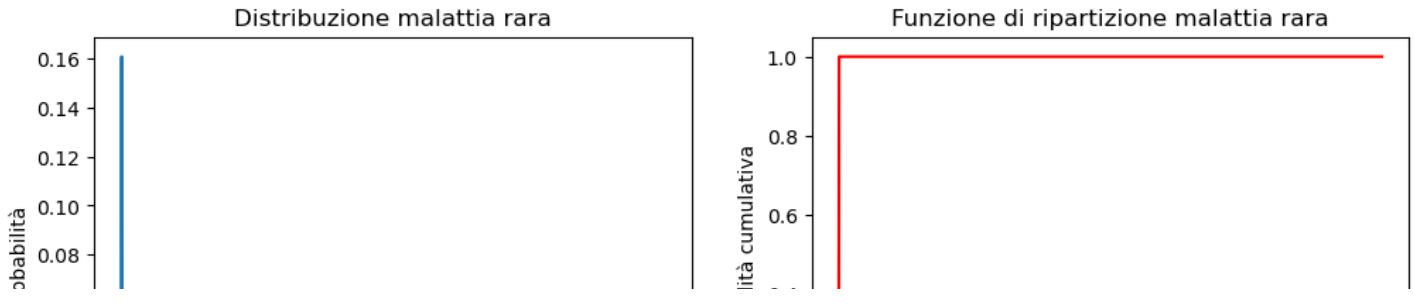
In [40]:

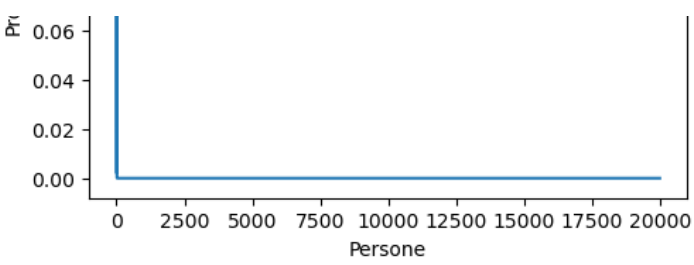
```
from scipy.stats import poisson

p = 3/10000
n = 20000
lam = n*p
PA = 0
for i in range(4):
    PA += poisson.pmf(i, lam)
print("Probabilità che vi siano meno di 4 persone che contraggono la malattia: ", PA)
```

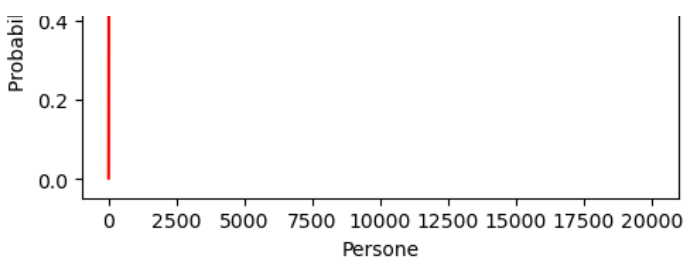
```
Probabilità che vi siano meno di 4 persone che contraggono la malattia:  0.15120388277664
804
```

In [41]:

```
import matplotlib.pyplot as plt

x = range(0, n+1)
y_pmf = poisson.pmf(x, lam)
y_cdf = poisson.cdf(x, lam)

plt.figure(figsize=(12, 4))

# Grafico della densità
plt.subplot(121)
plt.plot(x, y_pmf)
plt.title("Distribuzione malattia rara")
plt.xlabel("Persone")
plt.ylabel("Probabilità")

# Grafico della funzione di ripartizione
plt.subplot(122)
plt.plot(x, y_cdf, 'r')
plt.title("Funzione di ripartizione malattia rara")
plt.xlabel("Persone")
plt.ylabel("Probabilità cumulativa")

#plt.tight_layout()  # Ottimizza la disposizione dei subplot
plt.show()
```







Esercizio 4

Si lancia un dado equilibrato finché non esca un numero dispari.
A) Qual è la probabilità che ciò avvenga al quarto tentativo?
B) Quanti tentativi sono necessari affinché si abbia una probabilità maggiore del 95% che esca un numero dispari esattamente al tentativo successivo?
C) Sapendo che nei primi 4 lanci non si è avuto un numero dispari, qual è la probabilità che si abbia un numero dispari per la prima volta al settimo tentativo?

In [42]:

```
from scipy.stats import geom

t = 4
p = 1/2
PA = geom.pmf(t, p)
print("Probabilità che si ottenga un numero dispari al quarto tentativo: ", PA)
```

```
Probabilità che si ottenga un numero dispari al quarto tentativo:  0.0625
```

In [43]:

```
p_t = 95/100
n = 1
while True:
    pn = geom.pmf(n, p)
    if 1-pn > p_t:
        break
    n+=1
print("Tentativi necessari affinché si abbia una probabilità maggiore del 95% che esca un
numero dispari esattamente al tentativo successivo: ", n)
```

```
Tentativi necessari affinché si abbia una probabilità maggiore del 95% che esca un numero
dispari esattamente al tentativo successivo:  5
```

In [44]:

```
#P(T=7 | T>4) = P(T=3)
PC = geom.pmf(3, p)
print("Probabilità che si abbia un numero dispari per la prima volta al settimo lancio sa
pendo che nei primi 4 si è avuto un numero pari: ", PC)
```

```
Probabilità che si abbia un numero dispari per la prima volta al settimo lancio sapendo c
he nei primi 4 si è avuto un numero pari:  0.125
```

Esercizio 5

Si supponga che tre negozi della stessa tipologia attraggano rispettivamente il 20% della clientela, il 45% e il 35%. Scegliendo a caso 6 clienti, qual è la probabilità che 2 vadano nel primo negozio, 1 nel secondo e 3 nel terzo? Qual è la probabilità che nessun cliente vada nel primo negozio?

In [45]:

```
from scipy.stats import multinomial
import numpy as np

p = np.array([20/100, 45/100, 35/100])
n = 6
```

```
k = np.array([2, 1, 3])
PA = multinomial.pmf(k, n, p)
print("Probabilità che 2 clienti vadano nel primo negozio, 1 nel secondo e 3 nel terzo: "
, PA)
```

```
Probabilità che 2 clienti vadano nel primo negozio, 1 nel secondo e 3 nel terzo:  0.04630
500000000001
```

In [46]:

```
PB = 0
for i in range(7):
    k = np.array([0, i, 6-i])
    PB += multinomial.pmf(k, n, p)
print("Probabilità che nessun cliente vada nel primo negozio: ", PB)
```

```
Probabilità che nessun cliente vada nel primo negozio:  0.26214400000000004
```

3) Variabili aleatorie continue

Problema 1

Il numero di pezzi guasti di una fornitura segue una legge B(2, p) ove p è una variabile aleatoria di legge U([0.5, 1]). Qual è la probabilità che un solo pezzo sia guasto?

In [47]:

```
#X è la v. a. che conta i pezzi guasti
#Y è la v. a. che assume il valore di p

#B(2, p) è la legge della densità condizionale P(X/Y)
#P(X=x, Y=y) = P(X|y)*P(Y=y)

#dove P(X=x, Y=y) = f(x,y)
#P(Y=y) = f(y)

#Sappiamo che:
#f(y) = 2 se x € [0.5, 1]
#f(y) = 0 altrimenti

#P(X|Y) = binom.pmf(x, 2, y)

#f(x,y) = 2*binom.pmf(x, 2, y) se x=0,1,2 && y € [0.5, 1]
#f(x,y) = 0 altrimenti

#f(x) = 2*binom(2, x)*quad(y**x(1-y)**(2*x), 0.5, 1) se x=0,1,2
#f(x) = 0 altrimenti

#la risposta si ottiene per x=1
from scipy.special import binom
from scipy.integrate import quad

y = lambda x: x * (1 - x)
P = 2 * int(binom(2, 1)) * quad(y, 0.5, 1)[0]
print("Probabilità che un solo pezzo sia guasto: ", P)
```

```
Probabilità che un solo pezzo sia guasto:  0.3333333333333333
```

Problema 2

Un lago riceve acqua da due immissari e alimenta un emissario. Misurando la portata in base alla variazione di quota dell'acqua, i due immissari immettono con legge X1 ~ N(1,1) e X2 ~ N(2,2) mentre l'emissario viene alimentato con legge X3 ~ N(3/2,3). Si determini la legge seguita dall'altezza dell'acqua. Qual è la probabilità che la quota superi il livello di guardia pari a 2? Qual è la probabilità che la quota sia inferiore a 0.5?

In [48]:

```
#La legge seguita dall'altezza dell'acqua è:
#Y = X1 + X2 - X3 con Y che segue una legge normale
#Calcoliamo media e varianza

#E[Y] = E[X1 + X2 - X3] = E[X1] + E[X2] - E[X3]
media1 = 1
media2 = 2
media3 = 3/2

media = media1 + media2 - media3
print("Media: ", media)

#VAR(Y) = VAR(X1 + X2 - X3) = VAR(X1) + VAR(X2) + VAR(X3)
var1 = 1
var2 = 2
var3 = 3
var = var1 + var2 + var3
print("Varianza: ", var)
print(f'Y ~ N({media}, {var})')
```

```
Media:  1.5
Varianza:  6
Y ~ N(1.5, 6)
```

In [49]:

```
from scipy.stats import norm
import numpy as np
P = 1 - norm.cdf(2, 1.5, np.sqrt(6))
print("Probabilità che la quota superi il livello di guardia pari a 2: ", P)
```

```
Probabilità che la quota superi il livello di guardia pari a 2:  0.4191282431929131
```

In [50]:

```
P = norm.cdf(0.5, 1.5, np.sqrt(6))
print("Probabilità che la quota sia inferiore a 0.5: ", P)
```

```
Probabilità che la quota sia inferiore a 0.5:  0.34154569915480437
```

Problema 3

Siano A e B due elementi collegati in serie. Dette Ta e Tb le v. a. descriventi la durata di singoli elementi, si determini la legge dell'intero dispositivo supponendo Ta e Tb di legge esponenziale ed indipendenti tra loro.

In [51]:

```
#Ta ~ Exp(lam_a) con lam_a>0
#Tb ~ Exp(lam_b) con lam_b>0

#L'intero dispositivo si guasta non appena si guasta un elemento

#Se r<=0 allora FT=0
#Se r>0 allora
#FT(r) = P(T<=r) = P(min(Ta, Tb)<=r)
# = 1 - P(min(Ta, Tb)>=r) = 1 - P(Ta>=r, Tb>=r)
# = 1 - P(Ta>=r)*P(Tb>=r) = 1 - Sa(r)*Sb(r)
# = 1 - e**(-(lam_a + lam_b)*r)
```

Problema 4

Siano A e B due elementi collegati in parallelo. Dette Ta e Tb le v. a. descriventi la durata dei singoli elementi, si determini la legge dell'intero dispostivio supponendo Ta e Tb di legge esponenziale ed indipendenti tra loro.

In [52]:

```
#Ta ~ Exp(lam_a) con lam_a>0
#Tb ~ Exp(lam_b) con lam_b>0
```

```
#L'intero dispositivo si guasta quando si guastano entrambi

#Se r<=0 allora FT=0
#Se r>0 allora
#FT(r) = P(T<=r) = P(max(Ta, Tb)<=r)
#P(Ta<=r, Tb<=r) = P(Ta<=r)*P(Tb<=r)
#Fa(r)*Fb(r) = (1-e**(-lam_a*r))*(1-e**(-lam_b*r))
```

Esercizi variabili aleatorie continue

Esercizio 1

Il numero di anni di funzionamento di una radio ha una distribuzione esponenziale di parametro λ=1/8. Qual è la probabilità che una radio funzioni per più di dieci anni? Costruire il grafico della densità e della funzione di ripartizione della distribuzione in esame.

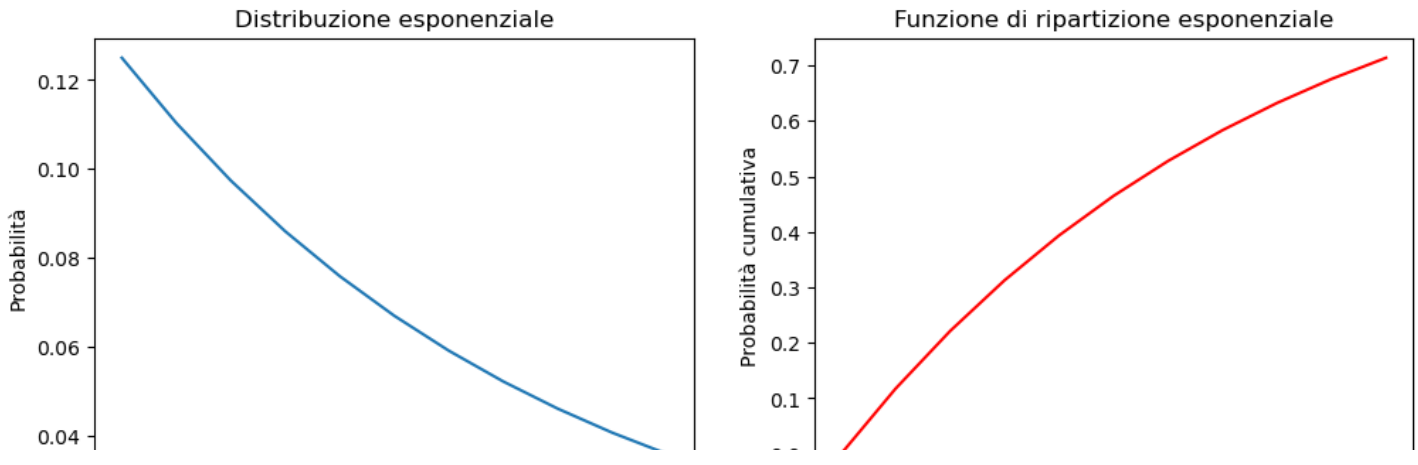
In [53]:

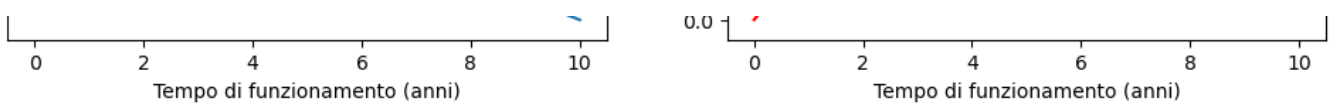
```
from scipy.stats import expon

lam = 1/8
t = 10
P = expon.sf(t, scale=1/lam)
print("Probabilità che una radio funzioni per più di dieci anni: ", P)
```

```
Probabilità che una radio funzioni per più di dieci anni:  0.2865047968601901
```

In [54]:

```
import matplotlib.pyplot as plt

x = range(0, t+1)
y_pdf = expon.pdf(x, scale=1/lam)
y_cdf = expon.cdf(x, scale=1/lam)

plt.figure(figsize=(12, 4))

# Grafico della distribuzione esponenziale
plt.subplot(121)
plt.plot(x, y_pdf)
plt.title('Distribuzione esponenziale')
plt.xlabel('Tempo di funzionamento (anni)')
plt.ylabel('Probabilità')

# Grafico della funzione di ripartizione esponenziale
plt.subplot(122)
plt.plot(x, y_cdf, 'r')
plt.title('Funzione di ripartizione esponenziale')
plt.xlabel('Tempo di funzionamento (anni)')
plt.ylabel('Probabilità cumulativa')

plt.show()
```





Esercizio 2

Il tempo (in ore) necessario per riparare un macchinario è una v.a. esponenziale di parametro λ=1. Qual è la probabilità che la riparazione superi le due ore di tempo? Qual è la probabilità che la riparazione richieda almeno tre ore, sapendo che ne richiede più di due?

