T.C.
ULUDAĞ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Kuantum Hesaplama ve Algoritmaları

Amin Hashemian
031890100

TASARIM DERSİ RAPORU

BURSA 2021

T.C.
ULUDAĞ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Kuantum Hesaplama ve Algoritmaları

Amin Hashemian
031890100

Projenin Danışmanı: Dr. Savaş Takan

İÇİNDEKİLER

ÖZET

Kuantum hesaplama, tüm pratik amaçlar için klasik bilgisayarlar tarafından çözülemeyen sorunlara çözüm hesaplama yeteneği vaat ediyor. Bununla birlikte, kuantum vaadi, pratik gerçekleştirmeye ulaşmaktan hala çok uzak. Kuantum mekaniğinin, kuantum bilgisayarların üstün performansını mümkün kılan bazı özellikleri, kuantum algoritmalarının tasarımını ve işlevsel donanımın inşasını da son derece zorlaştırmaktadır. Kübitlerin tutarlılık süresini ve kuantum işlemlerinin hızını artırarak kübit teknolojisinin kalitesini iyileştirmek için bazı çözümler ima edilmesi gerekiyor. Kuantum hata düzeltmesi için kübitin durumunu da düzeltilmesi gerekiyor.

1997'de dünyanın ilk küçük kuantum bilgisayarı yapıldı. Bugün, birçok bilim adamı en güçlü kuantum bilgisayarı oluşturmak için bu savaşa katılıyor. Bunlar arasında IBM, Google, Alibaba, Microsoft , Intel gibi bilinen bazı ticari kuruluşlar ve hepsi de kuantum bilişim geliştirme ve araştırma için milyarlarca dolar harcamaya istekli olan çok daha fazlası yer alıyor. Ama neden bu Kuantum Bilgisayarlarına bu kadar çok para harcayalıyorlar. Cevap basit: Farklı bağlamlarda klasik bilgisayarlara göre üstünlükleri nedeniyle

Kuantum bilgisayarları, güvenlik sistemleri, finans yönetimi, araştırma, ilaçlar ve çok daha fazlası gibi çeşitli alanlarda yeni atılımların gelişimini sebep olabilir. Ancak bunlar günlük Netflix izleme, e-posta yazma vb. görevlerimiz söz konusu olduğunda pek kullanışlı değil. Bunlar kuantum bilgisayarların bize gerçekten yardımcı olacağı yerler değil. Ancak, ilk klasik bilgisayarlar üretildiğinde bile, insanlar onların yalnızca bilim adamları tarafından kulanacağını düşünüyordu. Yani, Kuantum bilgisayarlarının bizim için ne sakladığını sadece zaman gösterecek.

ABSTRACT

Kuantum hesaplama, veriler üzerinde işlemler gerçekleştirmek için kuantum olaylarını kullanan yeni bilim alanıdır. Klasik hesaplamanın faydalarını kuşkusuz her gün yaşıyoruz. Ancak yine de günümüzün klasik sistemlerinin asla çözemeyeceği bazı zorluklar var. Belirli bir boyutu ve karmaşıklığı aşan problemler söz konusu olduğunda, onları çözmek için Dünya üzerinde yeterli hesaplama gücüne sahip değiliz. Bu karmaşık problemlerinden bazılarını çözme şansına sahip olmak için yeni bir tür hesaplamaya ihtiyacımız var. Ve Kuantum Bilgisayarları fikrinin ortaya geldiği yer burasıdır Kuantum hesaplamanın amacı, aynı sorunu çözen klasik algoritmalardan çok daha hızlı algoritmalar bulmaktır. Bilgisayar teknolojisinin tarihi, donanım açısında bir evrim yaşamakta röleler, valfler, transistörler, entegre devreler ve günümüzde kuantum teknolojileri içermiştir

Kuantum Hesaplama fikri gerçekten çok etkileyici, ancak gerçek dünyada uygulanması eşit derecede karmaşıktır. kübit'ler, bir Kuantum Bilgisayarının yapı taşıdır. Bu nedenle, bir Kuantum bilgisayarı oluşturmak için ilk görev kübit'leri oluşturmaktır. Bir kübit oluşturmanın birkaç farklı yolu vardır. Genel olarak, süperpozisyonda yaşayabilen şeyler kübit olarak kullanılabilir. Bu, gerçek dünyada kübit olarak uygulanabilen atomların, iyonların, elektronların, fotonların vb. Gibi düşünülebilir Kuantum durumunu daha uzun süre korumak, gerçek dünyadaki Kuantum hesaplamada karşılaşılması gereken bir büyük sorundur. Bilinen yöntemlerden biri, bir kuantum durumu oluşturmak ve sürdürmek için süper iletkenliği kullanır. Bu süper iletken kübitlerle uzun süre çalışmak için çok soğuk tutulmaları gerekir. Sistemdeki herhangi bir ısı hataya neden olabilir, bu nedenle kuantum bilgisayarlar mutlak sıfıra yakın, uzay boşluğundan daha soğuk sıcaklıklarda çalışır

Bu yazı kuantum hesaplamanın gerekliliğinden ve klasik bilgisayarlara kıyasla bize sunduğu avantajlardan bahsedecek. Kuantum hesaplamanın öğelerinin ne olduğunu tartışacak. Bununla birlikte Kuantum hesaplamanın temelleri incelenecektir ve son olarak David Deutsch un algoritma mantığı ispatlanacak

1.Giriş

2021 itibariyle, gerçek kuantum bilgisayarlar henüz beklendiği gibi dünyamızı etkisi altına almamıştır, gelişmeler umut verici ve gelecek daha iyi görünür gibi. Kuantum Hesaplama alanındaki araştırmalar, Kuantum Bilgisayarlarını geliştirmek için birçok askeri kurum ve ulusal hükümet tarafından finanse edilmektedir. Kuantum Hesaplama için teorik ve pratik araştırmalar devam ediyor. Mümkün olan en iyi algoritmalara sahip klasik bilgisayarlar tarafından çözülen problemler, Büyük ölçekli kuantum bilgisayarlar kullanılarak çok daha hızlı bir şekilde çözülebilir. Olası herhangi bir olasılıksal klasik algoritma, Simon'ın algoritması gibi Kuantum algoritmalarından daha yavaş çalışır. Kuantum hesaplaması Church-Turing tezini ihlal etmediği için herhangi bir klasik bilgisayar kuantum algoritmasını kullanabilir.

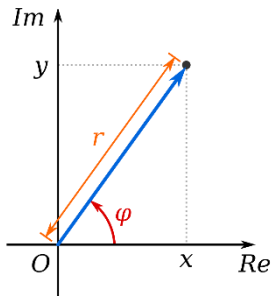
2. Kuantum hesaplama için temel matematik konuları

2.1. Karmaşık saylar

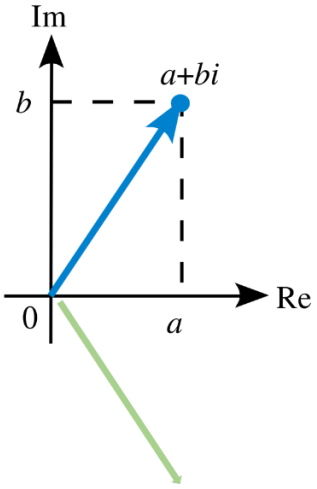
Karmaşık sayı a + ib biçiminde bir sayıdır
a ve b gerçek sayılardır ve i $i^2 = -1$ özelliğine sahip hayali bir birimdir.
a sayısı gerçek kısımdır ve b sayısı sanal kısım olarak adlandırılır ve şöyle yazılır: a + ib
aslında gerçek sayılrıda hayali kısmı sıfır olan bir karmaşık sayı bir karmaşık sayı düşünebiliriz .

karmaşık sayılar kümesini $C = \{a + i\, b \mid a, b \in R, i^2 = -1\}$ biçiminde tanımlaya biliriz
Karmaşık bir sayıyı göstermenin başka bir yolu, kutupsal bir koordinat sistemi kullanmaktır
Bu yöntemde x ve y kullanmak yerine P noktasından orijine olan uzaklığı ve gerçek eksenin pozitif yönü ile OP vektör açısını kullanırız.

$$r = |z| = \sqrt{x^2 + y^2}$$
$$re^{i\varphi} = r(cos\varphi + isin\varphi)$$



2.2.Eşlenek



Karmaşık bir c sayısının eşleniği şu şekilde tanımlanır:

$c = a + ib \;\; = re^{i\theta}$
$c^* = a - ib \;\; c^* = \; re^{-i\theta} = r(\cos\theta - i\sin\theta)$ (polar)
$cc^* = re^{-i\theta} \times re^{i\theta} \;\; = r^2 = |c|^2$

Bir karmaşık sayıyı eşleniği ile çarpmak gerçek bir sayı verir. Bu, kutup vektörünün uzunluğunun (norm) karesidir.

2.3. Vektör uzayı

Matematik, fizik ve mühendislikte, bir vektör uzayı (doğrusal uzay olarak da adlandırılır), matemariksel nesnelerden oluşan bir kümedir bu kümenin her üyesine vektör adı verilir aslında vektör uzayı ölçeklenebilir ve bir biri ile toplanabilir matematiksel nesneler kümesidir Matematikte en yaygın vektör uzayları ve uygulamaları, sırasıyla gerçek sayılar ve karmaşık sayılar alanlarında tanımlanan gerçek vektör uzayları ve karmaşık vektör uzaylarıdır. Vektör uzaylar axiom kurallarını üyeleri arasında ki işlemlerde sağlamalıdır bu kurallar kapalılık toplam ve çarpım şartlarını kapsar
Bir vektör $|a\rangle$ şeklinde gösterilir ve buna arc notation gösterimi denir burda $|a\rangle$ ya bir ”kat” denir
Bir vekör uzayı gösterilmiştir

$$V = \{ |a\rangle, |v\rangle, |w\rangle, |b\rangle, |c\rangle, \dots \}$$

2.3 Axiom kuralları:

toplam altında kapalılık: $|a\rangle + |v\rangle \in V$
$|n\rangle$ vektör uzayının null vektörü düşünülürse o zaman:
$|a\rangle + |n\rangle = |a\rangle$
Her vektrün bir scaler sayı ile çarpımı vektör uzayında kapalıdır
$s|a\rangle \in V$
Eğer vektöre çarpan scaler sayı gerçek bir sayı ise vektör uzayı gerçek bir vektör uzayı olarak adlandırılır
Eğer vektöre çarpan scaler sayı karmaşık bir sayı ise vektör uzayı **karmaşık** bir vektör uzayı olarak adlandırılır
Kuantum hesaplamalar **karmaşık** vektör uzayında çalışır yani $s|a\rangle$ vektör göteriminde s bir gerçek sayı olmaya bilir
Birinci ve üçüncü kurallarda şöyle bir sonuç alına bilir:

$\vec{A} = \alpha|a\rangle + \beta|b\rangle \in V$

$\vec{A} = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \in V \quad (2\ boyut)$

Bir karmaşık vektör uzayı örneği:

$$C^3: \quad V = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}2+i\\7\\0\end{bmatrix}, \begin{bmatrix}0\\-2i\\1\end{bmatrix}, \dots \right\} (3\ boyut)$$

2.4 Vektör uzayı

Bir vektör uzayının tüm vektörlerini scaler çarpım ve toplam ile üretebileceğimiz vektör kümesine Basis vektör kümesi denir

$V = \alpha_1|v_1\rangle + \alpha_2|v_2\rangle + \alpha_3|v_3\rangle$

Olursa $|v_1\rangle, |v_2\rangle, |v_3\rangle$ vektörlerine V vektör uzayını Taban vektörleri denir

Benzer bir şekilde aynı iki boyutlu bir vekörün gösteriminde:

$\overrightarrow{A} = X\hat{\imath} + Y\hat{\jmath}$ gösteriminde $\hat{\imath}\ ve\ \hat{\jmath}$ vektörleri Taban vektör olarak adlandırılır

