

ITIS-LS "Francesco Giordani" Caserta

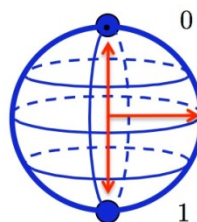
prof. Ennio Ranucci

a.s. 2019-2020

L'informatica quantistica



Classical Bit



Qubit

Bit
(Classical Computing)

0



1

Qubit
(Quantum Computing)

0



1

La rivoluzione scientifica del ventunesimo secolo?

A determinarla sarà l'informatica quantistica (Quantum Information). Ad affermarlo è una voce assolutamente autorevole, quella del Premio Nobel per la Fisica(1997) William Phillips. Tra le scoperte e le innovazioni lasciate in eredità dal secolo precedente due sono particolarmente importanti: la scoperta del comportamento quantistico del mondo microscopico e l'avvento della scienza dell'informazione. Il calcolo quantistico (quantum computation), che potrebbe diventare un nuovo paradigma nello sviluppo dei computer futuri. Pensiamo alla differenza tra un antico abaco e il nostro PC: è su questa scala che dobbiamo immaginare la differenza tra un computer di oggi e un quantum computer.

Nel mondo dei computer il quantum computing è l'avanguardia in termini di potenza e scalabilità e può trasformare in maniera radicale ogni ambito dell'informatica, dalla sicurezza delle informazioni alla scienza dei dati. Utilizzando i qubit, l'unità fondamentale dell'informazione quantistica, le macchine possono risolvere problemi neppure avvicinabili nei sistemi binari. Per le prime esperienze di coding esiste un simulatore quantistico in Python e Quantum Development Kit di Microsoft e del linguaggio Q#

Qubit o Quantum Bit è l'unità di informazione quantistica, analoga al "bit" nell'informatica classica. Per distinguere tra un bit classico e un qubit, viene utilizzata la notazione di Dirac. Quindi, i qubit sono rappresentati come $|0\rangle$ e $|1\rangle$ e sono spesso letti come 'stato zero' e 'stato uno'.

L'informatica quantistica è una tecnologia in rapida ascesa che sfrutta le leggi della meccanica quantistica per risolvere problemi troppo complessi per i computer classici.

Oggi, IBM Quantum rende disponibile a migliaia di sviluppatori un vero hardware quantistico, uno strumento che gli scienziati hanno iniziato a immaginare solo tre decenni fa.

Quando scienziati e ingegneri incontrano problemi difficili, si rivolgono ai supercomputer. Si tratta di computer classici molto grandi, spesso con migliaia di core CPU e GPU classici. Tuttavia, anche i supercomputer faticano a risolvere certi tipi di problemi.

Gli algoritmi quantistici adottano un nuovo approccio a questo tipo di problemi complessi, creando spazi multidimensionali che i computer classici non possono creare. Quando due qubit sono impigliati, le modifiche a un qubit hanno un impatto diretto sull'altro. Gli algoritmi quantistici sfruttano queste relazioni per trovare soluzioni a problemi complessi.

Il computer desktop utilizza una ventola per diventare abbastanza freddo da funzionare. I processori quantistici devono essere molto freddi – circa un centesimo di grado sopra lo zero assoluto. Per raggiungere questo obiettivo, si utilizzano superfluidi super-raffreddati per creare superconduttori.

Sparando fotoni a microonde ai qubit, si può controllare il loro comportamento e farli

trattenere, cambiare e leggere singole unità di informazioni quantistiche. In particolare, i quantum bit hanno alcune proprietà che derivano dalle leggi della fisica quantistica come:

- la **sovrapposizione di stati** (possono essere contemporaneamente 0 e 1) grazie alla quale si possono fare calcoli paralleli anziché sequenziali come avviene oggi con la capacità computazionale dei computer “tradizionali”;
- l’**entanglement**, cioè la correlazione (il legame) che c’è tra un qubit ed un altro, aspetto molto importante perché è da qui che deriva una forte accelerazione nel processo di calcolo grazie all’influenza che un qubit può produrre su un altro anche se distante;
- l’**interferenza quantistica**, che è di fatto l’effetto del primo principio (la sovrapposizione degli stati); l’interferenza quantistica permette di “controllare” la misurazione dei qubit basandosi sulla natura ondulatoria delle particelle (l’interferenza di fatto rappresenta la sovrapposizione di due o più onde e, a seconda che ci sia sovrapposizione o meno tra le parti più alte e quelle più basse dell’onda – si possono avere interferenze costruttive, quando creste o ventri coincidono e formano un’onda che è la somma delle onde che si sovrappongono, oppure interferenze distruttive, quando a sovrapporsi sono cresta di un’onda e ventre di un’altra, in questo caso le due onde si annullano a vicenda).

L’esempio di YK Sugi pubblicato su www.freecodecamp.org

Un computer **quantistico non** utilizza bit per memorizzare informazioni. Invece, usa qualcosa chiamato qubit. Ogni qubit non solo può essere impostato su 1 o 0, ma può anche essere impostato su 1 e 0.

Devi spostare solo 3 persone: Alice, Becky e Chris. Hai prenotato 2 taxi per questo scopo e vuoi capire chi entra in quale taxi, avendo informazioni su chi è amico di chi e chi è nemico di chi.

- Alice e Becky sono amiche
- Alice e Chris sono nemici
- Becky e Chris sono nemici

il tuo obiettivo è quello di dividere questo gruppo di 3 persone nei due taxi per raggiungere i seguenti due obiettivi:

- Massimizza il numero di **coppie di amici** che condividono la stessa auto
- Riduci al minimo il numero di **coppie nemiche** che condividono la stessa auto

Possiamo impostare i tre bit su **0, 0 e 1** per rappresentare:

- Alice entra nel Taxi #0
- Becky sale sul Taxi #0
- Chris sale sul Taxi #1

Poiché ci sono due scelte per ogni persona, ci sono $2^3 = 8$ modi per dividere questo gruppo di persone in due auto.

