Resolución de Tarea 10 - Algoritmos Cuánticos (Fecha: 12 de Diciembre de 2025)

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI5651 - Diseño de Algoritmos I
Septiembre - Diciembre 2025
Estudiante: Junior Miguel Lara Torres (17-10303)

Tarea 10 (9 puntos)

Indice

Pregunta 1

Este desglose explica detalladamente cada una de las simulaciones del Algoritmo de Shor para factorizar $N = 35$, centrándose en el rol simulado de la Transformada Discreta de Fourier (DFT) para encontrar el período r.

Cada simulación comienza con la selección aleatoria de una base x y el establecimiento de un tamaño de registro simulado de $M = 64$. La DFT se utiliza para simular el muestreo cuántico de Fourier (QFT), que encuentra el período r (el orden de x módulo N) con alta probabilidad. Se describen 5 simulaciones diferentes.

Simulación 1

Esta simulación requirió dos intentos de búsqueda de período para encontrar un factor no trivial.

Iteración 1	Iteración 2
Base x Seleccionada: 34 **Paso de DFT (QFT simulado):** El pico de amplitud (simulado j) fue 32, resultando en un *ratio de frecuencia* 32/64. **Orden r Encontrado:** $r = 2$ (el período de $34^a \pmod{35}$). **Verificación de Factores:** Se evalúa $x^{r/2} \pmod{35} = 34^1 \pmod{35} = 34$.	**Base x Seleccionada:** 27 **Paso de DFT (QFT simulado):** El pico de amplitud fue $j = 16$, resultando en un *ratio de frecuencia* 16/64. **Orden r Encontrado:** $r = 4$ (el período de $27^a \pmod{35}$). **Verificación de Factores:** Se evalúa $x^{r/2} \pmod{35} = 27^2 \pmod{35} = 729 \pmod{35} = 29$.

Iteración 1	Iteración 2
Análisis Lógico: El orden $r = 2$ es par, pero $34 \equiv -1 \pmod{35}$. Este es el **caso trivial** ($x^{r/2} \equiv N - 1$), que no garantiza un factor no trivial. El algoritmo **continúa**.	**Análisis Lógico:** El orden $r = 4$ es par, y $29 \not\equiv -1 \pmod{35}$. Se encontró una **raíz no trivial** de 1 módulo 35. **Resultado Final:** Factorización exitosa. Los factores se encuentran calculando $\gcd(35, 29 + 1) = 5$ y $\gcd(35, 29 - 1) = 7$.

Simulación 2

Esta simulación demuestra cómo el algoritmo de Shor se detiene inmediatamente si la base seleccionada comparte un factor no trivial con N.

Iteración 1
Base x Seleccionada: 14. **Chequeo Inicial:** Se calcula el Máximo Común Divisor (GCD) de x y N: $\gcd(\mathbf{14}, \mathbf{35}) = \mathbf{7}$. **Análisis Lógico:** Como $\gcd(x, N) > 1$ y $\gcd(x, N) < N$, el valor 7 es un factor no trivial (o factor inmediato). El proceso de DFT/QFT no es necesario. **Resultado Final:** Éxito. Factor no trivial encontrado inmediatamente.

Simulación 3

Similar a la simulación 1, el primer intento produce un resultado trivial, requiriendo un segundo intento para encontrar el factor.

Iteración 1	Iteración 2
Base x Seleccionada: 24. **Paso de DFT (QFT simulado):** Pico $j = 32$, resultando en un ratio 32/64. **Orden r Encontrado:** $r = 6$ (período de $24^a \pmod{35}$). **Verificación de Factores:** $x^{r/2} \pmod{35} = 24^3 \pmod{35} = 34$. **Análisis Lógico:** $r = 6$ es par, pero $34 \equiv N - 1 \pmod{35}$. **Caso trivial**. El algoritmo **continúa**.	**Base x Seleccionada:** 17. **Paso de DFT (QFT simulado):** Pico $j = 16$, resultando en un ratio 16/64. **Orden r Encontrado:** $r = 12$ (período de $17^a \pmod{35}$). **Verificación de Factores:** $x^{r/2} \pmod{35} = 17^6 \pmod{35} = 29$. **Análisis Lógico:** $r = 12$ es par, y $29 \not\equiv -1 \pmod{35}$. **Raíz no trivial** encontrada. **Resultado Final:** Factorización exitosa. Los factores son 5 y 7.

Simulación 4

En esta simulación, la primera base elegida cumple todas las condiciones para una factorización exitosa.

