Classic bit vs qubit

no-cloning theorem

Mario Cuomo

An article for Microsoft Q#Advent Calendar 2021

09.12.2021

CONTENUTO

1 ABSTRACT

Quando si parla di computer quantici erroneamente si associa tale tecnologia all'età moderna, sicuramente dopo gli anni 2000. In realtà gli studi su questo nuovo modello di calcolo nascono ancora prima. Si pensi per esempio che il ***no-cloning Theorem***, oggetto finale di questo articolo, è stato dimostrato negli anni '80 (anche se si ha una dimostrazione pratica del fenomeno 10 anni prima, nel 1970).
Questo perché?

Il modello di calcolo dei computer quantici si basa sulla meccanica quantistica che, come sappiamo, nasce nel 1900 con i primi studi di Max Planck e, successivamente, di Ewinr Schrödinger (vi siete mai chiesti perché molto spesso compaiono riferimenti a gatti quando si parla di quantum computing?) e Paul Dirac.
L'avvento di questo nuovo modello di calcolo pone una forte sfida nei confronti della ***strong Church Turing Thesis***. La *Church Turing Thesis* afferma che se esiste un problema umanamente calcolabile allora esiste anche una macchina probabilitstica di Turing che riesce a risolvere tale problema in maniera efficiente (o, detto in altri termini, la classe di funzioni calcolabili coincide con quella delle funzioni calcolabili da una macchina di Turing). L'aggettivo *strong* indica l'impossibilità dell'esistenza di un modello più espressivo delle TM.
Un esempio classico in letteratura di questa sfida è l'***Algoritmo di Shor***.

Dato un intero N esprimibile come il prodotto tra due numeri primi p e q, è possibile ottenere p e q in un tempo *polinomiale* rispetto a N.
I classici algoritmi ritornano tali valori in un tempo *super polinomiale* rispetto a N.

La teoria della computabilità si basa sulla *Church Turing Thesis*. Tale teoria purtroppo non è completamente espressiva in quanto si basa sulla fisica classica per modellare l'universo ma, come detto, l'universo è ***quantum physical***.
Possiamo affermare quindi che un computer quantico è una macchina realizzata **appositamente** per svolgere tutti quei compiti che una macchina classica non riesce a fare (per sua natura, non perché *'non è abbastanza potente'*). Allo stesso modo, però, comipiti svolti efficientemente da computer classici possono essere realizzati in maniera non efficiente da computer quantici: ne deriva che il quantum computing è uno strumento valido solo per alcune classi di problemi.
Anche se non non ancora disponibili, si sta lavorando per computer quantici general-purpose.

2 INTRODUZIONE ALLA SUPER-POSITION

L'unità di calcolo fondamentale per i computer quantici è il ***qubit***.
Per comprendere meglio cosa un *qubit* sia possiamo pensare alla fisica di un atomo, per esempio un *elettrone*.
Generalmente si afferma che un *elettrone* ruota su un'orbita intorno all'atomo. In base alla sua energia, ruota su un'orbita differente: gli stati di una particella sono quindi *quantizzati* in un insieme discreto.
La meccanica quantistica invece afferma che tali parametri, in un certo istante di tempo, non sono fissi ma **variano**.
Un *elettrone* quindi può trovarsi su più orbite nello stesso momento.
La proprietà appena introdotta è la ***super-position***.

Fin da subito è possibile proporre un paragone con il calcolo classico che utilizza come elemento di informazione il *bit*. Un *bit* è una cifra binaria che può assumere solamente due valori possibili: 0 oppure 1.
Il *qubit*, invece, può assumere ***simultaneamente*** entrambi gli stati oppure gli stati intermedi tra i due estremi.
Per *simulataneamente* si intende l'impossibilità di definire, in un determinato istante e con certezza, il valore del qubit. È possibile però definire la *probabilità* con la quale il qubit si trovi in uno stato piuttosto che in un altro.