Ecco l'elenco di tutte le possibili configurazioni:

Alice=A Becky=B C=Chris

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Ora, usando un normale computer, come determineremmo quale configurazione è la migliore? Per fare ciò, definiamo come possiamo calcolare il punteggio per ogni configurazione. Il punteggio rappresenterà la misura in cui ogni configurazione raggiungerà i due obiettivi che abbiamo menzionato in precedenza:

- Massimizza il numero di **coppie di amici** che condividono la stessa auto
- Riduci al minimo il numero di **coppie nemiche** che condividono la stessa auto

Definiamo semplicemente il nostro punteggio come segue:

punteggio di una determinata configurazione = numero di coppie di amici che condividono la stessa auto - numero di coppie nemiche che condividono la stessa auto

Ad esempio, supponiamo che Alice, Becky e Chris entrino tutti nel Taxi #1. Con tre bit, questo può essere espresso come **111**. In questo caso, c'è solo **una coppia di amici** che condivide la stessa auto: Alice e Becky e **due coppie nemiche** che condividono la stessa macchina: Alice e Chris, e Becky e Chris. Quindi, il punteggio totale di questa configurazione è $1-2 = -1$.

Con i punteggi possiamo trovare il caso migliore.

Con un normale computer, per trovare la configurazione migliore, dobbiamo essenzialmente passare attraverso tutte le configurazioni per vedere quale raggiunge il punteggio più alto.

Quindi, costruiamo la tabella:

A	B	C	Punteggio
0	0	0	-1
0	0	1	1
0	1	0	-1
0	1	1	-1
1	0	0	-1
1	0	1	-1
1	1	0	1
1	1	1	-1

Ci sono due soluzioni 001 e 110, entrambe con punteggio uguale ad 1.

Questo problema è abbastanza semplice. Diventa rapidamente difficile da risolvere con un normale computer se aumentiamo il numero di persone.

Abbiamo visto che con 3 persone, abbiamo bisogno di passare attraverso 8 possibili configurazioni. Cosa succede se ci sono 4 persone?

In tal caso, dovremo passare attraverso $2^4 = 16$ configurazioni.

Con n persone, dovremo passare attraverso 2^n configurazioni per trovare la soluzione migliore. Quindi, se ci sono 100 persone, dovremo passare attraverso: $2^{100} \approx 10^{30}$ = un milione di milioni di milioni di milioni di milioni di configurazioni.

Questo è semplicemente impossibile da risolvere con un normale computer.

Con un computer classico, utilizzando 3 bit, siamo stati in grado di rappresentare solo una di queste soluzioni alla volta, ad esempio 001.

Con un computer quantistico, utilizzando 3 **qubit**, possiamo rappresentare **le 8 configurazioni contemporaneamente**.

Impostiamo il primo qubit sia su 0 **che** su 1, è un po' come creare due mondi paralleli.

In uno di questi mondi paralleli, il qubit è impostato su 0. Nell'altro, è impostato su 1.

Ora, cosa succede se imposti anche il secondo qubit su 0 e 1? E' un po' come creare 4 mondi paralleli. Nel primo mondo, i due qubit sono impostati su 00. Nel secondo, sono 01. Nel terzo, sono 10. Nel quarto, sono 11.

Allo stesso modo, se imposti tutti e tre i qubit su 0 e 1, creerai 8 mondi paralleli: 000, 001, 010, 011, 100, 101, 110 e 111.

Quindi, invece di esaminare ciascuna configurazione in sequenza, possiamo calcolare i punteggi di tutte le soluzioni contemporaneamente.

Se diamo al tuo computer quantistico:

- Tutte le potenziali configurazioni rappresentate con qubit
- Una funzione che trasforma ogni potenziale soluzione in un punteggio.

Date queste due cose, il computer quantistico troverà uno dei migliori punteggi in pochi millisecondi. In realtà, ci sono errori durante l'esecuzione di un computer quantistico. Quindi, invece di trovare il primo punteggio migliore, potrebbe trovare il secondo punteggio migliore, il terzo e così via. Questi errori diventano più evidenti man mano che il problema diventa sempre più complesso. Quindi, in pratica, probabilmente dovremo eseguire la stessa operazione su un computer quantistico dozzine di volte o centinaia di volte. Infine scegliamo il miglior risultato tra i molti risultati che otteniamo.

Quando ci sono 3 persone che dobbiamo dividere in due auto, il numero di operazioni che dobbiamo eseguire su un computer quantistico è 1. Questo perché un computer quantistico calcola il punteggio di tutte le configurazioni contemporaneamente.

Quando ci sono 4 persone, il numero di operazioni è ancora 1.

Quando ci sono 100 persone, il numero di operazioni è ancora 1.

Con una singola operazione, un computer quantistico calcola i punteggi di tutte le $2^{100} \approx 10^{30}$ = **un milione di milioni di milioni di milioni di milioni di** configurazioni contemporaneamente.

La teoria quantistica, formulata da Max Planck agli inizi del Novecento, nasce da una ricerca condotta sulla radiazione emessa da un corpo nero. Tale corpo ha la capacità di assorbire tutte le radiazioni incidenti e di irradiarle a sua volta in maniera dipendente dalla temperatura ma indipendente dalla natura del materiale. Secondo Planck l'energia delle radiazioni elettromagnetiche assume sempre una potenza multipla intera di un'unità fondamentale, detta quanto (o quanto di energia). Questo processo viene chiamato quantizzazione.

Cosa significa quantizzare? Significa trattare un moto o un fenomeno che ci appare continuo come discreto. Ad esempio, le dune del deserto del Sahara ci appaiono come continue, cambiano forma e sembrano infinite, ma non lo sono. Nella realtà, le dune del potrebbero essere viste come un insieme finito "quantizzato" di un determinato numero determinato (quantità) di granelli. Ogni granello è un'unità discreta (non continua) e misura circa un millimetro.