In [55]:

```
from scipy.stats import expon

lam = 1
t = 2
P = expon.sf(t, scale=1/lam)
print("Probabilità che la riparazione superi le due ore di tempo: ", P)
```

```
Probabilità che la riparazione superi le due ore di tempo:  0.1353352832366127
```

In [56]:

```
tm = 3 #tempo necessario stimato
tn = 2 #tempo necessario minimo
P = expon.sf(tm-tn, scale=1/lam)
print("Probabilità che la riparazione superi le tre ore di tempo sapendo che ne richiede
almeno due: ", P)
```

```
Probabilità che la riparazione superi le tre ore di tempo sapendo che ne richiede almeno
due:  0.36787944117144233
```

Esercizio 3

Si suppone che l'altezza degli uomini in Italia segua approssimativamente una v.a. normale di media 175 cm e deviazione standard 9 cm. Quale sarebbe la percentuale di italiani di statura superiore al metro e 90? Alla visita di leva vengono scartate le reclute di altezza inferiore ai 153 cm. Quale sarebbe la percentuale di reclute scartate alla visita di leva? Costruire il grafico della densità e della funzione di ripartizione della distribuzione in esame

In [57]:

```
from scipy.stats import norm

media = 175
std = 9
h_1 = 190
P = 1 - norm.cdf(h_1, media, std)
print("Percentuale di italiani di statura superiore al metro e 90: ", round(P*100, 2), "%
")
```

```
Percentuale di italiani di statura superiore al metro e 90:  4.78 %
```

In [58]:

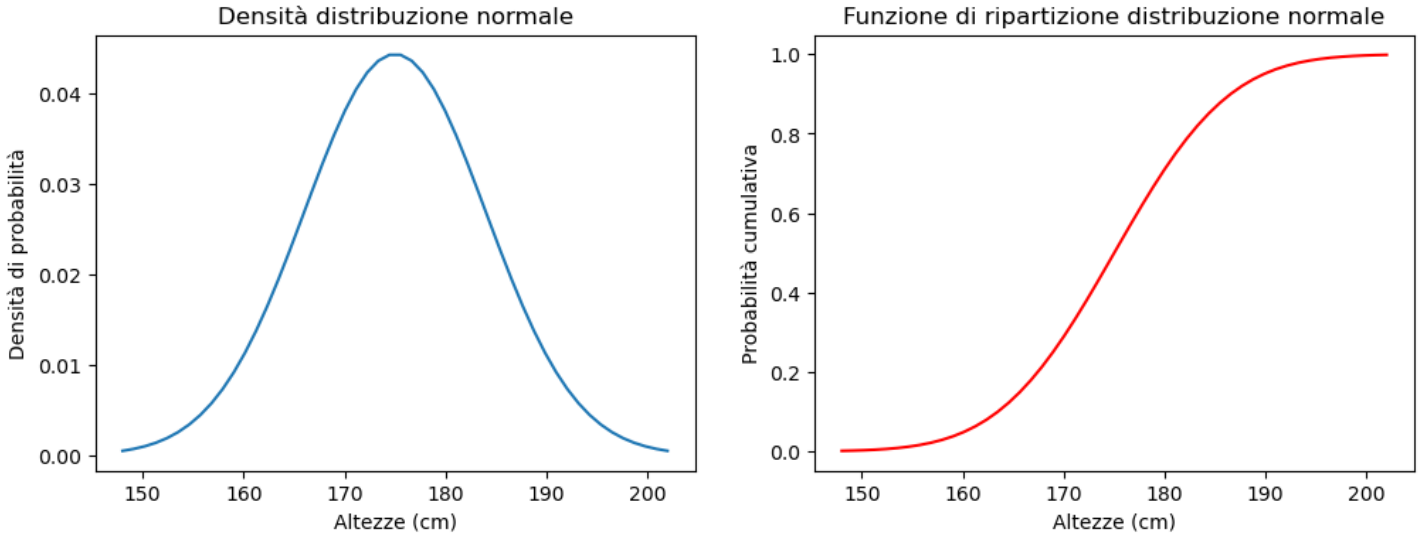
```
h_2 = 153
P = norm.cdf(h_2, media, std)
print("Percentuale di reclute scartate alla visita di leva: ", round(P*100, 2), "%")
```

```
Percentuale di reclute scartate alla visita di leva:  0.73 %
```

In [59]:

```
import matplotlib.pyplot as plt

x = np.linspace(media - 3 * std, media + 3 * std) #intervallo di copertura della normale
y_pdf = norm.pdf(x, media, std)
y_cdf = norm.cdf(x, media, std)
```

```
plt.figure(figsize=(12, 4))

# Grafico della distribuzione esponenziale
plt.subplot(121)
plt.plot(x, y_pdf)
plt.title('Densità distribuzione normale')
plt.xlabel('Altezze (cm)')
plt.ylabel('Densità di probabilità')

# Grafico della funzione di ripartizione esponenziale
plt.subplot(122)
plt.plot(x, y_cdf, 'r')
plt.title('Funzione di ripartizione distribuzione normale')
plt.xlabel('Altezze (cm)')
plt.ylabel('Probabilità cumulativa')

plt.show()
```



Esercizio 4

Si scelga a caso un punto X all'interno dell'intervallo [0,2]. Qual è la probabilità che il triangolo equilatero il cui lato ha lunghezza X abbia area maggiore di 1?

In [60]:

```
from scipy.stats import uniform

a = 0
b = 2
#A = x**2 * np.sqrt(3) / 4
#da qui ricavo il valore di x con A = 1
x = np.sqrt(4/(np.sqrt(3)))
P = 1 - uniform.cdf(x, loc=a, scale=b-a)
print("Probabilità che il triangolo equilatero abbia area maggiore di 1: ",P)
```

```
Probabilità che il triangolo equilatero abbia area maggiore di 1:  0.24016431434840735
```

Esercizio 5

Ad un esame universitario, il voto medio è stato 24 e la deviazione standard 4. Supponendo i voti distribuiti normalmente, calcolare:
A) la probabilità che uno studente abbia riportato un voto superiore a 27
B) la probabilità che uno studente abbia riportato un voto inferiore a 22
C) la probabilità che uno studente abbia riportato un voto compreso tra 23 e 25
D) il voto minimo riportato dal 70% degli studenti
E) il voto massimo non superato dal 90% degli studenti

In [61]:

```
from scipy.stats import norm

mu = 24
sig = 4
voto_1 = 27
PA = 1 - norm.cdf(voto_1, mu, sig)
print("Probabilità che uno studente abbia riportato un voto superiore a 27: ", PA)
```

```
Probabilità che uno studente abbia riportato un voto superiore a 27:  0.22662735237686826
```

In [62]:

```
voto_2 = 22
PB = norm.cdf(voto_2, mu, sig)
print("Probabilità che uno studente abbia riportato un voto inferiore a 22: ", PB)
```

```
Probabilità che uno studente abbia riportato un voto inferiore a 22:  0.3085375387259869
```

In [63]:

```
voto_3 = 23
voto_4 = 25
PC = norm.cdf(voto_4, mu, sig) - norm.cdf(voto_3, mu, sig)
print("Probabilità che uno studente abbia riportato un voto compreso tra 23 e 25: ", PC)
```

```
Probabilità che uno studente abbia riportato un voto compreso tra 23 e 25:  0.19741265136
58474
```

In [64]:

```
n = norm.ppf(1-0.7, mu, sig)
print("Voto minimo riportato dal 70% degli studenti: ", round(n))
```

```
Voto minimo riportato dal 70% degli studenti:  22
```

In [65]:

```
n = norm.ppf(0.9, mu, sig)
print("Voto massimo non superato dal 90% degli studenti: ", round(n))
```

```
Voto massimo non superato dal 90% degli studenti:  29
```

4) Legge dei grandi numeri e approssimazione normale

Problema 1

Un numero al lotto non esce da 324 estrazioni. Questo giustifica l'asserzione che tale ritardo dovrebbe comportare che il numero esca nelle imminenti estrazioni altrimenti si avrebbe una violazione della legge dei grandi numeri?

In [66]:

```
#Per la legge dei grandi numeri le frequenze si devono stabilizzare.
#324 estrazioni creano un forte squilibrio?

#Calcoliamo la probabilità di estrazione.
#Il numero di cinquine "buone" segue un'ipergeometrica
from scipy.stats import hypergeom

k = 1
b = 1
r = 89
n = 5
P = hypergeom.pmf(k, b+r, b, n)
print("Probabilità che venga estratto un dato numero: ", P)
```

```
Probabilità che venga estratto un dato numero:  0.05555555555555555
```

In [67]:

In [67]:

```
#Se consideriamo successo l'estrazione del nostro numero
#allora X_bar = 0

#verifichiamo che lim per n che tende a infinito di |X_bar - p| sia zero

#|X_bar - p| = |0 - 1/18| = 1/18 = 1/np.sqrt(324) = 1/np.sqrt(n)
#lim per n che tende a infinito di 1/np.sqrt(n) = 0
#Ciò è in accordo con la legge dei grandi numeri.
```

Problema 2

Si consideri un esame di ammissione composto da 10 domande, ciascuna con 3 risposte di cui una sola esatta. Per essere ammessi occorre fornire almeno il 70% di risposte esatte.
1) Qual è la probabilità di superare il test rispondendo a caso? 2) Qual è la probabilità di superare il test rispondendo a caso e se le domande fossero 100?

In [68]:

```
#Sia X la v. a. che conta il numero di risposte esatte.
#Rispondendo a caso, la probabilità di successo è:
from scipy.stats import binom

p = 1/3
n = 10
P = 0
for i in range(7, 11):
    P += binom.pmf(i, n, p)
print("Probabilità di superare il test rispondendo a caso: ", P)
```

```
Probabilità di superare il test rispondendo a caso:  0.01966163694558758
```

In [69]:

```
n = 100
P = 0
for i in range(70, 101):
    P += binom.pmf(i, n, p)
print("Probabilità di superare il test rispondendo a caso e se le domande fossero 100: ",
P)
```

```
Probabilità di superare il test rispondendo a caso e se le domande fossero 100:  7.733593
509556956e-14
```

In [70]:

```
#Il calcolo del coefficiente binomiale può risultare complicato.
#Applichiamo la regola empirica:
if((n*p > 5) and (n*(1-p) > 5)):
    print("È possibile usare l'approssimazione normale")
```

```
È possibile usare l'approssimazione normale
```

In [71]:

```
from scipy.stats import norm

P = 1 - norm.cdf(70)
print(P)
```

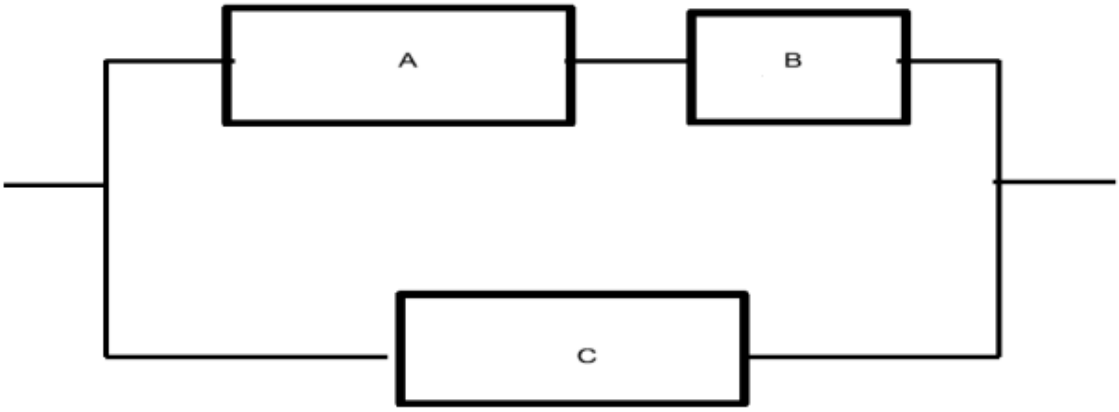
```
0.0
```

5) Statistica descrittiva e inferenziale

Esercizio 1

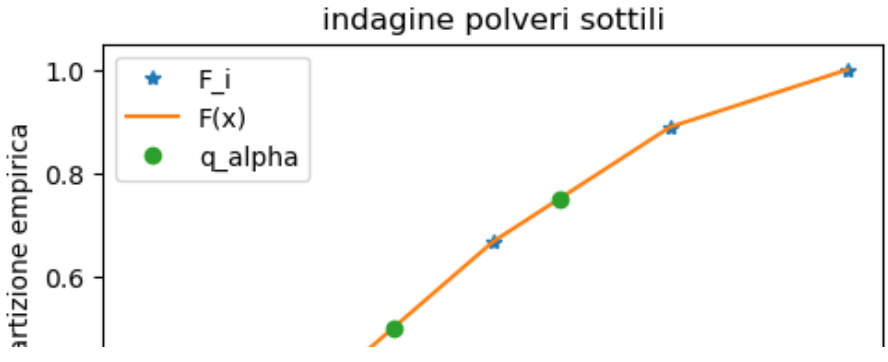
Un monitoraggio sulla densità di polveri sottili nell'aria ha condotto alle frequenze riportate nella tabella sotto, in

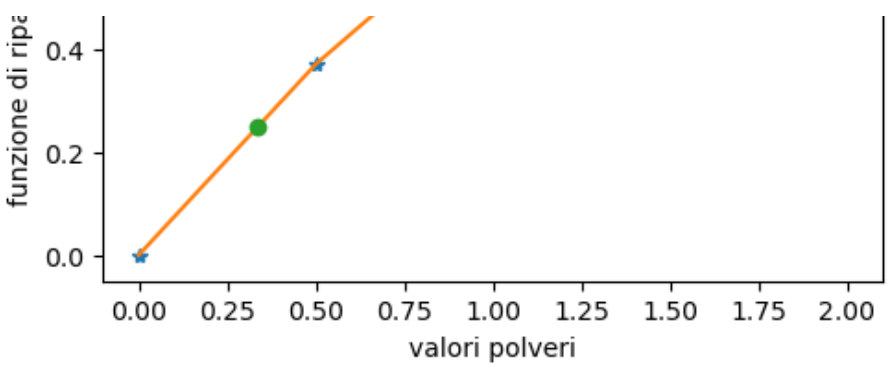
opportune unità di misura. Si calcolino i quartili empirici.



In [72]:

```
import numpy as np
import matplotlib.pyplot as plt

freq=np.array([10,8,6,3]) #frequenze relative
pp = freq/np.sum(freq)    #frequenze percentuali
freq_cum = np.cumsum(pp)  #frequenze cumulate
freq_cum=np.insert(freq_cum, 0, 0)

x = np.array([0, 0.5, 1, 1.5, 2])
alpha = np.array([0.25, 0.5, 0.75])

jj = np.zeros(3, dtype=int)
for k in range(3):
    for j in range(1,5):
        if alpha[k] > freq_cum[j-1] and alpha[k] <= freq_cum[j]:
            jj[k] = j

q_alpha = np.zeros(3)
for k in range(3):
    j = jj[k]
    q_alpha[k] = x[j-1] + (alpha[k]-freq_cum[j-1])*(x[j]-x[j-1])/(freq_cum[j]-freq_cum[j
-1])

plt.figure(figsize=(12, 4))

plt.subplot(121)

plt.plot(x, freq_cum, '*', label='F_i')
plt.plot(x, freq_cum, label='F(x)')
plt.plot(q_alpha, alpha, 'o', label='q_alpha')

legend = plt.legend(loc='upper left')

plt.title("indagine polveri sottili")
plt.xlabel("valori polveri")
plt.ylabel("funzione di ripartizione empirica")

plt.show()
```





Esercizio 2

L'altezza di 2000 individui di una popolazione è riportata nel file 'Data altezze.dat'.
1) Calcolare media e deviazione standard della popolazione. 2) Costruire un istogramma a 20 barre. 3) È possibile adattare ai dati una distribuzione normale?

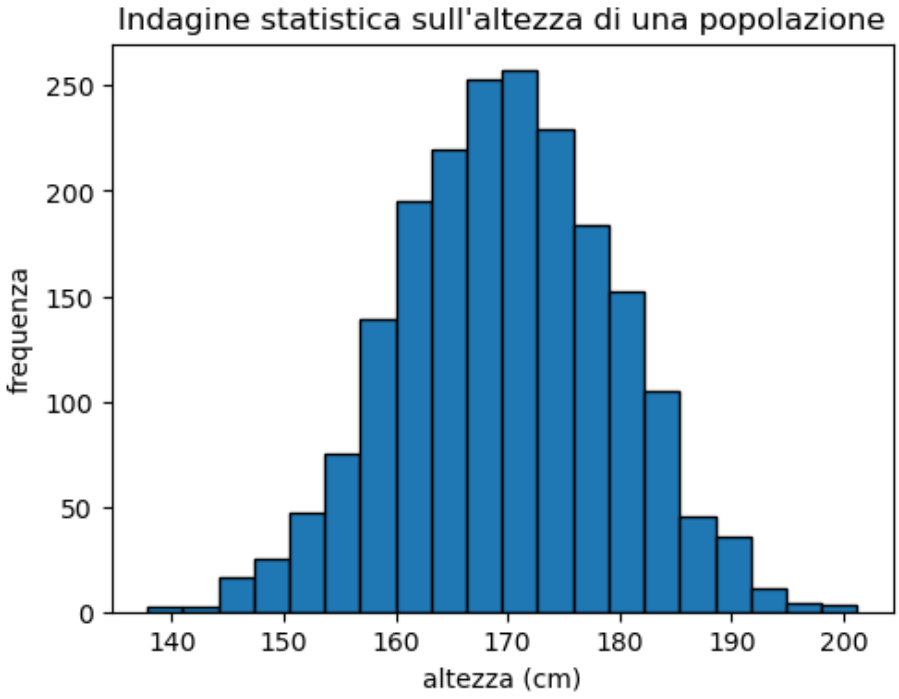
In [73]:

```
import numpy as np
import matplotlib.pyplot as plt

x=np.loadtxt('Data_altezze.dat')
print("Media: ", np.mean(x))
print("Deviazione standard: ", np.std(x))
```

```
Media:  169.88389694
Deviazione standard:  9.577702521955114
```

In [74]:

```
plt.figure(figsize=(12, 4))
plt.subplot(121)

plt.hist(x, bins=20, edgecolor="black")
plt.title("Indagine statistica sull'altezza di una popolazione")
plt.xlabel("altezza (cm)")
plt.ylabel("frequenza")

plt.show()
```

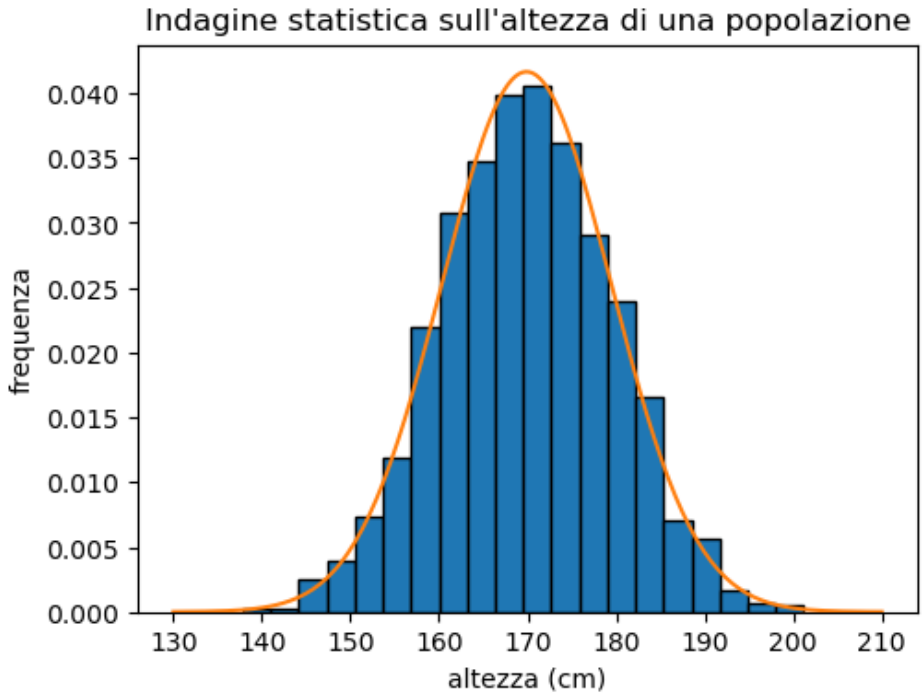


In [75]:

```
from scipy.stats import skew
```

```
from scipy.stats import kurtosis
from scipy.stats import norm

s = skew(x)
k = kurtosis(x, fisher=False)

print("Skewness: ", s)
print("Kurtosi: ", k)

plt.figure(figsize=(12, 4))
plt.subplot(121)

plt.hist(x, density=True, bins=20, edgecolor="black")
plt.title("Indagine statistica sull'altezza di una popolazione")
plt.xlabel("altezza (cm)")
plt.ylabel("frequenza")

xx = np.linspace(130, 210, num=200)
yy = norm.pdf(xx, loc=np.mean(x), scale=np.std(x))

plt.plot(xx, yy)

plt.show()
```

```
Skewness:  0.008198717711043949
Kurtosi:  2.8849900010804586
```



Esercizio 3 (intervalli di confidenza per la media)

Si vuole testare un dispositivo con uno strumento che fornice delle misure di voltaggio. Si eseguono 9 misurazioni registrando i valori in volt:
11, 13.2, 12.3, 10.9, 13, 10.5, 12.3, 13, 13.15. È nota la precisione dello strumento e si ha σ = 1 V. 1) Si determinino gli intervalli di confidenza al 95% e al 99%. 2) Determinare gli stessi intervalli di confidenza nel caso in cui si avesse σ = 1.4 V. 3) Sempre con precisione σ = 1 V, determinare gli stessi intervalli con la stessa media delle misure ma supponendo che essa provenga da un campione di 20 misurazioni.