Bir vektör uzayının Taban vektör kümesini üye sayısına o vektör uzayının boyutu denir

$$|v\rangle = \begin{bmatrix}2i\\3\\-i\end{bmatrix} = 2i\begin{bmatrix}1\\0\\0\end{bmatrix} + 3\begin{bmatrix}0\\1\\0\end{bmatrix} + (-i)\begin{bmatrix}0\\0\\1\end{bmatrix} \rightarrow |v_1\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix}, |v_2\rangle = \begin{bmatrix}0\\1\\0\end{bmatrix}, |v_3\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

$|v\rangle$ Üç boyutlu bir karmaşık vektör uzayıba ait vektör örneğidir

$$|v\rangle = \sum_{j=1}^{n} \alpha_j |v_j\rangle$$

Her vektör uzayının taban vektörleri orthogonal özalliğine sahiptir

2.5 Inner product

$$|v\rangle, |w\rangle \Longrightarrow \langle w | v\rangle = (\langle w|) \cdot (|v\rangle)$$

$$\langle w| = |w\rangle\dagger$$

†: anlama matrisin konjuke ve transposu alınmış halidir

$$|w\rangle = \begin{bmatrix}6\\-2i\\0\end{bmatrix} \Longrightarrow \langle w| = [6 \quad 2i \quad 0]$$

$$\langle w|v\rangle = [\, w_1^* \;\; w_2^* \;\; w_3^*] \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = w_1^*\, v_1 + \; w_2^*\, v_2 + \; w_3^* v_3 = \langle v|w\rangle^*$$

$$\langle v|v\rangle = v_1^*\, v_1 + \; v_2^*\, v_2 + \; v_3^* v_3 = (v_1)^2 + (v_2)^2 + (v_3)^2$$

Karmaşık vektör uzayında birm vektör hesabı:

$|v\rangle_{norm} = \dfrac{1}{\sqrt{\langle v|v\rangle}}\, |v\rangle = \; \left|v'\right\rangle$

$\left\langle v'\middle|v'\right\rangle = 1$ vektör uzayı orthogonal özelliği

Kuantum hesaplamada normalize egdilmiş vektör uzayı üzerinde çalışılır bu yüzden bu kavram çok önemli

2.6 Orthonormal taban vektörleri

$v_1\,, v_2\,, v_3$ vektör uzayının taban vektörleri olduğunu düşünürsek

$V = \{v_1\,, v_2\,, v_3\}$

$\langle v_1|v_1\rangle = 1\,, \langle v_2|v_2\rangle = 1, \langle v_3|v_3\rangle = 1$
$\langle v_1|v_2\rangle = 0\,, \langle v_1|v_3\rangle = 0, \langle v_2|v_3\rangle = 0$

$$\left\langle v_i\middle|v_j\right\rangle = \begin{cases} 0\,, & i \neq j \\ 1, & i = j \end{cases}$$

2.7 Hilbert space (Inner product space)

Hilbert Space vectör space karşın vektörler ile değil fonksyonlar ile uğraşmakta ve bu fonksyonlar sonsuz taban function olabilir Hilbert uzayı, iç çarpım, yani skaler çarpım işlemine sahip lineer bir uzaydır .Kuantum fiziğinde parçacık, fiziksel bir uzayda, yani 3 Boyutlu Öklid Uzayında lokalize olan bir nesne olarak kabul edilir Parçacık,
Hilbert Uzayında vektörler tarafından verilen durumlar, gözlenebilirler veya beklenen değerler cinsinden tanımlanır Böylece, bu uzay, uzaydaki bir kuantum parçacığının olasılık yoğunluğunu bulmaya yardımcı olur

Bir hilbert space in hayali basis foksyonları şöyle gösterilir

$\Psi_1(x), \Psi_2(x), \Psi_3(x), \Psi_4(x), \ldots$. Basisler orthonormal özelliğine sahip

$$f(x): \Psi(x) = C_1\, \Psi_2(x) + C_2\, \Psi_2(x) + C_3\, \Psi_3(x) + \cdots = \sum_{n=1}^{\infty} C_n\, \Psi_n(x)$$

$$f(x): \Psi(x) = \sum_{n=1}^{\infty} C_n \, \Psi_n(x)$$

$$\int \Psi_m^*(x)\, \Psi_n(x) dx = \delta_{mn} \, , \{^{m=1,2,3,4,....}_{n=1,2,3,4,....}$$

$$C_1: \int_{-\infty}^{\infty} \Psi_1^*(x) \Psi(x) dx \quad \Longrightarrow \quad C_n = \langle \Psi_n | \Psi \rangle$$

Fourier trick:

$$\sum_{n=1}^{\infty} |C_n|^2 = 1$$

Tüm kuantum sistemler hilbert space üzerinde çalıştırılabilir

2.8 Linear operatörler

Linear operatorler Quantum kapılarının tanıtımı için kullanılır
A" yı bir linear operatör düşünelim ozaman:

$$A|v\rangle = A|v_1\rangle + A|v_1\rangle + A|v_1\rangle + \cdots = \sum_{i=1}^{n} A|v_i\rangle$$

doğrusal bir operatörün bir matris ile temsil edildiği gösterilebilir

$$|v\rangle = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, A = \begin{bmatrix} a_{11} & \cdots & a_{13} \\ \vdots & \ddots & \vdots \\ a_{31} & \cdots & a_{33} \end{bmatrix}$$

İdentity operator

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

Not operator

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

$$|v\rangle = \alpha|v_1\rangle + \beta|v_2\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

$$|v_1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |v_2\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} ve \sqrt{\alpha^2 + \beta^2} = 1 \Longrightarrow X|v\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \beta|v_1\rangle + \alpha|v_2\rangle = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

$$Y = \begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix} Y - \ pauli \ operator \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} Z - pauli \ operator$$

$$\text{örnek: } \propto_i = \langle v_i|v\rangle \; ve \; |v_n\rangle = \sum_{n=1}^{n} \propto_i \; |v_i\rangle$$

$$\Longrightarrow \sum_{n=1}^{n} |v_i\rangle\langle v_i|v\rangle \Longrightarrow \sum_{n=1}^{n} |v_i\rangle\langle v_i| = I \; (identity \; vector)$$

2.9 Outer product

$$|v\rangle\langle \mathrm{w}| = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \begin{bmatrix} w_1^* & w_2^* & w_3^* \end{bmatrix} = \begin{bmatrix} v_1 w_1^* & v_1 w_2^* & v_1 w_3^* \\ v_2 w_1^* & v_2 w_2^* & v_2 w_3^* \\ v_3 w_1^* & v_3 w_2^* & v_3 w_3^* \end{bmatrix}$$

$$|v\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \; ve \; |w\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Longrightarrow |v\rangle\langle \mathrm{w}| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

2.10 Kuantum hesaplalamada taban vektörler

Hilber uzayında bir kübitin tüm pozisiyonları sadace iki taban vektör ile üretilebilir bu vektörlere kuantum sistemin taban vektörleri denir ve şöyle gösterilir:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \; ve \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$|v\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \alpha \; ve \; \beta \in C \;\; \alpha^2 + \beta^2 = 1$$

$$|0\rangle\langle 0| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$|0\rangle\langle 1| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$|1\rangle\langle 0| = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$|1\rangle\langle 1| = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Taban vektörlerin içsel çarpımının toplamından temel kuantum devrelerin elde edilmesi:

$$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

$$\mathrm{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| = \sum_{\iota=0}^{1} |i\rangle\langle i|$$

2.11 Tensor product

Tensor product İkili quantum sistemleri göstermemi için yardımcı olur

$$|v\rangle = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \; ve \; |w\rangle = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$\langle v|w\rangle \implies inner\ product$$
$$|v\rangle\langle w| \implies outer\ product$$

$$|v\rangle \otimes |w\rangle = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \otimes \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} v_1 w_1 \\ v_1 w_2 \\ v_2 w_1 \\ v_2 w_2 \end{bmatrix} \implies kat\ tensor\ product$$

$$\langle \mathrm{v}| \otimes \langle \mathrm{w}| = [v_1^* \; v_2^*] \otimes [w_1^* \;\; w_2^*] = [v_1^* w_1^* \; v_1^* w_2^* \; v_2^* w_1^* \; v_2^* w_2^*] \implies \text{bra tensor produc}$$

Iki farklı vekör uzayına ait $|v\rangle \; ve \; |w\rangle$ gibi iki farklı kuantum sistemin join state söz konusu olduğunda **tensor product** kullanılır
Tensor product kuantum operatörleri üzerindede uygulanabilir

$$X \otimes Y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ -i & 0 & 0 & 0 \end{bmatrix}$$

$$|0\rangle \otimes |0\rangle = |0\rangle|0\rangle = |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$|0\rangle \otimes |1\rangle = |0\rangle|1\rangle = |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$|1\rangle \otimes |0\rangle = |1\rangle|0\rangle = |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$|1\rangle \otimes |1\rangle = |1\rangle|1\rangle = |00\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$$

3.Kuantum devre tasarımına giriş

3.1 kübit

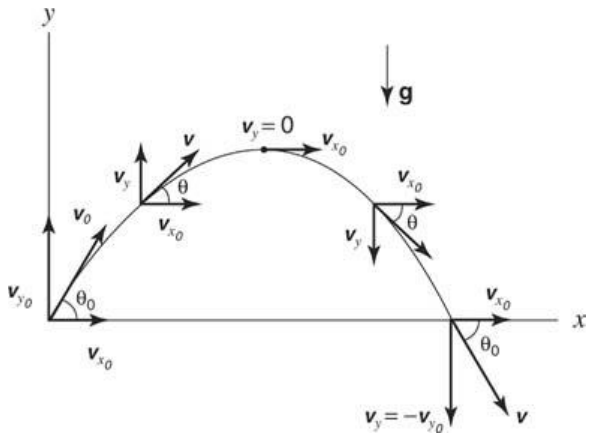
Bir kübit, ölçüm yoluyla erişilebilen bir bitin klasik bilgi içeren oluşumdaki en küçük kuantum birimini temsil eden temel kuantum durumudur. Bir kübiti, "bir" ve "sıfır" durumu olan matematiksel bir nesne olarak kabul ediyoruz bir kübitin değeri kuantum mekanik doğasıdan sürekli belirli aralıklar da değişimde klasik bitler sadece 1 ve 0 değerlerini alırken kübit 1ve 0 aralığındaki tüm değerleri bir alabilir $|q\rangle$, $|0\rangle$ ve $|1\rangle$ taban vektörlerine sahip bir kübitin öeneklemesidir.

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{bmatrix}1\\0\end{bmatrix} + \beta\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix} \quad \alpha^2 + \beta^2 = 1$$

α ve β karmaşık sayılardır. Bu karmaşık sayılara amplitudes (genlik) denir.

