# MULTIPLICACIÓN DE POLINOMIOS UTILIZANDO DFT/FFT

## 1. Algoritmo

El algoritmo que se utilizó es el que está en el libro de Cormen, segunda edición:

```
RECURSIVE-FFT(a)
n ← length[a]              ▷ n is a power of 2.
if n = 1
   then return a
ω_n ← e^{2πi/n}
ω ← 1
a^[0] ← (a_0, a_2, ..., a_{n-2})
a^[1] ← (a_1, a_3, ..., a_{n-1})
y^[0] ← RECURSIVE-FFT(a^[0])
y^[1] ← RECURSIVE-FFT(a^[1])
for k ← 0 to n/2 − 1
     do y_k ← y_k^[0] + ω y_k^[1]
        y_{k+(n/2)} ← y_k^[0] − ω y_k^[1]
        ω ← ω ω_n
return y                   ▷ y is assumed to be a column vector.
```

## 2. Código en python

```python
"""
Multiplication of polynomials using FFT
Cormen edtion 2
"""
from math import *
def PrintPolynomial(a):
  # a is array of polinomy coefficient, example: [1, -5, 0, 2] (1 - 5x +0x^2 + 2x^3)
  for k in range(len(a)):
    print str(a[k]) + "x^" + str(k),
  print

def PrintMatrix(m):
  print("-" * 10 * len(m[0]))
  for i in m:
    for j in i:
      print "| " + str(j).ljust(7, " "),
    print "|"
  print("-" * 10 * len(m[0]))

def MatrixVandermodeR(n):
  mvr = []
  for i in range(0, n):
    aux = [1]
    for j in range(1 , n):
      aux.append(pow(i, j))
    mvr.append(aux)
```

```python
    return mvr

def MatrixVandermodeI(n):
    mvr = []
    mvr.append([1] * (n))
    for i in range(1, n):
        aux = [1]
        for j in range(1 , n):
            aux.append(EulerNumber(i * j, n))
        mvr.append(aux)
    return mvr

def EulerNumber(k, n):
    u = (2 * pi * k) / n
    return int(cos(u)) + 1j * int(sin(u))

def MultiplicationPoint(a, b):
    # DFT(a) * DFT(b)
    c = []
    for k in range(len(a)):
        c.append(a[k] * b[k])
    return c

def GetEvenIndex(a):
    a_even = []
    for k in range(0, len(a), 2):
        a_even.append(a[k])
    return a_even

def GetOddIndex(a):
    a_odd = []
    for k in range(1, len(a), 2):
        a_odd.append(a[k])
    return a_odd

def RecursiveFFT(a):
    # a is array of polinomy coefficient, example: [1, -5, 0, 2] (1 - 5x +0x^2 + 2x^3)
    n = len(a)
    if n == 1:
        return a
    Wn = EulerNumber(1, n)
    W = 1
    a_even = GetEvenIndex(a)
    a_odd = GetOddIndex(a)
    y_even = RecursiveFFT(a_even)
    y_odd = RecursiveFFT(a_odd)
    y = []
    for k in range(0, n / 2):
        t = W * y_odd[k]
        y.insert(k, y_even[k] + t)
        y.insert(k + (n / 2), y_even[k] - t)
        W = W * Wn
    return y
```

