# EX - 4 EMPIRICAL ANALYSIS FOR POLYNOMIAL EVALUATION

T SADAKOPA RAMAKRISHNAN | 3122225002109 | IT - B

algorithms as well.

1. Let $p(x)$ be a polynomial of degree $n$, that is, $p(x) = \sum_{i=0}^{n} a_i x^i$.
   (a) Implement a simple $O(n^2)$-time algorithm using Python for computing $p(x)$, for a given value of $x$
   (b) Implement a $O(n \log n)$ algorithm for computing $p(x)$, based upon a more efficient calculation of $x^i$
   (c) Now, consider rewriting $p(x)$ as

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots + x(a_{n-1} + xa_n)\cdots)))$$

   which is known as the Horner's method. Write a Python function to compute $p(x)$ using this method. Analyze the time complexity of your code and express the same in asymptotic notation.

**a)**

```
# -*- coding: utf-8 -*-
"""
This module provides a function that evaluates polynomial
of n degree, given by user using the O(n^2) Time Complexity
or Aka Brute Force Method. This is a part
of the exercises given under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may
contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome

Created on Wed Apr 26 2023

Revised on Wed May 7 2023

Original Author: T. Sadakopa Ramakrishnan
<sadakopa2210221@ssn.edu.in>
"""
```

```
#importing random module for creating random coefficients
import random

def polynomial():
    '''
    This functions takes in the degree of polynomial from
user
    and uses random module to create coefficients and
appends
    to a list and returns the list.

    Returns : A list of coefficients
    '''
    n = int(input("Enter the degree of polynomial: "))
    coeff = []
    for i in range(n+1):
        coeffs = random.randint(1,100)
        coeff.append(coeffs)

    print("Coefficients are: ", coeff)
    return coeff

def getx():
    '''
    This function takes in the value of x from user
    to evaluate the polynomial.

    Returns the value of x to be used in main function.
    '''
    x = int(input("Enter the value of x: "))
    return x

def BruteForce(coeff, x):
    '''
    This function evaluates the polynomial using
    O(n^2) Time complexity.

    Takes in list and value of x as arguments.
```

```
    Returns the final evaluated value of the created
polynomial.
    '''
    fn = 0
    n = len(coeff)
    result = 0
    for i in coeff:
        fn += 1
        coeff = i
        for j in range(1,n):
            fn += 1
            coeff *= x
        result += coeff
        n -= 1

    print("f(n) for BruteForce method of degree", n, "is:",
fn)
    return result

#Running the above program.
print(BruteForce(polynomial(), getx()))
```

**Output:**

```
Enter the degree of polynomial: 2
Coefficients are:  [30, 30, 28]
Enter the value of x: 1
f(n) for BruteForce method of degree 0 is: 6
88
```

**b)**

```
# -*- coding: utf-8 -*-
"""
This module provides a function that evaluates polynomial
```

```
of n degree, given by the user using the O(nlogn) Time
Complexity.
This is a part of the exercises given under the course
UIT2201 (Programming and Data Structures).

In this source code I've executed my own logic and may
contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 26 2023

Revised on Wed May 7 2023

Original Author: T. Sadakopa Ramakrishnan
<sadakopa2210221@ssn.edu.in>
"""


import random
def polynomial():
    '''
    This functions takes in the degree of polynomial from
user
    and uses random module to create coefficients and
appends
    to a list and returns the list.

    Returns : A list of coefficients
    '''
    n = int(input("Enter the degree of polynomial: "))
    coeff = []
    for i in range(n+1):
        coeffs = random.randint(1,100)
        coeff.append(coeffs)

    print("Coefficients are: ", coeff)
```

```
    return coeff

def getx():
    '''
    This function takes in the value of x from user
    to evaluate the polynomial.

    Returns the value of x to be used in the main function.
    '''
    x = int(input("Enter the value of x: "))
    return x

def power(x, y):
    '''
    The given function calculates the value of x raised
    to the power y in time O(logn).

    The input is not modified in any way and there are no
    side effects.

    args:
        x: the base
        y: the power

    Returns:
        Value of x raised to the power y.

    '''

    fn = 0

    fn += 1

    if(y == 0):
        return 1
    temp = power(x, int(y / 2))
    if (y % 2 == 0):
        return temp * temp
```

```
    else:
        return x * temp * temp


def polyeval(coeff, x):
    '''
    This function evaluates the polynomial using
    O(nlogn) Time complexity.

    Takes in list and value of x as arguments.

    Returns the final evaluated value of the created
polynomial.
    '''
    fn = 0
    degree = len(coeff) - 1
    total_sum = 0
    for coeffs in coeff:
        fn += 1
        prod = power(x,degree)
        total_sum += prod*coeffs
        degree -= 1

    print("f(n):", fn)
    return total_sum

#Running the above program.
print(polyeval(polynomial(), getx()))
```

**Output:**

```
Enter the degree of polynomial: 2
Coefficients are:  [77, 87, 28]
Enter the value of x: 1
f(n): 3
192
```

c)

```
# -*- coding: utf-8 -*-
"""
This module provides a function that evaluates polynomial
of n degree, given by user using the O(n) Time Complexity
or Aka Horner's Method. This is a part
of the exercises given under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may
contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 26 2023

Revised on Wed May 7 2023

Original Author: T. Sadakopa Ramakrishnan
<sadakopa2210221@ssn.edu.in>
"""


import random
def polynomial():
    '''
    This functions takes in the degree of polynomial from
user
    and uses random module to create coefficients and
appends
    to a list and returns the list.

    Returns : A list of coefficients
    '''
    n = int(input("Enter the degree of polynomial: "))
    coeff = []
    for i in range(n+1):
```

```
        coeffs = random.randint(1,100)
        coeff.append(coeffs)

    print("Coefficients are: ", coeff)
    return coeff

def getx():
    '''
    This function takes in the value of x from user
    to evaluate the polynomial.

    Returns the value of x to be used in main function.
    '''
    x = int(input("Enter the value of x: "))
    return x

def Horner(coeff, x):
    '''
    This function evaluates the polynomial using
    O(n) Time complexity.

    Takes in the list and value of x as arguments.

    Returns the final evaluated value of the created
polynomial.
    '''
    fn = 0
    n = len(coeff)
    result = coeff[0]
    for i in range(1,n):
        fn += 1
        result = result * x + coeff[i]

    print("f(n) for Horner's method of degree", n, "is:",
fn)
    return result

#Running the above program.
```

```
print(Horner(polynomial(), getx()))
```

**Output:**

```
Enter the degree of polynomial: 2
Coefficients are:  [10, 41, 27]
Enter the value of x: 1
f(n) for Horner's method of degree 3 is: 2
78
```