3.2 Bloch sphere

önceden söylendiği gibi hilbert space inner product ve normalize vektörler kümesidir
Klasik fizikteki tipik problem, mermi hareketidir. Herhangi bir noktada topun "durumunu" pozisyon ve hıza göre tanımlayabiliriz.
Kuantum mekaniğinde, bir nesnenin durumunu her zaman bir vektörle tanımlarız kuntum vektörleri tanımlayan vektör space'e karmaşık vektör space denilir çünkü newton mekaniğinin aksine bu vektör uzayında karmaşık sayılarda var olur



Kuantum sistemde her zaman ölçümünde bir künbit durumu şöyle gösterilir bunu klasik fizikte bir mermini hareketi gibi eşleyebiliriz şekilde gösterildiği gibi her zaman ölçümünde mermini durumu x,y, ve hıza göre belirlenir bir kübitin durumu ise:

$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\\beta\end{bmatrix} \quad \alpha^2 + \beta^2 = 1$

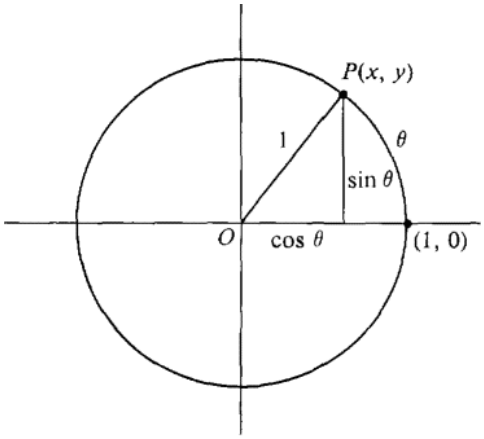
$\alpha, \beta \in C \quad ve\ \alpha = x + x'i \quad \beta = y + y'i$ görüldüğü gibi aslında bir kübitin durumu vektör uzayında iki basis vektör olan yani $|0\rangle$ ve $|1\rangle$ nin doğrusal bir superposisyonu dur Bunedenle $|\Psi\rangle$ förmülünde **dört** gerçek parametre var yani x, x', y, y' iki düzeyli bir kuantum sistemi gösterimi için normal geometrik n-boyutlu gösterimleri kullanılamaz bunu asıl nedeni
$|0\rangle$ ve $|1\rangle$ nin sonucu $\alpha^2 + \beta^2 = 1$ formulune göre birbirnden bağımsız **değildir** halbuki klasik fizkte x ve y birbirinden bağımsız dır .

bu sorunu çözmek için iki düzeyli bir kuantum sistemini parametreleri 4 den euler formulunu kullanarak 3 indigeye bilir

$$\alpha = x + x'i = r_1\, e^{i\varphi_1} \quad \beta = y + y'i = r_2\, e^{i\varphi_2}$$
$$|\Psi\rangle = r_1\, e^{i\varphi_1}|0\rangle + r_2\, e^{i\varphi_2}|1\rangle = e^{i\varphi_1}\,[\; r_1|0\rangle + r_2\, e^{i(\varphi_2-\varphi_1)}\,|1\rangle\;]$$
$$r_1^2 + r_2^2 = 1 \;\; ve\;\varphi = \varphi_2 - \varphi_1 \quad \Longrightarrow \; |\Psi\rangle = e^{i\varphi_1}\,[\; \cos\tfrac{\theta}{2}\,|0\rangle + \sin\tfrac{\theta}{2}\, e^{i\varphi}\,|1\rangle\;]$$

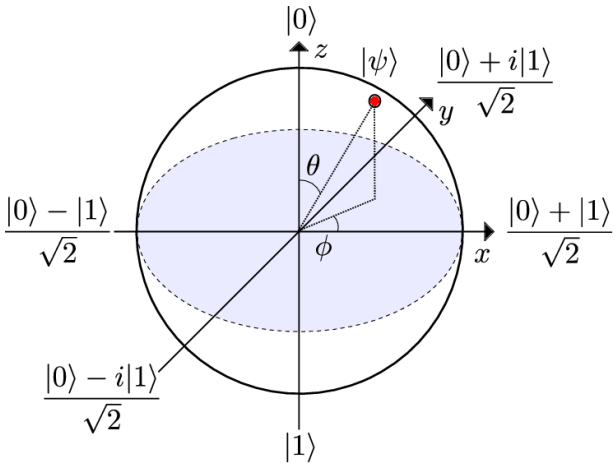
Elde edilen formulde toplam 3 değişken mevcut
$\varphi_1\,, \theta\,, \varphi$
$e^{i\varphi_1}$ çarpanına global phase denir ve hiç bir önemi yok çünkü olasılıkları hesaplarken karmaşık eşleniklerle çarpmamız gerektiğini biliyoruz ve üstel ortalama Ψ 'nin karmaşık eşleniği, üstel eksi i'dır Ψ ve çarpmayı bildiğimizde bu faktör ortadan kalkar

$$|\Psi\rangle = \;[\; cos\,\theta\;|0\rangle + \sin\theta\; e^{i\varphi}\;|1\rangle\;]$$
$$0 \le \varphi \le 2\pi \;\; ve\; 0 \le \theta \le \pi$$



Elde edilen formulde iki düzeyli bir kuantum sistemi sadece iki parametre ile göstermek olur
İki düzeyli kuantum sistemi iki parametre ile göstermek için kullanılan uzaya bloch sphere denir

$\theta = 0 \implies |\Psi\rangle = |0\rangle$
$\theta = 2\pi \implies |\Psi\rangle = |1\rangle \quad \langle 0|1\rangle = 0$
$e^{i\pi} = -1$
$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \langle +|-\rangle = 0$
$|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$
$|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \quad \langle i|-i\rangle = 0$



Normal vektör uzayında orthogonol vektörlerin birbiryle açısı 90 derceyken bloch sphere de bu açı 180 derece olur çünkü blok sphere, iki seviyeli sistemlerimizi görselleştirmek için inşa edilmiş yapay bir yapıdır.

3.3 Tek kübit işlemleri

kübitler vektörlerden başka bir şey değildir. Bazen belirli işlemleri başarılı bir şekilde gerçekleştirmek için bu vektörlerin yönünü değiştirmemiz gerekebilir, işte burada kuantum kapıları devreye giriyor! Kapılar, girdiyi ihtiyaçlarımıza göre manipüle eden işlemlerdir. Mantık, klasik hesaplama sisteminde tanımlanmış v OR, AND, NOT, XOR, Gibi birçok kapı vardır.

3.4 Unitary Transform

Klasik kapılar ile kuantum kapıları arasındaki en büyük fark budur. Klasik kapılar tersine çevrilemezken, kuantum kapıları tersine çevrilebilir
Görünüşe göre doğa keyfi durum dönüşümüne izin vermiyor. Kapalı bir kuantum mekanik sisteminin durumlarının değişmesine , matematiksel olarak üniter dönüşümler olarak bilinen ,belirli bir dönüşüm sınıfı neden olur. Üniter bir dönüşüm U : H → H bir izomorfizmadır, burada H bir iç çarpım uzayıdır (Hilbert uzayı).
Unitary Matrix:
Üniter bir dönüşüm, üniter bir matris ile temsil edilecektir. Eşlenik devriği (Hermityen devrik) U† olan , karmaşık bir U matrisini üniter olarak adlandırırsak ozaman:

$$U^{\dagger}U = UU^{\dagger} = I$$

$$U^{\dagger} = (U^{T})^{*} = U^{-1}$$

$$|\Psi'\rangle = U|\Psi\rangle \quad \langle\Psi'|\Psi'\rangle = 1 \;\; \langle\Psi|\Psi\rangle = 1$$

$$\langle\Psi'| = (U|\Psi\rangle)^{\dagger} = |\Psi\rangle^{\dagger}U^{\dagger} = \langle\Psi|U^{\dagger} \Longrightarrow \langle\Psi'|\Psi'\rangle = \langle\Psi|U^{\dagger}U|\Psi\rangle = \langle\Psi|I|\Psi\rangle = \langle\Psi|\Psi\rangle = 1$$

$$U|q_1\rangle = |q_2\rangle ve\; U^{\dagger}|q_2\rangle = |q_1\rangle \Longrightarrow reversible\; system$$

$$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0| \;\; X^{\dagger} = (X^{T})^{*} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Longrightarrow XX^{\dagger} = X^{2} = I \implies |0\rangle\, X^{2} = |0\rangle$$

Yani 1 kübitlik bir üniter dönüşüm veya 1-kübit kapısı, 2 × 2 üniter matris U'dur Üniter U operatörü tanım gereği doğrusaldır
Bir kuantum durumu $|\Psi\rangle$ 'nin bir süperpozisyonu ise

$$|\Psi\rangle = a_1|\Psi_1\rangle + a_2|\Psi_2\rangle + a_3|\Psi_3\rangle + a_4|\Psi_4\rangle + \cdots . + a_n|\Psi_n\rangle$$

$$U|\Psi\rangle = U(a_1|\Psi_1\rangle + a_2|\Psi_2\rangle + a_3|\Psi_3\rangle + a_4|\Psi_4\rangle + \cdots . + a_n|\Psi_n\rangle)$$

$$\Longrightarrow U|\Psi\rangle = a_1U|\Psi_1\rangle + a_2U|\Psi_2\rangle + a_3U|\Psi_3\rangle + a_4U|\Psi_4\rangle + \cdots . + a_nU|\Psi_n\rangle$$

Üniter bir operatör, bir durum vektörünün normunu veya uzunluğunu korur ve Bir ortonormal tabanı başka bir ortonormal tabana dönüştürür.

3.5 Hadamard kapısı

$$H = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} : \{|0\rangle, |1\rangle\} \rightarrow \{|+\rangle, |-\rangle\}$$

$$HH^{\dagger} = H^{\dagger}H = H^2 = I$$

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

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

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

$$H|-\rangle = |1\rangle$$

$$H|+\rangle = |0\rangle$$

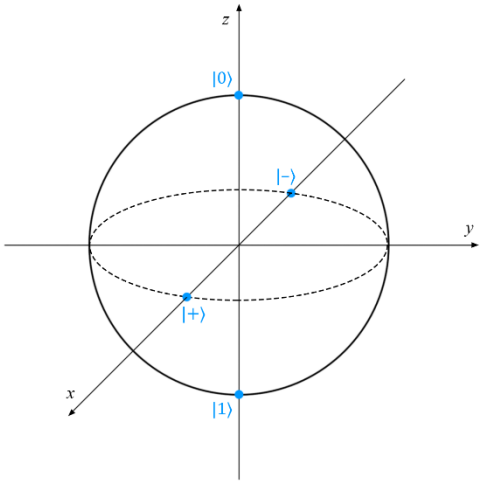
Hadamard kapısı bir kübiti superpozisiyon durumuna getirmek için kullanılır bu durumunda ikili quantum sisteminde her iki taban vektörün çarpanı $1/\sqrt{2}$ ye eşittir ve olasılık açısındanda her iki durumun %50 olasılıklığı var Hadamard kapısı, Y ekseni etrafında 90º dönüş ve ardından X ekseni etrafında 180º dönüş olarak da ifade edilebilir

$H|0\rangle = |+\rangle$

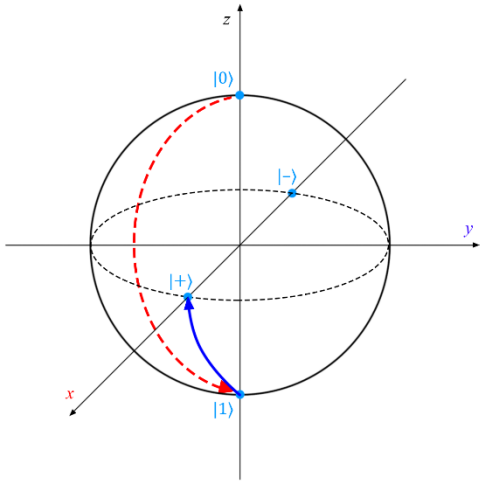
$H|1\rangle = |-\rangle$

$H|+\rangle = |0\rangle$

$H|-\rangle = |1\rangle$

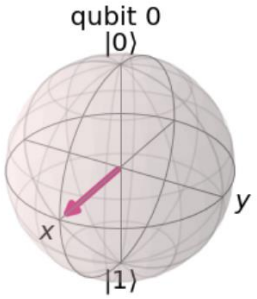


Placement on the Bloch Sphere



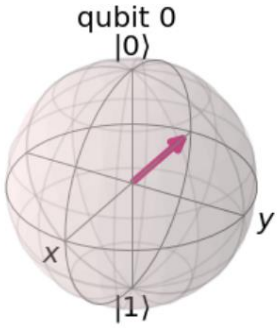
$H|0\rangle = |+\rangle$

$H = XY^{1/2}$

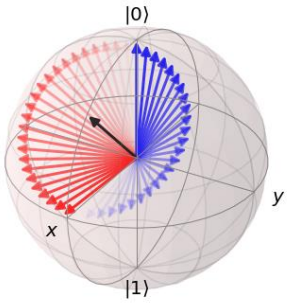


$Y^{1/2}$, Y ekseni etrafında 90 derecelik dönüş anlamına gelir (yarım Y dönüşü)

ikili sistemin $|0\rangle$ durumunda olduğunu düşünelim. Yarım Y dönüşü sistemin durumun $|+\rangle$ noktasına götürür. X dönüşü hiçbir şeyi değiştirmeyecek. Tam tersi: $|+\rangle$ konumundaysak, $Y^{1/2}$ $|1\rangle$'e ve X $|0\rangle$ durumuna yol açar.



ikili sistemin $|1\rangle$ durumunda olduğunu düşünelim. Yarım Y dönüşü sistemin durumun $|-\rangle$ noktasına götürür. X dönüşü hiçbir şeyi değiştirmeyecek.



