CS 3010 Programming Assignment 4 - Report and Test Runs

CS 3010.02-1 Numerical Methods

By Nicholas Magtangob

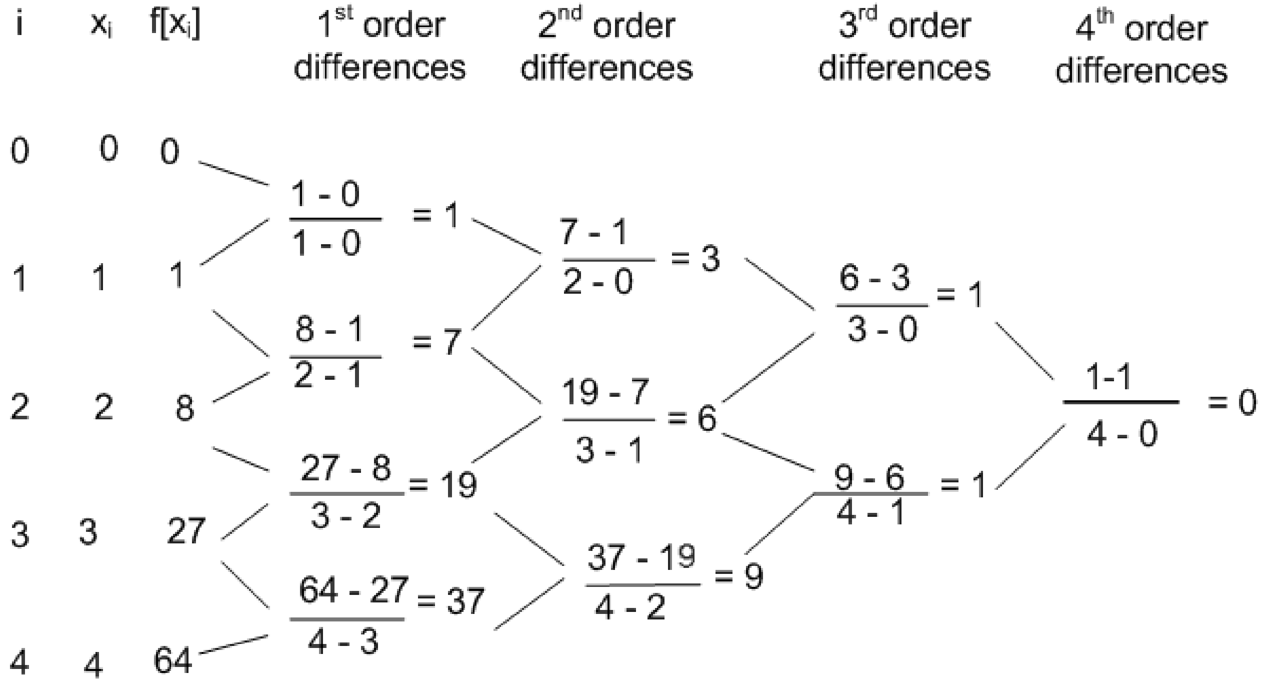
Due : 4 / 27 / 25

Introduction

For programming project 4, we are tasked with creating a divided difference table from a set of x values and corresponding y values, which can then be used to generate an interpolating polynomial.

For this program, it was originally created in Java but switched to Python because it features a library called Sympy which helps with the final phase of the project, where we have to simplify the polynomial we've generated via expanding it and collecting like terms.

The code follows the process outlined in creating a divided difference table from a set of points. We start by calculating the denominators, and store them into a 2D array which itself contains the denominators at each order. The denominators are simply the x values, but a distance apart, proportional to how far they are from each other on the triangle. The following image below was used for reference.



*Notice how the denominators at order 1 are 1 apart, and at order 2, 3, and 4, they are 2, 3, and 4 spaces apart.

Then, we create two similar pyramids by calculating the numerators and dividing difference values and storing them into their respective 2D arrays. We do both of these at the same time because each order of numerators requires the previous order of divided differences. To calculate the divided difference pyramid, we simply take the numerators from the 2D array in the previous order and divide them by the denominators at the 2D array in the previous order.