In [76]:

```
import numpy as np
from scipy.stats import norm

sig = 1.
X = np.array([11, 13.2, 12.3, 10.9, 13, 10.5, 12.3, 13, 13.15])
m = np.mean(X)
n = X.size
```

```
alpha=0.05
phi = norm.ppf(1-alpha/2.)

I_l = m - sig/np.sqrt(n)*phi
I_r = m + sig/np.sqrt(n)*phi

print(f'Intervallo di confidenza per la media al 95% [{I_l}, {I_r}]')

alpha=0.01
phi = norm.ppf(1-alpha/2.)

I_l = m - sig/np.sqrt(n)*phi
I_r = m + sig/np.sqrt(n)*phi

print(f'Intervallo di confidenza per la media al 99% [{I_l}, {I_r}]')
```

```
Intervallo di confidenza per la media al 95% [11.496678671819982, 12.803321328180019]
Intervallo di confidenza per la media al 99% [11.291390232150366, 13.008609767849634]
```

In [77]:

```
sig = 1.4

alpha=0.05
phi = norm.ppf(1-alpha/2.)

I_l = m - sig/np.sqrt(n)*phi
I_r = m + sig/np.sqrt(n)*phi

print(f'Intervallo di confidenza per la media al 95% (sigma=1.4) [{I_l}, {I_r}]')

alpha=0.01
phi = norm.ppf(1-alpha/2.)

I_l = m - sig/np.sqrt(n)*phi
I_r = m + sig/np.sqrt(n)*phi

print(f'Intervallo di confidenza per la media al 99% (sigma=1.4) [{I_l}, {I_r}]')
```

```
Intervallo di confidenza per la media al 95% (sigma=1.4) [11.235350140547975, 13.06464985
9452025]
Intervallo di confidenza per la media al 99% (sigma=1.4) [10.947946325010513, 13.35205367
4989488]
```

In [78]:

```
n = 20
sig = 1

alpha=0.05
phi = norm.ppf(1-alpha/2.)

I_l = m - sig/np.sqrt(n)*phi
I_r = m + sig/np.sqrt(n)*phi

print(f'Intervallo di confidenza per la media al 95% (campione di 20 misurazioni) [{I_l},
{I_r}]')

alpha=0.01
phi = norm.ppf(1-alpha/2.)

I_l = m - sig/np.sqrt(n)*phi
I_r = m + sig/np.sqrt(n)*phi

print(f'Intervallo di confidenza per la media al 99% (campione di 20 misurazioni) [{I_l},
{I_r}]')
```

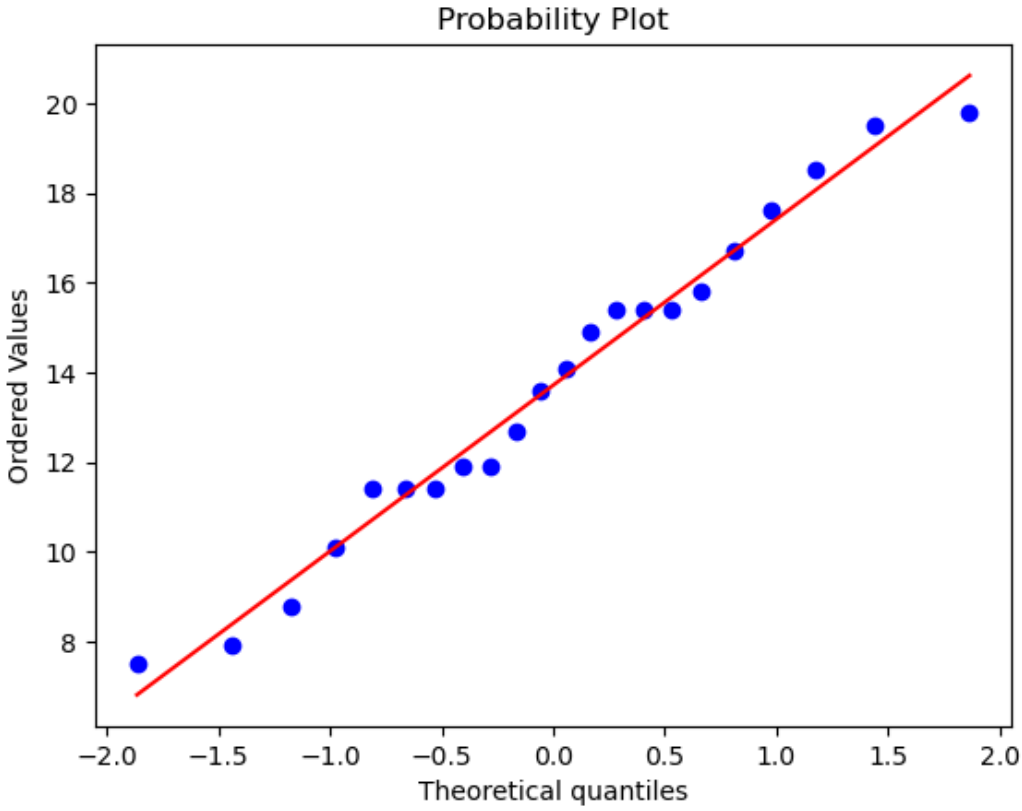
```
Intervallo di confidenza per la media al 95% (campione di 20 misurazioni) [11.71173872971
171, 12.58826127028829]
Intervallo di confidenza per la media al 99% (campione di 20 misurazioni) [11.57402705788
2873, 12.725972942117128]
```

Esercizio 4 (intervalli di confidenza per la media senza varianza)

Viene effettuato un test di rottura di un certo materiale ottenendo i seguenti valori in megapascal(MPa):
19.8 10.1 14.9 7.5 15.4 15.4 15.4 18.5 7.9 12.7 11.9 11.4 11.4 14.1 17.6 16.7 15.8 19.5 8.8 13.6 11.9 11.4
Dopo aver verificato graficamente che il campione proviene da una popolazione distribuita approssimativamente in modo normale, determinare l'intervallo di confidenza al 95% per la media.

In [79]:

```
import numpy as np
from scipy.stats import t
import matplotlib.pyplot as plt
from scipy.stats import probplot
from scipy.stats import norm

X = np.array([19.8, 10.1, 14.9, 7.5, 15.4, 15.4, 15.4, 18.5, 7.9, 12.7, 11.9, 11.4, 11.4
, 14.1, 17.6, 16.7, 15.8, 19.5, 8.8, 13.6, 11.9, 11.4])

m = np.mean(X)
n = X.size
S = np.std(X, ddof=1)

fig, ax = plt.subplots(1,1)
probplot(X, dist=norm, plot=ax)
plt.show()
print("Si può assumere che la distribuzione della popolazione è approssimativamente norma
le")
```




```
Si può assumere che la distribuzione della popolazione è approssimativamente normale
```

In [80]:

```
df = n-1
alpha = 0.05
t1 = t.ppf(1-alpha/2, df)

I_l = m - S/np.sqrt(n)*t1
I_r = m + S/np.sqrt(n)*t1

print(f'Intervallo di confidenza per la media al 95% [{I_l}, {I_r}]')
```

```
Intervallo di confidenza per la media al 95% [12.138069152904343, 15.289203574368383]
```

Intervallo di confidenza per la media al 95% [12.136069152904345, 13.269205574366365]

Esercizio 5 (intervalli di confidenza per la varianza)

Un macchinario riempie automaticamente delle bottiglie. Da un campione di 20 misurazioni siottengono i seguenti valori (in litri)
2.05, 2.04, 1.98, 1.96, 2.03, 2.01, 1.97, 1.99, 2.01, 2.05 1.96, 1.95, 2.04, 2.01, 1.97, 1.96, 2.02, 2.04, 1.98, 1.94
Se la varianza fosse troppo grande, la proporzione di bottiglie sotto o sovrariempite sarebbe non accettabile. Calcolare l'intervallo di confidenza al 95% per il limite superiore per la deviazione standard.

In [81]:

```
import numpy as np
from scipy.stats import chi2

X = np.array([2.05, 2.04, 1.98, 1.96, 2.03, 2.01, 1.97, 1.99, 2.01, 2.05, 1.96, 1.95, 2.
04, 2.01, 1.97, 1.96, 2.02, 2.04, 1.98, 1.94])
N = X.size
S = np.std(X, ddof=1)

alpha = 0.05
chi = chi2.ppf(1.-alpha, N-1)
sigma = np.sqrt(S**2*(N-1)/chi)
print(f"L'intervallo di confidenza al 95% per il limite superiore della deviazione standa
rd è [0, {sigma}]")
```

```
L'intervallo di confidenza al 95% per il limite superiore della deviazione standard è [0,
0.028752590846360084]
```

6) Test di ipotesi

Esercizio 1 (Test sulla media con varianza nota)

Le specifiche tecniche per la velocità di combustione di un propellente richiedono che deve essere di 50 cm/s. Sappiamo che la deviazione standard è di 2 cm/s. Si effettuano 24 misurazioni:
51., 50.2, 49.5, 48.7, 50.2, 50.5, 49.6, 51.1, 50.6, 49.1, 53.1, 50.4, 49.3, 48.9, 50.3, 51.8, 51.3, 48.5, 49.3, 55.1, 53.1, 52.5, 55.1, 50.6
Si può rigettare l'ipotesi nulla con un livello di significatività del 5%? E se invece si richiedesse l'1%? Calcolare infine il p-value.
Supponiamo che lo sperimentatore voglia impostare il test in modo che la reale velocità di combustione media differisca da 50 cm/s per al più 1 cm/s. Si vuole inoltre che il test affermerà questo fatto (cioè rigetterà H0 : μ = 50) con una probabilità del 90% e un livello di significatività del 5%. Determinare la dimensione campionaria.

In [82]:

```
#Ipotesi nulla       H_0: mu = mu_0
#Ipotesi alternativa H_1: mu != mu_0

import numpy as np
from scipy.stats import norm

X = np.array([51., 50.2, 49.5, 48.7, 50.2, 50.5, 49.6, 51.1, 50.6, 49.1, 53.1, 50.4,
49.3, 48.9, 50.3, 51.8, 51.3, 48.5, 49.3, 55.1, 53.1, 52.5, 55.1, 50.6])
sig = 2.
mu_0 = 50.

mu = np.mean(X)
n = X.size
Z_0 = (mu-mu_0)/sig*np.sqrt(n)

alpha = 0.05
phi = norm.ppf(1-alpha/2)

if (np.abs(Z_0)>phi):
```

```
    print("Poiché |Z_0| > phi, si rigetta l'ipotesi nulla. Quindi si afferma che il prope
llente non rispetta le specifiche con una significatività del 95%")
else:
    print("Poiché |Z_0| < phi, non si hanno elementi sufficienti per rigettare l'ipotesi
nulla. Ma non si afferma nulla.")
```

```
Poiché |Z_0| > phi, si rigetta l'ipotesi nulla. Quindi si afferma che il propellente non
rispetta le specifiche con una significatività del 95%
```

In [83]:

```
#Se si richiedesse un livello di signifativitá del 1%
alpha = 0.01
phi = norm.ppf(1-alpha/2)

if (np.abs(Z_0)>phi):
    print("Poiché |Z_0| > phi, si rigetta l'ipotesi nulla. Quindi si afferma che il prope
llente non rispetta le specifiche con una significatività del 99%")
else:
    print("Poiché |Z_0| < phi, non si hanno elementi sufficienti per rigettare l'ipotesi
nulla. Ma non si afferma nulla.")
```

```
Poiché |Z_0| < phi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma n
on si afferma nulla.
```

In [84]:

```
#Calcoliamo il p-value
p = 2.*(1.-norm.cdf(np.abs(Z_0)))
print("P-value: ", p)
```

```
P-value:  0.04329746577930438
```

In [85]:

```
#Calcoliamo la dimensione campionaria tramite l'errore di secondo tipo
alpha = 0.05
beta = 0.1
delta = 1

phi=norm.ppf(1.-alpha/2)
dim_camp = (phi+norm.ppf(1.-beta))**2*sig**2/delta**2.
print("Dimensione campionaria minima per poter rigettare l'ipotesi nulla con una probabil
ità del 90% e un livello di significatività del 95%: ", dim_camp)
```

```
Dimensione campionaria minima per poter rigettare l'ipotesi nulla con una probabilità del
90% e un livello di significatività del 95%:  42.029692245762476
```

Esercizio 2 (Test sulla media con varianza ignota)

Si vuole testare ad un livello di significatività $\alpha = 0.05$ se il carico di rottura di un materiale supera 10 MPa, tenendo presente che 22 prove hanno fornito i seguenti risultati:
19.8 18.5 17.6 16.7 15.8 15.4 14.1 13.6 11.9 11.4 11.4 8.8 7.5 15.4 15.4 19.5 14.9 12.7 11.9 11.4 10.1 7.9
Calcolare inoltre il p-value.

In [86]:

```
#Ipotesi nulla       H_0: mu = mu_0
#Ipotesi alternativa H_1: mu > mu_0

import numpy as np
from scipy.stats import t

X = np.array([19.8, 18.5, 17.6, 16.7, 15.8, 15.4, 14.1, 13.6, 11.9, 11.4, 11.4, 8.8, 7.5
, 15.4, 15.4, 19.5, 14.9, 12.7, 11.9, 11.4, 10.1, 7.9])
mu_0 = 10
alpha = 0.05

n = X.size
mu = np.mean(X)
```

```
S = np.std(X, ddof=1)

T_0 = (mu-mu_0)/S*np.sqrt(n)
T = t.ppf(1-alpha, n-1)

if (T_0>T):
    print("Poiché T_0 > T, si rigetta l'ipotesi nulla. Quindi si afferma che il carico di rottura supera significativamente il valore di 10 con una significatività del 95%")
else:
    print("Poiché T_0 < T, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché T_0 > T, si rigetta l'ipotesi nulla. Quindi si afferma che il carico di rottura supera significativamente il valore di 10 con una significatività del 95%
```

In [87]:

```
p = 1 - t.cdf(T_0, n-1)
print("P-value: ", p)
```

```
P-value:  3.781272593450513e-05
```

Esercizio 3 (Test sulla varianza)

Un macchinario riempie automaticamente delle bottiglie. Da un campione di 20 misurazioni si ottengono i seguenti valori (in litri):
2.05, 2.04, 1.98, 1.96, 2.03, 2.01, 1.97, 1.99, 2.01, 2.05 1.96, 1.95, 2.04, 2.01, 1.97, 1.96, 2.02, 2.04, 1.98, 1.94
Se la deviazione standard fosse superiore a 0.05 litri, la proporzione di bottiglie sotto o sovrariempite sarebbe non accettabile.
I dati del campione contengono prove che suggeriscono che il produttore abbia un problema con le bottiglie riempite troppo o troppo poco? Utilizzare $\alpha = 0.05$ e assumere che il volume di riempimento abbia una distribuzione normale.

In [88]:

```
#Ipotesi nulla       H_0 : sig^2 = sig_0^2
#Ipotesi alternativa H_1 : sig^2 < sig_0^2

import numpy as np
from scipy.stats import chi2

X = np.array([2.05, 2.04, 1.98, 1.96, 2.03, 2.01, 1.97, 1.99, 2.01, 2.05, 1.96, 1.95, 2.04, 2.01, 1.97, 1.96, 2.02, 2.04, 1.98, 1.94])
sig_0 = 0.05
alpha = 0.05

n = X.size
S = np.std(X, ddof=1)

W_0 = S**2/sig_0**2*(n-1)
chi = chi2.ppf(alpha, n-1)

if (W_0<chi):
    print("Poiché W_0 < chi, si rigetta l'ipotesi nulla. Quindi si afferma che la proporzione di bottiglie sotto o sovrariempite è accettabile con una significatività del 95%")
else:
    print("Poiché W_0 > chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché W_0 < chi, si rigetta l'ipotesi nulla. Quindi si afferma che la proporzione di bottiglie sotto o sovrariempite è accettabile con una significatività del 95%
```

Esercizio 4 (Test sulla proporzione)

In una catena di produzione si vuole mantenere il numero di pezzi difettosi al di sotto del 5%. Si analizza un campione di 200 pezzi e si trovano 4 pezzi difettosi. 1) Si può asserire ad un livello di significatività $\alpha = 0.05$ che la produzione rispetta le aspettative? 2) Supponendo che il valore vero sia $p* = 0.03$ e supponendo che il costruttore voglia accettare un valore dell'errore di secondo tipo $\beta = 0.1$, quale ampiezza dovrebbe avere il

costruttore voglia accettare un valore dell'errore di secondo tipo $\beta = 0.1$, quale ampiezza dovrebbe avere il campione?