Görüldüğü gibi hadamard kapısı hilbert uzayındaki her noktayı belirli bir harekete göre konumunu taşımakta

Hadamard kapısı Qiskit

kübite başlangıç değeri atanmazsa başlangıç değer $|-\rangle$ olur

In [167]:
```
from qiskit import QuantumCircuit, Aer, assemble
import numpy as np
from qiskit.visualization import plot_histogram, plot_bloch_multivector

qc = QuantumCircuit(1)
qc.h(0)

qc.draw(output='mpl')
```

Out[167]:

In [168]:
```
svsim = Aer.get_backend('aer_simulator')
qc.save_statevector()
qobj = assemble(qc)
final_state = svsim.run(qobj).result().get_statevector()
# Print the statevector neatly:
array_to_latex(final_state, prefix="\\text{Statevector = }")
```

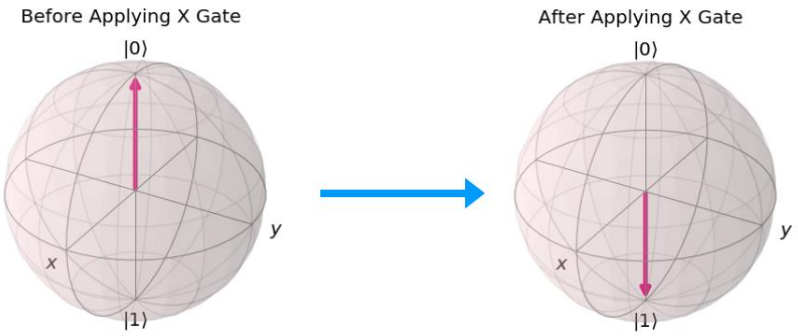
Out[168]: $\text{Statevector} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$

3.6 Pauli Operatörleri

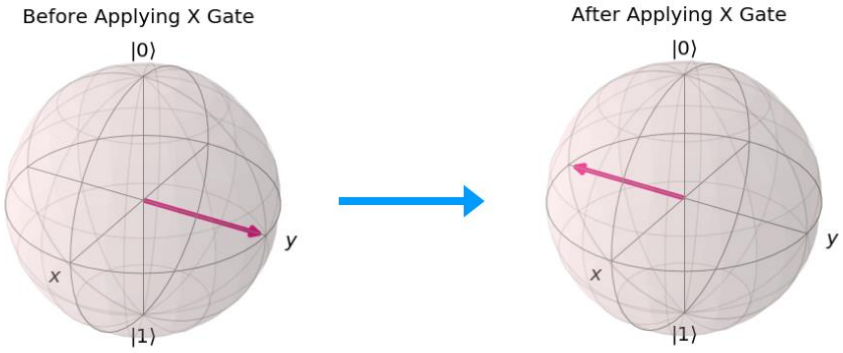
3.6.1 X Kapısı

Önceden söylendiği gibi Pauli X operatörü, girdi kübitini tersine çevirir. Klasik NOT kapısına benzer. Bu işlem aynı zamanda bit çevirme işlemi olarak da bilinir
Pauli-X kapısı, x ekseni etrafındaki π radyanlar boyunca tek kübitlik bir dönüştür.
X Kapısı, Bit flip Kapısı olarak da bilinir.
$\Psi = \alpha|0\rangle + \beta|1\rangle$ durumundaki bir Qubit için, X Gate uygulaması Qubit'in durumunu $\Psi = \beta|0\rangle + \alpha|1\rangle$ olarak değiştirecektir. X Kapısı Qubit'e uygulanmadan önce, $|0\rangle$ ve $|1\rangle$ alma olasılıkları sırasıyla $|\alpha|^2$ ve $|\beta|^2$ idi. X Kapısı Qubit'e uygulandıktan sonra, $|0\rangle$ ve $|1\rangle$alma olasılıkları sırasıyla $|\beta|^2$ ve $|\alpha|^2$ olur.

$$X|0\rangle = |1\rangle$$
$$X|1\rangle = |0\rangle$$



$|0\rangle$ durumundaki Qubit'lere X Kapısı Uygulamak



$$X|i\rangle = |-i\rangle$$
$$X|-i\rangle = |i\rangle$$

$|i\rangle$durumundaki Qubit'lere X Kapısı Uygulamak

X Kapısı, X ekseni etrafında dönüş yaptığından, X ekseni üzerinde bulunan bir vektör üzerinde hiçbir etkisi olmayacaktır
X kendi tersidir. Bu nedenle, X Kapısını aynı Qubit'e iki kez uygulamak, Qubit'in orijinal durumuna yol açacaktır

$$XX^{\dagger} = X^2 = I$$

$$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

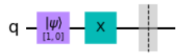
$$\mathrm{X}|0\rangle = |0\rangle\langle 1|0\rangle + |1\rangle\langle 0|0\rangle = |1\rangle$$
$$\langle 1|0\rangle = 0 \; ve \; \langle 0|0\rangle = 1$$

$$\mathrm{X}(\alpha|0\rangle + \beta|1\rangle) = \beta|0\rangle + \alpha|1\rangle$$

Qiskit'teki X Kapısı, Quantum Circuit (QuantumCircuit sınıfının bir örneği) üzerindeki x() yöntemini çağırarak ve X Gate'in uygulanacağı Qubit için bir tamsayı ileterek herhangi bir Qubit'e uygulanabilir.

In [42]:
```
from qiskit import QuantumCircuit, QuantumRegister,assemble, Aer, transpile
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt, pi
from qiskit.tools.visualization import circuit_drawer
from qiskit.visualization import *
from qiskit.quantum_info import state_fidelity
from qiskit import *
from qiskit import Aer

qreg_q = QuantumRegister(1,'q')  #register tanımlama
sim = Aer.get_backend('aer_simulator')  # imulasyon yöntemini belirleme
initial_state=[1,0]

qc = QuantumCircuit(qreg_q)   #devra tanım
qc.initialize(initial_state)
qc.x(0)
qc.save_statevector()   # simülator save
qobj = assemble(qc)     # simülator obje yaratım
qc.draw(output='mpl')
```

Out[42]:

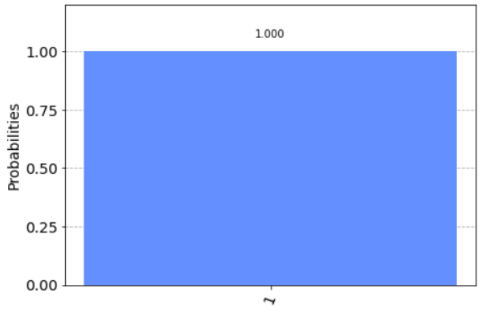


In [45]:
```
state = sim.run(qobj).result().get_statevector()
print("State of Measured Qubit = " + str(state))
```

```
State of Measured Qubit = [0.+0.j 1.+0.j]
```

In [46]:
```
qobj = assemble(qc)
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)
```

Out[46]:



3.6.2 Y Kapısı

Pauli-Y kapısı, y ekseni etrafında π radyan boyunca tek kübitlik bir dönüştür. Pauli Y kapısında Pauli X kapısıyla aynı sonucu elde ederiz, ancak gerçek uzayda hareket etmek yerine hayali uzayda hareket ederiz. Y Kapısı, flip ve Faz çevirme Kapısı olarak da bilinir. Bunun nedeni, X Gate gibi bir bit çevirme ve Z Gate gibi bir Faz çevirme gerçekleştirmesidir

$$YY^{\dagger} = Y^2 = I$$

$$\mathrm{X} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0|$$

$$\mathrm{Y}|0\rangle = -i|0\rangle\langle 1|0\rangle + i|1\rangle\langle 0|0\rangle = \ i|1\rangle$$

$$\mathrm{Y}|1\rangle = -i|0\rangle\langle 1|1\rangle + i|1\rangle\langle 0|1\rangle = \ -i|1\rangle$$

$$\mathrm{Y}|i\rangle = \ |-i\rangle$$

$$\mathrm{Y}|-i\rangle = \ |i\rangle$$

$$\langle 1|0\rangle = 0 \ ve \ \langle 0|0\rangle = 1$$

$$\mathrm{Y}(\alpha|0\rangle + \beta|1\rangle) = \ -i\beta|0\rangle + i\alpha|1\rangle$$

$\Psi = \alpha|0\rangle + \beta|1\rangle$ durumundaki bir Qubit için, Y Gate uygulaması Qubit'in durumunu $\Psi = -i\beta|0\rangle + i\alpha|1\rangle$ olarak değiştirecektir. Y Kapısı Qubit'e uygulanmadan önce, $|0\rangle$ ve $|1\rangle$ alma olasılıkları sırasıyla $|\alpha|^2$ ve $|\beta|^2$ idi. Y Kapısı Qubit'e uygulandıktan sonra, $|0\rangle$ ve $|1\rangle$alma olasılıkları sırasıyla $|\beta|^2$ ve $|\alpha|^2$ olur.

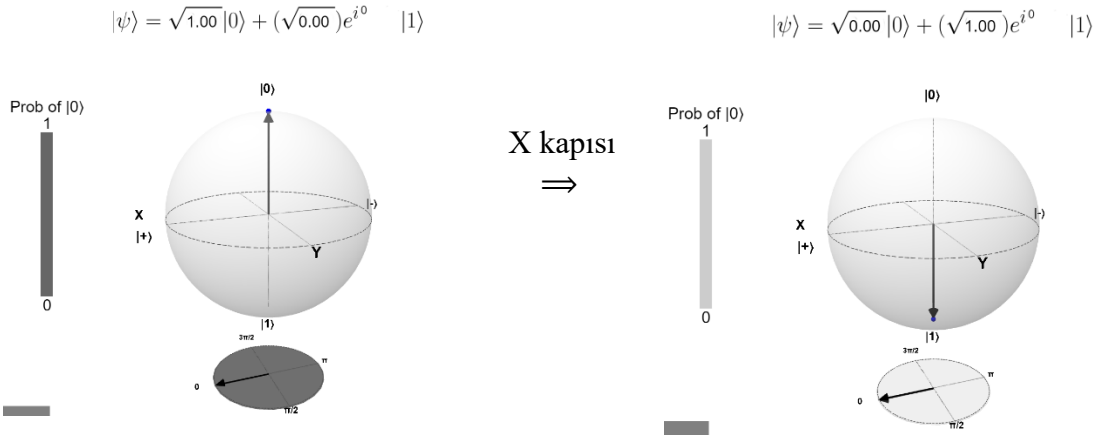
```
In [92]: from qiskit import QuantumCircuit, QuantumRegister,assemble, Aer, transpile
         import matplotlib.pyplot as plt
         import numpy as np
         from math import sqrt, pi
         from qiskit.tools.visualization import circuit_drawer
         from qiskit.visualization import plot_histogram, plot_bloch_vector
         from qiskit.quantum_info import state_fidelity
         from qiskit import BasicAer

         qreg_q = QuantumRegister(1,'q')  #register tanımlama
         sim = Aer.get_backend('aer_simulator')  # imulasyon yöntemini belirleme
         qc = QuantumCircuit(qreg_q)   #devra tanımı
         initial_state = [1,0]   # kübitin ilk durumunu belirleme |0>
         qc.initialize(initial_state, 0)
         qc.y(qreg_q)           #devreye Y kapısı uygulama
         qc.save_statevector()   # simülator save
         qobj = assemble(qc)     # simülator obje yaratımı
         qc.draw(output='mpl')
```

Out[92]:

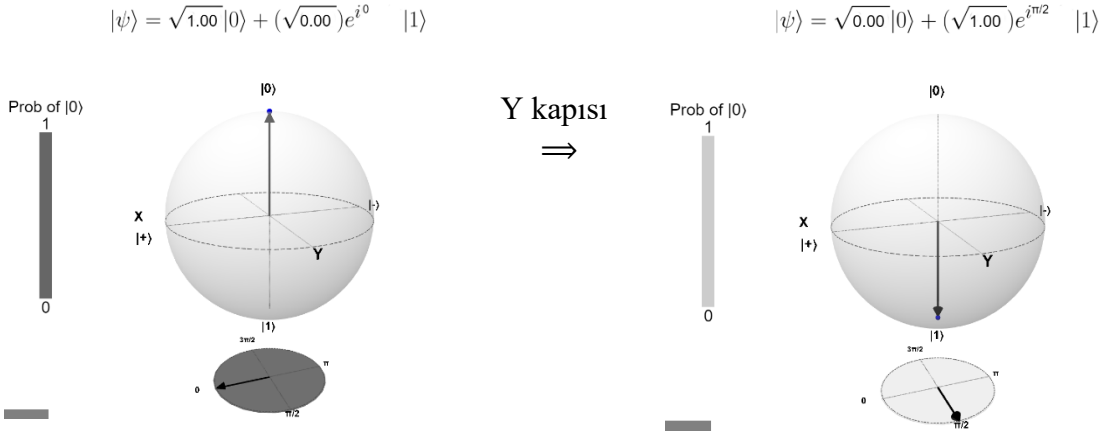

```
In [93]: out_state = result.get_statevector()
         print(out_state) # kubit durum vektor gosterimi

         [0.-0.j 0.+1.j]
```

Python karmaşık sayılarda i simgesi yerine j kullanıyor yukarıdaki kod örneğinde $\mathrm{Y}|0\rangle = \ i|1\rangle$ uygulanmış oluşan registera ilk $|0\rangle$ durum değeri atanmış ve sonra pauli-Y kapısı uygulanmış sonuç [0.-0.j 0.+1.j] şeklinde gösterilmiş bu aslında $\begin{bmatrix} 0 - 0.i \\ 0. + 1.i \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix}$

Y-Pauli ve X-Pauli arasındaki fark nedir?
Her iki kapıda 180 derecelik bir rotasyona neden olur Y kapısı Y ekseni etrafında ve X kapısı X ekseni etrafında fark olan faz açısı değişimidir bu anlamak için durum vektörlerini bloch sphere uzayında göstermek daha yararlı olur



Görüldüğü gibi X kapısı uygulandığında faz açısında hiç bir değişim yok



Y kapısı uygulandığında faz açısı $\frac{\pi}{2}$ değişmekte başlangıç durumuna bağlı Y kapısı rotasyonlarında faz açısı değişimi farklı olabilir

3.6.3 Z Kapısı

Pauli-Y kapısı, y ekseni etrafında π radyan boyunca tek kübitlik bir dönüştür Standart temel çiftleri $|0\rangle$ ve $|1\rangle$cinsinden temsil edilen bir durum üzerindeki etkisi bakımından, $|0\rangle$ ila $|1\rangle$katsayıları üzerinde hiçbir etkisi yoktur. Bu nedenle, Z Kapısının uygulanması, kubit'in durumunun ölçüm üzerine $|1\rangle$ veya $|0\rangle$'e superpozisyon olasılığını değiştirmez.

$$ZZ^{\dagger} = Z^2 = I$$

$$\mathrm{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

$$\mathrm{Z}|0\rangle = |0\rangle\langle 0|0\rangle - |1\rangle\langle 1|0\rangle = |0\rangle$$

$$\mathrm{Z}|1\rangle = |0\rangle\langle 0|1\rangle - |1\rangle\langle 1|1\rangle = -|1\rangle \equiv |1\rangle$$

$$\mathrm{Z}|i\rangle = |-i\rangle$$

$$\mathrm{Z}|-i\rangle = |i\rangle$$

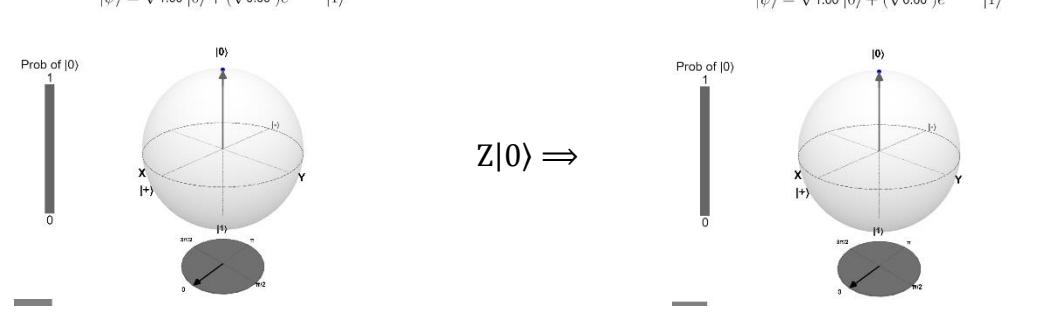
$$\langle 1|0\rangle = 0 \; ve \; \langle 0|0\rangle = 1$$

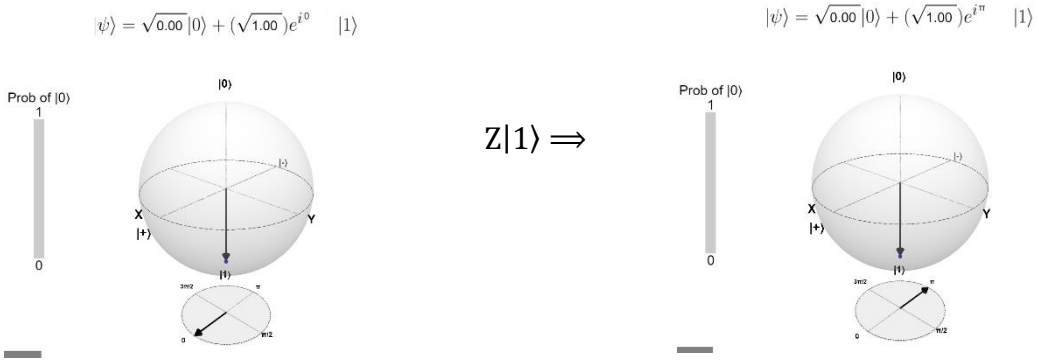
$$\mathrm{Z}(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle$$

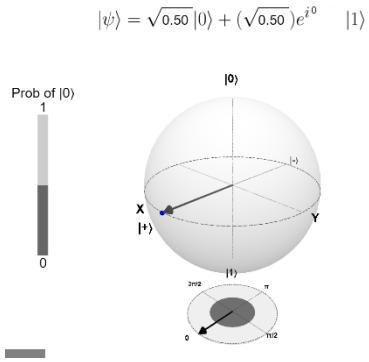
$$Z\begin{bmatrix}\alpha \\ \beta\end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}\ e^{i\varphi} \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ -\sin\frac{\theta}{2}\ e^{i\varphi} \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ -\sin\frac{\theta}{2}(\cos\varphi + i\sin\varphi) \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}(-\cos\varphi - i\sin\varphi) \end{bmatrix}$$

$$\begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}(\cos(\varphi+\pi) + i\sin(\varphi+\pi)) \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta'}{2} \\ \sin\frac{\theta'}{2}(\cos(\varphi'+\pi) + i\sin(\varphi'+\pi)) \end{bmatrix} \Longrightarrow \theta' = \theta \;\; ve \;\; \varphi' = \varphi + \pi$$

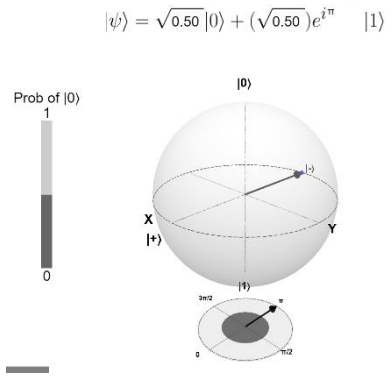
İspatlandığı gibi Pauli-Z kapısı, kuantum durumlarını Z ekseni etrafında 180º döndürür. Pauli-Z kapısının doğası gereği, genellikle faz flip kapısı olarak adlandırılır.

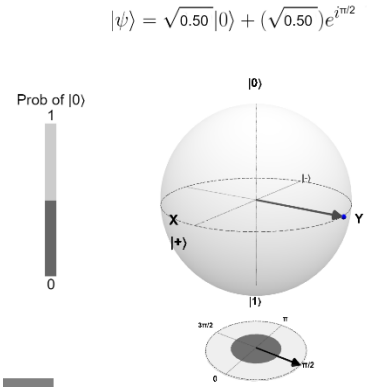




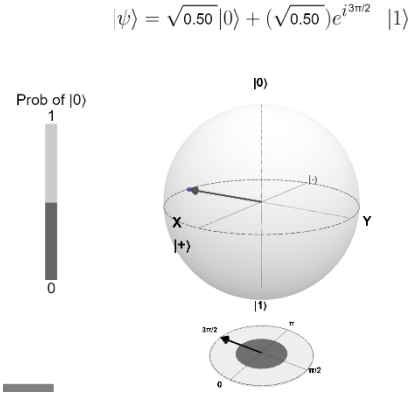


$Z|+\rangle \Longrightarrow$





$Z|i\rangle \Longrightarrow$



Örnek kodda $Z|+\rangle = |-\rangle$ incelenmiştir

In [132]:
```
from qiskit import QuantumCircuit, QuantumRegister,assemble, Aer, transpile
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt, pi
from qiskit.tools.visualization import circuit_drawer
from qiskit.visualization import plot_histogram, plot_bloch_vector
from qiskit.quantum_info import state_fidelity
from qiskit import BasicAer

qreg_q = QuantumRegister(1,'q')  #register tanımlama
sim = Aer.get_backend('aer_simulator')  # imulasyon yöntemini belirleme
qc = QuantumCircuit(qreg_q)   #devra tanımı
initial_state = [1/sqrt(2), 1/sqrt(2)]   # kübitin ilk durumunu belirleme |+>
qc.initialize(initial_state, 0)
qc.z(qreg_q)            #devreye Z kapısı uygulama
qc.save_statevector()   # simülator save
qobj = assemble(qc)     # simülator obje yaratımı
qc.draw(output='mpl')
```

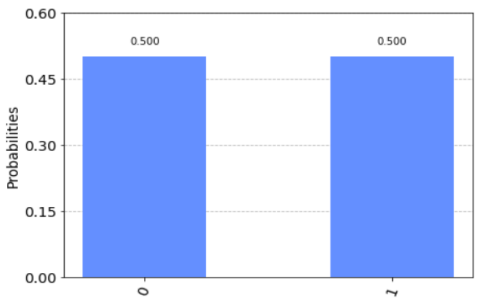
Out[132]:

In [133]:
```
state = sim.run(qobj).result().get_statevector()
print("State of Measured Qubit = " + str(state))
```

```
State of Measured Qubit = [ 0.70710678+0.j -0.70710678+0.j]
```

```
In [134]: qobj = assemble(qc)
          result = sim.run(qobj).result()
          counts = result.get_counts()
          plot_histogram(counts)
```

Out[134]:



$|-\rangle$ durumunda her iki taban vektörün olasılığı eşittir

3.7 Phase kapısı

Faz kapısı, Qubit'in fazını belirli bir miktarda değiştirir. Ancak, standart temel çiftleri $|0\rangle$ ve $|1\rangle$cinsinden temsil edilen bir durum üzerindeki etkisi bakımından, $|0\rangle$ila $|1\rangle$katsayıları üzerinde hiçbir etkisi yoktur. Bu nedenle, Faz Kapısının uygulanması, Qubit'in durumunun ölçüm üzerine $|0\rangle$veya $|1\rangle$'e çökme olasılığını değiştirmez. Z Kapısı, parametrenin (faz değişimi) 180° veya π radyan olduğu özel bir Faz Kapısı durumudur. P kapısı, Z ekseni yönü etrafında φ dönüşü gerçekleştirir. Qiskit kütüphanesinde ki p(phi, qubit) fonksyonunda phi açı ve reel sayıdır qubit parameter değeri ise seçilen qubitin indexi dir.