La teoria dei quanti e i successivi studi condotti da Einstein sull'effetto fotoelettrico portano alla scoperta della natura corpuscolare della luce.

La teoria dei quanti si basa sul criterio della quantizzazione: quantità fisiche come l'energia non possono essere scambiate in modo continuo ma attraverso 'pacchetti' (quanti); un sistema può pertanto possedere valori di energia specifici, e non illimitati come invece sostenevano le leggi della fisica classica. Un computer quantistico sfrutta alcune tra le proprietà più bizzarre e controintuitive della meccanica quantistica per ottenere una potenza di calcolo di gran lunga superiore rispetto a quella di un computer (e di un supercomputer) classico. L'unità minima di informazione di un processore convenzionale è il bit, un'entità binaria che può assumere i valori zero e uno a seconda del passaggio o meno di corrente. Dal canto loro, i processori quantistici usano i qubit, in genere particelle subatomiche come fotoni o elettroni, che invece possono immagazzinare molte più informazioni.

L'informatica quantistica (o *quantum computing*) è la branca di studi che si concentra sullo sviluppo di una tecnologia informatica basata sulla teoria dei quanti. In questo modo sarà possibile sviluppare computer che, seguendo i principi della fisica quantistica, abbiano un'enorme capacità di calcolo.

Alla base del funzionamento dei computer quantistici troviamo il qubit, "alter ego" del bit dell'informatica classica.

Se quest'ultimo può assumere solo due valori ben determinati (convenzionalmente indicati con "0" e "1" e corrispondenti allo stato di carica di un transistor del chip), il bit quantistico può assumere diversi valori (risponde al "Principio di indeterminazione" di Heisenberg) ed essere contemporaneamente sia "0" sia "1". Proprio per questo il qubit è maggiormente versatile e più potente rispetto alla sua alternativa "digitale", permettendo di processare una quantità maggiore di informazioni.

Nasce così l'approccio teorico al computer quantistico: anziché calcolare con dei bit (binary digit), le unità di informazione che codificano i due stati aperto e chiuso (1 e 0) di un interruttore, si sfruttano quelli che vengono chiamati qubit, le unità dell'informazione quantistica che sono codificati non da 1 o 0 ma dallo stato quantistico in cui si trova una particella o un atomo, che può avere contemporaneamente sia il valore 1 sia il valore 0.

Non solo: possono considerare una varietà di combinazioni che producono differenti stati quantistici (una particella può essere per 70% allo stato 1 e per il 30% allo stato 0, o 40% e 60%, o 15 e 85, e così via). Una condizione detta sovrapposizione quantistica e che assume un significato incredibile se si pensa alla progressione matematica. Infatti, proprio per via di questo principio 2 qubit possono avere 4 stati contemporaneamente, 4 qubit hanno 16 stati, 16 qubit hanno 256 stati e via dicendo. Se pensiamo al fatto che la quantità di informazione contenuta in n qubit è pari a 2 alla n bit classici, ci rendiamo conto che si sta parlando di numeri astronomici.

Il qubit in pratica è costituito da una particella come ad esempio un elettrone. L'elettrone può trovarsi in due stati: spin-su e spin-giù. I simboli degli spin \uparrow e \downarrow in qualche modo corrispondono ai tradizionali 0 ed 1.

Nella meccanica quantistica emerge però un fatto per molti versi straordinario. Gli stati di spin sono distinti e nella realtà coesistono, si dice che si trovano in sovrapposizione. Il singolo elettrone ha dunque uno stato indeterminato ed inimmaginabile nella logica classica, il quale comprende sia \uparrow sia \downarrow . Quindi se si procede a costruire un registro costituito da due elettroni, i dati registrati sono quattro in corrispondenza delle possibili configurazioni degli spin:

$\uparrow \uparrow, \uparrow \downarrow, \downarrow \uparrow, \downarrow \downarrow,$

che corrispondono ai quattro dati binari normali:

00, 01, 10, 11

Dunque in un registro a due celle convivono in sovrapposizione tutti e quattro i valori sopra riportati, estraibili con opportuni strumenti. In ambito tradizionale avremmo bisogno di 4 registri a 2 celle, nel computer quantistico è sufficiente 1 registro con 2 celle. Continuando si può affermare che un registro a 3 celle contiene 8 valori, a 4 celle 16 valori e così via. Il numero di operazioni che un sistema quantistica può eseguire è 2^n quando n sono i qubits utilizzati. Da questo viene una

significativa conseguenza, un sistema che ha per esempio 500 qubit ha le potenzialità di calcolo di 2500 operazioni in un solo passo. Dettagli che fanno chiaramente intuire le potenzialità elaborative di una macchina quantistica.

Sul piano pratico tuttavia i problemi sono enormi. La costruzione di un solo registro quantistico è una sfida tecnologica molto impegnativa. Infatti una particella per stare nello stato di sovrapposizione deve essere isolata nello spazio vuoto e non deve interagire con strumenti o circuiti o componenti perché altrimenti perde questa proprietà. Si dice che interviene un fenomeno di decoerenza quantistica : la particella collassa, non è più in sovrapposizione ed assume uno solo dei suoi stati quando subisce una interferenza esterna.

<https://systemscue.it/elaborare-qubit-computer-quantistico/8984/>

I qubit

Un qubit è rappresentato da un **vettore di dimensione 2 e modulo 1**.

Ci sono **2 qubit base** ortogonali:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ (ground state)}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ (excited state)}$$

Un qubit non deve per forza essere in uno stato di $|0\rangle$ o $|1\rangle$ ma può essere anche in **uno stato di sovrapposizione** (superposition) del tipo:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Dove il modulo rimane 1, quindi:

$$|\alpha|^2 + |\beta|^2 = 1$$

Altri qubit standard in sovrapposizione sono quindi:

$$|+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$|-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$|\odot\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix}$$

$$|\ominus\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -i \end{pmatrix}$$

<https://quantum-computing.ibm.com/>

“**IBM Quantum Experience**”: permette a studenti, ricercatori e appassionati di informatica e programmazione di sviluppare e testare i propri algoritmi su un processore quantistico reale, direttamente dalla poltrona di casa o dal laboratorio.

