605.621 Foundations of Algorithms Spring 2024
Project Assignment #4

Pseudocode

```
INTERWEAVED(s, x, y)
result_x, result_y, result_noise = {}          # Final result to be added to
potential_x, potential_y, potential_noise = []          # potential values for x,y, and noise
idx_x, idx_y, idx_noise = 0          # Indexes for x, y, and noise
for i from 0 to length of s − 1
    if idx_x >= idx_y
        if idx_x < length of x and s[i] == x[idx_x]
            insert i+1 to potential_x
            increment idx_x
        else if idx_y < length of y and s[i] == y[idx_y]
            insert i+1 to potential_y
            increment idx_y
        else
            insert i+1 to potential_noise
            increment idx_noise
    else if idx_y > idx_x
        if idx_y < length of y and s[i] == y[idx_y]
            insert i+1 to potential_y
            increment idx_y
        else if idx_x < length of x and s[i] == x[idx_x]
            insert i+1 to potential_x
            increment idx_x
        else
            insert i+1 to potential_noise
            increment idx_noise
    if length of x == length of potential_x
        add potential_x into result_x
        clear potential_x
        reset idx_x
        if potential_noise is not empty
            add potential_noise to result_noise
            clear potential_noise
        reset idx_noise
    if length of y == length of potential_y
        add potential_y into result_y
        clear potential_y
        reset idx_y
        if potential_noise is not empty
            add potential_noise to result_noise
            clear potential_noise
        reset idx_noise
    if idx_noise ≥ 1 and length of potential_y ≤ 1 and length of potential_x ≤ 1
        clear potential_x
        clear potential_y
        reset idx_noise, idx_y, and idx_x
Check for Interweaving
insert remaining potential_noise into result_noise
return result_x, result_y, result_noise, is_interweave
```

Natural Language Description of Algorithm

The algorithm starts by accepting the inputs s, x, y. In a real system the overarching for loop and s would be the stream coming into the system. After creating result, potential, and indexing variables for x, y, and noise it enters the for loop. If index of x is greater than or equal to the index of y it will run through the first loop which checks for matches in the correct index of x and y and inserts to noise if nothing is found. It will only enter the else if statement if index of y is greater

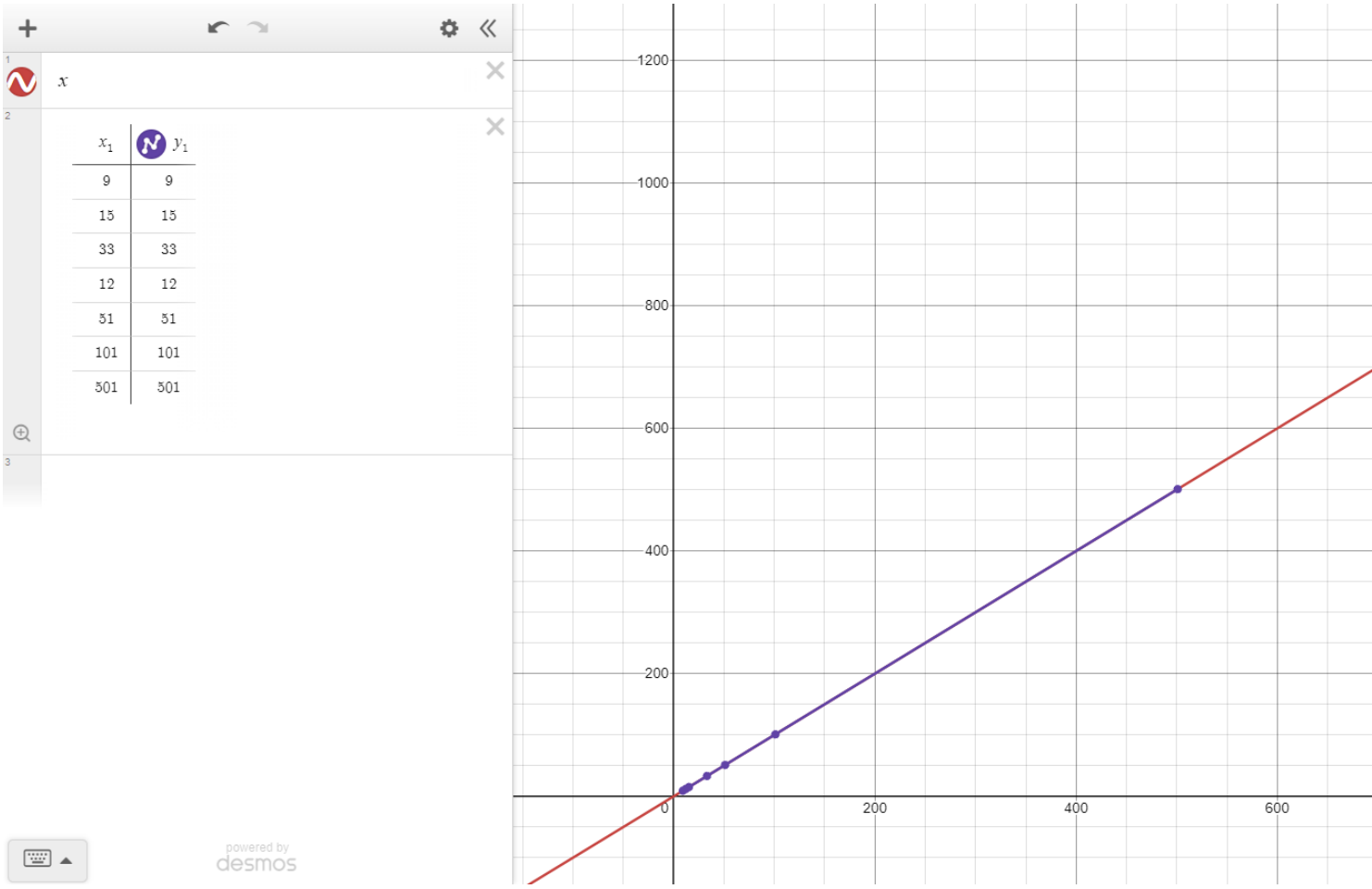
than index of x. Finally we check to ensure that the potential x and y don't have a found item, if they do it is pushed into their respective result variables to be passed back at the end of code. It will also push potential noise into result noise if necessary. Finally we hit the last if statement that is entered if potential x and y only have at most 1 variable in them and noise index is greater than 1. If this statement is entered it clears potential x and y and resets the index's. Finally before returning we insert the remaining items in potential noise into result noise. It will then return result x, y, and noise.