Iteración 1
Base x Seleccionada: 32. **Paso de DFT (QFT simulado):** Pico $j = 16$, resultando en un ratio 16/64. **Orden r Encontrado:** $r = 12$ (período de $32^a \pmod{35}$). **Verificación de Factores:** $x^{r/2} \pmod{35} = 32^6 \pmod{35} = 29$. **Análisis Lógico:** $r = 12$ es par, y $29 \not\equiv -1 \pmod{35}$. **Raíz no trivial** encontrada.

Iteración 1

Resultado Final: Factorización exitosa. Los factores son 5 y 7.

Simulación 5

Al igual que en la Simulación 2, la base elegida contiene inmediatamente un factor de N.

Iteración 1

Base x Seleccionada: 25.
Chequeo Inicial: Se calcula el Máximo Común Divisor (GCD) de x y N: $\gcd(\mathbf{25}, \mathbf{35}) = \mathbf{5}$.
Análisis Lógico: Como $\gcd(x, N) > 1$ y $\gcd(x, N) < N$, el valor 5 es un factor no trivial (factor inmediato). El proceso se detiene.
Resultado Final: Éxito. Factor no trivial encontrado inmediatamente.

Puntos Relevantes y Conexiones Teóricas

- La DFT (implementada clásicamente como FFT) procesa la superposición periódica (que se representa como un arreglo de impulsos) y produce amplitudes en el *dominio de la frecuencia*. El valor j del **pico de amplitud más alto** (ej. $j = 16$ en Simulación 4) simula el resultado probabilístico de la medición cuántica. Este valor j es una aproximación al valor $\frac{M}{r}$.
- Aunque la simulación usa la DFT para obtener una pista ($r_{candidato}$), la lógica subsiguiente requiere encontrar el **orden real** r (el período) de $x^a \pmod N$ para garantizar la corrección.
- La clave para el éxito en Shor, una vez encontrado el período r, reside en dos condiciones:
  - r debe ser **par**.
  - $x^{r/2} \pmod N$ no debe ser 1 ni $-1 \pmod N$ (es decir, $x^{r/2} \not\equiv N - 1$).

  En las simulaciones 1 y 3, el primer intento falló porque, aunque r era par, la base x resultó ser una **raíz trivial** de 1 módulo 35 ($x^{r/2} \equiv 34 \pmod{35}$), obligando al algoritmo a seleccionar una nueva base x para la siguiente iteración hasta hallar un caso **no trivial** (ej. $x^{r/2} \equiv 29 \pmod{35}$ en las iteraciones exitosas).
- Las simulaciones 2 y 5 ilustran que, en el contexto de un simulador clásico, el paso más eficiente es el **chequeo inicial de** gcd, que es lineal en el número de bits de N. El valor del algoritmo de Shor se manifiesta cuando este chequeo inicial falla y es necesario encontrar el orden r, una tarea que es exponencialmente lenta para la computadora clásica, pero polinomial ($\mathbf{O}((\log \mathbf{N})^{\mathbf{3}} \log(\log \mathbf{N}))$) para la computadora cuántica.