Formalmente indichiamo la ***super-position*** come segue

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle , \alpha_0, \alpha_1 \in C$$

Oltre alla *super-position*, un altro assioma fondamentale del quantum computing è il ***decadimento dopo l'osservazione***.
Tornando all'esempio dell'elettrone che ruota intorno all'atomo, se proviamo a misurarlo e chiederci su quale orbita sia, esso perde la proprietà di *super-position* e collassa su uno dei due stati noti.
La probabilità con la quale collassa su un valore piuttosto che su un altro dipende da i due parametri α_0 e α_1. In particolare si ha che il qubit, dopo l'osservazione, collassa a 0 con probabilità $|\alpha_0|^2$, mentre collassa a 1 con probabilità $|\alpha_1|^2$.
Ne deriva che deve essere verificato il vincolo di normalizzazione per la *super-position*.

$$|\alpha_0|^2 + |\alpha_1|^2 = 1$$

La *super-position* sopra descritta utilizza la *ket-notation* ideata da Dirac per rappresentare vettori colonna.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} e \, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Il qubit $|\psi\rangle$ si può rappresentare quindi anche come

$$|\psi\rangle = \alpha_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

3 MISURAZIONE DI UN QUBIT - Q#

È possibile utilizzare il linguaggio Q# per sviluppare applicazioni e algoritmi che sfruttano il calcolo quantico. L'idea di base è quella di avere due sistemi separati: il primo è quello del simulatore quantico che esegue il codice Q#, il secondo è quello dell'applicazione (per esempio Python, come negli esempi seguenti) che sfrutta i risultati ottenuti dalla computazione.
Qui è presente la documentazione di come configurare un ambiente Q#.

https://docs.microsoft.com/it-it/azure/quantum/install-overview-qdk

Un primo esperimento che è possibile effettuare riguarda la misurazione di un qubit che ha esattamente la stessa probabilità di trovarsi in uno dei due stati base.
Un qubit così descritto è il seguente:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\end{bmatrix}$$

La probabilità che $|\psi\rangle$ si trovi in 0 è $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$.
La probabilità che $|\psi\rangle$ si trovi in 1 è $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$.

```
import qsharp
from Qrng import SampleQuantumRandomNumberGenerator

i=0
while i<10:
    print(SampleQuantumRandomNumberGenerator.simulate())
    i=i+1
```

```
namespace Qrng {
    open Microsoft.Quantum.Intrinsic;

    operation SampleQuantumRandomNumberGenerator() :
        Result {
        use q = Qubit(); // Allocate a qubit.
        H(q);            // Put the qubit to
            superposition. It now has a 50% chance of
            being 0 or 1.
        let r = M(q);    // Measure the qubit value.
        Reset(q);
        return r;
    }
}
```

Nell'esmpio proposto è generato un qubit. Il costruttore Qubit() ritorna un qubit che si trova nello stato $|\psi\rangle = |0\rangle$.
Se misuriamo il qubit in questa situazione esso ritornerà sempre il valore 0.

Per porre il qubit in uno stato intermedio equiprobabile tra $|0\rangle$ e $|1\rangle$ si applica l'operatore di *Hadamard* H, descritto nei prossimi paragrafi.
Una possibile esecuzione dello script potrebbe produrre l'output:

0 1 1 1 0 0 1 1 0 0

Se non si fosse stato applicato l'operatore H, l'output sarebbe stato:

0 0 0 0 0 0 0 0 0 0

4 OPERATORI SU UN QUBIT

Così come avviene con i bit classici, è possibile definire degli operatori fondamentali che operano sui qubit. In questo capitolo il focus è relativo agli operatori che lavorano sul singolo qubit.
Applicare un operatore a un qubit significa moltiplicare per una matrice U il qubit di partenza. La matrice U deve essere una *trasformazione unitaria*.
Un operatore è *unitario* quando il prodotto tra esso e l'*operatore aggiunto* restituisce la matrice *identità*.

$$UU^* = I$$

Si hanno 4 importanti operatori a singolo qubit, gli *operatori di Pauli*.

- operatore $\boldsymbol{X}$, *bit flip*
  L'operatore X, spesso in letteratura definito come σ_X, è descritto dalla matrice quadrata:
  $$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
  Come suggerisce il nome, il risultato finale dopo l'applicazione dell'operatore è quello di scambiare tra loro i valor di ampiezza α_0 e α_1.

  $$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |0\rangle$$
  $$X |\psi\rangle = \alpha_1 |0\rangle + \alpha_0 |0\rangle$$

- operatore $\boldsymbol{Z}$, *phase flip*
  L'operatore Z, spesso in letteratura definito come σ_Z, è descritto dalla matrice quadrata:
  $$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$
  Come suggerisce il nome, il risultato finale dopo l'applicazione dell'operatore è quello di invertire la fase del qubit.

  $$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |0\rangle$$
  $$Z |\psi\rangle = \alpha_0 |0\rangle - \alpha_1 |1\rangle$$

- operatore $\boldsymbol{Y}$, $Y = XZ$
  L'operatore Y può essere ricavato a partire dagli operatori X e Z come segue:
  $$Y = XZ = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- operatore $\boldsymbol{I}$, *identity*
  L'applicazione di questo operatore non modifica la *super-position* del qubit.