In [89]:

```
#ipotesi nulla       H_0 : p = p0
#ipotesi alternativa H_1 : p < p0

import numpy as np

p0 = 0.05
difettosi = 4
n = 200
alpha = 0.05

mu = difettosi / n
Z_0 = (mu - p0)/np.sqrt(p0*(1-p0))*np.sqrt(n)
phi = norm.ppf(alpha)

if (Z_0<phi):
    print("Poiché Z_0 < phi, si rigetta l'ipotesi nulla. Quindi si afferma che la produzi
one rispetta le aspettative con una significatività del 95%")
else:
    print("Poiché Z_0 > phi, non si hanno elementi sufficienti per rigettare l'ipotesi nu
lla. Ma non si afferma nulla.")
```

```
Poiché Z_0 < phi, si rigetta l'ipotesi nulla. Quindi si afferma che la produzione rispett
a le aspettative con una significatività del 95%
```

In [90]:

```
p_star = 0.03
beta = 0.1

phi_beta = norm.ppf(beta)
dim_camp = ((phi_beta*np.sqrt(p_star*(1-p_star))+phi*np.sqrt(p0*(1-p0)))/(p0-p_star))**2
print("Dimensione campionaria minima per poter rigettare l'ipotesi nulla con una probabil
ità del 90% e un livello di significatività del 95%: ", dim_camp)
```

```
Dimensione campionaria minima per poter rigettare l'ipotesi nulla con una probabilità del
90% e un livello di significatività del 95%:  832.6221546780524
```

Esercizio 5 (Test sulla media per coppie di popolazioni)

Si vogliono confrontare due tipi di preparati per pittura. Ci si aspetta un diverso tempo di essiccamento. Si può supporre che la deviazione standard del tempo di essiccamento per ciascun tipo di essiccamento sia 8 minuti. 10 pareti vengono tinteggiate con il trattamento 1 e altrettamente pareti con il trattamento 2. Si rilevano le medie campionare $\bar{X} = 121$ minuti e $\bar{Y} = 112$ minuti. Si può trarre la conclusione che il tempo di essiccamento del campione 1 sia maggiore di quello del campione 2 assumendo $\alpha = 0.05$? Calcolare l'intervallo di confidenza per la differenza dei tempi medi di essiccamento.

In [91]:

```
#Ipotesi nulla       H_0 : mu1 = mu2
#Ipotesi alternativa H_1 : mu1 > mu2

import numpy as np
from scipy.stats import norm

sig = 8
n = 10
mu1 = 121
mu2 = 112
alpha = 0.05

Z_0 = (mu1-mu2)/np.sqrt((sig**2/n)+(sig**2/n))
phi = norm.ppf(1-alpha)

if (Z_0>phi):
```

```
    print("Poiché Z_0 > phi, si rigetta l'ipotesi nulla. Quindi si afferma che il tempo d
i essiccamento del campione 1 è maggiore di quello del campione 2 con un livello di signi
fatività del 95%.")
else:
    print("Poiché Z_0 < phi, non si hanno elementi sufficienti per rigettare l'ipotesi nu
lla. Ma non si afferma nulla.")
```

```
Poiché Z_0 > phi, si rigetta l'ipotesi nulla. Quindi si afferma che il tempo di essiccame
nto del campione 1 è maggiore di quello del campione 2 con un livello di signifatività de
l 95%.
```

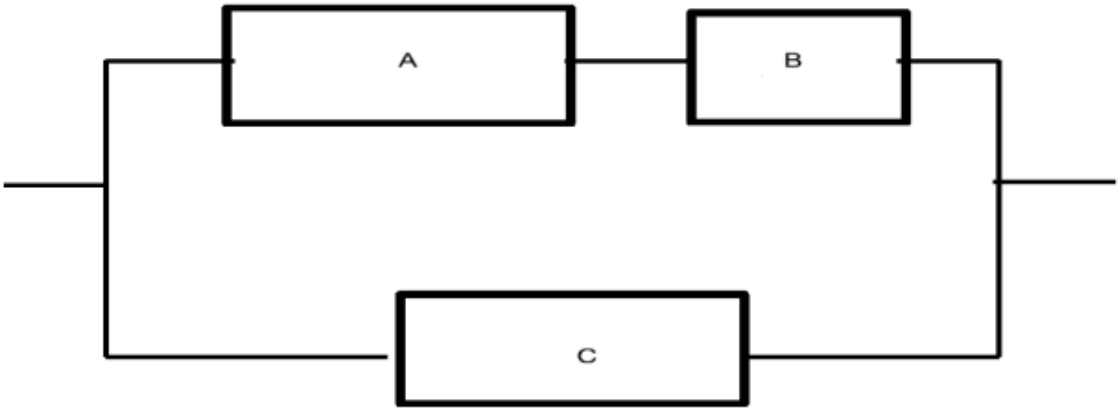
In [92]:

```
I_conf = mu1 - mu2 - phi*np.sqrt(sig**2/n+sig**2/n)
print(f"L'intervallo di confidenza per la differenza dei tempi di essiccamento è ]{I_conf
}, +inf[")
print("Vale a dire che il tempo di essiccamento del primo preparato è di circa almeno 3 m
inuti maggiore di quello del secondo preparato con una significatività del 95%")
```

```
L'intervallo di confidenza per la differenza dei tempi di essiccamento è ]3.1151927633590
85, +inf[
Vale a dire che il tempo di essiccamento del primo preparato è di circa almeno 3 minuti m
aggiore di quello del secondo preparato con una significatività del 95%
```

Esercizio 6 (Test per dati accoppiati)

Quindici adulti di età compresa tra 35 e 50 anni parecipano ad uno studio per valutare gli effetti di dieta alimentare ed esercizio fisico sul livello di colesterolo nel sangue. In ogni individuo il livello di colesterolo è stato misurato inizialmente e tre mesi dopo la dieta e l'allenamento. Con un livello di significatività $\alpha = 0.05$, è possibile avallare l'ipotesi che dieta ed esercizio fisico portino a ridurre il livello medio di colesterolo?



In [93]:

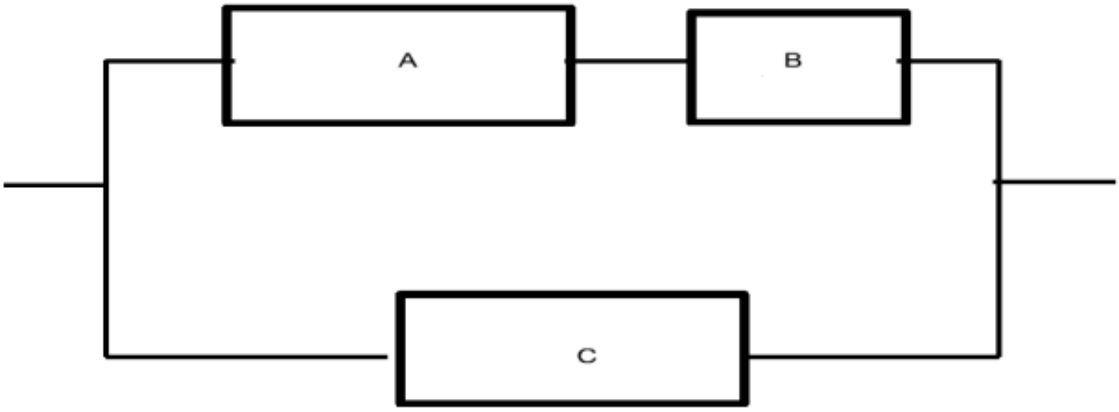
```
#Ipotesi nulla       H_0 : mu_d=0
#Ipotesi alternativa H_1 : mu_d>0

import numpy as np
from scipy.stats import t

X = np.array([[265, 240, 258, 295, 251, 245, 287, 314, 260, 279, 283, 240, 238, 225, 247]
,
              [229, 231, 227, 240, 238, 241, 234, 256, 247, 239, 246, 218, 219, 226, 233
]])
alpha=0.05

D = X[0,:]-X[1,:] #array delle differenze
n=D.size
D_bar = np.mean(D)
S = np.std(D, ddof=1)
T_0 = D_bar/S*np.sqrt(n)
T = t.ppf(1-alpha, n-1)
```

```
if (T_0>T):
    print("Poiché T_0 > T, si rigetta l'ipotesi nulla. Quindi si afferma che il trattamento dieta + esercizio fisico ha prodotto una riduzione del livello medio di colesterolo con un livello di signifatività del 95%.")
else:
    print("Poiché T_0 < T, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché T_0 > T, si rigetta l'ipotesi nulla. Quindi si afferma che il trattamento dieta + esercizio fisico ha prodotto una riduzione del livello medio di colesterolo con un livello di signifatività del 95%.
```

Esercizio 7 (Test del chi-quadro)

Si vuole testare se un dado sia equilibrato o meno ad un livello di significatività $\alpha = 0.05$. Si effettuano 100 lanci, registrando i risultati riportati nella seguente tabella.

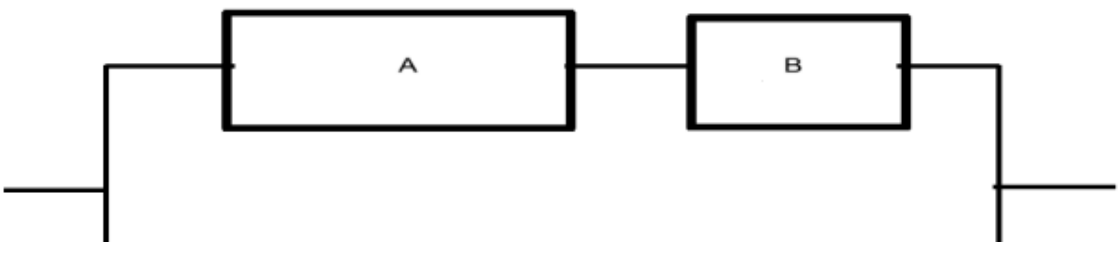


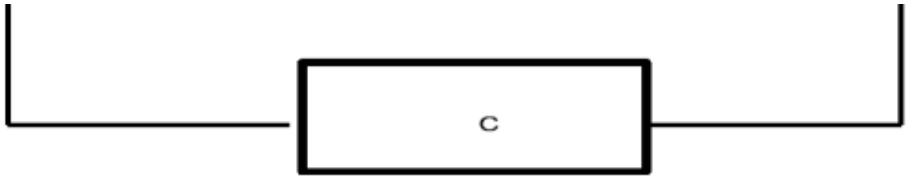
In [94]:

```
#Ipotesi nulla       H_0 : il dado è equilibrato
#Ipotesi alternativa H_1 : il dado non è equilibrato

import numpy as np
from scipy.stats import chi2

X = np.array([20, 7, 12, 18, 20, 23])
E = np.array([100, 100, 100, 100, 100, 100])/6
n = 6
alpha = 0.05

Tn = sum((X-E)**2/E)
chi = chi2.ppf(1-alpha, n-1)

if(Tn > chi):
    print("Poiché Tn > chi, si rigetta l'ipotesi nulla. Quindi si afferma che il dado non è equilibrato con un livello di signifatività del 95%")
else:
    print("Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.
```

Dopo aver aumentato la dimensione campionaria, si registrano i risultati riportati nella seguente tabella. Si ripeta il test di cui sopra con i nuovi dati.





Si ripeta quest'ultimo test con α = 0.01.

In [95]:

```
Y = np.array([388, 322, 314, 316, 344, 316])
E = np.array([2000, 2000, 2000, 2000, 2000, 2000])/6

Tn = sum((Y-E)**2/E)
chi = chi2.ppf(1-alpha, n-1)

if(Tn > chi):
    print("Poiché Tn > chi, si rigetta l'ipotesi nulla. Quindi si afferma che il dado non è equilibrato con un livello di signifativitá del 95%")
else:
    print("Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché Tn > chi, si rigetta l'ipotesi nulla. Quindi si afferma che il dado non è equilibrato con un livello di signifativitá del 95%
```

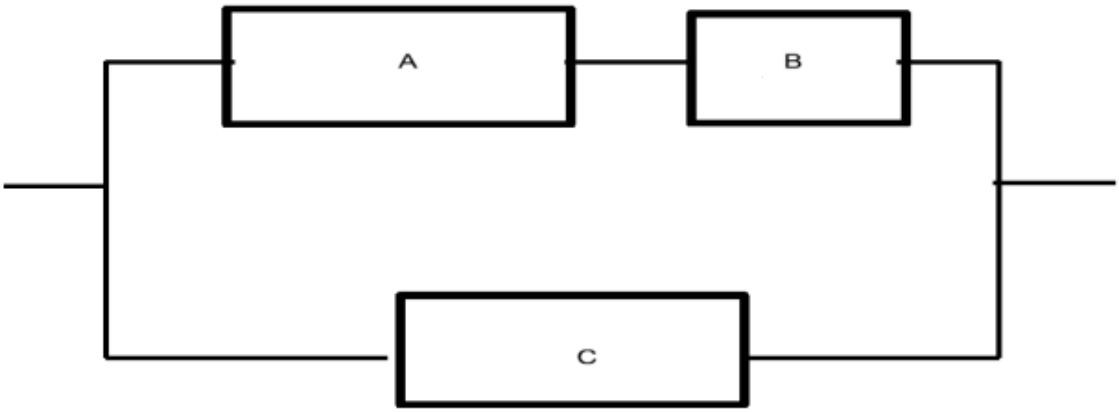
In [96]:

```
alpha = 0.01

chi = chi2.ppf(1-alpha, n-1)

if(Tn > chi):
    print("Poiché Tn > chi, si rigetta l'ipotesi nulla. Quindi si afferma che il dado non è equilibrato con un livello di signifativitá del 99%")
else:
    print("Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.
```

Esercizio 8 (Test di adattamento ai dati di una legge teorica di probabilità)

Si può adattare una distribuzione di Poisson ai dati della seguente tabella?



In [97]:

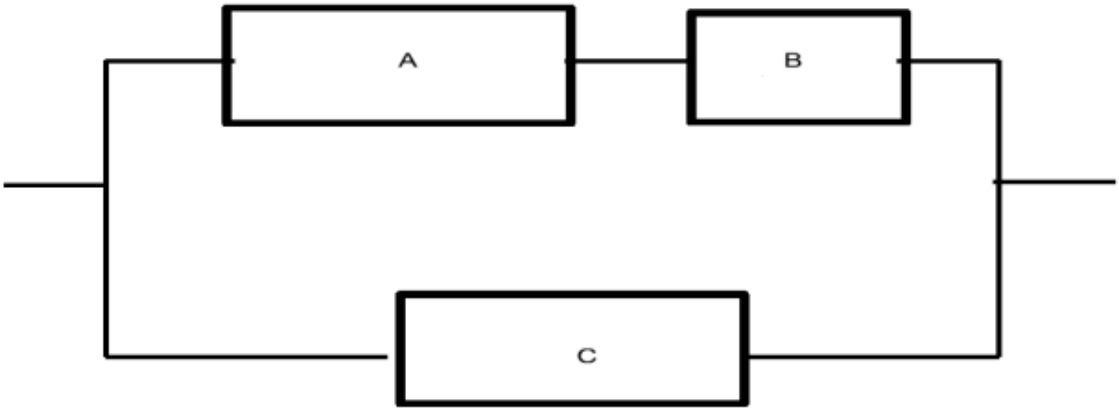
```
#Ipotesi nulla       H_0: i dati seguono una legge teorica con f.r. Fx
#Ipotesi alternativa H_1: i dati non seguono una legge teorica con f.r. Fx

import numpy as np
from scipy.stats import chi2
```

```
from scipy.stats import poisson

X = np.array([584, 398, 165, 35, 15])
alpha=0.05
r=1 #numero di parametri stimati

m = X.size
N = np.sum(X)
p=X/N #frequenze relative empiriche

lam = np.sum(X*np.array([0,1,2,3,4]))/N

p0 = np.zeros(m)
for i in range(m-1):
    p0[i] = poisson.pmf(i, lam)
p0[m-1]=1-np.sum(p0) #frequenze relative teoriche

Tn = N*np.sum((p-p0)**2/p0)
chi = chi2.ppf(1-alpha, m-r-1)

if(Tn > chi):
    print("Poiché Tn > chi, si rigetta l'ipotesi nulla. Quindi si afferma che i dati non seguono una distribuzione di Poisson con un livello di signifativitá del 99%")
else:
    print("Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nul la. Ma non si afferma nulla.")
```

```
Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.
```

Esercizio 9 (Test del chi-quadro di indipendenza)

Si vuole testare se un antibiotico è efficace. Si considerano 170 pazienti. I dati ottenuti sono stati raccolti nella tabella seguente, detta tabella di contingenza,



I due effetti, trattamento e guarigione, sono indipendenti?

In [98]:

```
#Ipotesi nulla       H_0: trattamento e guarigione sono indipendenti
#Ipotesi alternativa H_1: trattamento e guarigione non sono indipendenti

import numpy as np
from scipy.stats import chi2

X = np.array([[44, 10], [81, 35]])
n = 170
m = 2 #possibili esiti per trattamento
r = 2 #possibili esiti per guarigione
alpha = 0.05

p = np.array([54, 116])/n #frequenze realtive empiriche per trattamento
q = np.array([125, 45])/n #frequenze realtive empiriche per guarigione
pi = X/n #probabilità congiunte empiriche
```

```
Tn=0
for h in range(m):
    for k in range(r):
        Tn+=(p[h]*q[k]-pi[h][k])**2/pi[h][k]
Tn*=n
chi = chi2.ppf(1-0.05, (m-1)*(r-1))

if(Tn > chi):
    print("Poiché Tn > chi, si rigetta l'ipotesi nulla. Quindi si afferma che trattamento
e guarigione non sono indipendenti con un livello di signifatività del 99%")
else:
    print("Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nul
la. Ma non si afferma nulla.")
```

```
Poiché Tn < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non
si afferma nulla.
```

Esercizio 10 (Test della mediana)

Un motore a reazione è formato legando insieme un propellente di accensione e un propellente di sostegno all'interno di un alloggiamento metallico. La resistenza al taglio del legame tra i due tipi di propellente è una caratteristica importante. Vogliamo testare l'ipotesi che la mediana della resistenza al taglio sia 2000 psi con una significatività $\alpha = 0.05$. I dati sono riportati nel file Dataset_motore.dat.

