# Q11

**QX11-ShorsAlgorithmClassical.ipynb**
File

Necesito que me expliques detalladamente cada una de las celdas del siguiente proyecto

A continuación tienes una explicación detallada, línea por línea, de cada una de las celdas del notebook **QX11-ShorsAlgorithmClassical.ipynb**, el cual implementa la parte clásica del algoritmo de Shor para factorizar un entero compuesto $N$.

---

## Celda 1: Importación de bibliotecas

```
python

import math
import matplotlib.pyplot as mpl
import numpy as np
```

- **`import math`**
  Carga el módulo estándar de Python para funciones matemáticas de bajo nivel (máximo común divisor, raíces, etc.).
- **`import matplotlib.pyplot as mpl`**
  Importa la librería Matplotlib (submódulo **`pyplot`**) bajo el alias **`mpl`**, que se utiliza para generar gráficas 2D de forma sencilla.
- **`import numpy as np`**
  Importa NumPy bajo el alias **`np`**, la librería fundamental para cálculo numérico en Python (vectores, matrices, operaciones aritméticas eficientes).

  **Contexto teórico**
  Estas tres librerías son un estándar en Python para computación científica: **`math`** cubre funciones básicas, **`numpy`** permite trabajar con arrays y operaciones vectorizadas, y **`matplotlib`** se encarga de la visualización.

---

## Celda 2: Definición de parámetros $N$ y $a$

```
python

N = 15
a = 13
```

```
#N = 21
#a = 2
```

- **`N = 15`**
  Establece el entero compuesto que queremos factorizar ($N = 15$).
- **`a = 13`**
  Escoge un número aleatorio $a$ tal que $1 < a < N$. En Shor, se elige al azar y luego se comprueba $\gcd(a, N) = 1$.
- Las dos líneas comentadas (**`#N = 21`**, **`#a = 2`**) muestran cómo cambiar $N$ y $a$ para probar otro ejemplo ($N = 21,\ a = 2$).

  **Por qué elegimos $a$**
  Shor requiere un $a$ coprimo con $N$. Si no lo fuera, $\gcd(a, N)$ arrojaría directamente un factor no trivial de $N$.

---

## Celda 3: Cálculo del máximo común divisor

```
python

math.gcd(a, N)
```

- Llama a **`math.gcd(a, N)`** para calcular $\gcd(a, N)$.
  - Si el resultado es mayor que 1, significa que hemos encontrado un factor trivial de $N$ (caso degenerado).
  - Si el resultado es 1, seguimos con el procedimiento de orden (periodo).

  **Interpretación**

$$d = \gcd(a, N).$$

  - Si $d > 1$, se ha hallado un divisor común de $a$ y $N$, y hemos terminado (pues $d$ es un factor de $N$).
  - Si $d = 1$, procedemos a buscar el periodo de la función $f(z) = a^z \bmod N$.

---

## Celda 4: Generación de la secuencia $a^z \bmod N$ y graficado

```
python

z = list(range(N))
y = [a**z0 % N for z0 in z]

mpl.plot(z, y)
mpl.xlabel('z')
mpl.ylabel(f'{a}^z (mod {N})')
mpl.show()
```

1. `z = list(range(N))`
   Crea la lista de exponentes $[0, 1, 2, \ldots, N-1]$.
2. `y = [a**z0 % N for z0 in z]`
   Para cada $z_0$ en `z`, calcula $a^{z_0} \bmod N$. Esto genera la secuencia periódica que usaremos para encontrar el orden.
3. `mpl.plot(z, y)`
   Dibuja la curva $z \mapsto a^z \bmod N$.
4. **Etiquetado de ejes**
   - `mpl.xlabel('z')` marca el eje horizontal con "$z$".
   - `mpl.ylabel(f'{a}^z (mod {N})')` marca el vertical con "$a^z \bmod N$".
5. `mpl.show()`
   Muestra la gráfica en pantalla.

   **Objetivo**
   Visualizar cómo los residuos $\{a^z \bmod N\}$ repiten un ciclo. El siguiente paso será detectar el periodo $r$ de esta secuencia.