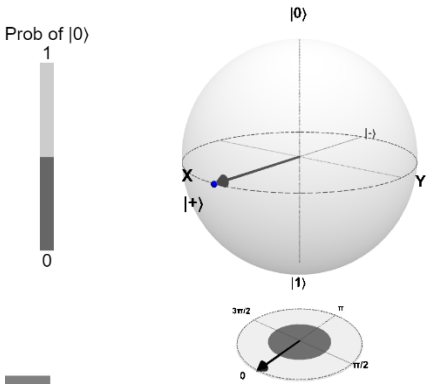
$$P(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}$$

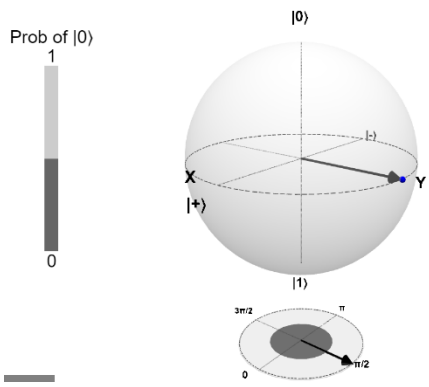
φ,uygulanması gereken açı faz farkını temsil eder.

$$\Psi = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = |+\rangle \; ve \; \left(\varphi = \frac{\pi}{2}\right) olsun \implies \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ i/\sqrt{2} \end{bmatrix} = |i\rangle$$

$$e^{i\frac{\pi}{2}} = i$$

$$|\psi\rangle = \sqrt{0.50}\,|0\rangle + (\sqrt{0.50})e^{i0} \quad |1\rangle \qquad\qquad |\psi\rangle = \sqrt{0.50}\,|0\rangle + (\sqrt{0.50})e^{i\pi/2} \quad |1\rangle$$





In [135]:
```
from qiskit import QuantumCircuit, QuantumRegister,assemble, Aer, transpile
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt, pi
from qiskit.tools.visualization import circuit_drawer
from qiskit.visualization import plot_histogram, plot_bloch_vector
from qiskit.quantum_info import state_fidelity
from qiskit import BasicAer

qreg_q = QuantumRegister(1,'q')  #register tanımlama
sim = Aer.get_backend('aer_simulator')  # imulasyon yöntemini belirleme
qc = QuantumCircuit(qreg_q)   #devra tanımı
initial_state = [1/sqrt(2), 1/sqrt(2)]   # kübitin ilk durumunu belirleme |+>
qc.initialize(initial_state, 0)
qc.p(pi/2, 0)            #devreye P kapısı uygulama
qc.save_statevector()   # simülator save
qobj = assemble(qc)     # simülator obje yaratımı
qc.draw(output='mpl')
```

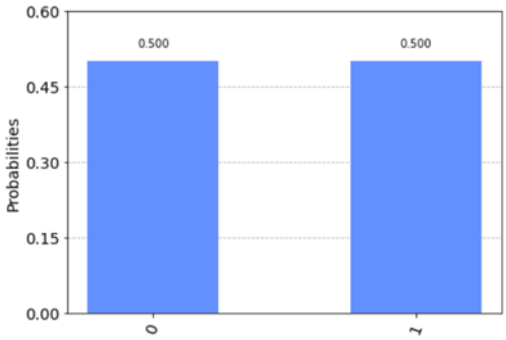
Out[135]:

In [137]:
```
state = sim.run(qobj).result().get_statevector()
print("State of Measured Qubit = " + str(state))
```

```
State of Measured Qubit = [7.07106781e-01+0.j         4.32978028e-17+0.70710678j]
```

In [138]:
```
qobj = assemble(qc)
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)
```

Out[138]:



Yukardaki kodda gösterilmiştir

$$\Psi = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = |+\rangle \; ve \; \left(\varphi = \frac{\pi}{2}\right) olsun \Longrightarrow \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ i/\sqrt{2} \end{bmatrix} = |i\rangle$$

3.8 S kapısı

S kapısı, Qubit'in fazını 90° veya π/2 radyan değiştirir. |0⟩ ila |1⟩ katsayıları üzerinde hiçbir etkisi yoktur. Bu nedenle, S Gate uygulaması, Qubit'in durumunun ölçüm üzerine |0⟩ veya |1⟩'e superposition olasılığını değiştirmez.S kapısı $\sqrt{Z}$ kapısı olarakta tanımlanır.Skapısı aslında 90 faz deşişimine sahip olan bir P kapısıdır yukarıda örneklenmiştir.

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

Bir S Kapısının tersi, $S^\dagger$ Kapısıdır. S Kapısı, Bloch Sphere üzerindeki Z ekseni etrafında saat yönünün tersine 90° veya π/2 radyan dönüş üretir.
$S^\dagger$ Kapısı, Bloch Sphere üzerindeki Z ekseni etrafında saat yönünde 90° veya π/2 radyan dönüş efekti üretir. S Kapısı uygulandıktan sonra $S^\dagger$ Kapısı uygulanması, Qubit'in durumunu orijinal durumuna değiştirdiğinden, $S^\dagger$ Kapısı, S Kapısının tersidir . S kapısı qiskit kütüphanesinde
s (qubit) ve $S^\dagger$ kapısı sdg(qubit) fonksyonlarını kullanarak uygulanır.

In [143]:
```
from qiskit import QuantumCircuit, QuantumRegister,assemble, Aer, transpile
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt, pi
from qiskit.tools.visualization import circuit_drawer
from qiskit.visualization import plot_histogram, plot_bloch_vector
from qiskit.quantum_info import state_fidelity
from qiskit import BasicAer

qreg_q = QuantumRegister(1,'q')  #register tanımlama
sim = Aer.get_backend('aer_simulator')  # imulasyon yöntemini belirleme
qc = QuantumCircuit(qreg_q)   #devra tanım
initial_state = [1/sqrt(2), 1/sqrt(2)]   # kübitin ilk durumunu belirleme |+>
qc.initialize(initial_state, 0)
qc.s(0)   #devreye S kapısı uygulama
qc.sdg(0)  #devreye sdg kapısı uygulama
qc.save_statevector()   # simülator save
qobj = assemble(qc)     # simülator obje yaratım
qc.draw(output='mpl')
```

Out[143]:

In [144]:
```
state = sim.run(qobj).result().get_statevector()
print("State of Measured Qubit = " + str(state))
```

```
State of Measured Qubit = [0.70710678+0.j 0.70710678+0.j]
```

$$|\Psi\rangle = |+\rangle \implies SS^\dagger|+\rangle = |+\rangle$$

3.9 T Kapısı

T kapısı, kubit'in fazını 45° veya π/4 radyan değiştirir. |0⟩ ila |1⟩ katsayıları üzerinde hiçbir etkisi yoktur. Bu nedenle, T Gate uygulaması, Qubit'in durumunun ölçüm üzerine |0⟩ veya |1⟩'e superposition olasılığını değiştirmez T kapısı $\sqrt[4]{Z} = \sqrt[2]{S}$ kapısı olarakta tanımlanır.T kapısı aslında 45 faz deşişimine sahip olan bir P kapısıdır

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$$

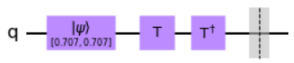
$$e^{i\frac{\pi}{4}} = \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}$$

$$|q\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \Longrightarrow T|q\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1+i}{\sqrt{2}} \end{bmatrix}$$

Bir T Kapısının tersi, tdg Kapısı ve yaT† Kapısı olarak tanımlanır. T Kapısı, Bloch Sphere üzerindeki Z ekseni etrafında saat yönünün tersine 45° veya π/4 radyan dönüş üretir. T† Kapısı, Bloch Sphere üzerindeki Z ekseni etrafında saat yönünde 45° dönüş üretir. T† Kapısı uygulandıktan sonra T† Kapısı uygulanması, Qubit'in durumunu orijinal durumuna değiştirdiğinden, T† Kapısı, T Kapısının tersidir

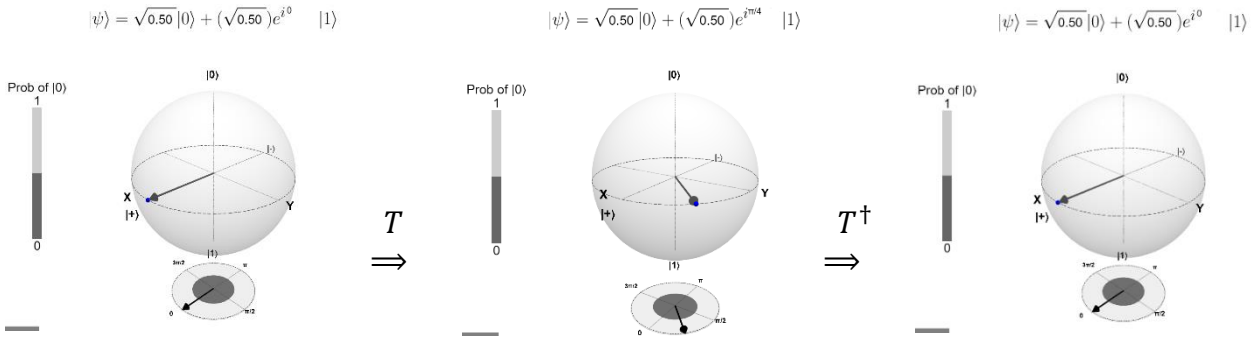
In [145]:
```
from qiskit import QuantumCircuit, QuantumRegister,assemble, Aer, transpile
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt, pi
from qiskit.tools.visualization import circuit_drawer
from qiskit.visualization import plot_histogram, plot_bloch_vector
from qiskit.quantum_info import state_fidelity
from qiskit import BasicAer

qreg_q = QuantumRegister(1,'q')  #register tanımlama
sim = Aer.get_backend('aer_simulator')  # imulasyon yöntemini belirleme
qc = QuantumCircuit(qreg_q)   #devra tanımı
initial_state = [1/sqrt(2), 1/sqrt(2)]   # kübitin ilk durumunu belirleme |+>
qc.initialize(initial_state, 0)
qc.t(0)   #devreye S kapısı uygulama
qc.tdg(0)  #devreye sdg kapısı uygulama
qc.save_statevector()   # simülator save
qobj = assemble(qc)     # simülator obje yaratımı
qc.draw(output='mpl')
```

Out[145]:



In [146]:
```
state = sim.run(qobj).result().get_statevector()
print("State of Measured Qubit = " + str(state))
```

```
State of Measured Qubit = [0.70710678+0.j 0.70710678+0.j]
```



$$|\Psi\rangle = |+\rangle \implies TT^{\dagger}|+\rangle = |+\rangle$$

3.10 U kapısı

U kapısı, tüm tek kübit kuantum kapılarının en genelidir.

