Análisis de Algoritmo de Euclides MCD VS Mi Algoritmo MCD V1, V2 y Algoritmo de Stein

Jesús Manuel De Dios López

20 de septiembre de 2023

Vamos a analizar los tiempos para ver la eficiencia del tiempo maquina para el algoritmo de Euclides VS mi algoritmo con el fin de calcular el MCD de dos números.

1. Algoritmo de Euclides

```python
#MCD Auclides
def MCD_euclides(a,b):
  while a != b:
    if a > b:
      a = a - b
    else:
      b = b - a
  return a

```

Listing 1: Algoritmo de Euclides para MCD

```python
  #Cuantficador Universal Para Todo. tema de Matematicas Discretas
  def paraTodo(P, L):
    if L == []:
      return True
    elif P(L[0]):
      return paraTodo(P, L[1:])
    else:
      return False

  #Cuantficador Universal Existe Un. tema de Matematicas Discretas
  def existeUn(P,L):
    if L == []:
      return False
    elif P(L[0]):
      return True
    else:
      return existeUn(P, L[1:])

  def es_primo(n, divisor=2):
    if n <= 1:
      return False
    if divisor * divisor > n:
      return True
    if n % divisor == 0:
      return False
    return es_primo(n,divisor+1)


  def MCD_C(L,n,mult=1):
    if existeUn(es_primo, L) or existeUn(lambda x: n >= x,L):
      return mult
    elif paraTodo(lambda x: x%n == 0, L):
      return MCD_C(list(map(lambda x: x//n, L)), n, mult * n)
    elif n == 2:
      return MCD_C(L, n + 1, mult)
    else:
      return MCD_C(L, n + 2, mult)

```

Listing 2: Mi algoritmo V2 para MCD

2. Resultados

Resultados:

- Algoritmo de Euclides: El tiempo promedio de ejecución para calcular el MCD de dos números usando el algoritmo de Euclides fue de aproximadamente 0.02098 segundos en 10,000 iteraciones.
- Algoritmo personalizado (basado en factorización y recursión): El tiempo promedio de ejecución para calcular el MCD de dos números utilizando tu implementación personalizada fue de aproximadamente 0.25857 segundos en 10,000 iteraciones.

Resumen de los algoritmos:

1. Algoritmo de Euclides:
    - Basado en la división y el módulo de números enteros.
    - Altamente eficiente y ampliamente utilizado para calcular el MCD de dos números.
    - Requiere solamente una serie de divisiones y módulos.
    - Eficiente incluso para números grandes.
    - Es el enfoque preferido en la mayoría de los casos debido a su velocidad y simplicidad.
2. Algoritmo personalizado (basado en factorización y recursión):
    - Basado en la factorización de números enteros en la lista L y recursión.
    - Menos eficiente que el algoritmo de Euclides, especialmente para números grandes.
    - Utiliza funciones adicionales como es_primo, $existeUn$, y $paraTodo$.
    - Puede requerir un tiempo considerablemente mayor para calcular el MCD en comparación con el algoritmo de Euclides.
    - Aunque es una implementación válida, podría no ser la opción más eficiente en la mayoría de los casos.

En resumen, los resultados indican que el algoritmo de Euclides es mucho más rápido y eficiente que mi implementación personalizada para calcular el MCD de dos números. Por lo tanto, en la práctica, se recomienda el uso del algoritmo de Euclides cuando se necesita calcular el MCD, especialmente para números grandes o cuando la eficiencia es importante.

3. Analisis de mi versión 1 del algoritmo MCD

```
#Cuantficador Universal Para Todo. tema de Matematicas Discretas
def paraTodo(P, L):
  if L == []:
    return True
  elif P(L[0]):
    return paraTodo(P, L[1:])
  else:
    return False

def MCD_B(L,n,res):
  if paraTodo(lambda x: x%n == 0, L):
    return MCD_B(list(map(lambda x: x/n, L)), n, res+[n])
  elif n == L[0] or n > L[0]:
    resultado = 1
    for elemento in res:
      resultado *= elemento
    return resultado
  elif n == 2:
    n += 1
    return MCD_B(L, n, res)
  elif n > 2:
    n += 2
    return MCD_B(L, n, res)

```

Listing 3: Mi algoritmo V1 para MCD

4. Resultados

1. Algoritmo personalizado (versión V1): El tiempo promedio de ejecución para calcular el MCD de dos números utilizando tu segunda implementación personalizada (V1) fue de aproximadamente 13.897 segundos en 10,000 ejecuciones para los números $(997, 48)$

Estos resultados indican claramente que el algoritmo de Euclides es mucho más rápido y eficiente que ambas versiones de tu implementación personalizada para calcular el MCD. El tiempo de ejecución de la versión V1 es notablemente más largo que el de la versión original, lo que sugiere que la versión V1 es menos eficiente en términos de tiempo de ejecución.

5. Algoritmo de Stein

El algoritmo de Stein, también conocido como el .algoritmo binario de Euclides.º .algoritmo de Euclides binario extendido". Este algoritmo es una variante del algoritmo de Euclides que puede ser más rápido en algunos casos.

6. Analisis del Algoritmo de Stein

```
    def MCD_Stein(a, b):
      if a == b:
        return a
      if a == 0:
        return b
      if b == 0:
        return a

      if a % 2 == 0 and b % 2 == 0:
        return 2 * MCD_Stein(a // 2, b // 2)
      elif a % 2 == 0:
        return MCD_Stein(a // 2, b)
      elif b % 2 == 0:
        return MCD_Stein(a, b // 2)
      else:
        if a > b:
          return MCD_Stein((a - b) // 2, b)
      else:
        return MCD_Stein((b - a) // 2, a)

```

Listing 4: Algoritmo de Stein para MCD

7. Resultados

- Algoritmo de Euclides: El tiempo promedio de ejecución para calcular el MCD de dos números utilizando el algoritmo de Euclides fue de aproximadamente 0.083 segundos en 10,000 ejecuciones.
- Tu algoritmo personalizado (versión original): El tiempo promedio de ejecución para calcular el MCD de dos números utilizando tu implementación personalizada fue de aproximadamente 0.261 segundos en 10,000 ejecuciones.
- Algoritmo de Stein: El tiempo promedio de ejecución para calcular el MCD de dos números utilizando el algoritmo de Stein fue de aproximadamente 0.148 segundos en 10,000 ejecuciones.

Los resultados indican que, en este conjunto particular de datos y ejecuciones, el algoritmo de Euclides es más rápido que tu implementación personalizada, y el algoritmo de Stein se encuentra en medio en términos de tiempo de ejecución. Sin embargo, la eficiencia de los algoritmos puede depender de los números específicos utilizados y de otros factores, por lo que es importante tener en cuenta que los resultados pueden variar en diferentes situaciones.