Così come i computer convenzionali sono costituiti da bit, i computer quantistici sono costituiti da bit quantistici, o qubit.

Un qubit ha uno stato.

Tuttavia, mentre lo stato di un bit è un numero 0 oppure 1 lo stato di un qubit è un vettore.

Più specificamente, lo stato di un qubit è un vettore in uno spazio vettoriale bidimensionale.

Questo spazio vettoriale è noto come spazio di stato.

Due stati quantici speciali corrispondono a 0 e 1 come quelli del bit tradizionale.

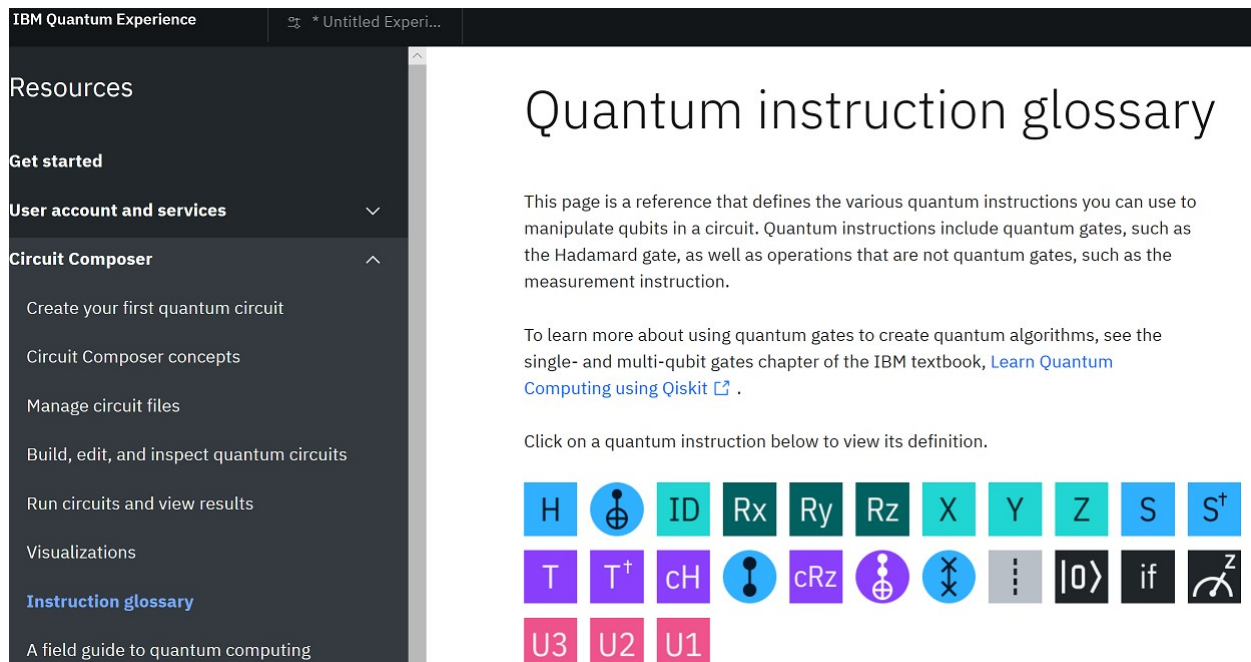
Lo stato quantico corrispondente a 0 si scrive $|0\rangle$

Questo stato speciale è denominato stato di base computazionale.

C'è un altro stato di base computazionale, scritto come $|1\rangle$

Per il calcolo quantistico, è necessario essere in grado di modificare lo stato dei qubit con le porte di logica quantistica.

Nella pagina indicata nella seguente immagine vengono descritte tutte le porte quantistiche:



Tutti i qubit iniziano nello stato fondamentale. Ciò significa che se misuriamo un qubit senza applicare operazioni su di esso, ci aspettiamo di ricevere 0 come risultato.

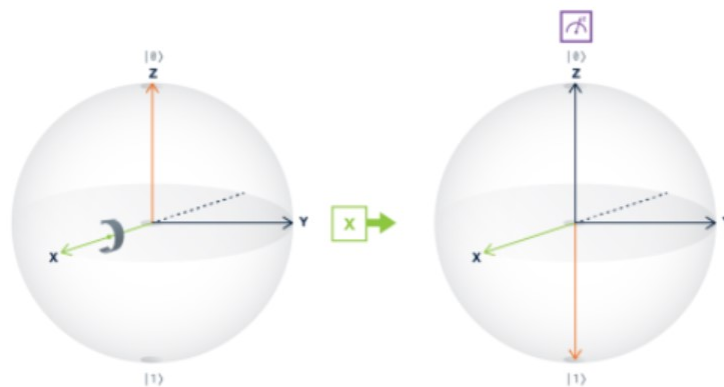
Gli operatori pauli

Lo spin di una particella è il suo momento angolare, la rotazione della particella sul suo asse, puoi immaginarla come una piccola sfera. Lo spin Rappresenta le probabilità delle posizioni che può assumere, lo stato fisico della particella, in un determinato periodo di tempo.

Wolfgang Ernst Pauli, fisico austriaco, enunciò il principio di esclusione, per il quale vinse il Premio Nobel nel 1945, secondo il quale due elettroni in un atomo non possono avere tutti i numeri quantici uguali.

Proviamo il gate **X** (corrispondente al NOT)

Il gate X ruoterà il qubit 0 dallo stato fondamentale allo stato eccitato, quindi una misurazione immediatamente dopo dovrebbe restituire un risultato 1.