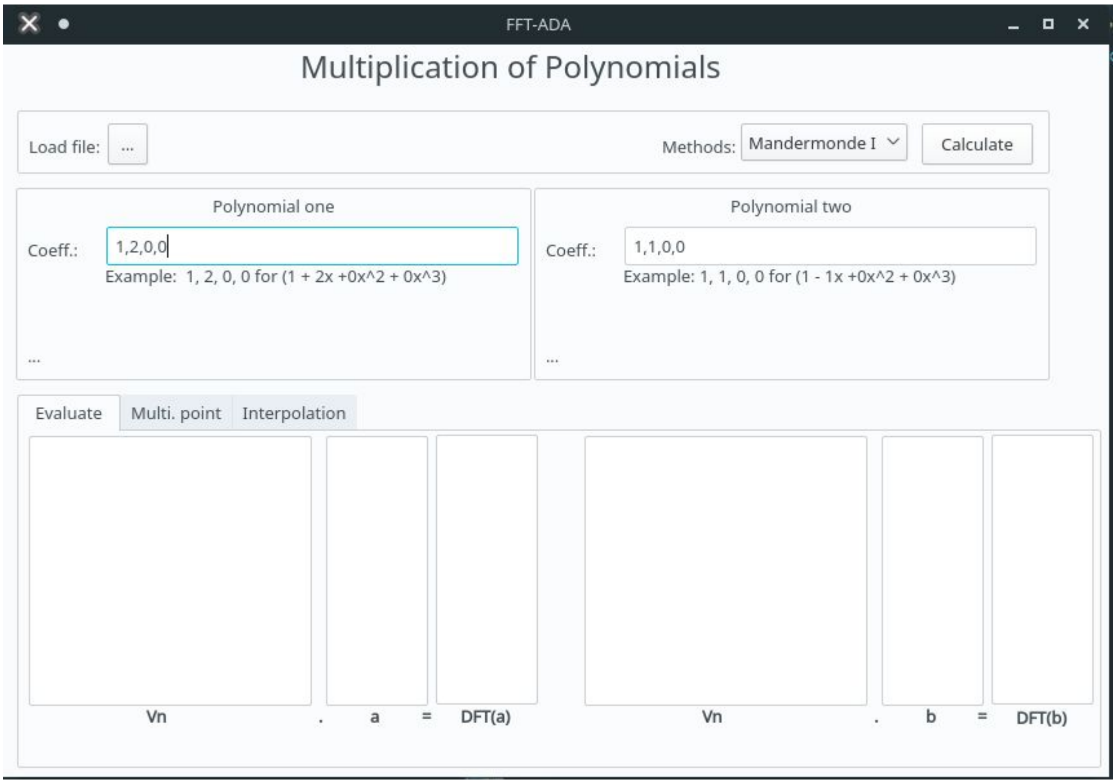
```python
def MultiplicationPolynomial(a, b, n):
 # Step 1: Evaluation
 dft_a = RecursiveFFT(a)
 dft_b = RecursiveFFT(b)

 print("\nSTEP 1: EVALUATION")
 print("*****************")
 print("\nDFT(a): ")
 print(dft_a)
 print("\nDFT(B): ")
 print(dft_b)

 # Step 2: Multiplication point
 # Yn = DFT(a) * DFT(b)
 Yn =  MultiplicationPoint(dft_a, dft_b)
 print("\nSTEP 2: MULTIPLICACION POINT")
 print("**************************")
 print("\nYn = DFT(a) * DFT(b): ")
 print(Yn)

 # Step 3: Interpolation
 # Vn = c * Yn
 # Vn^-1 = 1/Vn
 # c = (1/n) * Vn^-1 * Yn

 VnI = MatrixVandermodeI(n)
 print("\nSTEP 3: INTERPOLATION")
 print("**********************")
 print("\nMatrix Vandermode I")
 PrintMatrix(VnI)
 VnI_inv = []
 for i in VnI:
   aux = []
   for j in i:
     aux.append(1 / j)
   VnI_inv.append(aux)
 print("\nMatrix Vandermode I Inverse")
 PrintMatrix(VnI_inv)
 c = []
 for i in VnI_inv:
   aux = []
   for k in range(n):
     aux.append((i[k] * Yn[k]))
   c.append(sum(aux).real / n)
 print("\nc = (1/n) * Vn^-1 * Yn")
 print(c)
 return c

import sys
if __name__ == "__main__":
 if len(sys.argv) > 1:
   # Input: n, a, b
   # n is number of coefficient
   # a is array of polinomy coefficient, example: [1, 2, 0, 0] (1 + 2x +0x^2 + 0x^3)
```

```python
    # b is array of polinomy coefficient, example: [1, 1, 0, 0] (1 - 1x +0x^2 + 0x^3)

    f = open(sys.argv[1], "r")
    n = int(f.readline())
    a = list(map(int, f.readline().split()))
    b = list(map(int, f.readline().split()))

    print("\nINPUT")
    print("*****")
    print("a(x):")
    PrintPolynomial(a)
    print("b(x):")
    PrintPolynomial(b)
    c = MultiplicationPolynomial(a, b, n)
    print("\nCoefficient of polinomy solution c(x):")
    PrintPolynomial(c)

    f.close()
  else:
    print "Pass file name ..."

# Input: In file name: poly_4
# Compile and execute:
# python2 FFT.py poly_4
```

## 3. Ejecución

El script está con el nombre de : m*ultiplication.py*

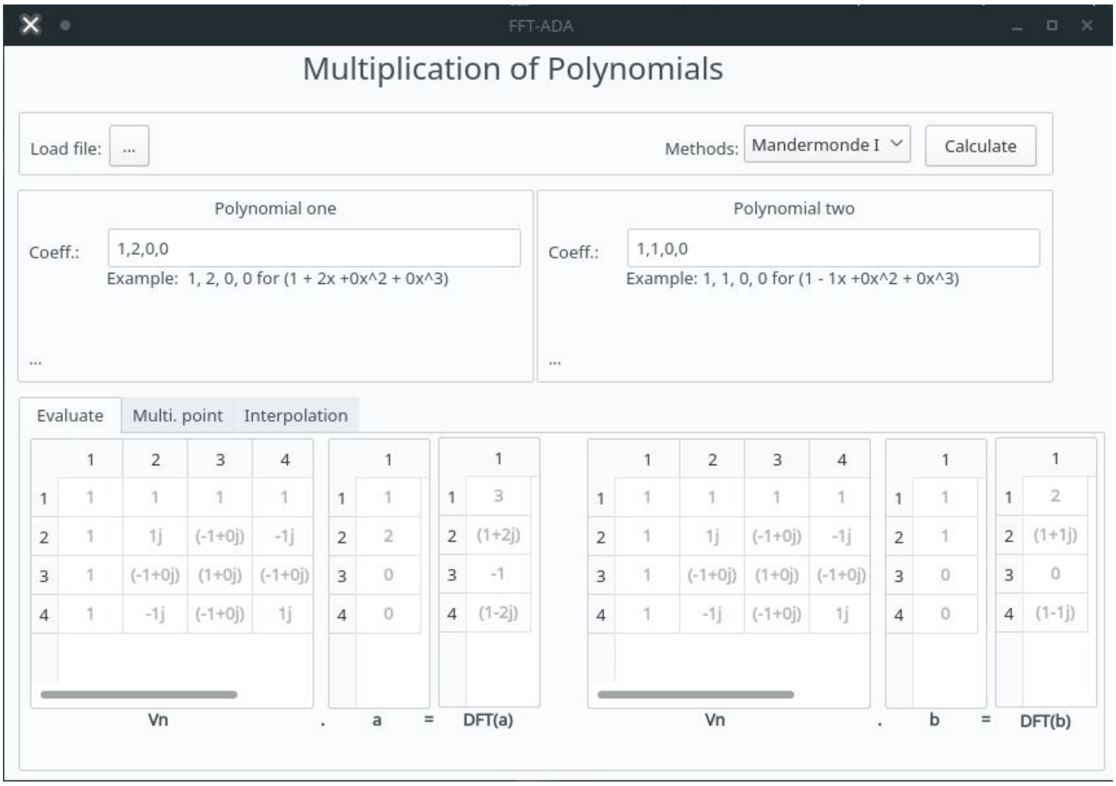
- Ingresamos los coeficientes de los polinomios a(x) y b(x)