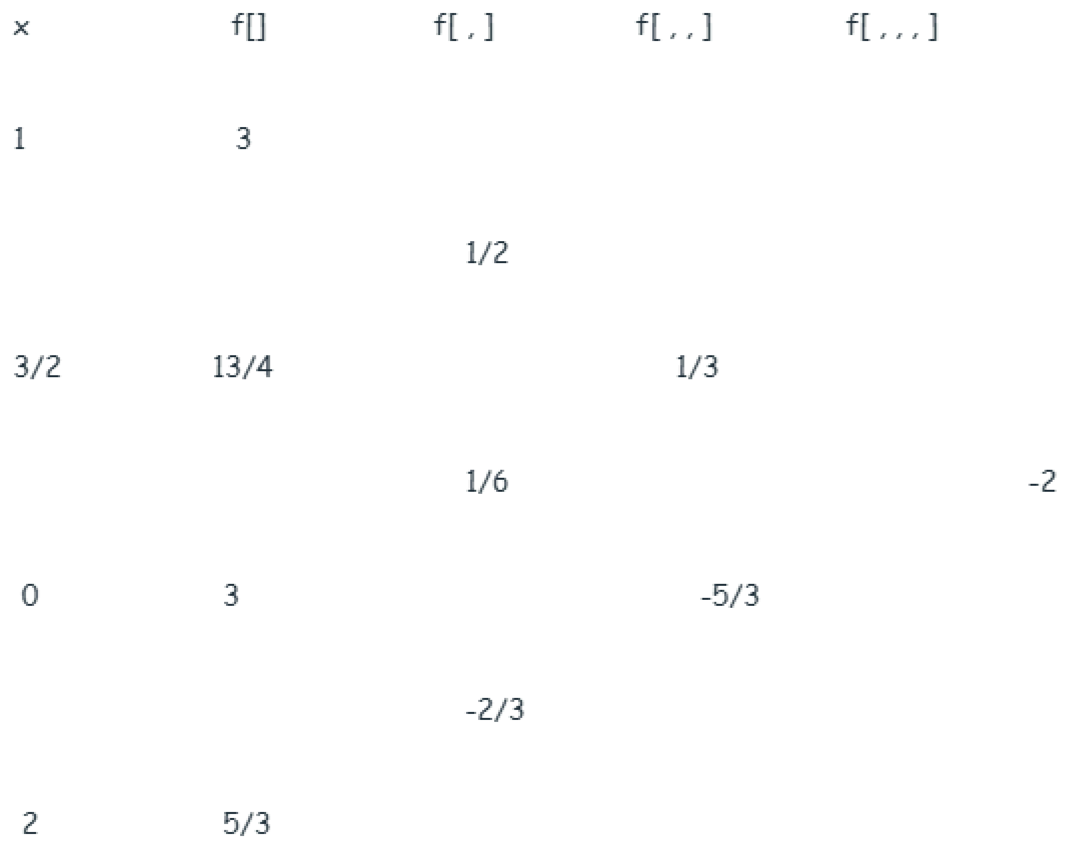
The final step is to print out both the polynomial generated from the divided difference values, and then the simplified form.

To print out the polynomial before simplifying, we simply need to take the first value of the divided differences we calculated, also including the 0th order 1st value, which in total is akin to taking the upper diagonal of the triangle formed by the divided difference pyramid. Then we get the respective values for "x - given x value" as dictated by the original set of points given by the user. For example, if the points were (0, 1), (2,3), and (5,6), then our interpolating polynomial will feature some combination of "x-0", "x-2", and "x-5". From here, we can easily generate the polynomial.

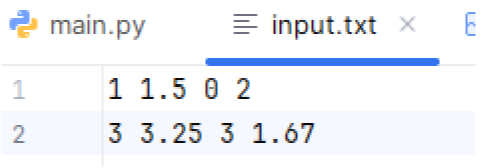
Then, to finally get the simplified version of the polynomial, we use a set of features provided in Python's library Sympy, which greatly helps in expanding, simplifying, and collecting like terms for most equations and polynomials when they are written as string. We only need to expand the polynomial we've generated, and then we collect like terms and print out the new polynomial.

Test Run #1 - Using the set of points listed on Canvas

For the first test run, we use the “input.txt” text file given on canvas. We can benchmark our program’s final result with the listed answers on canvas to see if things worked out.

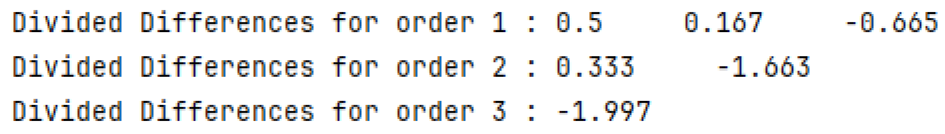


x	f[]	f[,]	f[, ,]	f[, , ,]
1	3			
		1/2		
3/2	13/4		1/3	
		1/6		-2
0	3		-5/3	
		-2/3		
2	5/3			



```
1 1.5 0 2
3 3.25 3 1.67
```

After running our program, we get the following output:
The divided differences were listed as :

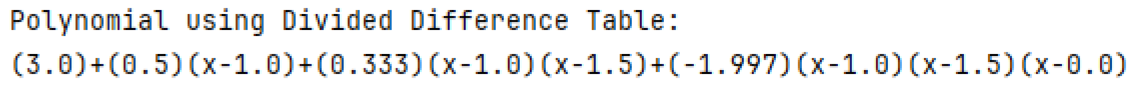


```
Divided Differences for order 1 : 0.5     0.167     -0.665
Divided Differences for order 2 : 0.333     -1.663
Divided Differences for order 3 : -1.997
```

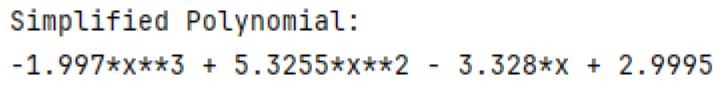
From these values, the upper diagonal of the table would be 3 (which is the first element at order 0, as it is a unique case where there was no real set of divided differences as we had no previous order to calculate numerators from), and then 0.5, 0.333, and -1.997

As we can see, our values are exceptionally close to the correct actual answer as listed on canvas, and we've rounded off to 3 decimal places.