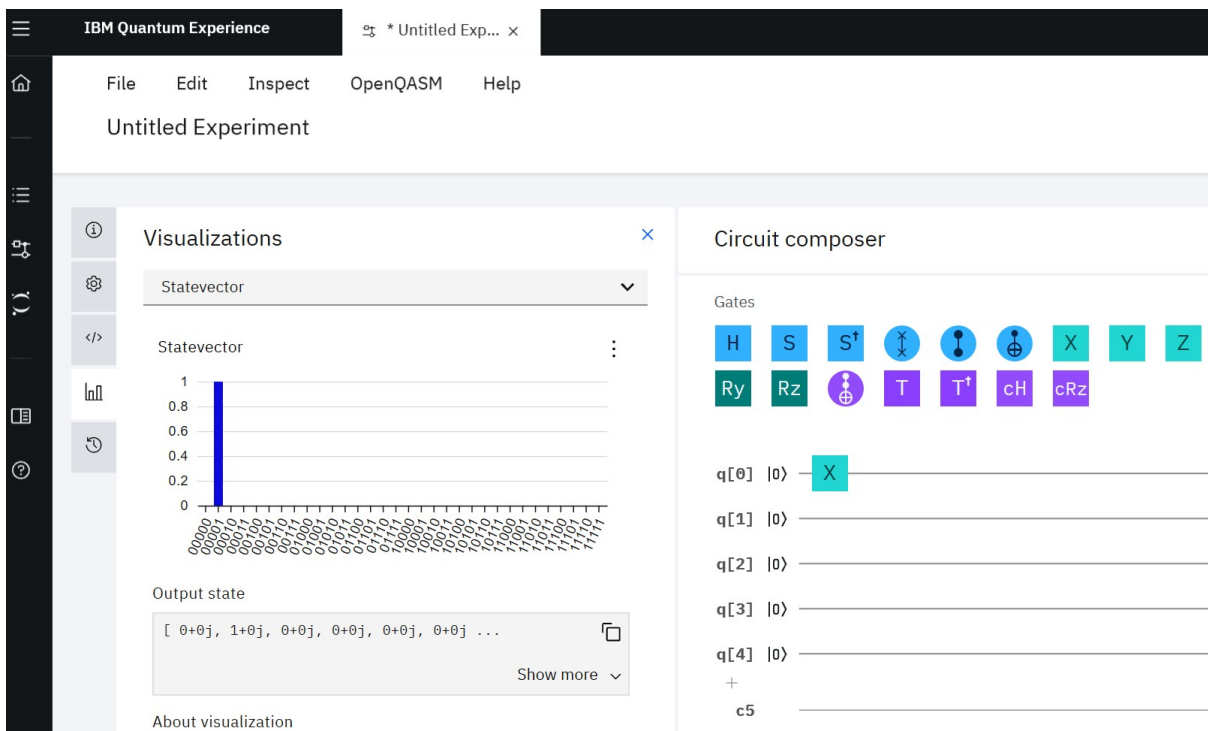
X gate

The Pauli X gate has the property of flipping the $|0\rangle$ state to $|1\rangle$, and vice versa. It is equivalent to R_x for the angle π .



Clicchiamo su Circuits Composer (colonna a sx) ed apriamo un nuovo Circuito.

Clicchiamo sull'icona "Istogramma" e trasciniamo la X sul qubit 0 -> l'istogramma indica il passaggio a 1



The screenshot shows the IBM Quantum Experience interface. On the left, the 'Visualizations' panel is active, displaying a histogram for the statevector. The histogram shows a single bar at the state $|0000\rangle$ with a probability of 1.0. Below the histogram, the 'Output state' is shown as $[\theta+0j, 1+0j, \theta+0j, \theta+0j, \theta+0j, \theta+0j \dots]$. On the right, the 'Circuit composer' panel shows a circuit with 5 qubits. The first qubit, $q[0]$, starts in the $|0\rangle$ state and has an X gate applied to it. The other qubits, $q[1]$ through $q[4]$, also start in the $|0\rangle$ state. The classical register $c5$ is also visible.

Se aggiungiamo un'altra X sulla linea del qubit 0 l'istogramma passa a 0

Clicchiamo sull'icona Editor di Circuito per consultare il codice asm quantum corrispondente al nostro circuito grafico.


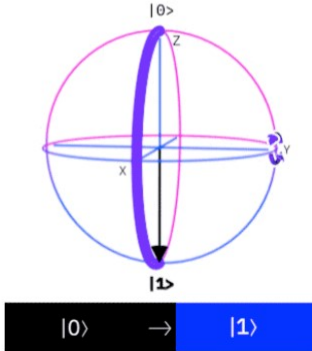
```

OPENQASM 2.0;
include "qelib1.inc";
qreg q[5];
creg c[5];
x q[0];

```

Y gate

The Pauli Y gate is equivalent to R_y for the angle π . It is also equivalent to the combined effect of X and Z.

Composer reference	OpenQASM reference	Bloch sphere rotation
	<code>y q[0];</code>	

Z gate

The Pauli Z gate has the property of flipping the $|+\rangle$ to $|-\rangle$, and vice versa. It is equivalent to R_z for the angle π .

Composer reference	OpenQASM reference	Bloch sphere rotation
	<code>z q[0];</code>	

The Hadamard gate is a single-qubit operation that maps the basis state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, thus creating an equal superposition of the two basis states.

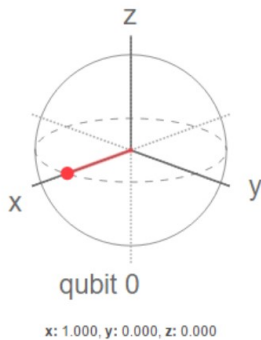
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Clifford gates

I "Clifford Gates" sono gli operatori secondari che servono a passare in uno stato di sovrapposizione.