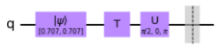
$$U(\theta,\varphi,\gamma) = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\gamma}\sin\frac{\theta}{2} \\ e^{i\varphi}\sin\frac{\theta}{2} & e^{i(\varphi+\gamma)}\cos\frac{\theta}{2} \end{bmatrix}$$

$$U\left(\frac{\pi}{2},0,\pi\right) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H$$

$$U(0,0,\gamma) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\gamma} \end{bmatrix} = P$$

In [150]:
```
from qiskit import QuantumCircuit, QuantumRegister,assemble, Aer, transpile
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt, pi
from qiskit.tools.visualization import circuit_drawer
from qiskit.visualization import plot_histogram, plot_bloch_vector
from qiskit.quantum_info import state_fidelity
from qiskit import BasicAer

qreg_q = QuantumRegister(1,'q')  #register tanımlama
sim = Aer.get_backend('aer_simulator')  # imulasyon yöntemini belirleme
qc = QuantumCircuit(qreg_q)   #devra tanımı
initial_state = [1/sqrt(2), 1/sqrt(2)]   # kübitin ilk durumunu belirleme |+>
qc.initialize(initial_state, 0)
qc.t(0)   #devreye T kapısı uygulama
qc.u(pi/2, 0, pi, 0)   #devreye U kapısı uygulama
qc.save_statevector()   # simülator save
qobj = assemble(qc)     # simülator obje yaratımı
qc.draw(output='mpl')
```

Out[150]:

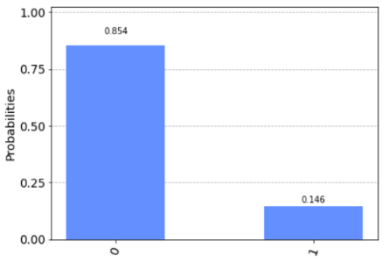


In [151]:
```
state = sim.run(qobj).result().get_statevector()
print("State of Measured Qubit = " + str(state))
```

```
State of Measured Qubit = [0.85355339+0.35355339j 0.14644661-0.35355339j]
```

In [149]:
```
qobj = assemble(qc)
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)
```

Out[149]:



Verilen kod örneğinde devreye ilk olarak T kapısı ve sonra U kapısı uygulanmış en sonda ise kubitin son durumu ölçülmüş qubitin ilk durum vektörü $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ son durum vektörü $\begin{bmatrix} 0.85 + 0.35i \\ 0.14 - 0.35i \end{bmatrix}$ olur

3.11 Multi-Qubit kapılar

İki farklı uzaydan olan iki kübit için karşılık gelen iki kübitlik durum tensör çarpımı tarafından oluşturulur

$$|v\rangle \otimes |w\rangle = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \otimes \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} v_1 w_1 \\ v_1 w_2 \\ v_2 w_1 \\ v_2 w_2 \end{bmatrix} \Longrightarrow kat\ tensor\ product$$

iki kübitin durumunu tanımlamak için dört karmaşık genlik gerekir. Bu genlikleri şu şekilde bir 4lu sütün vektörde saklarız

$$|v\rangle = \begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix} = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$$

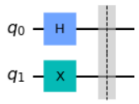
$$|a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1$$

$$|ba\rangle = |b\rangle \otimes |a\rangle = \begin{bmatrix} b_0 a_0 \\ b_0 a_1 \\ b_1 a_0 \\ b_1 a_1 \end{bmatrix}, |bac\rangle = |b\rangle \otimes |a\rangle \otimes |c\rangle = \begin{bmatrix} b_0 a_0 c_0 \\ b_0 a_0 c_1 \\ b_0 a_1 c_0 \\ b_0 a_1 c_1 \\ b_1 a_0 c_0 \\ b_1 a_0 c_1 \\ b_1 a_1 c_0 \\ b_1 a_1 c_1 \end{bmatrix}$$

Örnek kodda iki reg tanımlanmış ve bir kuantum devre yaratılmış for döngüsü kullanarak sayacın çift olduğunda devre kübitine H operatörü ve tek olduğunda X kapısı uygulanır

In [194]:
```
from qiskit import QuantumCircuit, QuantumRegister,assemble, Aer, transpile
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt, pi
from qiskit.tools.visualization import circuit_drawer
from qiskit.visualization import *
from qiskit.quantum_info import state_fidelity
from qiskit import BasicAer

qreg_q = QuantumRegister(2,'q')  #register tanımlama

sim = Aer.get_backend('aer_simulator')  # imulasyon yöntemini belirleme
qc = QuantumCircuit(qreg_q)   #devra tanım
for qubit in range(2):
    if qubit%2==0:
        qc.h(qubit)
    else:
        qc.x(qubit)
qc.save_statevector()   # simülator save
qobj = assemble(qc)     # simülator obje yaratım
qc.draw(output='mpl')
```

Out[194]:



$$X|q_1\rangle \otimes H|q_0\rangle = X \otimes H|q_1 q_0\rangle$$

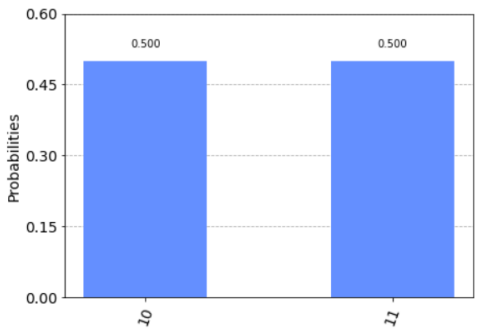
$$X \otimes H = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & H \\ H & 0 \end{bmatrix}$$

In [198]:
```
state = sim.run(qobj).result().get_statevector()
print("State of Measured Qubit = " + str(state))
```

```
State of Measured Qubit = [0.        +0.j 0.        +0.j 0.70710678+0.j 0.70710678+0.j]
```

In [5]:
```
qobj = assemble(qc)
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)
```

Out[5]:



Print(state) fonksyonundan elde ettiğimiz sonuç şu şekilde hesaplanır

$$q_0 = |0\rangle \; ve \; q_1 = |0\rangle$$

$$|0\rangle \otimes |0\rangle = |0\rangle|0\rangle = |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$X \otimes H|q_1 q_0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0.707 \\ 0.707 \end{bmatrix} = \begin{bmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix}$$

$$|a_{10}|^2 = \frac{1}{2}$$

$$|a_{11}|^2 = \frac{1}{2}$$

3.11.1 CNOT kapısı

Çok kübit kapıları çoklu kubit devresi oluşumuna deden olur.
CNOT kapısı, birinci kübitin (kontrol) durumu $|1\rangle$ ise, ikinci kübit (hedef) üzerinde bir X kapısı gerçekleştiren koşullu bir kapıdır. CNOT kapısı için qiskit kütüphanesinde
cx(kontrol,hedef) şeklinde tanımlanır
CNOT kapısının, hangi kübitin hedeflendiğine bağlı olarak iki gösterimi vardır.

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

In [19]:
```
import numpy as np
from qiskit import *
from qiskit import Aer

backend = Aer.get_backend('unitary_simulator')
#prepare 2qubits
circ = QuantumCircuit(2)
circ.cx(0,1)

circ.draw(output='mpl')
```

Out[19]:

In [20]:
```
job = execute(circ, backend)
result = job.result()
print(result.get_unitary(circ, decimals=3))
```

```
[[1.+0.j 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 0.+0.j 1.+0.j]
 [0.+0.j 0.+0.j 1.+0.j 0.+0.j]
 [0.+0.j 1.+0.j 0.+0.j 0.+0.j]]
```

Devre sonuç matrisinde $CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ elde edilmiş

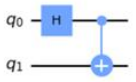
circ.cn(0,1) yerine circ.cn(1,0)

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

$$\text{CNOT}|00\rangle = |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \text{CNOT}|01\rangle = |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \text{CNOT}|10\rangle = |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \text{CNOT}|11\rangle = |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

```
In [132]: from qiskit import QuantumCircuit, Aer, assemble
          import numpy as np
          from qiskit.visualization import plot_histogram, plot_bloch_multivector

          qc = QuantumCircuit(2)
          qc.h(0)
          qc.cx(0,1)
          qc.draw(output='mpl')
```

Out[132]:




```
In [133]: svsim = Aer.get_backend('aer_simulator')
          qc.save_statevector()
          qobj = assemble(qc)
          final_state = svsim.run(qobj).result().get_statevector()
          # Print the statevector neatly:
          array_to_latex(final_state, prefix="\\text{Statevector = }")
```

Out[133]: $\text{Statevector} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$

$$\text{CNOT}|0+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \text{ ispat: } |0+\rangle = |q_1 q_0\rangle = |0\rangle \otimes |+\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) \Longrightarrow$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

Yukardaki kodda görüldüğü gibi kontrol kübiti q_0 alınmıştır q_1 in kontrol kübit alındığı taktirde CNOT matrix inin değişimi için statevector sonucu farklı olur

3.11.2 Entanglement

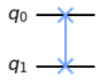
Bir kuantum bilgisayarında, entanglement, kübitler için bir tür hesaplama çarpanı olarak kullanılır. Gittikçe daha fazla kübiti birbirine entanglement oldulça, sistemin hesaplama yapma yeteneği doğrusal bir şekilde değil, katlanarak büyür *Bell* state ($\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ bir kuantum entanglement durumudur Bu birleşik durum, iki ayrı kübit durumu olarak yazılamaz yani bell state durumu
iki bağımsız kübitin tensör çarpımı olarak ifade edilemez bu durm daki kuantum sistemlere entanglement durmunda olan stateler denir

3.11.3 SWAP kapısı

Swap kapısının amacı, iki kübitin durumlarını değiştirmektir. Böylece, örneğin $|10\rangle \Longrightarrow |01\rangle$ olur.
Swap kapısı, üç CNOT kapısı ile uygulanabilir

In [8]:
```
import numpy as np
from qiskit import *
from qiskit import Aer

backend = Aer.get_backend('unitary_simulator')

circ = QuantumCircuit(2)

circ.swap(0,1)
circ.draw(output='mpl')
```

Out[8]:



In [9]:
```
job = execute(circ, backend)
result = job.result()
print(result.get_unitary(circ, decimals=3))
```

```
[[1.+0.j 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 1.+0.j 0.+0.j]
 [0.+0.j 1.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 0.+0.j 1.+0.j]]
```

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

4.Kuantum algoritmalarına giriş

Bir matematiksel problemı nasıl kuantum hesaplama ile ilişkilenirebiliriz?
Kuantum hesaplama,da bazı dezavantajlar olur. Bir kuantum devresinde devrenin tersi sağlanmalıdır Kübitler daima değişimde olduklarından keyfi istenilmiş bir durumda kopyalanmaları mümkün değil ve en önemlisi, bir kübiti ölçmek için süperpozisyon durumun bozulması gerekir
Ancak bir kübit, klasik bir bitin yapamadığı şeyleri yapabilir. Bir kübit, 0 veya 1 ile sınırlı değildir. Her iki durumun bir kombinasyonu olabilir aslında kübit veri kümesi daima değişimde olduğndan dolayı klasik bit sistemine göre tüm olasılıkları kendinde barındırmış durumda, bit sisteminde bir registerin değeri yanlış olduğu zaman sistemin yinede ana bilgi hafızasına geri dönmesi gerekiyor ta ki doğru değer bulunsun ama kuantum bilgi sisteminde bir bilgi birimi tüm değerleri kendinde barındırmış ve otomatik bir şekilde bu değerler değişimde yapılacak iş sadece istenilen register degerine göre uygun devre tasarlanmasıdır bir kübit bir birim zamanda iki değer almıyor her kübit her birim zamanıda sadece bir değer almakta sadece bit sistemi ile olan fark şu ki bu değerlerin aralığı daha geniştir ve en önemlisi otomatik bir değer değişimi var artık eski bir bit sistimi gibi bir birim bilginin değerini değiştirmek için sistem kendisi zaman ve enerji harcamaz bunu kuantum fiziğin doğası yapıyor hemde sürekli bir şekilde bu sistemin hızlanmasına neden olur bir açıdan bilgi transferi açısından zaman kaybı artık yok doğru bilgiyi bilgi deposunda arama zaman kaybı da en az a indirgemiş oluyor

Günümüzdeki quantum bilgisayarlar belkı bazı hasaplama problemleri için elektronik bilgisayarladan saat frekansı açısında daha yavaş olabilir hemde şu ana kadar var olan kübit sayısı az bir miktardır yanlız matematik ve quantum dunyası arasında olan ilginç ilişkiler her şeyi değiştirebilir çözulmeyen problemler çözülür hale gelir yada eskisinden daha hızlı olabilir kuantum algoritmaları bilgisayar bilimlerinin geleceği olarak görünebilir biligisayar bilimlerinin asıl sıkıntısı gelişmeyen alt yapılardır problemler kuantum hasaplama yoluyla çözülür ama çalışma aşamasında her şey artık fizik mühendislerine bağlıdır.
kuantum hesaplamayı bir problemi çözmek için kullanmak istiyorsak eğer, her şeyden önce matematiksel olarak problem çözülmuş oıması gerek ve sonra kuantum hesaplama dünyasına taşınmalıdır yani ilk olarak matematik ve kuantum hesaplama arasındaki ilişkiyi bulmalıyız eşit durumlarda kuanum bir bilgisayar bir problemi klasik bilgisayara göre daha az adımda çözebilmesi mümkündür mesela çarpanlarına ayırma probleminde klasik bilgisayarlar çok büyük bir sıkıntı yaşıyorlar halbuki kuantum bilgisayarlar uygun bir algortitmayla hızlı bir şekilde bu problemi çözüyorlar
örnek olarak Bir f fonksiyonumuz olduğunu varsayalım. Giriş olarak tek bir bit Ya 0 ya da 1 alır. Ve çıktısı olarak da tek bir bit sağlar 0 veya 1.
Bu durum için dört farklı olası fonksyon düşünülebilir.