---

## Celda 5: Cálculo del periodo $r$

```
python

r = z[y[1:].index(1) + 1]
print(r)
```

- `y[1:].index(1)`
  Busca el primer índice (después de $z = 0$) donde la secuencia toma valor 1. Como `y[0] = a^0 mod N = 1`, omitimos la posición 0.
- `+ 1`
  Ajusta el índice al vector original `z`.
- `r = ...`
  Guarda en `r` el periodo, es decir, el menor entero positivo $r$ tal que

$$a^r \equiv 1 \pmod{N}.$$

- `print(r)`
  Muestra el valor del periodo en consola.

  **Importancia del periodo**
  En Shor, una vez que tenemos $r$, el resto de la factorización es puramente algebraico:
  - Si $r$ es impar, no sirve: volvemos a elegir otro $a$.
  - Si $r$ es par, vamos al siguiente paso.

---

## Celda 6: Extracción de factores usando $r$

```python
if r % 2 == 0:
    x = (a**(r/2.)) % N
    print(f'x: {x}')
    if ((x+1) % N) != 0:
        print(math.gcd(int(x)+1, N),
              math.gcd(int(x)-1, N))
    else:
        print('x+1 is 0 (mod N)')
else:
    print('r is odd')
```

1. **`if r % 2 == 0:`**
   Verifica que el periodo $r$ sea **par**. Sólo en ese caso procede a factorizar.
2. **`x = (a**(r/2.)) % N`**
   Calcula
   $$x = a^{r/2} \bmod N.$$
   En teoría de números, si $r$ es par y $a^r \equiv 1 \pmod N$, entonces
   $$a^{r/2} \equiv \pm 1 \pmod N.$$
3. **`if ((x+1) % N) != 0:`**
   - Si $x \not\equiv -1 \pmod N$, entonces tanto $\gcd(x+1, N)$ como $\gcd(x-1, N)$ producirán factores no triviales de $N$.
4. **`print(math.gcd(int(x)+1, N), math.gcd(int(x)-1, N))`**
   Muestra en consola los dos posibles factores de $N$:
   $$\gcd(x+1,\, N), \quad \gcd(x-1,\, N).$$
5. En caso contrario, informa que $x \equiv -1 \pmod N$ (no sirve este $a$), o bien que $r$ es impar y hay que reintentar con otro $a$.

   **Resultado final**
   Se obtienen dos divisores no triviales de $N$, que al multiplicarse dan $N$. Así completamos la factorización clásica que necesita Shor tras la parte cuántica de detección de periodo (aquí simulada por búsqueda pura).

---

## Celda 7: (vacía)

- Esta última celda está vacía y no realiza ninguna acción. Puede usarse para añadir comentarios o código adicional.

---

### Resumen del flujo

1. **Importar** libs fundamentales (`math`, `numpy`, `matplotlib`).
2. **Elegir** $N$ (entero a factorizar) y $a$ (aleatorio, coprimo con $N$).
3. **Verificar** $\gcd(a, N) = 1$.
4. **Generar** la secuencia $\{a^z \bmod N\}$ y **graficar** para observar periodicidad.
5. **Detectar** el periodo mínimo $r$.
6. **Calcular** $x = a^{r/2} \bmod N$ y **extraer** factores $\gcd(x \pm 1, N)$.
7. **Mostrar** los factores de $N$.

Este notebook implementa la **parte clásica** de Shor (detección de periodo por fuerza bruta). En su versión cuántica real, ese periodo se obtendría con transformada de Fourier cuántica; aquí lo calculamos directamente para fines didácticos.

Podrias darme todas las teorias matematicas y fisicas aplicadas en el siguiente proyecto

Aquí tienes un desglose de las principales teorías matemáticas y físicas que subyacen al proyecto de la parte clásica del algoritmo de Shor (factorización de un entero $N$ mediante búsqueda del periodo de la función $f(z) = a^z \bmod N$):

---

# 1. Teorías Matemáticas

## 1.1 Aritmética modular

- **Definición de congruencia**

$$a \equiv b \pmod{N} \quad \Longleftrightarrow \quad N \mid (a - b).$$

- **Operaciones modulares**: suma, resta, multiplicación y, en contexto cuántico, exponenciación rápida (modular exponentiation) para calcular $a^z \bmod N$.