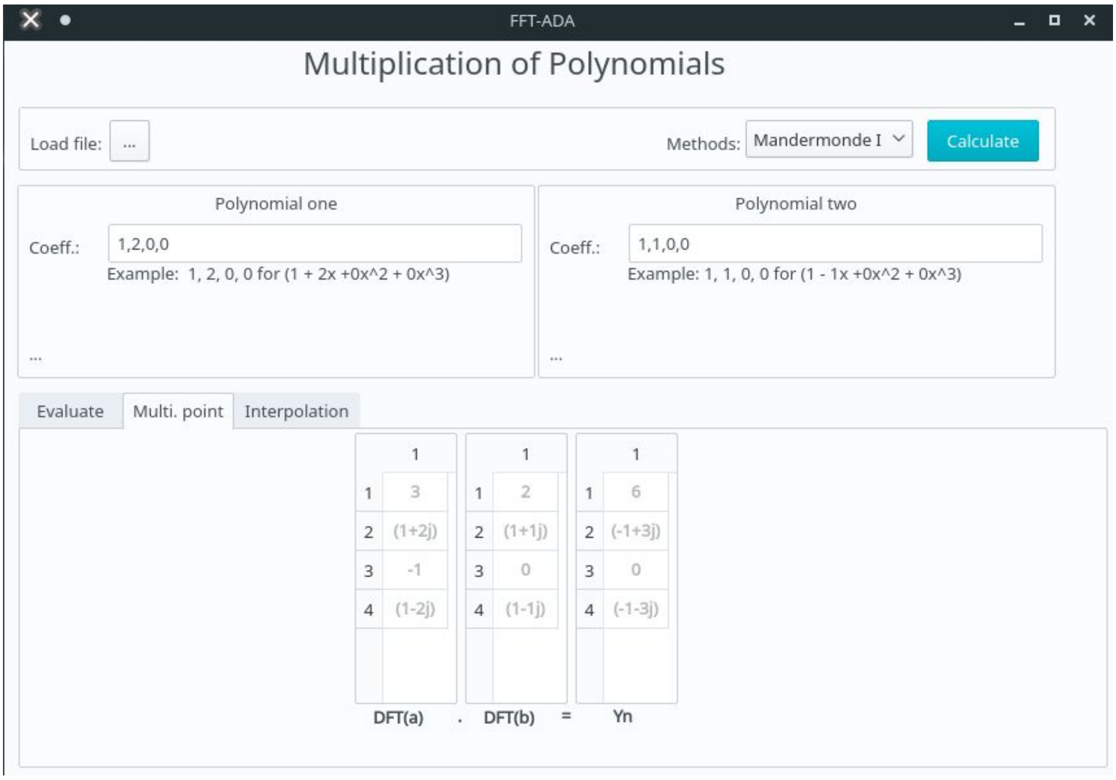
- Elegimos el método para multiplicar y damos click en el botón Calcular:

## 4. Resultados

- Evaluación




- Producto punto


FFT-ADA
Multiplication of Polynomials
Load file:
...
Methods:
Mandermonde I
Calculate
Polynomial one
Coeff.:
1,2,0,0
Example: 1, 2, 0, 0 for (1 + 2x +0x^2 + 0x^3)
Polynomial two
Coeff.:
1,1,0,0
Example: 1, 1, 0, 0 for (1 - 1x +0x^2 + 0x^3)
Evaluate
Multi. point
Interpolation
1 3
2 (1+2j)
3 -1
4 (1-2j)
1 2
2 (1+1j)
3 0
4 (1-1j)
1 6
2 (-1+3j)
3 0
4 (-1-3j)
DFT(a) . DFT(b) = Yn

- Interpolación