Il primo gate, noto come **Hadamard Gate**, serve a creare una nuova base detta **diagonale**, portando il nostro qubit $|0\rangle$ in uno stato di sovrapposizione, con le due probabilità (di essere $|0\rangle$ o $|1\rangle$) a $1/2$ se misurato con la base canonica.

Questo qubit si indica con $|+\rangle$




Se lo andiamo a rappresentare, capiamo cosa succede:
Il qubit non sta più sull'asse Z. Abbiamo quindi un qubit che è sia $|0\rangle$ che $|1\rangle$ per metà del tempo.

Questo qubit è il risultato di questa operazione:
 $H|0\rangle$

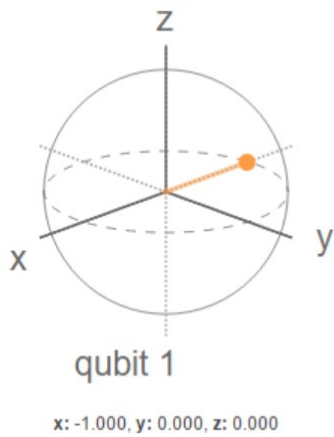
Abbiamo quindi individuato la base canonica e la base diagonale, esiste anche la base circolare.

H gate

The H or Hadamard gate rotates the states $|0\rangle$ and $|1\rangle$ to $|+\rangle$ and $|-\rangle$, respectively. It is useful for making superpositions. As a Clifford gate, it is useful for moving information between the x and z bases.

Composer reference	OpenQASM reference	Bloch sphere rotation
	<code>h q[0];</code>	

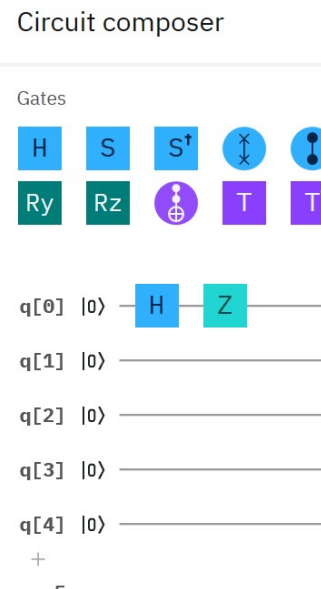
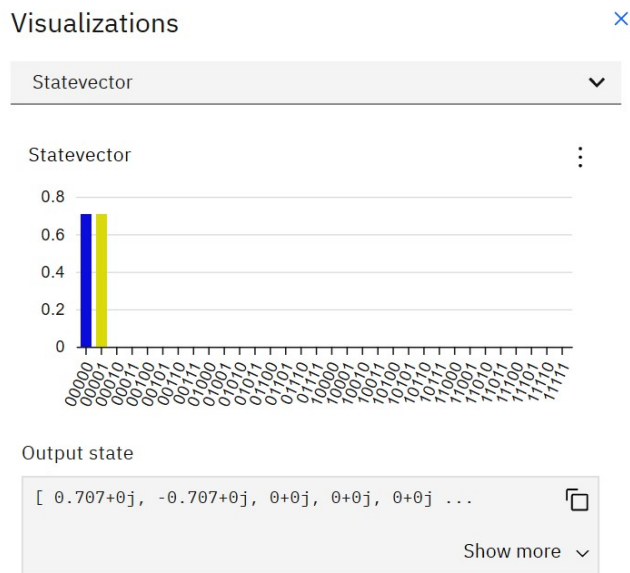
Se neghiamo H con Z otteniamo:



Abbiamo anche qui un qubit che è sia $|0\rangle$ che $|1\rangle$ per metà del tempo, ma differisce da quello precedente.

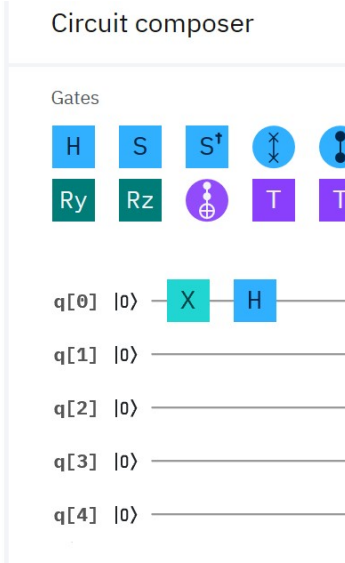
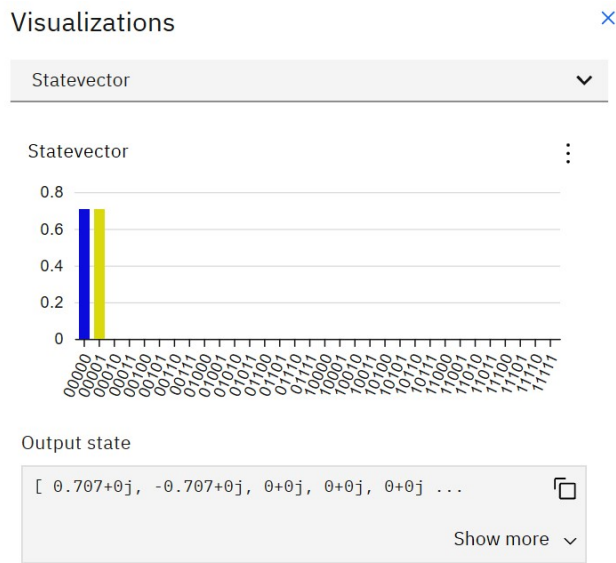
Questo qubit è dato da questa operazione:
 $ZH|0\rangle$

E possiamo facilmente asserire che i due qubit appena creati siano ortogonali tra loro.

