LVquicksort

Il codice lo ho fatto in python

Definisco la funzione LV QuickSort

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

# LVQuickSort
def quicksort(arr, count=0):
    # e una funzione ricorsiva qindi prima cosa faccio il caso base
    if len(arr) <= 1:
        return arr, count

    # scelgo un pivot casuale
    pivot = arr[random.randint(0, len(arr) - 1)]
    less_than_pivot, equal_to_pivot, greater_than_pivot = [], [], [] # Inizializzo le liste

    #faccio i confronti
    for x in arr:
        count += 1  # Conta ogni confronto
        if x < pivot:
            less_than_pivot.append(x)
        elif x == pivot:
            equal_to_pivot.append(x)
        else:
            greater_than_pivot.append(x)

    # chiamo ricorsivamente la funzione
    left_sorted, left_count = quicksort(less_than_pivot, count)
    right_sorted, right_count = quicksort(greater_than_pivot, left_count)
    sorted_arr = left_sorted + equal_to_pivot + right_sorted

    # Ritorno l'array ordinato e il numero di confronti
    return sorted_arr, right_count
```

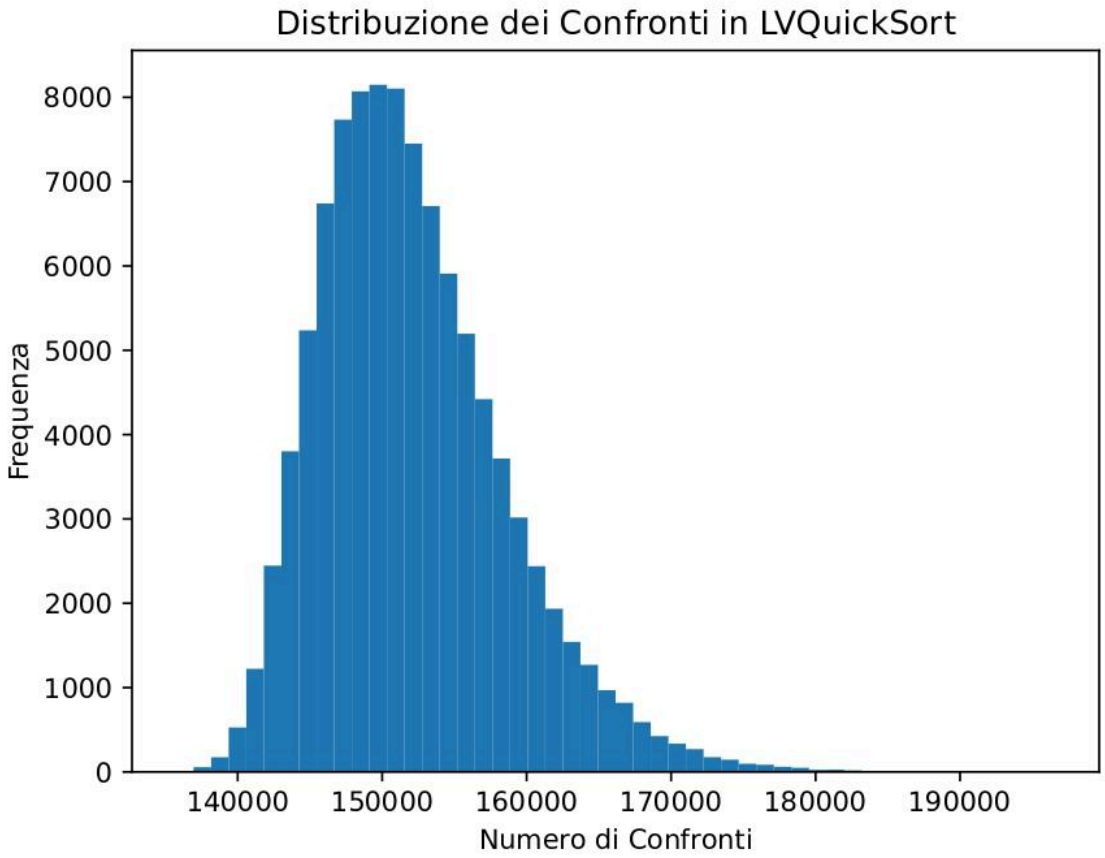
Faccio le 10^5 run e salvo il numero di confronti in una lista

```
# creo l'array S di 10000 elementi casuali
S = [random.randint(1, 10000) for _ in range(10000)]

# fai 100000 run di quicksort
R = 100000
counts = []
for i in range(R):
    # ogni creo una nuova lista casuale
    # e salvo il numero di confronti (_ indica che la lista ordinata non mi serve)
    _, count = quicksort(list(S))
    print(f"Number of comparisons: {count} for run {i + 1}")
    counts.append(count)
```

Faccio analisi statistiche sui risultati ottenuti e creo il istogramma

```
#Calcolo di μ^ e σ^
media = np.mean(counts)
deviazione = np.std(counts, ddof=1)

# creo l'istogramma
plt.hist(counts, bins=50)
plt.title('Distribuzione dei Confronti in LVQuickSort')
plt.xlabel('Numero di Confronti')
plt.ylabel('Frequenza')
plt.savefig('LVQuickSort_Distribuzione.pdf')
plt.show()


# Calcolo dei limiti superiori per le probabilità
# Utilizzando le disuguaglianze di Chebyshev per 2μ^ e 3μ^ confronti
prob_2mu = 1 / 4  # Disuguaglianza di Chebyshev per 2μ
prob_3mu = 1 / 9  # Disuguaglianza di Chebyshev per 3μ

#Confronto delle frequenze empiriche
freq_2mu = len([c for c in counts if c > 2 * media]) / R
freq_3mu = len([c for c in counts if c > 3 * media]) / R

print(f"μ^: {media}, σ^: {deviazione}")
print(f"Limite probabilità per 2μ^ confronti: {prob_2mu}, Frequenza empirica: {freq_2mu}")
print(f"Limite probabilità per 3μ^ confronti: {prob_3mu}, Frequenza empirica: {freq_3mu}")
```

Questo e l'istogramma risultante risultante come possiamo notare ha una distribuzione normalizzata come previsto visto che il caso peggiore diventa meno probabile e si tende al caso medio



μˆ: 152103.92773, σˆ: 6444.289233790204
Limite probabilità per 2μˆ confronti: 0.25, Frequenza empirica: 0.0
Limite probabilità per 3μˆ confronti: 0.1111111111111111, Frequenza empirica: 0.0

Frequenza empirica e 0 perche LVQuickSort e un algoritmo efficiente e non ci sono casi 2 o 3 volte la media.