LAB 2.1

Per attuare l'implementazione dell'algoritmo LVQuickSort, è stato impiegato per la rioridinazione di una sequenza S costituita da 10^4 numeri. La sequenza è stata previamente generata, e durante ogni singola esecuzione, è stato registrato il numero di confronti effettuati. Successivamente, mediante un programma elaborato in linguaggio Python(sfruttando il multithreading), sono stati condotti 10^5 run del suddetto algoritmo. Al fine di valutare le prestazioni, sono stati calcolati il valore medio e la deviazione standard del numero di confronti effettuati nei 10^5 run.

```
import random
from math import sqrt, pow
from concurrent.futures import ThreadPoolExecutor
from concurrent.futures import as_completed

def markov(v):
    print(f"Probability of LVQuickSort performing at least {v} times the expected comparisons is
<= {1/v}")

def chebyshev(expected_value, variance, v):
    print(f"Probability of LVQuicksort performing at least {v} times the expected comparisons is
<= {variance / (pow(v-1, 2) * pow(expected_value, 2))}")

def mean_std_dev(run):
    summ=0
    ris = []
    with ThreadPoolExecutor(max_workers=1000) as executor:
        results = [executor.submit(LVQuickSort) for _ in range(run)]
    for result in as_completed(results):
        try:
            print(result.result(), file=fout)
            Tempcomp = result.result()
            ris.append(Tempcomp)
            summ += Tempcomp
        except Exception as e:
            print(f"Error during processing: {e}")
    mean = summ / run
    #variance = summation / run-1
    variance = sum([pow(x-mean,2) for x in ris])  / (run-1)
    print(f"\nNumber of mean comparisons made: {mean}")
    print(f"Standard deviation is: {sqrt(variance)}")
    print("\nProbabilities calculated with the inequality derived from Chebyshev's inequality:")
    chebyshev(mean, variance, 2)
    chebyshev(mean, variance, 3)
    print("\nProbabilities calculated with the inequality derived from Markov's inequality:")
    markov(2)
    markov(3)
    print("\n")
```

```python
def create_seq():
    seq = [random.randint(0, 100000) for _ in range(10000)]
    return seq

def swap(v, i, j):
    tmp = v[j]
    v[j] = v[i]
    v[i] = tmp

def quick_sort(v, start, end):
    temp_comp = 0
    pivot_index = start + random.randint(0, end - start)
    swap(v, pivot_index, start)
    i = start + 1
    for j in range(start+1, end+1):
        temp_comp  += 1
        if v[j] < v[start]:
            swap(v, i, j)
            i += 1
    swap(v, start, i-1)
    return i-1, temp_comp

def lvqs(v, start, end):
  comp=0
  if start < end:
    pivot_index, comp = quick_sort(v, start, end)
    comp += lvqs(v, start, pivot_index-1)
    comp += lvqs(v, pivot_index+1, end)
  return comp

def LVQuickSort():
    v = create_seq()
    LVcomp = lvqs(v, 0, len(v)-1)
    return LVcomp

if __name__ == '__main__':
    with open("output.txt", "w") as fout:
        mean_std_dev(100000)
    fout.close()
```

Come può essere dedotto dal programma, esso è in grado di registrare il numero di confronti effettuati in ogni esecuzione all'interno di un file di testo denominato "output.txt". Questo file sarà successivamente utilizzato da uno script in linguaggio Python come input per la generazione del grafico a 50 bin richiesto. In aggiunta, il software si impegna anche nella stima della probabilità che l'algoritmo LVQuickSort abbia eseguito il doppio e il triplo del valore atteso del numero di confronti, basandosi sulle due disuguaglianze illustrate nelle diapositive.