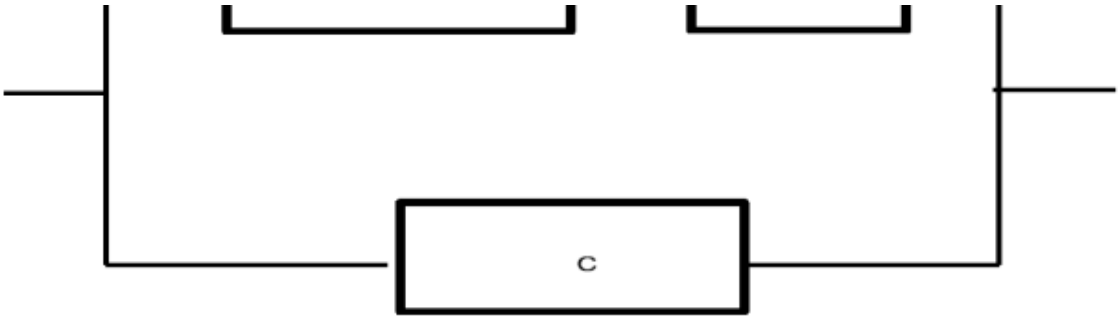
In [99]:

```
#Ipotesi nulla       H_0: mediana=2000 psi
#Ipotesi alternativa H_1: mediana!=2000 psi

import numpy as np
from scipy.stats import binom

X = np.loadtxt('Dataset_motore.dat')
alpha = 0.05

m = np.median(X) #mediana campionaria
n = X.size
Y = X -2000
r = sum((Y>0))

if(r>n/2):
    k = range(r, n+1)
    P = binom.pmf(k, n, 0.5)
else:
    k = range(r, n+1)
    P = 1-binom.pmf(k, n, 0.5)

p_value = 2*np.sum(P)

if(p_value<=alpha):
    print("Poiché p-value <= alpha, si rigetta l'ipotesi nulla. Quindi si afferma che la
mediana si discosta dal valore atteso con un livello di significatività del 95%.")
else:
    print("Poiché p-value > alpha, non si hanno elementi sufficienti per rigettare l'ipot
esi nulla. Ma non si afferma nulla.")
```

```
Poiché p-value > alpha, non si hanno elementi sufficienti per rigettare l'ipotesi nulla.
Ma non si afferma nulla.
```

Esercizio 11 (Test di Kruskal-Wallis)

In un esperimento si confrontano quattro diverse tecniche di mescolamento per il cemento e si misura la resistenza alla trazione. Si può affermare che la tecnica di mescolamento influisca sulla resistenza alla trazione? Si usi $\alpha = 0.05$.



In [100]:

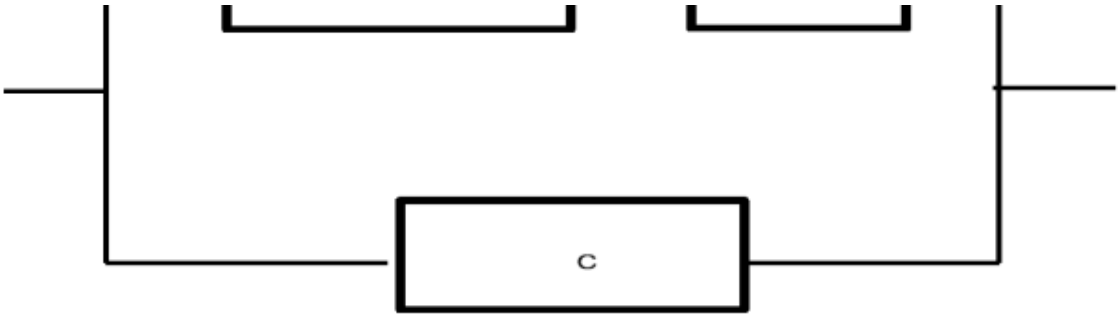
```
#Ipotesi nulla       H_0: la tecnica di mescolamento non influisce
#Ipotesi alternativa H_1: la tecnica di mescolamento influisce

import numpy as np
from scipy.stats import rankdata
from scipy.stats import chi2

X_1 = np.array([3129, 3000, 2865, 2890])
X_2 = np.array([3200, 3000, 2975, 3150])
X_3 = np.array([2800, 2900, 2985, 3050])
X_4 = np.array([2600, 2700, 2600, 2765])
m = 4
alpha=0.05

n= np.array([X_1.size, X_2.size, X_3.size, X_4.size])
N = np.sum(n) #numero totale di osservazioni
Y = np.concatenate([X_1, X_2, X_3, X_4])

R = rankdata(Y) #ranghi
S2 = (np.sum(R**2)- N*(N+1)**2/4)/(N-1)

RR = np.zeros(m)
for i in range(m):
    RR[i]=np.sum(R[m*i:m*(i+1)]) #somma dei ranghi per ogni campione

H = (np.sum(RR**2/n)-N*(N+1)**2/4)/S2
chi = chi2.ppf(alpha, m-1)

if(H>=chi):
    print("Poiché H >= chi, si rigetta l'ipotesi nulla. Quindi si afferma che la tecnica di mescolamento influisce sulla resistenza alla trazione con un livello di significatività del 95%.")
else:
    print("Poiché H < chi, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché H >= chi, si rigetta l'ipotesi nulla. Quindi si afferma che la tecnica di mescolamento influisce sulla resistenza alla trazione con un livello di significatività del 95%.
```

7) Regressione lineare

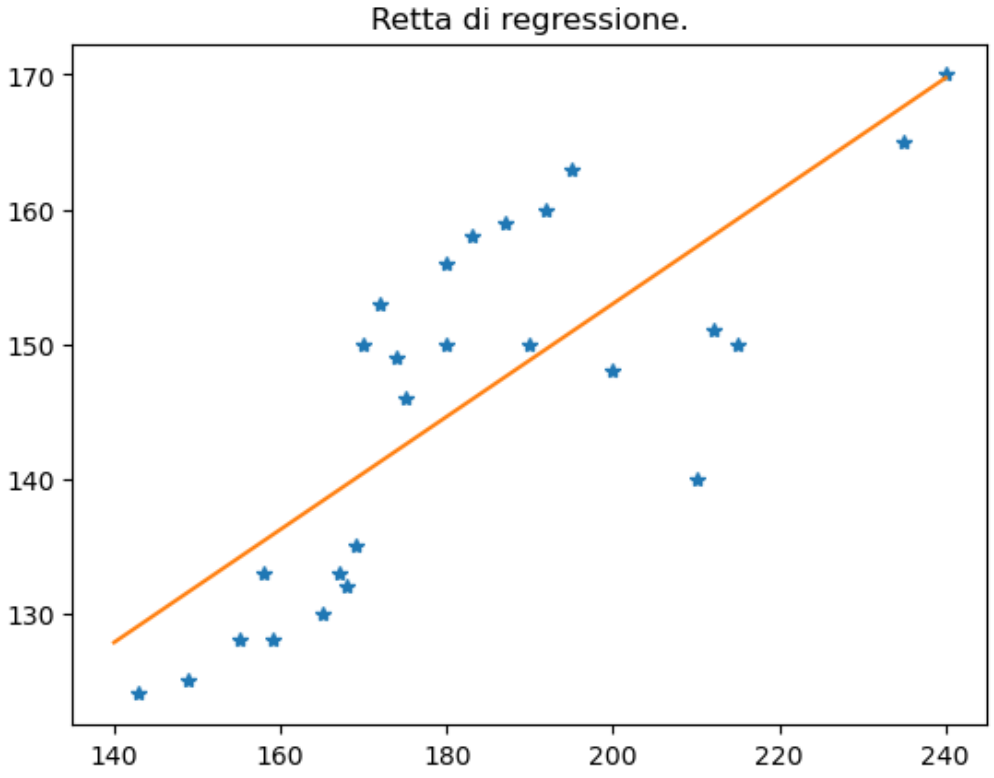
Esercizio 1

Il peso corporeo e la pressione sistolica del sangue di 26 individui maschi selezionati in modo casuale nella fascia d'età che va da 25 a 30 anni sono mostrati in tabella. Assumiamo che il peso e la pressione sanguigna siano normalmente distribuiti. 1) Si determini la retta di regressione. 2) Si calcolino gli intervalli di confidenza per i coefficienti di regressione. 3) Si testi la significatività della regressione usando $\alpha = 0.05$. 4) Si calcoli il coefficiente di determinazione.



In [3]:

```
import numpy as np
import matplotlib.pyplot as plt

Data = np.loadtxt("DATA_reg_lin.dat")
x = Data[:,1]
y = Data[:,2]

y_bar = y.mean()
x_bar = x.mean()
sig_xy = sum((x - x_bar)*(y - y_bar))/x.size
sig_x_2 = sum((x - x_bar)**2)/x.size

b_0 = y_bar - (sig_xy / sig_x_2) * x_bar
b_1 = sig_xy / sig_x_2

xx =  np.linspace(140, 240, 1000)
yy = b_0 + b_1*xx

plt.plot(x,y,'*')
plt.plot(xx, yy)
plt.title("Retta di regressione.")
plt.show()
```



In [10]:

```
#Intervalli di confidenza
from scipy.stats import t

#Calcolo dei residui
y_hat = b_0 + b_1 * x
```

```
r = y - y_hat

s2 = np.sum(r**2)/(x.size-2)
n = x.size
alpha = 0.05
T = t.ppf(1-alpha/2, n-2)
b_0_l = b_0 - np.sqrt(s2)*np.sqrt(1/n+x_bar**2/(n*sig_x_2))*T
b_0_r = b_0 + np.sqrt(s2)*np.sqrt(1/n+x_bar**2/(n*sig_x_2))*T
b_1_l = b_1 - np.sqrt(s2)/np.sqrt(n*sig_x_2)*T
b_1_r = b_1 + np.sqrt(s2)/np.sqrt(n*sig_x_2)*T
print("Valore del parametro b_0:", b_0)
print(f"Intervallo di confidenza per il parametro b_0: [{b_0_l}, {b_0_r}].\n")
print("Valore del parametro b_1:", b_1)
print(f"Intervallo di confidenza per il parametro b_1: [{b_1_l}, {b_1_r}].")
```

```
Valore del parametro b_0: 69.10437279118662
Intervallo di confidenza per il parametro b_0: [42.459175605734316, 95.74956997663892].

Valore del parametro b_1: 0.41941520291569634
Intervallo di confidenza per il parametro b_1: [0.2746281144802123, 0.5642022913511804].
```

In [11]:

```
#Significatività della regressione
#ipotesi nulla       H_0 : beta_1 = 0
#ipotesi alternativa H_1 : beta_1 != 0

T1 = np.sqrt(n)*b_1*np.sqrt(sig_x_2)/np.sqrt(s2)

if(np.abs(T1)>=T):
    print("Poiché |T1| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che la pressio
ne sistolica dipende dal peso con un livello di significatività del 95%.")
else:
    print("Poiché |T1| < T, non si hanno elementi sufficienti per rigettare l'ipotesi nul
la. Ma non si afferma nulla.")
```

```
Poiché |T1| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che la pressione sistolic
a dipende dal peso con un livello di significatività del 95%.
```

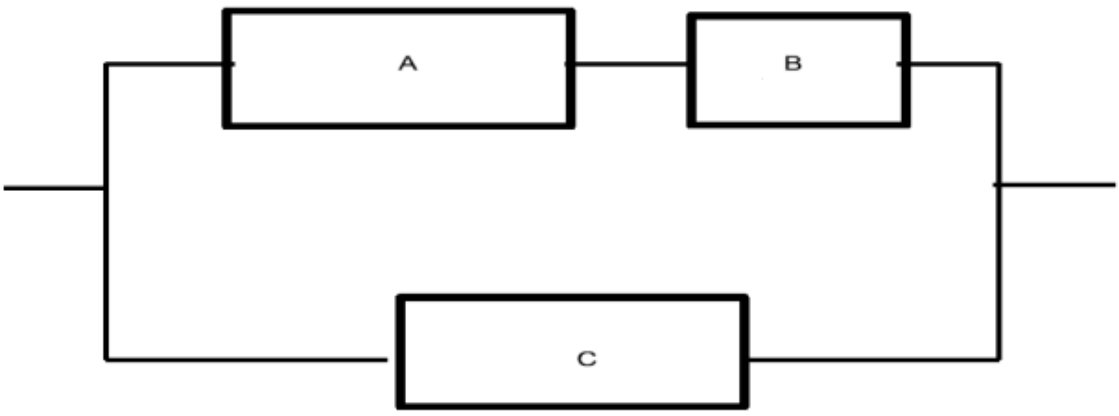
In [13]:

```
#Coefficiente di determinazione
sig_y_2 = np.sum((y-y_bar)**2)/n
R2 = sig_xy**2/(sig_y_2*sig_x_2)
print("Coefficiente di determinazione:", R2)
```

```
Coefficiente di determinazione: 0.5982872450148402
```

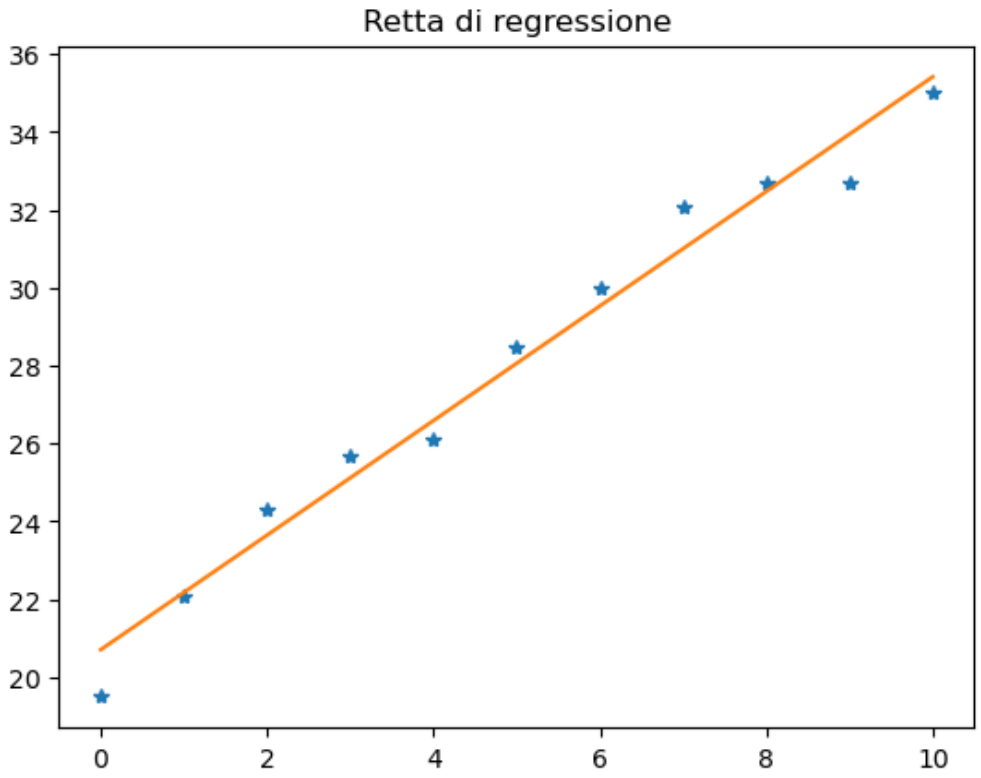
Esercizio 2

L'ossigeno consumato da una persona che cammina è funzione della sua velocità. La seguente tabella riporta il volume di ossigeno consumato a varie velocità di cammino. Ipotizzando una relazione lineare, scrivere l'equazione della retta di regressione. Si testi la significatività della regressione usando $\alpha = 0.05$.