Once we print out our polynomials and simplified form, we get:



```
Polynomial using Divided Difference Table:
(3.0)+(0.5)(x-1.0)+(0.333)(x-1.0)(x-1.5)+(-1.997)(x-1.0)(x-1.5)(x-0.0)
```

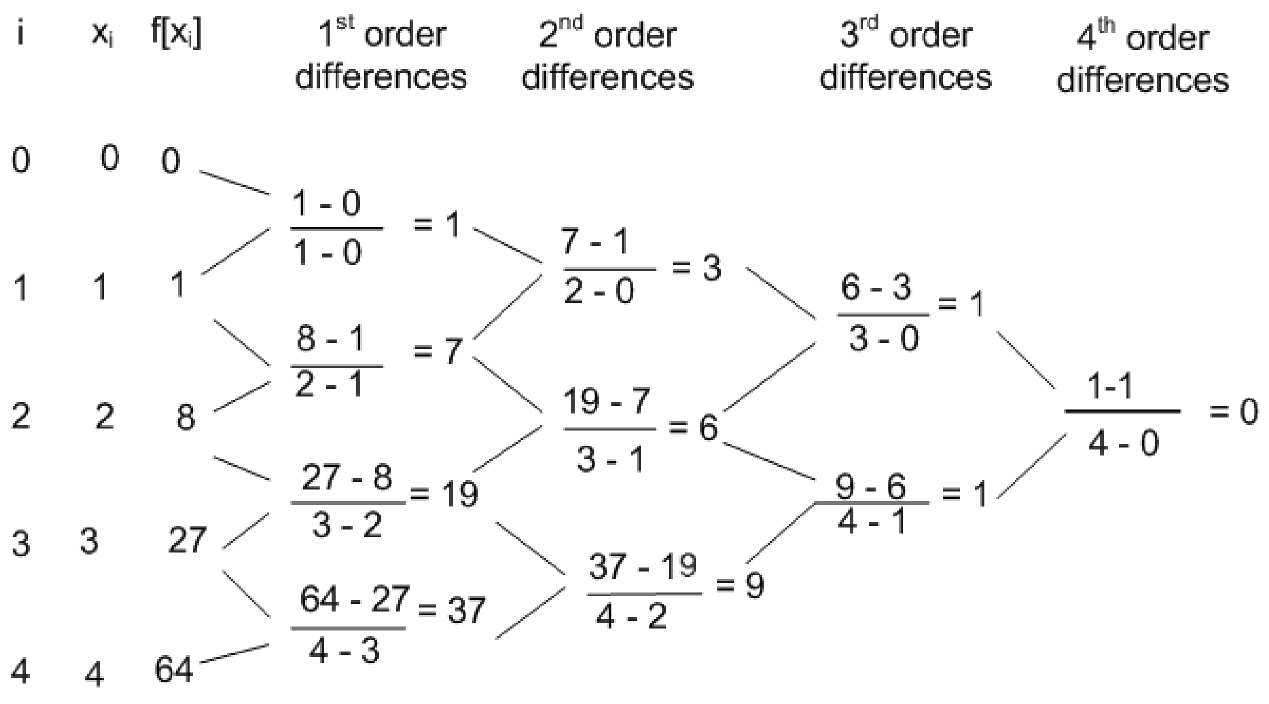


```
Simplified Polynomial:
-1.997*x**3 + 5.3255*x**2 - 3.328*x + 2.9995
```

Again, this is correct as it is similar to the answer listed on Canvas. The only difference here is that the -1.997 in the divided differences make the answers very close to the same answer, but only a few decimal places off. This is likely an error due to rounding, and if we extend the program a bit more, it can account for rounding values such as -1.997 to a close whole number such as -2.

Test run #2 - Using a set of points from an online example

For the second test run, we use a set of points as defined in the image below



As we can see, this is a great example because we can directly compare our answers to the image to see if we got the right divided differences.

```
Divided Differences for order 1 : 1.0     7.0     19.0     37.0
Divided Differences for order 2 : 3.0     6.0     9.0
Divided Differences for order 3 : 1.0     1.0
Divided Differences for order 4 : 0.0
```

Comparing the image to our console output, we can see that the same set of code we used for the previous test run has resulted in answers that are exactly correct. The first array indexes including the 0th order would be 0, 1, 3, 1, 0.

The resulting polynomial and simplified version outputted by the code are as follows:

```
Polynomial using Divided Difference Table:
(0.0)+(1.0)(x-0.0)+(3.0)(x-0.0)(x-1.0)+(1.0)(x-0.0)(x-1.0)(x-2.0)+(0.0)(x-0.0)(x-1.0)(x-2.0)(x-3.0)


Simplified Polynomial:
x**3
```