## 1.2 Algoritmo de Euclides

- **Máximo común divisor (mcd)**
  Se basa en la recursión

$$\gcd(a, b) = \gcd(b,\ a \bmod b),$$

  hasta que uno de los operandos sea cero.
- En el proyecto se usa para verificar $\gcd(a, N) = 1$ y, en caso contrario, extraer un factor trivial.

## 1.3 Teoremas fundamentales de teoría de números

- **Pequeño teorema de Fermat**
  Para $p$ primo y $a \not\equiv 0 \pmod{p}$:

$$a^{p-1} \equiv 1 \pmod{p}.$$

- **Teorema de Euler**
  Para $N$ cualquiera y $\gcd(a, N) = 1$:

  $$a^{\varphi(N)} \equiv 1 \pmod{N},$$

  donde $\varphi(N)$ es la función totiente de Euler.
- **Orden de un elemento en un grupo**
  El **orden** $r$ de $a$ módulo $N$ es el menor entero positivo tal que

  $$a^r \equiv 1 \pmod{N}.$$

### 1.4 Teoría de grupos

- Se considera el grupo multiplicativo $(\mathbb{Z}/N\mathbb{Z})^\times$, cuyas propiedades garantizan que la búsqueda del periodo de $a^z$ termine en tiempo polinomial (en la parte cuántica).

### 1.5 Funciones periódicas y análisis de señales

- La secuencia $\{a^z \bmod N\}_{z=0}^{N-1}$ es **periódica** con periodo $r$.
- Visualizar $(z,\ a^z \bmod N)$ ayuda a "ver" un ciclo que se repite cada $r$ pasos.

### 1.6 Complejidad Algorítmica

- **Factorización clásica** de enteros es, hasta donde se sabe, de tiempo **subexponencial** en el mejor caso (e.g. método de criba cuadrática), pero **no polinomial**.
- El interés de Shor es que, al combinar estas ideas con la parte cuántica, se consigue factorización en **tiempo polinomial**.

---

## 2. Teorías Físicas

Aunque esta implementación hace la parte clásica de Shor, el algoritmo completo descansa en los principios de la mecánica cuántica:

### 2.1 Mecánica cuántica fundamental

- **Postulados de la mecánica cuántica**:
  - Estados descritos por vectores en un espacio de Hilbert.
  - Evolución temporal dada por operadores unitarios (gates cuánticos).
  - Medición que colapsa el estado cuántico.

### 2.2 Qubit y superposición

- Un **qubit** general se escribe

  $$|\psi\ = \alpha|0\ + \beta|1\ , \quad |\alpha|^2 + |\beta|^2 = 1.$$

- Permite procesar simultáneamente múltiples valores de $z$ gracias a la superposición.

### 2.3 Transformada de Fourier cuántica (QFT)

- Núcleo del paso cuántico de Shor: aplica una QFT sobre la superposición de $|z\rangle$ para extraer información espectral (el periodo $r$).
- Matemáticamente es análoga a la transformada discreta de Fourier, pero implementada con puertas cuánticas en $O(n^2)$ operaciones.

### 2.4 Interferencia cuántica

- Tras la QFT, diferentes términos de la superposición interfieren constructivamente en los estados que contienen la información del periodo.
- La amplitud de probabilidad se concentra en múltiplos de $2^n/r$, permitiendo estimar $r$.

### 2.5 Decoherencia y errores cuánticos

- En un sistema real, la **decoherencia** limita la fidelidad de los operadores unitarios, y por tanto la exactitud de la QFT y la medición.
- Para la factorización práctica se requieren corrección de errores cuánticos y hardware tolerante a fallos.

---

## Flujo conceptual general

1. **Seleccionar** un $a$ coprimo con $N \rightarrow$ **Euclides**.
2. **Preparar** superposición de estados $\sum_z |z\rangle$.
3. **Calcular** en paralelo $|z\rangle \mapsto |a^z \bmod N\rangle$.
4. **Aplicar** QFT para revelar el periodo $r$.
5. **Medir** y **extraer** $r \rightarrow$ **aritmética modular**.
6. **Obtener** factores $\gcd(a^{r/2} \pm 1,\ N)$.

Con esto quedan claras las bases matemáticas (teoría de números, álgebra, análisis de señales) y físicas (mecánica cuántica, QFT e interferencia) que sostienen el algoritmo de Shor y su simulación clásica.