5 MATRICI DI PAULI PER UN QUBIT - Q#

È possibile testare facilmente gli operatori di Pauli utilizzando Q#.
Di seguito alcuni esempi.

```
import qsharp
from Qrng import X_application
from Qrng import Z_application
from Qrng import Y_application
from Qrng import YasXZ_application


print("By measuring X|0> we obtain " + str(X_application.simulate()))
print("By measuring ZZX|0> = ZZ|1> = Z(-|1>) we obtain " +
                                    str(Z_application.simulate()))
print("By measuring Y|0> we obtain " + str(Y_application.simulate())+
                                    " that is the same of XZ|0>:
                                    "+str(YasXZ_application.simulate()))
```

```
namespace Qrng {
    open Microsoft.Quantum.Intrinsic;

    operation X_application() : Result {
        use q = Qubit(); // Allocate a |0> qubit.
        X(q);            // Apply X to the qubit, we get
            |1>
        let r = M(q);    // Measure the qubit value.
        Reset(q);
        return r;
    }

    operation Z_application() : Result {
        use q = Qubit(); // Allocate a |0> qubit.
        X(q);            // Apply X to the qubit, we get
            |1>
        Z(q);            // Apply Z to the qubit, we get
            -|1>
        Z(q);            // Apply Z to the qubit, we get
            |1>
        let r = M(q);    // Measure the qubit value.
        Reset(q);
        return r;
    }
```

```
    operation YasXZ_application() : Result {
        use q = Qubit(); // Allocate a |0> qubit.
        X(q);            // Apply X to the qubit, we get
            |1>
        Z(q);            // Apply Z to the qubit, we get
            -|1>
        let r = M(q);    // Measure the qubit value.
        Reset(q);
        return r;
    }

    operation Y_application() : Result {
        use q = Qubit(); // Allocate a |0> qubit.
        Y(q);            // Apply Z to the qubit, we get
            -|1>
        let r = M(q);    // Measure the qubit value.
        Reset(q);
        return r;
    }
}
```

L'operatore di *Hadamard* H è caratterizzato dalla matrice

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Applicare l'operatore a un qubit sbilanciato su uno stato base, per esempio $|0\rangle$, pone il qubit in una *super-position* bilanciata tra i due stati base.

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

6 QUBIT ENTANGLED

Così come avviene con i computer tradizionali, è possibile pensare a più qubit assieme nel quantum computing.
Dato un insieme di due qubit qualsiasi essi devono essere pensati come una unica entità e non separati.
Il principio alla base di questa idea è l'***entanglement***.

$$\alpha_0 |0\rangle + \alpha_1 |0\rangle$$
$$\beta_0 |0\rangle + \beta_1 |0\rangle$$

Lo stato complessivo descritto dai due qubit è il prodotto tensore tra i due.

$$\begin{bmatrix}\alpha_0\\ \alpha_1\end{bmatrix} \otimes \begin{bmatrix}\beta_0\\ \beta_1\end{bmatrix} = \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle$$

Gli stati base sono $|00\rangle, |01\rangle, |10\rangle$ e $|11\rangle$ e rappresentano una base *ortonormale*.
Il vincolo di normalizzazione rimane valido ed è

$$|\alpha_0\beta_0|^2 + |\alpha_0\beta_1|^2 + |\alpha_1\beta_0|^2 + |\alpha_1\beta_1|^2 = 1$$

Dopo aver osservato uno dei due qubit, la descrizione dello stato complessivo cambia.
Se si misura il primo qubit, e si osserva per esempio un 1, il sistema cambia il proprio stato.

$$\frac{\alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle}{\sqrt{|\alpha_1\beta_0|^2 + |\alpha_1\beta_0|^2}}$$