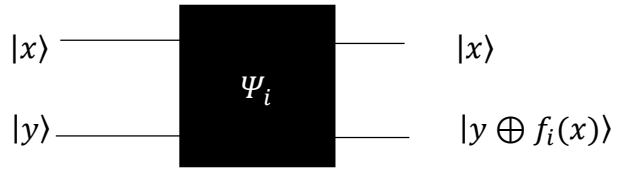
f_0 fonksyonu her zaman 0 üretir.
f_1 fonksyonu , giriş 0 ise 0, giriş 1 ise 1 üretir.
f_2 fonksyonu, giriş 0 ise 1, giriş 1 ise 0 üretir.
f_3 fonksyonu her zaman 1 üretir

Görüldüğü gibi girdi ne olursa olsun bazı fonksyonlar herzaman aynı çıktıyı üretiyor diğer taraftan $f_1$ve $f_2$fonksyonları , girişlerin yarısı için 0 (giriş 0 ise $f_1$ve giriş 1 ise f_2) ve diğer yarısı için 1 (giriş 1 ise $f_1$ve f_2 ise) döndürdükleri için dengeli çıkışlar üretiyorlar
şimdi şöyle düşünelim Bu fonksyonlar rastgele verilirse, bu fonksyonların sabit (f_0 veya f_3) veya dengeli ($f_1$veya f_2) olup olmadığını nasıl belirleyebiliriz?
Klasik olarak, fonksyonu iki kez çalıştırmamız gerekir.fonksyon 0 girdisiyle çalışırsa ve Sonuç olarak 0 üretilirse, fonksyon, her zaman 0 döndüren $f_0$sabit fonksyonu olabilir ve ya giriş 0 ise 0 döndüren dengeli $f_1$fonksyonuda olabilir. fonksiyonun sabit mi yoksa dengeli mi olduğuna karar vermek için fonksiyonu 1 girişi ile tekrar çalıştırmak gerekir.
Budurumda f_0 sabit fonksyonu hala 0 döndürür. Ancak dengeli fonksyon f_1, giriş 1 için 1 döndürür
0 girişi için 1 alınırsa yinede fonksyonu f_2 yada f_3 olduğuna kara vermek için 1 girdisi ile de fonksyonu çalıştırmalıyız.
Kuantum hesaplamada, aynı fonksyon yalnızca **bir** kez çalıştırılır.
Ψ adında bir kuantum kapısı düşünelim bu kapı yukardaki anlatılan dört fonksyonu f_i şekilde temsil eder

$$\Psi_i(|x\rangle) = |f_i(x)\rangle$$

Ψ bir kuantum dönüşüm kapısı olduğundan, tersine çevrilebilir olmalıdır. Bu nedenle, aynı boyutta girdi ve çıktıya sahip olmalıdır. Ve her çıktı, kaynaklandığı girdiyle benzersiz bir şekilde özdeş olmalıdır.
Ama burda sabit fonksyonlar için bir sorun oluşmakta çıktıyı bilirken girdinin ne olduğu net söylenemez
Bu sorunu ikinci bir kübitin eklenmesi ile hall edebiliriz

$$\Psi_i(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |y \oplus f_i(x)\rangle$$



Diyelim ki, i=0 Böylece f_0 fonksiyonunu uygularız. Tanım başına, f_0 (x)=0. Bunu yukarıdaki denkleme eklediğimizde, Ψ'nun çıktısının girdisine eşit olduğunu görebiliriz

$$\Psi_0(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |y \oplus f_0(x)\rangle = |x\rangle \otimes |y\rangle \oplus |0\rangle = |x\rangle \otimes |y\rangle$$

$i = 1$ olduğunda olulan fonksyon 0 girdisi için 0 ve 1 girdisi için 1 üretir

$$\Psi_1(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |y \oplus f_1(x)\rangle = |x\rangle \otimes |y \oplus x\rangle$$

X	Y
0	0
0	1
1	0
1	1

X	$y \oplus x$
0	0
0	1
1	0
1	1

|x⟩⊗|y⊕x⟩ teriminin doğruluk tablosu bunun tersine çevrilebilir olduğunu gösterir.

x=0 için 1 ve x=1 için 0 döndüren f_2 uyguladığımızda, f_2 (x)=x XOR 1 diyebiliriz

$$\Psi_2(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |y \oplus f_2(x)\rangle = |x\rangle \otimes |y \oplus x \oplus 1\rangle$$

X	Y
0	0
0	1
1	0
1	1

X	$y \oplus x \oplus 1$
0	1
0	0
1	0
1	1

Ve f_3 ki her zaman 1 değeri dondürecektir

$$\Psi_3(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |y \oplus f_3(x)\rangle = |x\rangle \otimes |y \oplus 1\rangle$$

X	Y
0	0
0	1
1	0
1	1

X	$y \oplus 1$
0	1
0	0
1	1
1	0

Aşağıdaki doğruluk tablosu, Ψ -kapısının temel durumlardaki kübit çiftlerini nasıl dönüştürdüğünü gösterir

x	**y**	$\|\boldsymbol{x}\rangle \otimes \|\boldsymbol{y} \oplus \boldsymbol{f_i}(\boldsymbol{x})\rangle$
0	0	$\|0\rangle \otimes \|f_i(x)\rangle$
0	1	$\|0\rangle \otimes \|f_i(x) \oplus 1\rangle$
1	0	$\|1\rangle \otimes \|f_i(1)\rangle$
1	1	$\|1\rangle \otimes \|f_i(1) \oplus 1\rangle$

$$H(|0\rangle) + H(|1\rangle) = \frac{1}{\sqrt{2}}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

$$\Psi(H(|0\rangle) + H(|1\rangle) = \frac{1}{2}(|0\rangle \otimes |f_i(x)\rangle - |0\rangle \otimes |f_i(x) \oplus 1\rangle + |1\rangle \otimes |f_i(1)\rangle - |1\rangle \otimes |f_i(1) \oplus 1\rangle)$$

$$\Psi(H(|0\rangle) + H(|1\rangle) = \frac{1}{2}(|0\rangle \otimes |f_i(x)\rangle - |f_i(x) \oplus 1\rangle + |1\rangle \otimes |f_i(1)\rangle - |f_i(1) \oplus 1\rangle)$$

$|f_i(x)\rangle - |f_i(x) \oplus 1\rangle$,ifadesini göz önüne alırsak:

$$|f_i(x)\rangle - |f_i(x) \oplus 1\rangle = \begin{cases} |0\rangle - |1\rangle = (-1)^{f_i(0)}(|0\rangle - |1\rangle, & f_i(x) = 0 \\ -|0\rangle + |1\rangle \ = (-1)^{f_i(1)}(|0\rangle - |1\rangle & f_i(x) = 1 \end{cases}$$

Elde ederiz ozaman $\Psi(H(|0\rangle) + H(|1\rangle)$ formülünü yeniden yazarsak şu sekilde olur:

$$\Psi(H(|0\rangle) + H(|1\rangle) = \frac{1}{2}(|0\rangle \otimes (-1)^{f_i(0)}(|0\rangle - |1\rangle) + |1\rangle \otimes (-1)^{f_i(1)}(|0\rangle - |1\rangle)$$

Ve $(-1)^{f_i(0)}$ ve $(-1)^{f_i(1)}$terimlerini parantezlerin dışına koyarak yeniden düzenleriz Ardından, $|0\rangle - |1\rangle$ terimi dışındaki her şeyi parantezlerin dışına da taşırız

$$\Psi(H(|0\rangle) + H(|1\rangle) = \frac{1}{2}\left((-1)^{f_i(0)}|0\rangle + (-1)^{f_i(1)}|1\rangle\right) \otimes (|0\rangle - |1\rangle))$$

$$\frac{1}{2} = \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$$

$$\Psi(H(|0\rangle) + H(|1\rangle) = \frac{1}{\sqrt{2}}\left((-1)^{f_i(0)}|0\rangle + (-1)^{f_i(1)}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle))$$

Elde edilen sonuç |x⟩⊗|y⟩ biçiminde iki kübitlik bir durumu düşünülebilir |x⟩ ve |y⟩ iki çıktı kübitidir
$|x\rangle=\frac{1}{\sqrt{2}}\left((-1)^{f_i(0)}|0\rangle+(-1)^{f_i(1)}|1\rangle\right)$ ve $|y\rangle=\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle))$

Aşağıdaki tablo, f_i'ye bağlı olarak |x⟩ kübitinin olası değerlerini gösterir

$\boldsymbol{f_i(0)}$	$\boldsymbol{f_i(1)}$	$(-\mathbf{1})^{f_i(\mathbf{0})}\|\mathbf{0}\rangle+(-\mathbf{1})^{f_i(\mathbf{1})}\|\mathbf{1}\rangle$
0	0	$\frac{1}{\sqrt{2}}(\|0\rangle+\|1\rangle))=\ \|+\rangle$
0	1	$\frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle))=\|-\rangle$
1	0	$\frac{1}{\sqrt{2}}(-\|0\rangle+\|1\rangle))=\|-\rangle$
1	1	$\frac{1}{\sqrt{2}}(-\|0\rangle-\|1\rangle))=\|+\rangle$

Tablo_5

Tablo_5 e göre

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

Bu nedenle, eğer Ψ-kapısı çıktısı |+⟩ ise, kübiti kesinlikle 0 olarak ölçeriz. Bu $f0$ için geçerlidir çünkü $f0$ (0)=0 ve $f0$ (1)=0. ve $f3$ için durum aynı şekilde böyle çünkü $f3$ (0)=1 ve $f3$ (1)=1.
Bu nedenle Bunlar iki sabit fonksiyondur

Yani devremize hangi fi fonksiyonunu kullanırsa, sadece bir kere çalıştırarak onun sabit mi yoksa dengeli mi olduğunu anlayabiliriz. Klasik olarak elde edemeyeceğimiz bir şey

Bu algoritma David Deutsch tarafından geliştirilmiştir. Kuantum algoritmalarının belirli bir problemi çözmenin sorgu karmaşıklığını azaltabileceğini kanıtlayan ilk kişiydi. Sorgu karmaşıklığı, bir yanıt almak için bir işlevi değerlendirmemiz gereken sayıdır.

KAYNAKLAR

1. **https://people.eecs.berkeley.edu/~vazirani**
2. **https://qiskit.org/textbook/ch-states/representing-qubit-states.html**
3. **https://pyqml.medium.com/**

ÖZGEÇMİŞ

Amin Hashemian

amin.hashemian.de@gmail.com
+905539534137

JR.SOFTWARE DEVELOPER

About Amin

Junior Software Developer with 3+ years of experience in the Technology domain looking for a Internship position. Supportive and enthusiastic team player dedicated to efficiently resolving project issues. Ability and willingness to innovate and learn new technologies, quick learner passionate about development.

Work Experience

Jr Software Developer
Nov 2020 – April2021, NetCentrics Worked on research projects as volunteer staff

Full Stack Developer
Apr2020 – Oct2020 Comma Soft collaborated with the backend and frontend projects team

Full Stack Developer
Jan 2019 – Jan2020 freelance E-commerce projects

Education

Bursa Uludag University
Sep 2018 - June 2022, Bachelor
Computer Engineering
GPA: 3.10

Marmara University
Sep 2017 - June 2018, Bachelor
Mechatronics, Robotics, Automation Eng

Islamic Azad University
Sep 2013 - May 2015, Doctorate
Medical student

Languages

English:C1,German:B2,Turkish:Native,Persian:Native

Skills

Creative Web design
Html5,Css3,JavaScript,Angular10
Node.js
PHP, Laravel
Java, Spring Core
SQL,PostgreSQL,MySQL
Python,pandas,django
Firebase,GraphQL,AWS EC2
C,C++,OOP,DataStructure
C#,.NET Core

Web: http://www.aminhashemian.com
linkedin : https://www.linkedin.com/in/amin-hashemian/
whatsapp : +905539534137