Quantum Circuit Simulator

Generazione di numeri e stringhe casuali basati su un vettore di stato complesso

Valerio Recano

Sommario

[QCSGenerator 2](#_Toc167807743)

[**Costruttore** 2](#_Toc167807744)

[**Metodi** 2](#_Toc167807745)

[**GenerateRandom[PUBLIC]** 2](#_Toc167807746)

[**WriteStringToFile[PUBLIC]** 2](#_Toc167807747)

[**GenerateString[PUBLIC]** 3](#_Toc167807748)

[**GenerateStrings[PUBLIC]** 3](#_Toc167807749)

[**ApplyMatrix[PRIVATE]** 3](#_Toc167807750)

[**ConjugateTranspose[PRIVATE]** 3](#_Toc167807751)

[**SampleFromDistribution[PRIVATE]** 3](#_Toc167807752)

[Flow Chart 5](#_Toc167807753)

# **QCSGenerator**

Porting della libreria Python QCSGenerator per la generazione di numeri e stringhe casuali basati su calcoli della meccanica quantistica (rif. <https://www.linkedin.com/posts/ivan-d-4ba2b664_una-nuova-libreria-per-la-generazione-di-activity-7200889180221014016-w4no?utm_source=share&utm_medium=member_desktop> )

Autore della libreria Python (Ivan Di Giacomo profilo alla url: [www.linkedin.com/in/ivan-d-4ba2b664](http://www.linkedin.com/in/ivan-d-4ba2b664) )

La classe **QCSGenerator** in C# utilizza metodi per generare numeri e stringhe casuali basati su un vettore di stato complesso e alcune trasformazioni matematiche. Ecco una spiegazione dettagliata dei vari metodi all'interno della classe:

## **Costruttore**

**Public QCSGenerator(int? seed = null)**

Il costruttore inizializza l'oggetto **QCSGenerator**. Se non viene fornito un seme (**seed**), viene utilizzato il tempo corrente in millisecondi per generarlo. Il seme viene usato per selezionare una posizione iniziale nel vettore di stato complesso.

* **Parametri**: **int? seed** (facoltativo)
* **Funzionalità**:
  + Se il seme non è fornito, viene calcolato utilizzando il tempo corrente.
  + Il seme viene poi ridotto modulo **lengthOfAllCharacters** per assicurarsi che rientri nei limiti dell'array **state**.
  + Inizializza il vettore **state** con zeri e imposta la posizione indicata dal seme a **1 + 0i**.

## **Metodi**

### **GenerateRandom[PUBLIC]**

**public int GenerateRandom()**

Questo metodo genera un numero casuale utilizzando la trasformazione di Hadamard su un vettore di stato complesso.

* **Funzionalità**:
  + Definisce una matrice di Hadamard.
  + Applica la matrice di Hadamard a coppie di elementi nel vettore di stato.
  + Calcola le probabilità di misura come il quadrato della magnitudine dei numeri complessi.
  + Seleziona un valore casuale basato su queste probabilità.
  + Applica la trasposta coniugata della matrice di Hadamard per invertire la trasformazione.
  + Ritorna il valore casuale generato.

### **WriteStringToFile[PUBLIC]**

**public void WriteStringToFile(string str)**

Questo metodo scrive una stringa in un file.

* **Parametri**: **string str** (la stringa da scrivere)
* **Funzionalità**:
  + Apre il file **strings.txt** in modalità append.
  + Scrive la stringa seguita da una nuova linea nel file.

### **GenerateString[PUBLIC]**

**public string GenerateString(int length = 30)**

Genera una stringa casuale di una lunghezza specificata.

* **Parametri**: **int length** (la lunghezza desiderata della stringa, predefinito a 30)
* **Funzionalità**:
  + Se la lunghezza è inferiore a 20, viene impostata a 20.
  + Crea un array di caratteri casuali della lunghezza specificata.
  + Utilizza un generatore di numeri casuali per selezionare caratteri dall'insieme **allCharacters**.
  + Ritorna la stringa formata dai caratteri selezionati.

### **GenerateStrings[PUBLIC]**

**public void GenerateStrings(int count = 50000)**

Genera un numero specificato di stringhe casuali di lunghezza variabile tra 25 e 30 caratteri e le scrive in un file.

* **Parametri**: **int count** (numero di stringhe da generare, predefinito a 50000)
* **Funzionalità**:
  + Utilizza un ciclo per generare il numero specificato di stringhe.
  + Per ogni stringa generata, chiama **WriteStringToFile** per scriverla nel file.

### **ApplyMatrix[PRIVATE]**

**private Complex[] ApplyMatrix(Complex[,] matrix, Complex[] vector)**

Applica una matrice a un vettore di numeri complessi.

* **Parametri**: **Complex[,] matrix** (la matrice da applicare), **Complex[] vector** (il vettore di stato)
* **Funzionalità**:
  + Moltiplica la matrice per il vettore e ritorna il vettore risultante.

### **ConjugateTranspose[PRIVATE]**

**private Complex[,] ConjugateTranspose(Complex[,] matrix)**

Calcola la trasposta coniugata di una matrice.

* **Parametri**: **Complex[,] matrix** (la matrice di input)
* **Funzionalità**:
  + Calcola e ritorna la trasposta coniugata della matrice.

### **SampleFromDistribution[PRIVATE]**

**private int SampleFromDistribution(double[] probabilities)**

Seleziona un indice basato su una distribuzione di probabilità.

* **Parametri**: **double[] probabilities** (un array di probabilità)
* **Funzionalità**:
  + Calcola la somma delle probabilità.
  + Genera un numero casuale tra 0 e la somma delle probabilità.
  + Scorre le probabilità cumulative fino a trovare l'indice corrispondente al numero casuale generato.
  + Ritorna l'indice selezionato.

# Flow Chart

Immagine che contiene testo, diagramma, documento, schermata

Descrizione generata automaticamente