Due qubit sono *entangled* quando non è possibile separarli e identificare univocamente le caratteristiche di ognuno. Essi si influenzano reciprocamente, anche a distanza.
Un esempio classico in letteratura è lo stato di *Bell*. Uno stato di *Bell* esprime molto semplicemente la correlazione quantistica ed è realizzabile a partire da due stati base con l'applicazione di Hadamard e $CNOT$ (operatore applicabile a due qubit).

$$|\psi\rangle = \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$$

Se si volesse rappresentare lo stato di Bell come combinazione di due qubit si avrebbe un sistema di equazioni impossibile.

$$\alpha_0\beta_0 = \frac{1}{\sqrt{2}} \quad \alpha_0\beta_1 = 0 \quad \alpha_1\beta_0 = 0 \quad \alpha_1\beta_1 = \frac{1}{\sqrt{2}}$$

7 OPERATORI SU DUE QUBIT

Una delle caratteristiche degli operatori unitari U è che l'applicazione consecutiva di due volte l'operatore a un qubit $|\psi\rangle$ restituisce il qubit stesso. Si parla quindi di operatori *invertibili*.
Se applichiamo un operatore U a due qubit sono prodotti in output due qubit. Ciò non avviene con operatori classici che si applicano ai bit (*AND*, *OR*, ...).

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$
$$U|\psi\rangle = \alpha'_{00}|00\rangle + \alpha'_{01}|01\rangle + \alpha'_{10}|10\rangle + \alpha'_{11}|11\rangle$$
$$U(U|\psi\rangle) = |\psi\rangle$$

Consideriamo due operatori ampiamente utilizzati tra due (o più) qubit.

- operatore $CNOT$, *Controlled Not*
  L'operatore $CNOT$, nella sua versione base, prende in input due qubit: un qubit a che è il ***qubit di controllo*** e un qubit b che è il ***qubit target***. L'idea è quella di applicare un *bit flip* sul qubit target b se e solo se il qubit di controllo a è $|1\rangle$.

  $$CNOT|00\rangle = |00\rangle$$
  $$CNOT|01\rangle = |01\rangle$$
  $$CNOT|10\rangle = |11\rangle$$
  $$CNOT|11\rangle = |10\rangle$$

  Più in generale si ha che

  $$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$
  $$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{11}|10\rangle + \alpha_{10}|11\rangle$$

  La matrice descrittiva dell'operatore $CNOT$ è

  $$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

  L'operatore $CNOT$ è utilizzato per generare due qubit entangled quando si utilizza come bit target $|0\rangle$.
  Dato un qubit di controllo generico $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ con il qubit target $|0\rangle$ si ha un sistema che si trova in uno stato complessivo $(\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes |0\rangle = \alpha_0|00\rangle + \alpha_1|10\rangle$. Se si applica a questo l'operatore $CNOT$ si ottiene lo stato entangled $\alpha_0|00\rangle + \alpha_1|11\rangle$.

- operatore H, operatore di *Hadamard*
  Come si è visto, l'operatore di Hadamard permette di mettere in *super-position* un qubit completamente sbilanciato in uno dei due stati base.
  Lo stesso ragionamento può essere applicato quando in input si hanno due qubit.
  È possibile pensare all'operatore di Hadamard su due qubit come un unico ***operatore composto*** che contiene al suo interno due operatori di *Hadamard* (H_1 e H_1) che lavorano ognuno sul singolo qubit.
  L'operatore risultante è il prodotto tensore tra le due matrici rappresentanti i due operatori.

$$H_2 = H_1 \otimes H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ \end{bmatrix}$$

  Se si applica H_2 a uno stato completamente sbilanciato, per esempio $|00\rangle$, si ottiene

$$\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

  che è uno stato che si trova in ogni possibile stato base con una probabilità del 25%

Gli operatori $H, X, Z, CNOT$ sono utilizzati nel ***Quantum Teleportation Protocol*** per teletrasportare informazioni di un qubit.
Il protocollo si basa sulla condivisione di una coppia di qubit *entangled*.