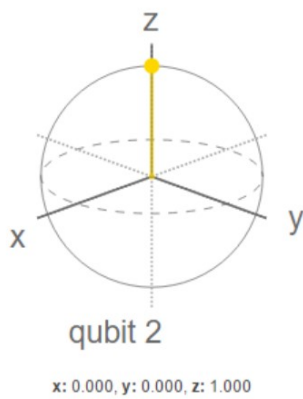


Otteniamo lo stesso risultato con X H :

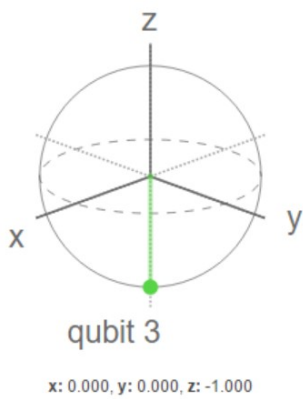




Nel modello di computer quantistico che ci fornisce IBM, non possiamo scegliere in che base misurare un qubit, ma basta reinvertire gli assi applicando ancora la matrice H per ottenere lo stesso risultato che misurare i qubit nella nuova base diagonale:



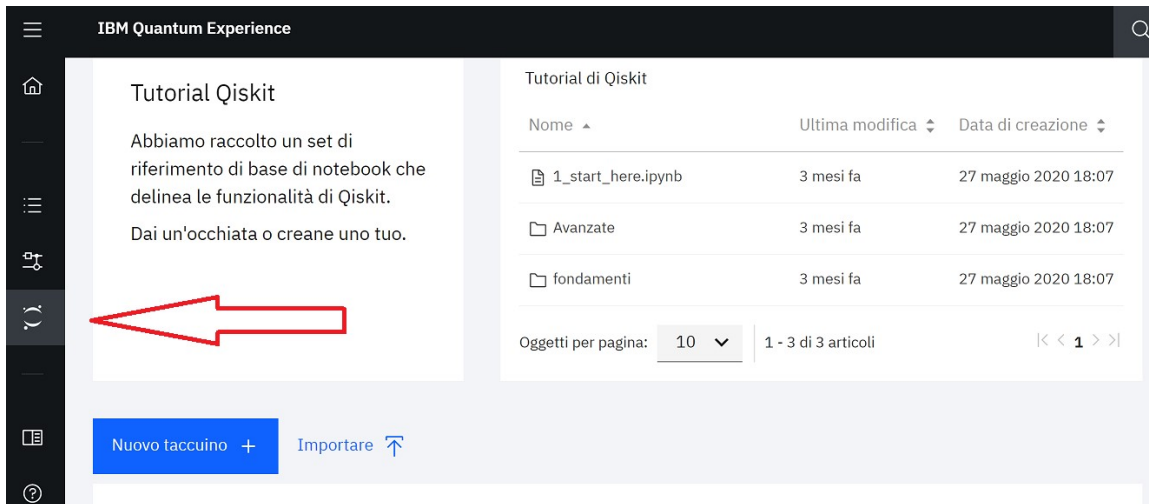
Questo qubit è dato da questa operazione:
 $HH|0\rangle$



Questo qubit è dato da questa operazione:
 $HZH|0\rangle$

Entriamo in ambiente Qiskit

Siamo entrati in ambiente Qiskit cliccando sull'icona indicata dalla freccia:



Clicchiamo su nuovo taccuino.

Possiamo provare le istruzioni Python:

- 1) se scriviamo `1+3` e clicchiamo su run otteniamo 4
- 2)

```
an_integer = 42 # Just an integer
a_float = 0.1 # A non-integer number, up to a fixed precision
a_boolean = True # A value that can be True or False
a_string = "just enclose text between two 's, or two "s, or do what we did for this string" # Text
none_of_the_above = None # The absence of any actual value or variable type
print(a_string)
```
- 3)

```
a_tuple = ( 42, 0.5, True, [0,1], None, 'Banana' )
a_tuple[0]
```
- 4)

```
for j in range(5):
    print(j)
```
- 5)

```
a_list = [ 42, 0.5, True, [0,1], None, 'Banana' ]
if 'strawberry' in a_list:
    print('We have a strawberry!')
elif a_list[5]=='apple':
    print('We have an apple!')
else:
    print('Not much fruit here!')
```
- 6)

```
import numpy as np
np.sin( np.pi/2 )
```
- 7)

```
def somma ( num1, num2 ):
    risultato = num1 + num2
    return risultato
x = somma(1,72)
print(x)
```



```

8) import random
a_list = [ 42, 0.5, True, [0,1], None, 'Banana' ]
for j in range(5):
    print('* conteggio: ',j+1)
    print('\n numero casuale compreso tra 0 to 1: ', random.random() )
    print("\n elemento scelto casualmente nella lista :", random.choice( a_list ) )
    print('\n')

```

Dividere le informazioni in bit

I bit sono l'alfabeto più semplice del mondo. Con solo due caratteri, 0 e 1, possiamo rappresentare qualsiasi informazione.

Un esempio sono i numeri. Rappresentiamo un numero attraverso una stringa di dieci cifre 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. In questa stringa di cifre, ogni cifra rappresenta quante volte il numero contiene una potenza del dieci. Ad esempio, quando scriviamo 9213, intendiamo

$$9000 + 200 + 10 + 3$$

Oppure $(9 \times 10^3) + (2 \times 10^2) + (1 \times 10^1) + (3 \times 10^0)$

Di solito utilizziamo questo sistema basato sul numero 10, possiamo altrettanto facilmente usarne uno basato su qualsiasi altro numero. Il sistema di numeri binari, ad esempio, si basa sul numero due. Questo significa usare i due caratteri 0 e 1 per esprimere numeri come multipli di potenze di due. Ad esempio, 9213 diventa 1000111111101,

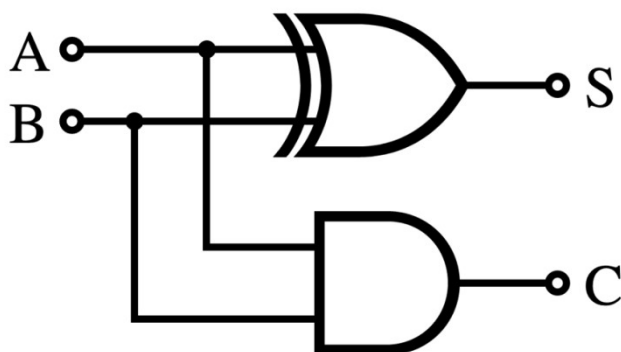
$$9213 = (1 \times 2^{13}) + (0 \times 2^{12}) + (0 \times 2^{11}) + (0 \times 2^{10}) + (1 \times 2^9) + (1 \times 2^8) + (1 \times 2^7) + (1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

In questo stiamo esprimendo numeri come multipli di 2, 4, 8, 16, 32, ecc. Anziché 10, 100, 1000, ecc.