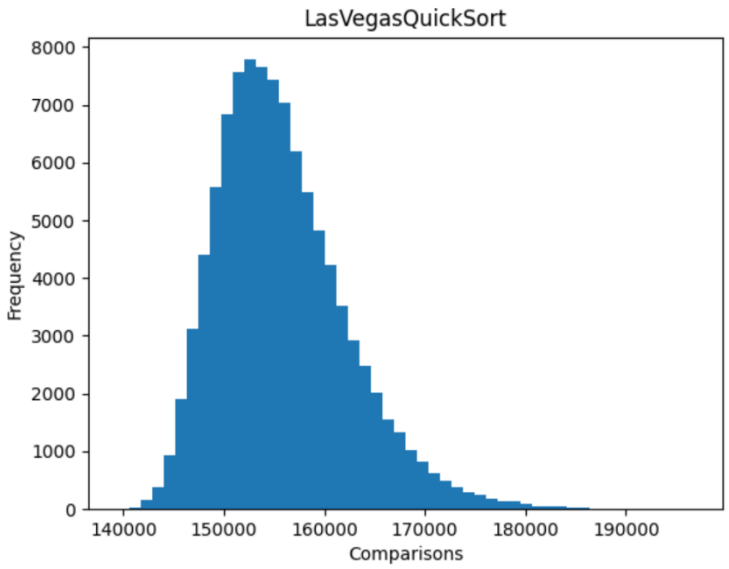
Output del programma :

```
Number of mean comparisons made: 155782.78475
Standard deviation is: 6480.029216738421

Probabilities calculated with the inequality derived from Chebyshev's inequality:
Probability of LVQuicksort performing at least 2 times the expected comparisons is <= 0.0017302744788073405
Probability of LVQuicksort performing at least 3 times the expected comparisons is <= 0.0004325686197018351 3

Probabilities calculated with the inequality derived from Markov's inequality:
Probability of LVQuickSort performing at least 2 times the expected comparisons is <= 0.5
Probability of LVQuickSort performing at least 3 times the expected comparisons is <= 0.3333333333333333
```

Mentre il programma utilizzato per creare il plot a partire dal file output.txt è il seguente:

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

# Matplotlib configuration
plt.title("LasVegasQuickSort")
plt.xlabel("Comparisons")
plt.ylabel("Frequency")

# Reading data
with open("output.txt") as f:
  comparisons = [float(line) for line in f]
comparisons = np.array(comparisons)

# Calculating range and bin width
bin_max = np.max(comparisons)
bin_min = np.min(comparisons)
bin_width = (bin_max - bin_min) / 10

# Creating histogram
counts, bins, patches = plt.hist(comparisons, bins=np.arange(bin_min, bin_max +
bin_width, bin_width), density=True)

# Saving the image
plt.savefig("histo.png")
print("The histogram has been created successfully.")
```

Il suo output:



L'attento esame del grafico rivela chiaramente che la frequenza con cui si verificano il doppio e il triplo del valore atteso del numero di confronti è costantemente pari a zero. In alcuni casi, il numero di confronti supera leggermente i 180,000, ma mai oltre i 190,000. Considerando il valore atteso/medio di 155782, come indicato nell'output del programma, si osserva che il doppio di questo valore, tanto meno il triplo, non viene mai raggiunto.

Questo risultato è in sintonia con le probabilità ottenute applicando la disuguaglianza (7). L'output del programma attesta che la probabilità di superare il doppio del valore atteso dei confronti è ≤ 0.001730, mentre la probabilità di superare il triplo è ≤ 0.0004325. Entrambe le probabilità sono notevolmente basse, giustificando quindi l'osservazione nel grafico che tali eventi non si verificano mai.

Per quanto concerne la disuguaglianza (6), essa suggerisce che il doppio del valore atteso dei confronti si verificherà con una probabilità inferiore o uguale a 0.5. Sebbene tale stima sia formalmente accurata (0 ≤ 0.5), non fornisce una valutazione significativa. D'altra parte, la probabilità stimata che il triplo o più del valore atteso dei confronti si verifichi è massimo 1/3, secondo quanto indicato.