Asymptotic Analysis

	INTERWEAVED(s, x, y)		
1	result_x, result_y, result_noise = {}	c_1	1
2	potential_x, potential_y, potential_noise = []	c_2	1
3	idx_x, idx_y, idx_noise = 0	c_3	1
4	**for** i = 0 to length of s − 1	c_4	n
5	**if** idx_x >= idx_y	c_5	1
6	**if** idx_x < length of x and s[i] == x[idx_x]	c_6	1
7	insert i+1 to potential_x	c_7	1
8	increment idx_x	c_8	1
9	**Else If** idx_y < length of y and s[i] == y[idx_y]	c_9	1
10	insert i+1 to potential_y	c_{10}	1
11	increment idx_y	c_{11}	1
12	**Else**	c_{12}	1
13	insert i+1 to potential_noise	c_{13}	1
14	increment idx_noise	c_{14}	1
15	**Else if** idx_y > idx_x	c_{15}	1
16	**if** idx_y < length of y and s[i] == y[idx_y]	c_{16}	1
17	insert i+1 to potential_y	c_{17}	1
18	increment idx_y	c_{18}	1
19	**Else if** idx_x < length of x and s[i] == x[idx_x]	c_{19}	1
20	insert i+1 to potential_x	c_{20}	1
21	increment idx_x	c_{21}	1
22	**Else**	c_{22}	1
23	insert i+1 to potential_noise	c_{23}	1
24	increment idx_noise	c_{24}	1
25	**if** length of x == length of potential_x	c_{25}	1
26	add potential_x into result_x	c_{26}	1
27	clear potential_x	c_{27}	1
28	reset idx_x	c_{28}	1
29	**if** potential_noise is not empty	c_{29}	1
30	add potential_noise to result_noise	c_{30}	1
31	clear potential_noise	c_{31}	1
32	reset idx_noise	c_{32}	1
33	**if** length of y == length of potential_y	c_{33}	1
34	add potential_y into result_y	c_{34}	1
35	clear potential_y	c_{35}	1
36	reset idx_y	c_{36}	1
37	**if** potential_noise is not empty	c_{37}	1
38	add potential_noise to result_noise	c_{38}	1
39	clear potential_noise	c_{39}	1
40	reset idx_noise	c_{40}	1
41	**if** idx_noise ≥ 1 and length of potential_y ≤ 1 and length of potential_x ≤ 1	c_{41}	1
42	clear potential_x	c_{42}	1
43	clear potential_y	c_{43}	1
44	reset idx_noise, idx_y, and idx_x	c_{44}	1
45	check for interweaving	c_{45}	1
46	insert remaining potential_noise into result_noise	c_{46}	1
47	**return** result_x, result_y, result_noise, is_interweave	c_{47}	1

As we can see in the line by line list of runtime affecting's the runtime of this program would be **O(n)**. Each character is only looked at one time as it is a "stream" and cannot be looked at again once it is gone. Besides that each if/elif/else statement has only linear effect on the runtime, same with inserting the index's into their final items.

Asymptotic Analysis Comparison



While there are other items we can look at while determining the runtime of the code there is only one piece of the code that actually affects the runtime, how many times the loop runs. As we can see x is the input length and y would be the amount of loops. We can see there is an exact 1 to 1 relation between input size and loop runs.