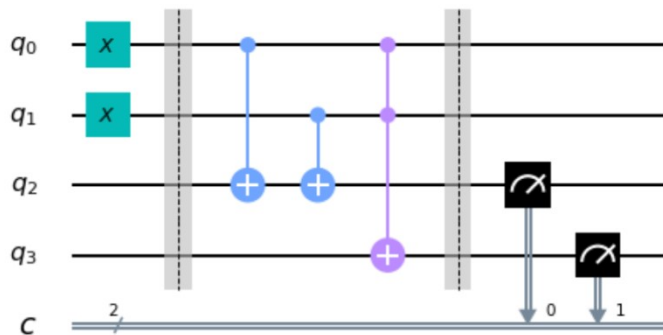
Queste stringhe di bit, note come stringhe binarie, possono essere utilizzate per rappresentare più di semplici numeri. Ad esempio, c'è un modo per rappresentare qualsiasi testo usando i bit. Per ogni lettera, numero o segno di punteggiatura che si desidera utilizzare, è possibile trovare una stringa corrispondente di massimo otto bit utilizzando la tabella **ASCII**. Che si tratti di numeri, lettere, immagini o suoni, tutto viene rappresentato sotto forma di stringhe binarie.

Come i nostri computer digitali standard, i computer quantistici si basano su questa stessa idea di base. La differenza principale è che usano i *qubit*, un'estensione del bit alla meccanica quantistica.

Usiamo i qubit come se fossero bit:



Ecco il circuito quantistico che rappresenta lo stesso processo di cui sopra:



Circuiti Quantum in ambiente QisKit

Prepariamo due qubit inizializzati a zero

```
qc = QuantumCircuit()
qr = QuantumRegister(2,'qreg')
qc.add_register( qr )
qc.qregs
qc.draw(output='mpl')
```

Applichiamo la porta H al qubit zero

```
qc = QuantumCircuit()
qr = QuantumRegister(2,'qreg')
qc.add_register( qr )
qc.qregs
qc.h(qr[0])
qc.draw(output='mpl')
```

<https://qiskit.org/textbook/ch-states/atoms-computation.html>

Creiamo con Qiskit un circuito con otto qubit e otto uscite.

```
from qiskit import QuantumCircuit, execute, Aer
dim = 8
dim_quantum = dim
dim_bit = dim
qc_output = QuantumCircuit(dim_quantum,dim_bit)
for j in range(dim):
    qc_output.measure(j,j)
qc_output.draw()
```

Questo circuito, che abbiamo chiamato `qc_output`, viene creato da Qiskit utilizzando `QuantumCircuit`. Il numero `dim_quantum` definisce il numero di qubit nel circuito. Con `dim_bit` definiamo il numero di bit di uscita che estrarremo dal circuito alla fine.

L'estrazione delle uscite in un circuito quantistico viene eseguita utilizzando un'operazione chiamata misura.

Ogni misurazione indica a un qubit specifico di fornire un output a un bit di output specifico.

Il codice aggiunge un'operazione di misurazione a ognuno dei nostri otto qubit. I qubit e i bit sono entrambi etichettati dai numeri da 0 a 7. Il comando `qc.measure(j,j)` aggiunge una misura al nostro circuito `qc` che indica a qubit `j` di scrivere un output nel bit `j`.

Eseguendo molte volte il codice sopra scritto e mostrando il risultato come un istogramma è sempre lo stesso perché i computer quantistici possono avere qualche casualità nei loro risultati, invece in questo caso, dal momento che non stiamo facendo nulla di quantico, otteniamo solo il risultato 00000000 con certezza.

Si noti che questo risultato proviene da un simulatore quantistico, che è un computer standard che calcola ciò che farebbe un computer quantistico ideale. Le simulazioni sono possibili solo per un piccolo numero di qubit, ma sono comunque uno strumento molto utile quando si progettano i primi circuiti quantistici. Per funzionare su un dispositivo reale è sufficiente sostituire `Aer.get_backend('qasm_simulator')` con l'oggetto back-end del dispositivo che si desidera utilizzare.

```
from qiskit.visualization import plot_histogram
counts = execute(qc_output,Aer.get_backend('qasm_simulator')).result().get_counts()
plot_histogram(counts)
```

Oppure

```
from qiskit.visualization import plot_bloch_multivector
backend = Aer.get_backend('statevector_simulator')
out = execute(qc_output,backend).result().get_statevector()
plot_bloch_multivector(out)
```

Creazione di un circuito sommatore

Un solo ingresso

Usiamo la porta X (NOT)

```
qc_encode = QuantumCircuit(8)
qc_encode.x(7)
qc_encode.draw()
qc = qc_encode + qc_output
qc.draw(output='mpl',justify='none')
counts = execute(qc,Aer.get_backend('qasm_simulator')).result().get_counts()
plot_histogram(counts)
```

```
print("Esecuzione del codice")
```

```
from qiskit import QuantumCircuit
```

```
qc = QuantumCircuit(2) # Create circuit with 2 qubits
```

```
qc.h(0) # Do H-gate on q0
```

```
qc.cx(0,1) # Do CNOT on q1 controlled by q0
```

```
qc.measure_all()
```

```
qc.draw()
```