Implementación en Python

El archivo funcional se encuentra en shor.py

- Crear un environment:
- Instalar numpy:
- Ejecutar el programa:

```python
import random
import math
import numpy as np
from fractions import Fraction

# Definimos N como constante para el problema y M como el tamaño del registro cuántico simulado
N_TO_FACTOR = 35
K_ITERATIONS = 10
```

```python
M_REGISTRY = 64  # Tamaño del registro M, potencia de 2 (64 >= 35^2 no es necesario aquí, pero
SIMULATIONS = 5  # Número de simulaciones independientes a ejecutar

# ----------------------------------------------------------------------
# Funciones Aritméticas (Polinomiales en el tamaño de los bits)
# ----------------------------------------------------------------------

def modular_exponentiation(base, exp, mod):
    """Calcula (base^exp) mod mod, usando la función nativa de Python."""
    return pow(base, exp, mod)

def find_order_classic(x, N):
    """Función auxiliar para encontrar el periodo r de x mod N por fuerza bruta clásica."""
    r = 1
    # Pow(x, r, N) calcula x^r mod N
    while pow(x, r, N) != 1:
        r += 1
        if r > N * N:
            return 0  # Previene bucles infinitos en casos raros
    return r

# ----------------------------------------------------------------------
# Simulación de QFT/DFT (Núcleo del requisito)
# ----------------------------------------------------------------------

def find_period_dft(x, N, M):
    """
    Simula la QFT mediante DFT (usando FFT de numpy) para encontrar el periodo r.
    """
    # 1. Definir el vector de amplitud de entrada (vector de "impulsos")
    # Este vector representa el patrón periódico de f(a) = x^a mod N.
    amplitudes = np.zeros(M, dtype=complex)

    # Calculamos la función f(a) para a en [0, M-1]
    for a in range(M):
        result = modular_exponentiation(x, a, N)
        # Marcamos solo los puntos donde el resultado es 1, ya que f(a)=1 es periódico con peri
        if result == 1:
            amplitudes[a] = 1.0

    # 2. Aplicar la Transformada Discreta de Fourier (FFT)
    dft_output = np.fft.fft(amplitudes)

    # 3. Encontrar la frecuencia dominante (simulación de la medición/muestreo)
    # Buscamos el pico de mayor magnitud (potencia) a una frecuencia j != 0.
    # Excluimos el índice 0 (frecuencia DC) que siempre es alto.
    magnitudes = np.abs(dft_output[1:])

    if np.any(magnitudes):
        # Encontramos el índice de frecuencia (j) con mayor potencia.
        j_peak = np.argmax(magnitudes) + 1  # +1 porque excluimos el índice 0 original
    else:
        # Caso de seguridad si la FFT es plana (ej. si x no es coprimo)
        return 0, 0, 0

    # 4. Inferir el periodo r a partir de la frecuencia j/M
    # La frecuencia j_peak/M debe ser una aproximación a s/r (donde r es el periodo).

    # Simplificamos la fracción j_peak/M para obtener la estimación del periodo r.
```

```
    ratio = Fraction(j_peak, M)
    candidate_r = ratio.denominator

    # Como la DFT solo da una PISTA PROBABILÍSTICA (s/r), usamos una verificación clásica
    # para encontrar el periodo mínimo real (r_actual) que corresponde a esa pista.
    r_actual = find_order_classic(x, N)

    return j_peak, r_actual, candidate_r

# ----------------------------------------------------------------------
# Bucle Principal del Algoritmo de Shor (Simulación)
# ----------------------------------------------------------------------

def shor_simulation_dft(N, M, k_max):

    print(f"--- Ejecutando Simulación de Shor para N = {N} ---")
    print(f"Tamaño del registro cuántico simulado (M): {M}")

    for i in range(1, k_max + 1):
        # 1. Seleccionar una base aleatoria x en [2, N-1]
        x = random.randint(2, N - 1)

        # 2. Verificar la coprimalidad (GCD inicial)
        d_init = math.gcd(x, N)
        if d_init != 1:
            print(f"\nIteración {i}/{k_max}: Base x = {x}. GCD(x, N) = {d_init}.")
            if 1 < d_init < N:
                print(f"Éxito: ¡Factor no trivial encontrado en el paso inicial! Factor = {d_i
                return True, i
            continue

        # 3. Simulación de QFT/DFT para obtener la pista del periodo
        j_peak, r_actual, r_candidate = find_period_dft(x, N, M)

        # Usamos r_actual (el periodo verdadero) ya que la DFT solo da la pista probabílistica
        # y esta simulación busca el resultado determinista del algoritmo.
        r = r_actual

        print(f"\nIteración {i}/{k_max}: Base x = {x}.")
        print(f"  DFT aplicada. Pico medido (simulado j): {j_peak} (Implica ratio j/M = {j_peal
        print(f"  Periodo inferido/actual r: {r} (candidato de fracción: {r_candidate})"

        # 4. Verificar condiciones de factorización: r debe ser par
        if r == 0:
             print("Resultado: Periodo r no encontrado (x es trivial). Continuar.")
             continue

        if r % 2 != 0:
            print(f"Resultado: Orden r={r} no es par/válido. Continuar.")
            continue

        # 5. Verificar condición de no-trivialidad de la raíz
        r_half = r // 2
        x_r_half_mod_N = modular_exponentiation(x, r_half, N)

        print(f"  Verificación: x^(r/2) mod N = {x_r_half_mod_N}")

        if x_r_half_mod_N == N - 1:
            print(f"Resultado: x^(r/2) = N-1 ({N-1}). Raíz trivial. Continuar.")
```

```
            continue

        # 6. Calcular factores no triviales
        d1 = math.gcd(N, x_r_half_mod_N + 1)
        d2 = math.gcd(N, x_r_half_mod_N - 1)

        if (1 < d1 < N) or (1 < d2 < N):
            factor = d1 if 1 < d1 < N else d2
            print(f"Éxito: ¡Factor no trivial encontrado! {N} = {factor} * {N // factor}"
            return True, i
        else:
            print("Resultado: Factor trivial obtenido. Continuar.")

    print(f"\n--- Simulación Finalizada ---")
    return False, k_max

random.seed(1257)
for i in range(1, SIMULATIONS + 1):
    print(f"\n================================== Simulación: {i} ==========================
    shor_simulation_dft(N_TO_FACTOR, M_REGISTRY, K_ITERATIONS)
```

Pregunta 2

Por mi súbalo a la NAZA, ya es cuestión de usted si permite mi atrevimiento xD. Memardo aquí.