8 OPERATORI SU DUE QUBIT - Q#

È possibile testare il funzionamento dell'operatore *CNOT* utilizzando Q#.
Come si è visto, si hanno in input due qubit: il *controller* e il *target*. Il target cambia valore in base al valore del controller: se questo è $|1\rangle$ allora si inverte il valore del qubit target, altrimenti rimane invariato.

```
namespace Qrng {
    open Microsoft.Quantum.Intrinsic;

    operation CNOT_application() : Result {
        use control = Qubit(); // Allocate a |0> qubit.
        use target = Qubit(); // Allocate a |0> qubit.
        X(control);            // Apply X to the control
            qubit, we get |1>

        CNOT(control,target);
        let r = M(target);    // Measure the qubit
            value, it returns 1
        Reset(control);
        Reset(target);
        return r;
    }
}
```

Si noti come se non fosse stato applicato l'operatore X al qubit di controllo, il risultato sarebbe stato 0.

È possibile testare anche l'*entanglement*.
Nell'esempio che segue si dichiarano due qubit, uno nello stato $|0\rangle$ e l'altro nello stato $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$.
È effettuato l'entanglement attraverso l'operatore *CNOT*.
Lo stato finale è

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Si noti come il comportamento di un qubit è descritto dal comportamento dell'altro.
Se si misura il primo qubit e si ottiene 0, allora anche il secondo qubit avrà valore 0. Se si misura il primo qubit e si ottiene 1, allora anche il secondo qubit avrà valore 1.

```
namespace Qrng {
    open Microsoft.Quantum.Intrinsic;

    operation Entangle_application() : Int {
        use control = Qubit(); // Allocate a |0> qubit.
        use target = Qubit(); // Allocate a |0> qubit.
        H(control);           // Apply H to the control
            qubit, we get a superposition in the middle

        CNOT(control,target);
        let m1 = M(target);    // Measure the qubit
            value, it now has a 50% chance of being 0 or
            1.
        let m2 = M(control);   //if m1==0 -> m2==0, if
            m1==1 -> m2==1
        Reset(control);
        Reset(target);
        mutable x = -1;
        if(m1==m2){
            if(m1==One){
                set x=1;
            }
            else{
                set x=0;
            }
        }
        return x;
    }
}
```

9 NO-CLONING THEOREM

Nei computer che utilizzano il modello di calcolo tradizionale una delle operazioni maggiormente eseguite è quella della *copia* di bit.
Purtroppo non è presente una operazione analoga nel modello di calcolo dei computer quantici: non è presente, e ***non può*** nemmeno esserci.
Dato un qubit $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ non è possibile costruire nessuna macchina che a partire da $|\psi\rangle$ produca due copie identiche di $|\psi\rangle$ nello stato complessivo $|\psi\rangle \otimes |\psi\rangle$.
Si ricorda che i valori di α_0 e α_1 non sono noti: se si effettua la misurazione si perde la proprietà di *super-position*.
Il ***No Cloning Theorem*** impone quindi forti restrizioni al programmatore.

La dimostrazione è molto semplice, realizzata per assurdo.
Si ipotizza l'esistenza di un operatore unitario U che possa effettuare la seguente operazione:

$$U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle$$

Se tale operatore esistesse, esso sarebbe valido anche per gli stati base.

$$U(|0\rangle \otimes |0\rangle) = |0\rangle \otimes |0\rangle$$
$$U(|1\rangle \otimes |0\rangle) = |1\rangle \otimes |1\rangle$$

Sia il qubit da clonare, per esempio, $|\psi\rangle = |+\rangle$

$$|\psi\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$$

Si applica l'operatore U a $|\psi\rangle \otimes |0\rangle$

$$U((\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle) \otimes |0\rangle) = \frac{1}{\sqrt{2}} U((|0\rangle \otimes |0\rangle) + (|1\rangle \otimes |0\rangle)) =$$
$$= \frac{1}{\sqrt{2}} (U(|0\rangle \otimes |0\rangle) + U(|1\rangle \otimes |0\rangle)) = \frac{1}{\sqrt{2}} (|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$$

Quello che si ottiene è diverso da ciò che si vorrebbe avere

$$|\psi\rangle \otimes |\psi\rangle = (\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle) \otimes (\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle)$$
$$= \frac{1}{2} (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)$$

Ne deriva che è ***impossibile*** realizzare l'operatore U.

10 NOTE

Il codice presentato è disponibile al seguente indirizzo
https://github.com/mariocuomo/Microsoft-Q-Advent-Calendar-2021

Questo articolo trae forte ispirazione dal corso di *Next Generation Computing Models - Università degli Studi Roma Tre.*