In [25]:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
y = np.array([19.5, 22.1, 24.3, 25.7, 26.1, 28.5, 30, 32.1, 32.7, 32.7, 35])

y_bar = y.mean()
x_bar = x.mean()
sig_xy = sum((x - x_bar)*(y - y_bar))/x.size
sig_x_2 = sum((x - x_bar)**2)/x.size

b_0 = y_bar - (sig_xy / sig_x_2) * x_bar
b_1 = sig_xy / sig_x_2

xx =  np.linspace(0, 10, 1000)
yy = b_0 + b_1*xx

plt.plot(x,y,'*')
plt.plot(xx, yy)
plt.title("Retta di regressione")
plt.show()
```



In [17]:

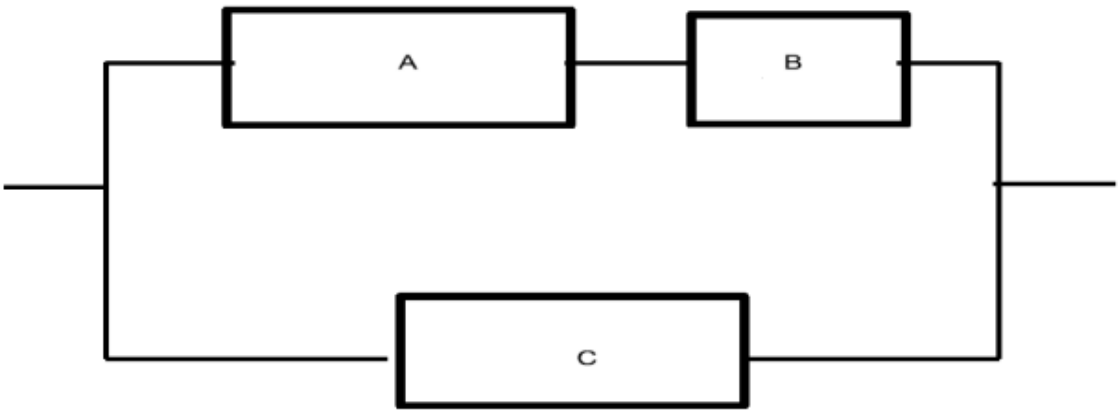
```
#Significatività della regressione
#ipotesi nulla       H_0 : beta_1 = 0
#ipotesi alternativa H_1 : beta_1 != 0

from scipy.stats import t

#Calcolo dei residui
y_hat = b_0 + b_1 * x
r = y - y_hat

s2 = np.sum(r**2)/(x.size-2)
n = x.size
alpha = 0.05
T = t.ppf(1-alpha/2, n-2)
T1 = np.sqrt(n)*b_1*np.sqrt(sig_x_2)/np.sqrt(s2)

if(np.abs(T1)>=T):
    print("Poiché |T1| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che l'ossigeno consumato dipende dalla velocità con un livello di significatività del 95%.")
```

```
else:
    print("Poiché |T1| < T, non si hanno elementi sufficienti per rigettare l'ipotesi nul
la. Ma non si afferma nulla.")
```

```
Poiché |T1| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che l'ossigeno consumato
dipende dalla velocità con un livello di significatività del 95%.
```

Esercizio 3

Consideriamo i dati di tabella. Formuliamo degli appropriati modelli di regressione.



In [27]:

```
import numpy as np

DATA = np.loadtxt("DATA_reg_lin_2.dat")
x = DATA[:,0]
y = DATA[:,1]

n = x.size
x_bar = x.mean()
y_bar = y.mean()

sig_xy = sum((x - x_bar)*(y - y_bar))/x.size
sig_x_2 = sum((x - x_bar)**2)/x.size

#Calcolo dei coefficienti di regressione
b_0 = y_bar - (sig_xy / sig_x_2) * x_bar
b_1 = sig_xy / sig_x_2

print("Coefficiente b_0:", b_0)
print("Coefficiente b_1:", b_1)
```

```
Coefficiente b_0: -4.391330475499407
Coefficiente b_1: 3.010167753127515
```

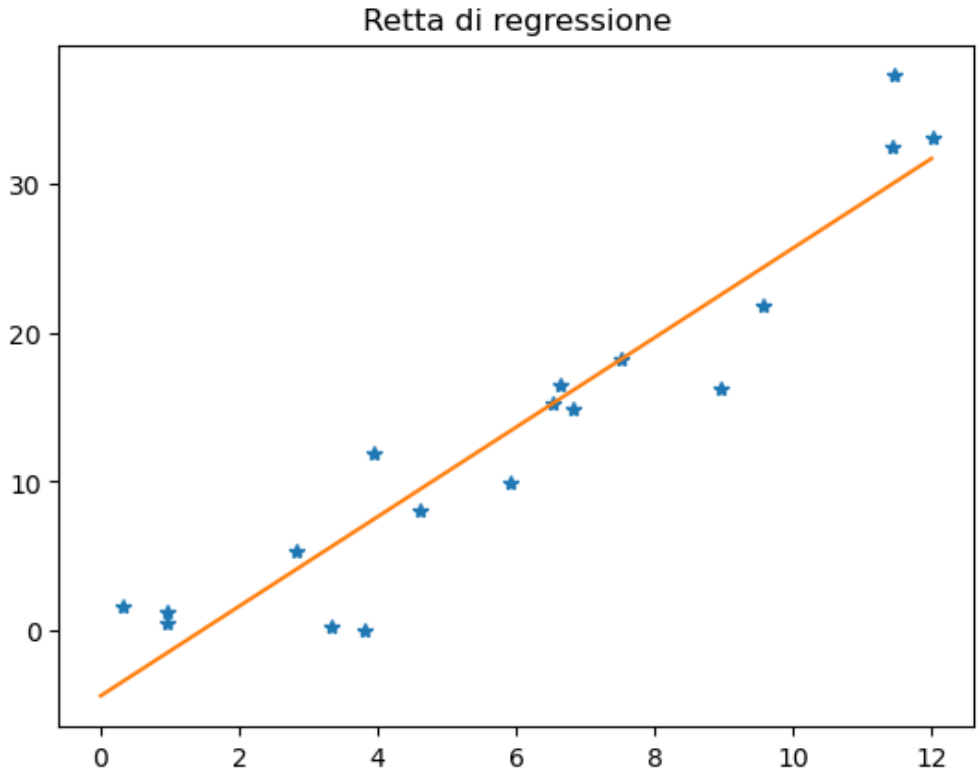
In [28]:

```
#Calcolo dei coefficienti di regressione usando le formule della rigressione lienare mult
ipla
x1 = np.ones(n)
x2 = x
X = np.zeros((n, 2))
X[:,0] = x1
X[:,1] = x2
XX = np.linalg.pinv(X)
b = np.dot(XX, y)
print("Coefficiente b_0:", b[0])
print("Coefficiente b_1:", b[1])
```

```
Coefficiente b_0: -4.391330475499409
Coefficiente b_1: 3.0101677531275146
```

In [29]:

```
import matplotlib.pyplot as plt

xx =  np.linspace(0, 12, 1111111)
yy = b[0] + b[1]*xx

plt.plot(x,y,'*')
plt.plot(xx, yy)
plt.title("Retta di regressione")
plt.show()
```



In [30]:

```
#Coefficiente di determinazione
y_hat = np.dot(X, b)
R2 = np.sum((y_hat-y_bar)**2)/np.sum((y-y_bar)**2)
print("Coefficiente di determinazione:", R2)
```

```
Coefficiente di determinazione: 0.8905996203945862
```

In [31]:

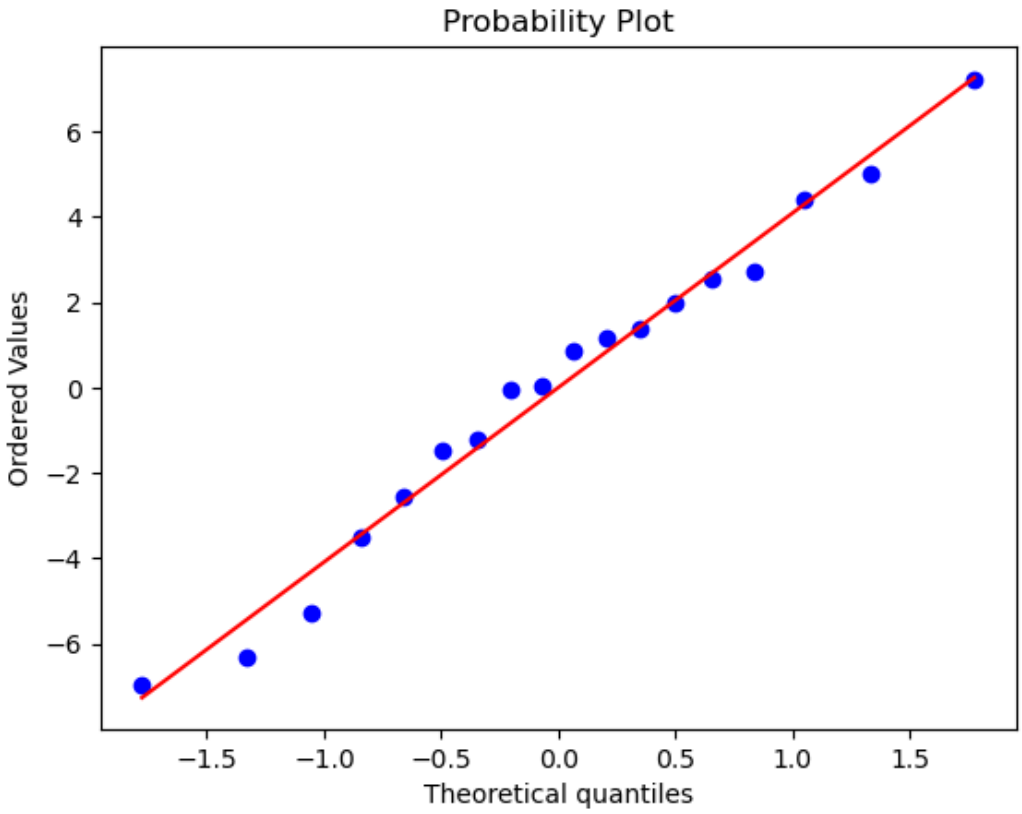
```
#Significatività della regressione utilizzando le formule della rigressione lineare multi
pla
#ipotesi nulla       H_0 : beta_1 = 0
#ipotesi alternativa H_1 : beta_1 != 0

from scipy.stats import t

alpha = 0.05
T = t.ppf(1-alpha/2, n-2)

M = np.linalg.inv(np.dot(X.T,X))
m = M[1][1]
r = y-y_hat
s2 = np.sum(r**2)/(n-2)
T1 = b[1]/np.sqrt(s2*m)

if(np.abs(T1)>=T):
    print("Poiché |T1| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che c'è una di
pendenza tra y e x con un livello di significatività del 95%.")
else:
    print("Poiché |T1| < T, non si hanno elementi sufficienti per rigettare l'ipotesi nul
la. Ma non si afferma nulla.")
```

```
Poiché |T1| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che c'è una dipendenza tr
```

a y e x con un livello di significatività del 95%.

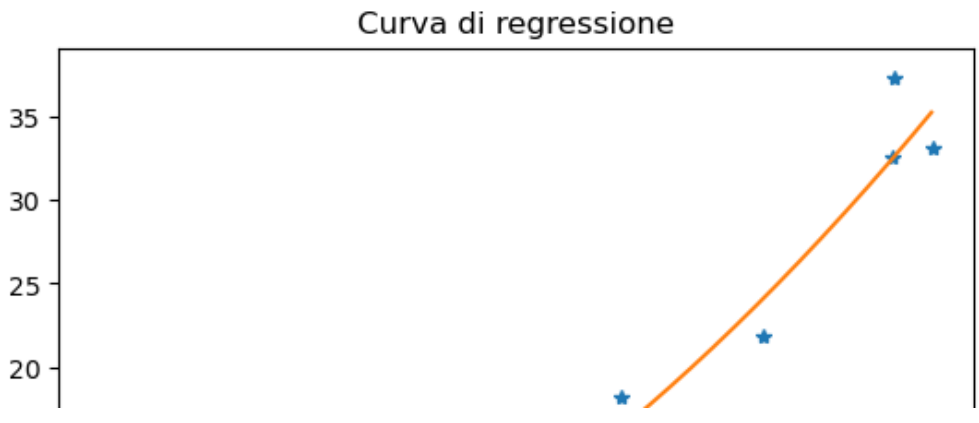
In [32]:

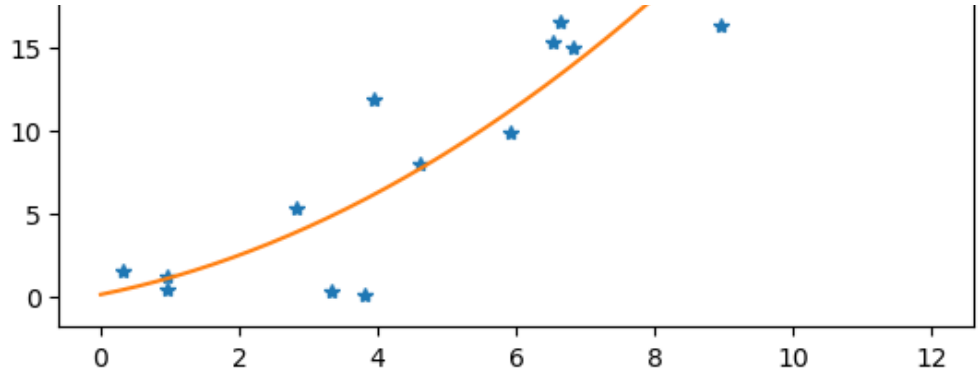
```
#I residui seguono una distribuzione normale?

from scipy.stats import norm, probplot

fig, ax = plt.subplots(1, 1)
probplot(r, dist = norm, plot = ax)
plt.show()
```



In [33]:

```
import matplotlib.pyplot as plt

X2 = np.zeros((n,3))
X2[:,0] = x1
X2[:,1] = x2
X2[:,2] = x2**2
XX2 = np.linalg.pinv(X2)
b2 = np.dot(XX2, y)

xx =  np.linspace(0, 12, 1111111)
yy = b2[0] + b2[1]*xx + b2[2]*xx**2

plt.plot(x,y,'*')
plt.plot(xx, yy)
plt.title("Curva di regressione")
plt.show()
```





In [34]:

```
#Coefficiente di determinazione
y_hat_2 = np.dot(X2, b2)
R2 = np.sum((y_hat_2-y_bar)**2)/np.sum((y-y_bar)**2)
print("Coefficiente di determinazione:", R2)
```

```
Coefficiente di determinazione: 0.9268534244836729
```

In [35]:

```
#Test di indipendenza per x
#H_0 : beta_1 = 0
#H_1 : beta_1 != 0
from scipy.stats import t

alpha = 0.05
T = t.ppf(1-alpha/2, n-2)

M2 = np.linalg.inv(np.dot(X2.T,X2))
m2 = M2[1][1]
m3 = M2[2][2]
r2 = y-y_hat_2
s2_2 = np.sum(r2**2)/(n-2)
T1_2 = b2[1]/np.sqrt(s2_2*m2)
if(np.abs(T1_2)>=T):
    print("Poiché |T1_2| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che c'è una dipendenza tra y e x con un livello di significatività del 95%.")
else:
    print("Poiché |T1_2| < T, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché |T1_2| < T, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.
```

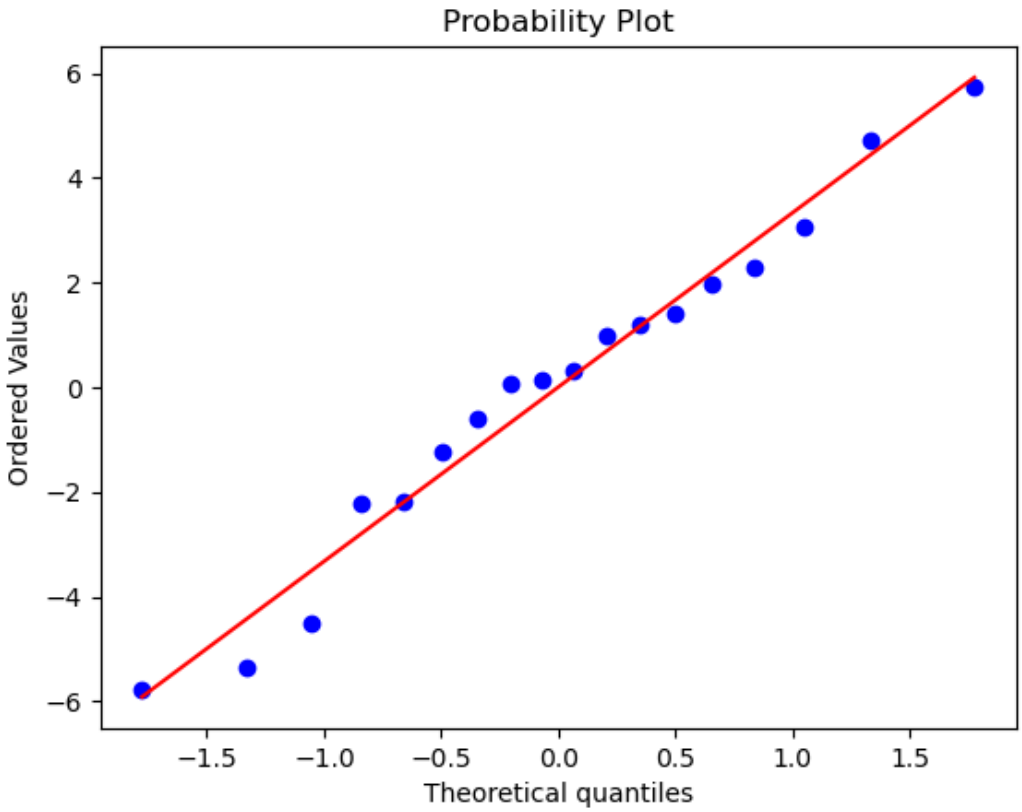
In [37]:

```
#Test di indipendenza per x**2
#H_0 : beta_2 = 0
#H_1 : beta_2 != 0
T2_2 = b2[2]/np.sqrt(s2_2*m3)
if(np.abs(T2_2)>=T):
    print("Poiché |T2_2| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che c'è una dipendenza tra y e x^2 con un livello di significatività del 95%.")
else:
    print("Poiché |T2_2| < T, non si hanno elementi sufficienti per rigettare l'ipotesi nulla. Ma non si afferma nulla.")
```

```
Poiché |T2_2| >= T, si rigetta l'ipotesi nulla. Quindi si afferma che c'è una dipendenza tra y e x^2 con un livello di significatività del 95%.
```

In [38]:

```
from scipy.stats import norm, probplot

fig, ax = plt.subplots(1, 1)
probplot(r2, dist = norm, plot = ax)
plt.show()
```



Osserviamo che il coefficiente di determinazione è maggiore rispetto a quello del modello lineare, suggerendo una migliore adeguatezza del modello quadratico.

8) Numeri pseudo-casuali

Esercizio 1

Scrivere un algoritmo per generare numeri pseudo casuali con distribuzione binomiale B(n, p) con n e p a piacere.

In [13]:

```
import numpy as np

#Numeri pseudo casuali con distribuzione di Bernoulli
p = 0.2
xi = np.random.rand()
if xi < p:
    X = 1
else:
    X = 0
display(X)
```

0

In [15]:

```
N = 100000
#Contiamo il numero di successi e di insuccessi su N tentativi
N_succ = 0
N_insucc = 0
for i in range(N):
    xi = np.random.rand()
    if xi < p:
        N_succ+=1
    else:
        N_insucc+=1

# Probabilità empiriche
```

```
p_succ = N_succ/N
p_insucc = N_insucc/N

print("Probabilità empirica di successo:",p_succ)
print("Probabilità empirica di insuccesso:",p_insucc)

# Probabilità teoriche
print("Probabilità teorica di successo:",p)
print("Probabilità teorica di insuccesso:",1.-p)
```

```
Probabilità empirica di successo: 0.19803
Probabilità empirica di insuccesso: 0.80197
Probabilità teorica di successo: 0.2
Probabilità teorica di insuccesso: 0.8
```

Esercizio 2

Scrivere un algoritmo per generare numeri pseudo casuali con distribuzione multinomiale B(1/4, 1/2, 1/4). Generare 2000 di questi numeri e costruire un istogramma verificando l'accordo con la distribuzione teorica di probabilità.

In [16]:

```
import numpy as np
import matplotlib.pyplot as plt

def rand_multi(p):
    F = np.cumsum(p)
    xi = np.random.rand()
    X = np.nonzero(xi<F)[0][0]+1
    return X

p = np.array([1/4, 1/2, 1/4])
```

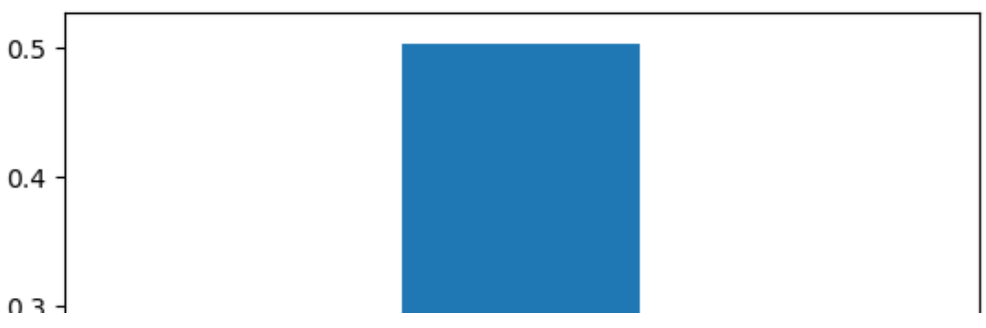
In [17]:

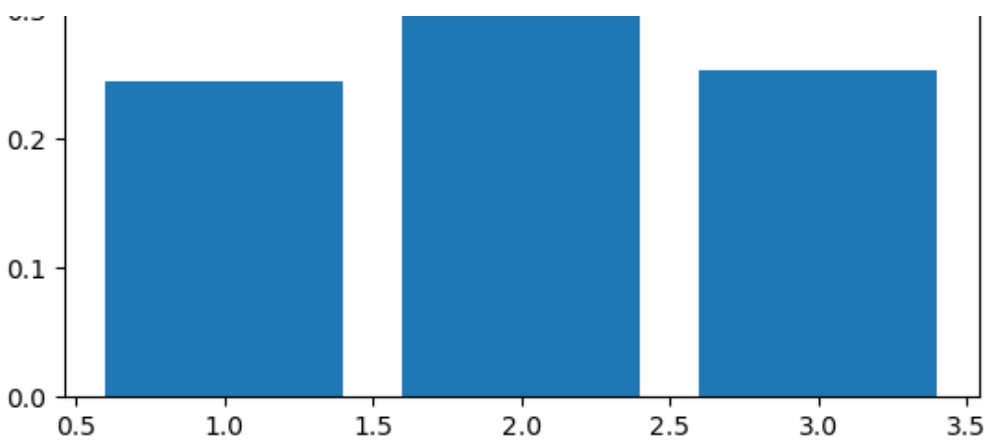
```
N = 2000
X = np.zeros(N)
for i in range(N):
    X[i] = rand_multi(p)

m = p.size
F = np.zeros(m)
for i in range(m):
    F[i] = np.sum(X==i+1)

print("Probabilità empiriche:",F/np.sum(F))
print("Probabilità teoriche:",p)
```

```
Probabilità empiriche: [0.244  0.5025 0.2535]
Probabilità teoriche: [0.25 0.5  0.25]
```

In [18]:

```
# Istogramma
x = np.arange(m)+1
F = F/np.sum(F)
plt.bar(x,F)
plt.show()
```



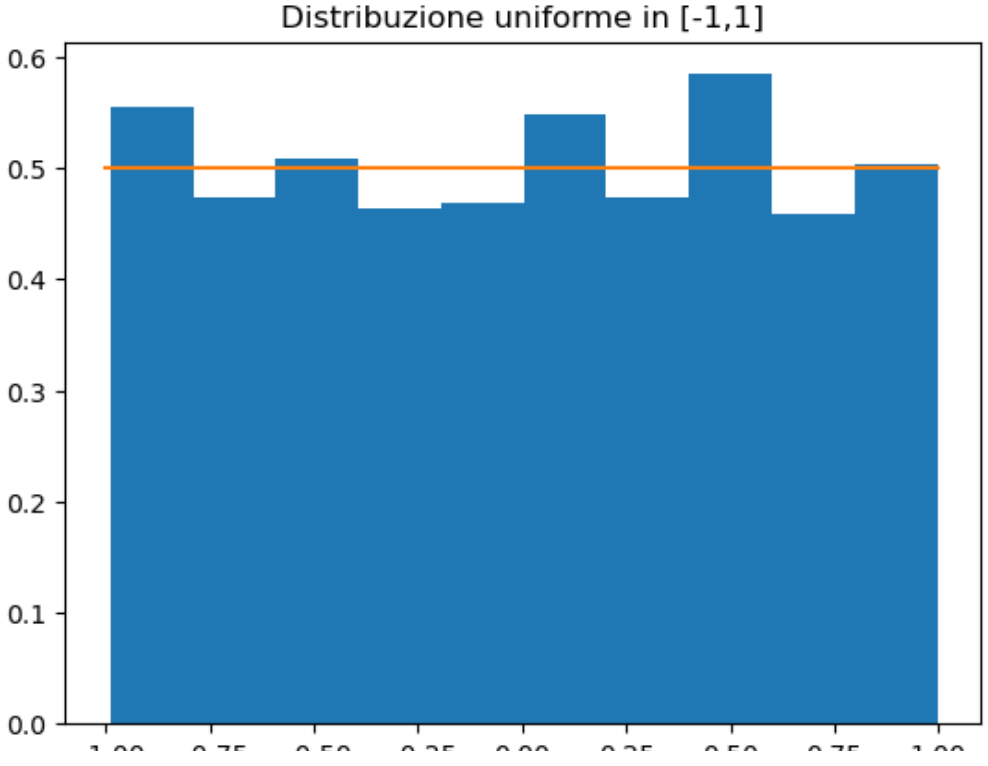


Esercizio 3

Si scrivano delle funzioni di Python per generare numeri random con distribuzione U([–1, 1]), EXP(3), N(1, 2), $\chi 2$ (5).
Si generino N = 1000 di questi numeri e si confrontino i risultati con le distribuzioni teoriche mediante istogramma e mediante grafico quantile-quantile.

In [38]:

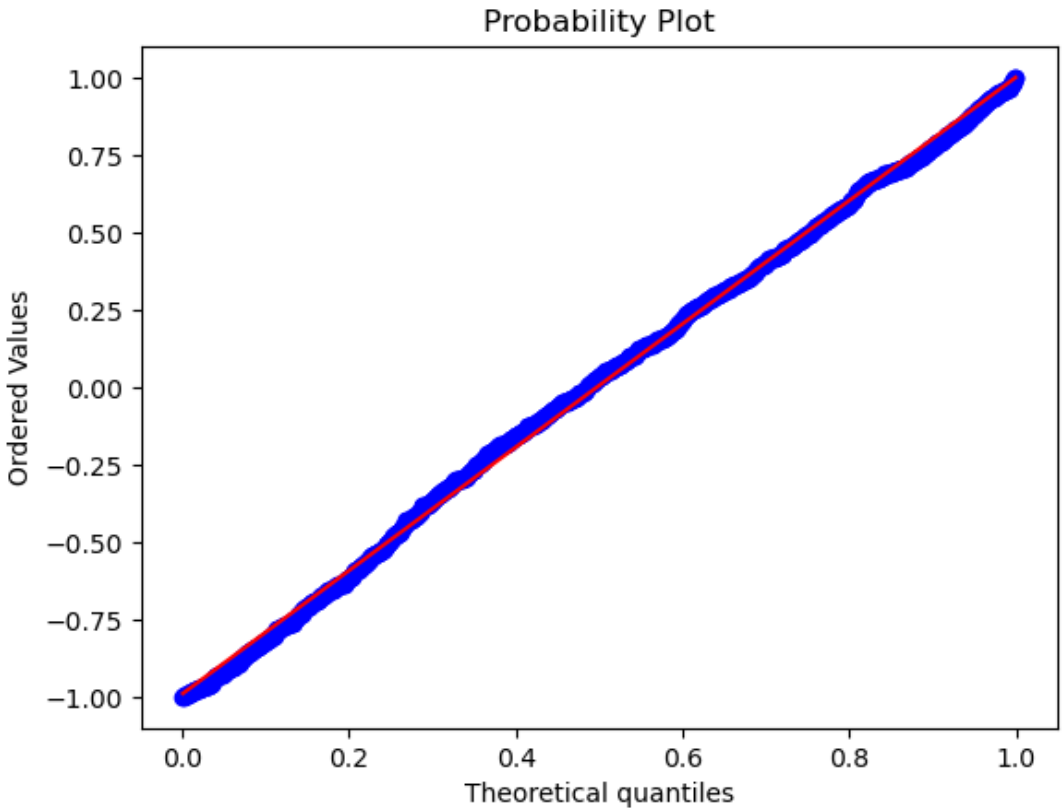
```
#Distribuzione uniforme
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import uniform
from scipy.stats import probplot

a = -1.
b = 1.
N = 1000
X = np.random.rand(N)
Y = a + X*(b-a)

xx = np.linspace(a,b,1000)
yy = uniform.pdf(xx, a, b-a)

fig, ax = plt.subplots(1, 1)
ax.hist(Y, density=True)
plt.plot(xx,yy)
ax.set_title("Distribuzione uniforme in [-1,1]")
plt.show()
```



−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

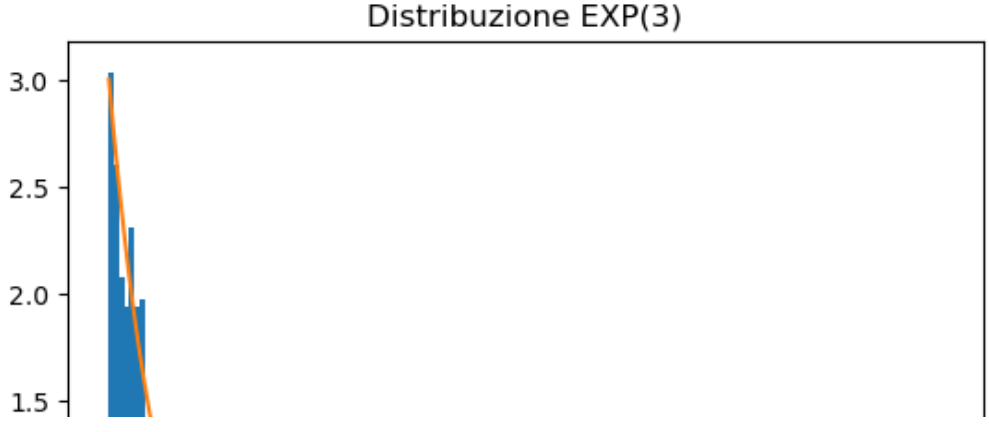
In [21]:

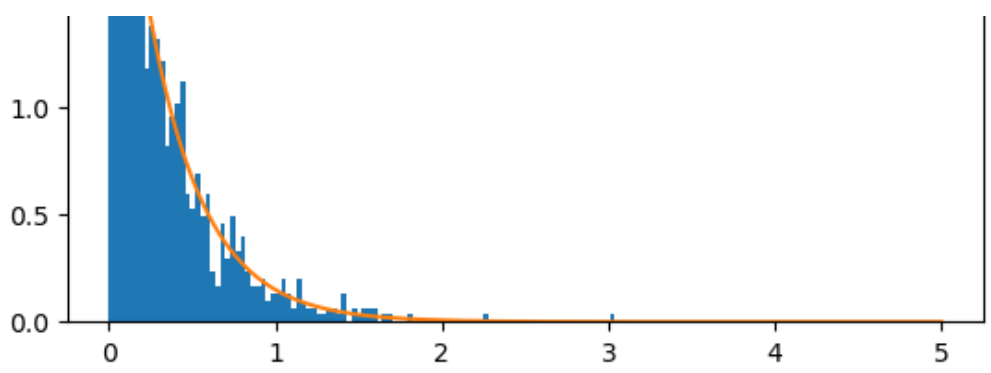
```
fig, ax = plt.subplots(1, 1)
probplot(Y, dist=uniform, plot=ax)
plt.show()
```



In [37]:

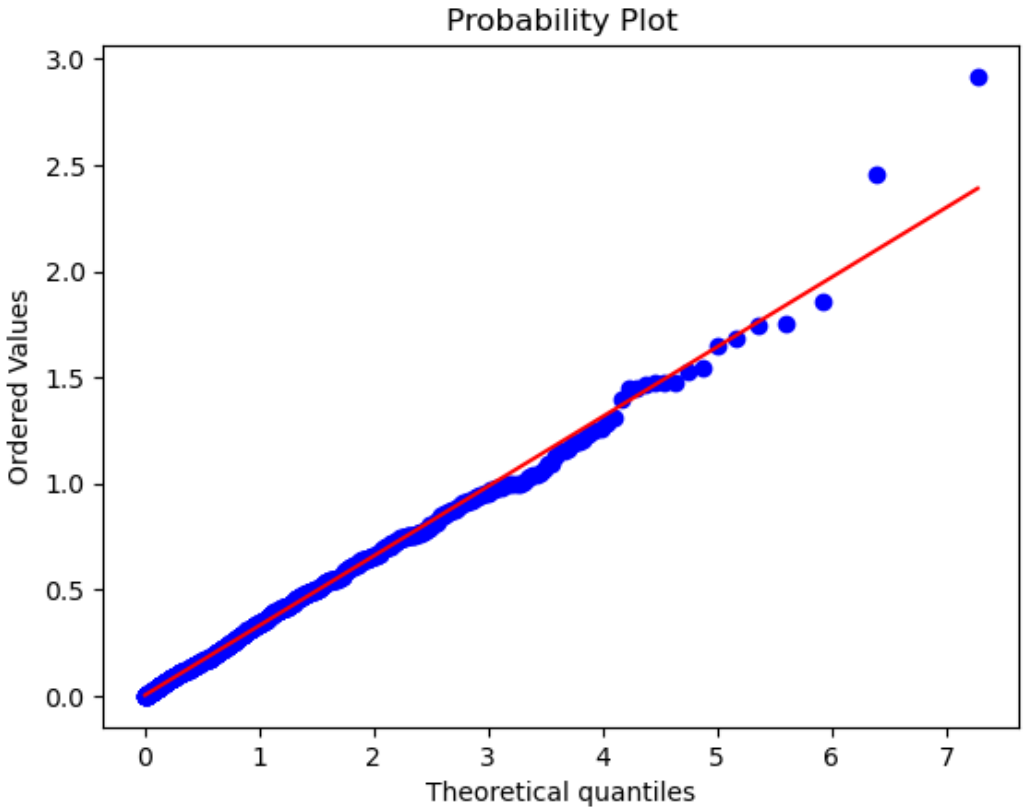
```
#Distribuzione esponenziale
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import expon
from scipy.stats import probplot

lam = 3.
N = 1000
X = np.random.rand(N)
Y = -np.log(X)/lam

xx = np.linspace(0.,5.,1000)
yy = expon.pdf(xx, 0., 1./lam)

fig, ax = plt.subplots(1, 1)
ax.hist(Y, bins=100, density=True)
plt.plot(xx,yy)
ax.set_title("Distribuzione EXP(3)")
plt.show()
```





In [24]:

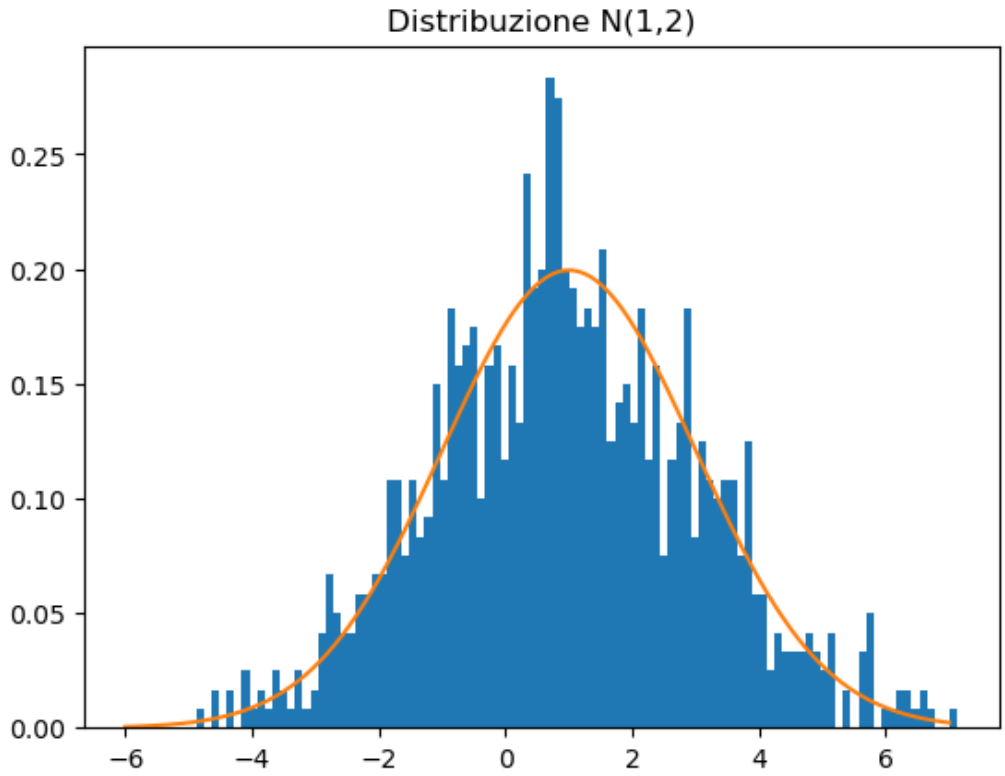
```
fig, ax = plt.subplots(1, 1)
probplot(Y, dist=expon, plot=ax)
plt.show()
```



In [35]:

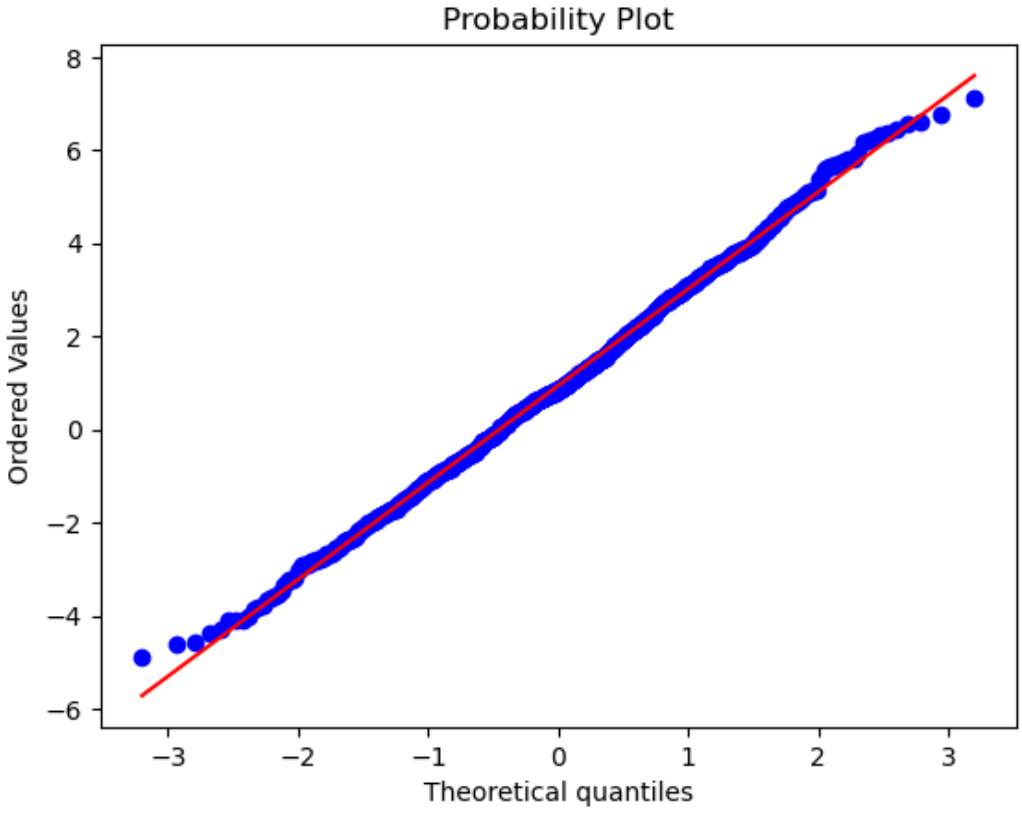
```
#Distribuzione normale
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy.stats import probplot

mu = 1
sig = 2
N = 1000
X = np.random.rand(N)
xi1 = X[0:int(N/2)]
xi2 = X[int(N/2):N]
eta1 = np.sqrt(-2*np.log(xi1))*np.cos(2.*np.pi*xi2)
eta2 = np.sqrt(-2*np.log(xi1))*np.sin(2.*np.pi*xi2)
Y = np.zeros(N)
Y[0:int(N/2)] = mu + sig*eta1
Y[int(N/2):N] = mu + sig*eta2

xx = np.linspace(-6.,7.,1000)
yy = norm.pdf(xx, loc=1, scale=2)

fig, ax = plt.subplots(1, 1)
```

```
ax.hist(Y, bins=100, density=True)
plt.plot(xx,yy)
ax.set_title("Distribuzione N(1,2)")
plt.show()
```



In [36]:

```
fig, ax = plt.subplots(1, 1)
probplot(Y, dist=norm, plot=ax)
plt.show()
```

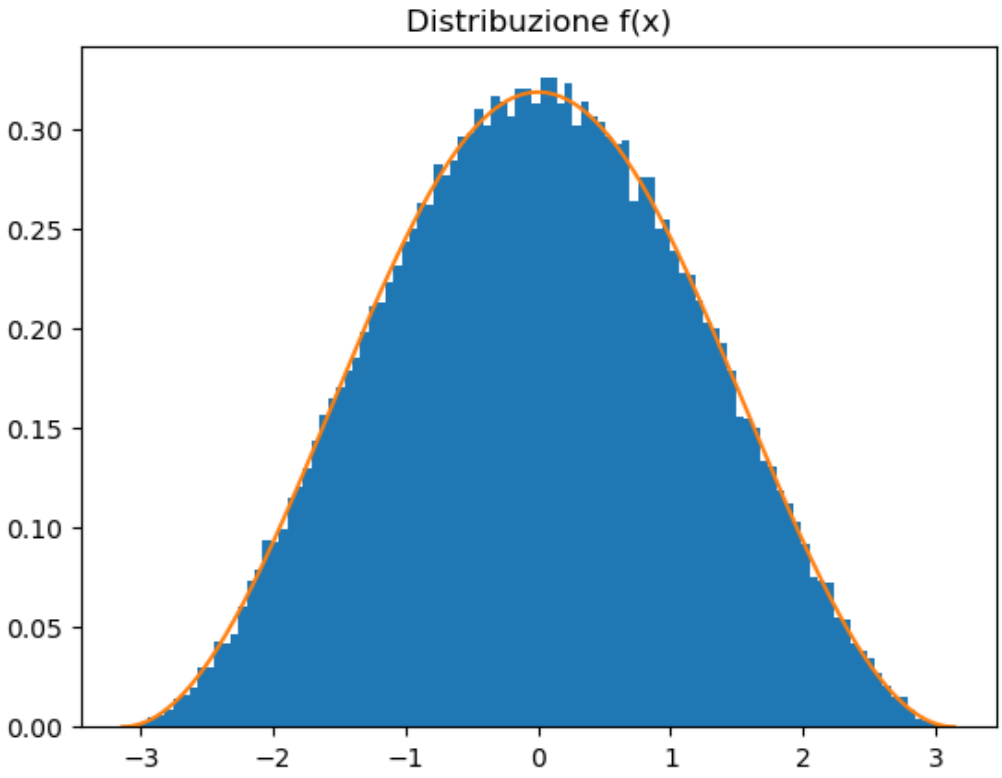


Esercizio 4

Generare numeri pseudo-casuali con distribuzione f(x) = (1 + cos x) / 2π , x ∈ [−π, π]. Eseguire un confronto statistico tra i numeri generati e la distribuzione teorica.

In [39]:

```
import numpy as np
import matplotlib.pyplot as plt

def fun(x):
    y = (1.+np.cos(x))/(2.*np.pi)
    return y

def rigetto(a,b,M):
    while True:
        r1 = np.random.rand()
        r2 = np.random.rand()
        xi = a + r1*(b-a)
        eta = M*r2
        if eta <= fun(xi):
            break
    return xi

a = -np.pi
b = np.pi
M = 1./np.pi

N = 100000
X = np.zeros(N)
for i in range(N):
    X[i] = rigetto(a,b,M)

xx = np.linspace(a,b,1000)
yy = fun(xx)

fig, ax = plt.subplots(1, 1)
ax.hist(X, bins=100, density=True)
plt.plot(xx,yy)
ax.set_title("Distribuzione f(x)")
plt.show()
```



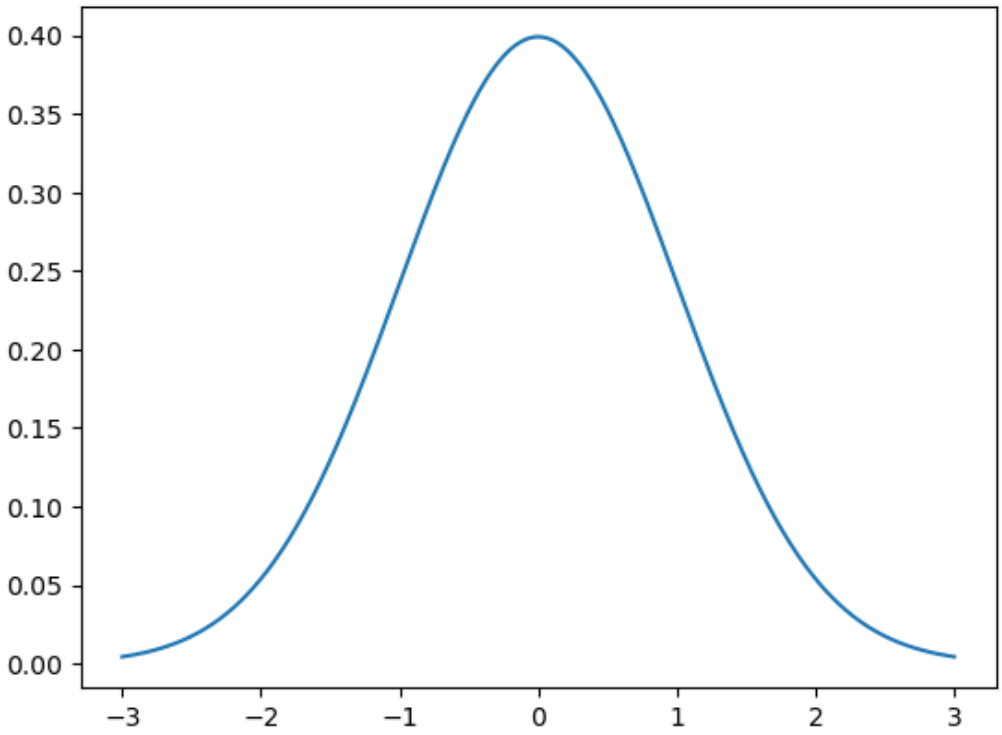
Esercizio 5

Sia X ~ N(0, 1). Si calcoli numericamente con il metodo hit or miss la probabilità p = P(0.5 ≤ x ≤ 2).

In [45]:

```
import numpy as np
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

def fun(x):
    y = np.exp(-x**2./2.)/np.sqrt(2.*np.pi)
    return y

a = 0.5
b = 2.
xx = np.linspace(-3.,3.,1000)
yy = norm.pdf(xx)
plt.plot(xx,yy)
plt.show()
```



In [46]:

```
phi = norm.cdf(b)-norm.cdf(a)
print("Probabilità teorica:",phi)

M = 0.4 #Massimo raggiunto dalla curva
N = 1000000
NS = 0 #Numero di punti che ricadono al di sotto della curva
for i in range(N):
    r1 = np.random.rand()
    r2 = np.random.rand()
    xi = a + r1*(b-a)
    eta = r2*M
    if fun(xi) > eta:
        NS = NS+1
p = NS/N
I = p*M*(b-a)
print("Probabilità con il metodo hit or miss:",I)
```

```
Probabilità teorica: 0.2857874067778077
Probabilità con il metodo hit or miss: 0.286095
```

CATENE DI MARKOV

Sia p sia la seguente matrice

[0.5 0.2 0.3]
[0.1 0.6 0.3]
[0.0 0.0 1.0]

1)Sia Pij la probabilità di transizione per passare dallo stato i allo stato j
2) STATI ASSORBENTI : una colonna che ha solo zeri tranne sulla diagonale,stati assorbenti
3) Gli stati i e j comunicano se Pij>0
4) classe irriducibile : è un insisme di stati in cui è possibile raggiungere da ogni stato
5) se la catena di markov è irriducibile e ha una distribuzione stazionaria, la riga corrispondente sarà un vettore invarianete per P

$$P = \begin{pmatrix} 0 & 1/4 & 3/4 \\ 0 & 1/2 & 1/2 \\ 3/4 & 0 & 1/4 \end{pmatrix}$$

dimostrare che ha un'unica distribuzione stazionaria π e determinarla sia analiticamente che con il metodo Monte Carlo confrontando infine i risultati.

In [3]:

```
# TUTTA LA TEORIA PER  DETERMINARE CHE HA UN'UNICA DISTRIBUZIOEN STAZIONARIA
# 1->2  1->3
# 2->2  2->3
# 3>1   3->3
# 1->2 E 2->3 PROPRIETà TRANSITIVA

#tutti gli stati comunicano fra di loro quindi la classe è irriducibile
#dato che la catena è regolare e inoltre esiste phh > 0 allora la catena è REGOLARE
# dato che p ha tutti gli elementi > 0 - P è regolare
# Per il teorema di Markov ha un unica distribuzione stazionaria
```

In [9]:

```
import numpy as np

X = np.array([[0,1/4,3/4],[0,1/2,1/2],[3/4,0,1/4]])
print(X@X)
```

```
[[0.5625 0.125  0.3125]
 [0.375  0.25   0.375 ]
 [0.1875 0.1875 0.625 ]]
```

In [17]:

```
#METODO ANALATICO

lamb, V = np.linalg.eig(X.T) #mi torna gli autovalori e autovettori
print("autovalori: \n", lamb)
print("autovettori: \n",V)
```

```
autovalori:
 [-0.57569391  1.          0.32569391]
autovettori:
```

```
 [[ 0.78010553 -0.57469577  0.55506939]
 [-0.18130286 -0.28734789 -0.79611302]
 [-0.59880267 -0.76626103  0.24104363]]
```

In [27]:

```
v = V[:,1]
N= np.sum(v)
v/N
#unica distribuzione stazionaria con metodo analitico
```

Out[27]:

```
array([0.35294118, 0.17647059, 0.47058824])
```

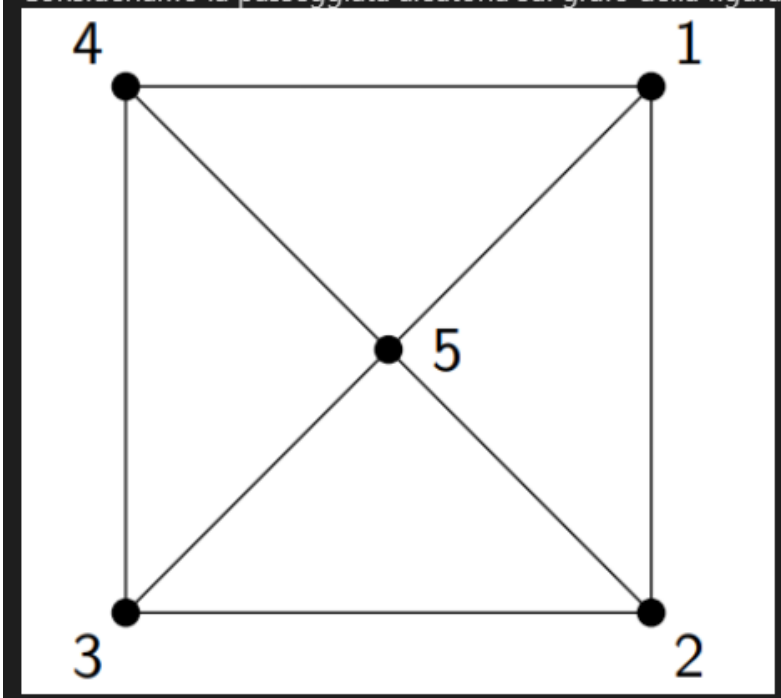
In [106]:

```
#METODO MONTE CARLO

n = 3
F = np.zeros(n)
j = np.random.randint(n)
F[j]= 1


N = 10000
for i in range(N):
    jk = np.random.multinomial(1,X[j,:])
    j = np.nonzero(jk)[0][0]
    F[j] += 1

v= F/N
v
```

Out[106]:

```
array([0.3517, 0.1768, 0.4716])
```

Consideriamo la passeggiata aleatoria sul grafo della figura seguente.



Determinare la matrice di transizione della catena di Markov associata, dimostrare che la catena è regolare e determinare la distribuzione invariante sia analiticamente che con il metodo Monte Carlo.

In [109]:

```
matrix = np.array([[0,1/3,0,1/3,1/3],[1/3,0,1/3,0,1/3],[0,1/3,0,1/3,1/3],[1/3,0,1/3,0,1
/3],[1/4,1/4,1/4,1/4,0]])
print(matrix)
```

```
[[0.         0.33333333 0.         0.33333333 0.33333333]
 [0.33333333 0.         0.33333333 0.         0.33333333]
 [0.         0.33333333 0.         0.33333333 0.33333333]
 [0.33333333 0.         0.33333333 0.         0.33333333]
 [0.25       0.25       0.25       0.25       0.        ]]
```

In [116]:

```
P = np.dot(matrix,matrix)
print(P)
```

```
[[0.30555556 0.08333333 0.30555556 0.08333333 0.22222222]
 [0.08333333 0.30555556 0.08333333 0.30555556 0.22222222]
 [0.30555556 0.08333333 0.30555556 0.08333333 0.22222222]
 [0.08333333 0.30555556 0.08333333 0.30555556 0.22222222]
 [0.16666667 0.16666667 0.16666667 0.16666667 0.33333333]]
```

In [117]:

```
#la catena è regolare dato che tutti i pij > 0
# dato che è regolare e ha un insieme ifnito di stati possiede osolo una distribuzione st
azionaria
```

In [131]:

```
lamb, V = np.linalg.eig(P.T)
print("autovalori: \n",lamb)
print("autovettori: \n", V)
```

```
autovalori:
 [ 1.00000000e+00  4.44444444e-01  1.11111111e-01  3.23558624e-18
 -9.93072636e-17]
autovettori:
 [[ 4.16025147e-01  5.00000000e-01  2.23606798e-01 -2.09073343e-02
  -6.90269127e-01]
 [ 4.16025147e-01 -5.00000000e-01  2.23606798e-01 -7.06797625e-01
  -1.53390131e-01]
 [ 4.16025147e-01  5.00000000e-01  2.23606798e-01  2.09073343e-02
   6.90269127e-01]
 [ 4.16025147e-01 -5.00000000e-01  2.23606798e-01  7.06797625e-01
   1.53390131e-01]
 [ 5.54700196e-01  1.54741386e-17 -8.94427191e-01  1.56562997e-16
   1.39587131e-16]]
```

In [148]:

```
vi = V[:,0]
N = np.sum(vi)

vi/N
```

Out[148]:

```
array([0.1875, 0.1875, 0.1875, 0.1875, 0.25  ])
```

In [146]:

```
n = 5
F = np.zeros(n)
j = np.random.randint(0,n)
F[j]= 1


N = 10000
for i in range(N):
    jk = np.random.multinomial(1,P[j])
    j = np.nonzero(jk)[0][0]
    F[j] += 1

v= F/N
v
```

Out[146]:

```
array([0.1823, 0.193 , 0.1777, 0.1933, 0.2538])
```