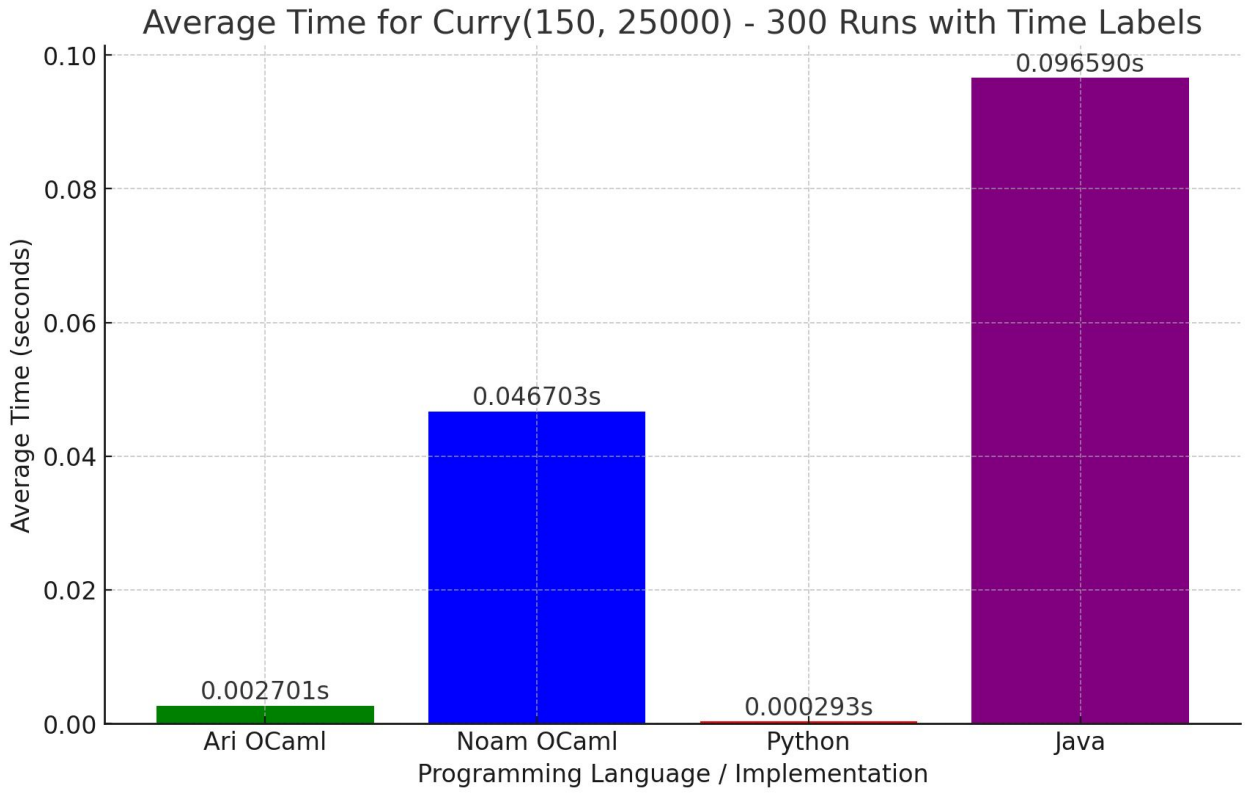
linear-expansion
פרוייקט בOcaml
מצגת השוואות ביצועים

מגישים: ארי פייגלין, נועם קפלינסקי

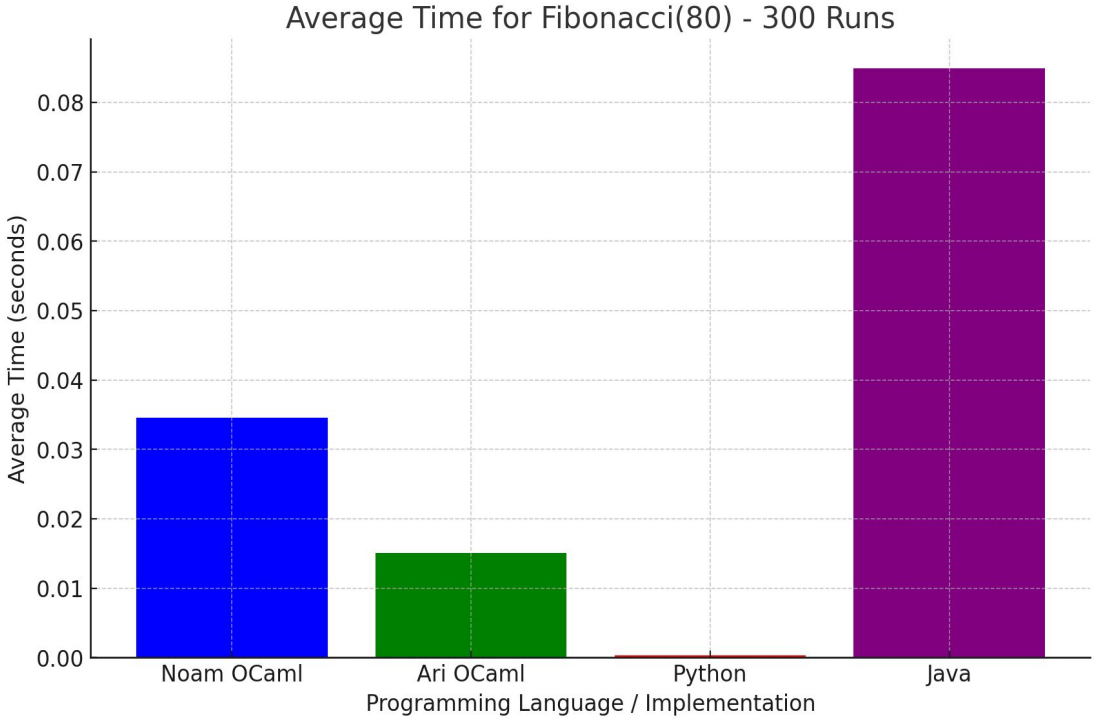
השוואות בין המודלים השונים,java וpython

דוגמא ראשונה:



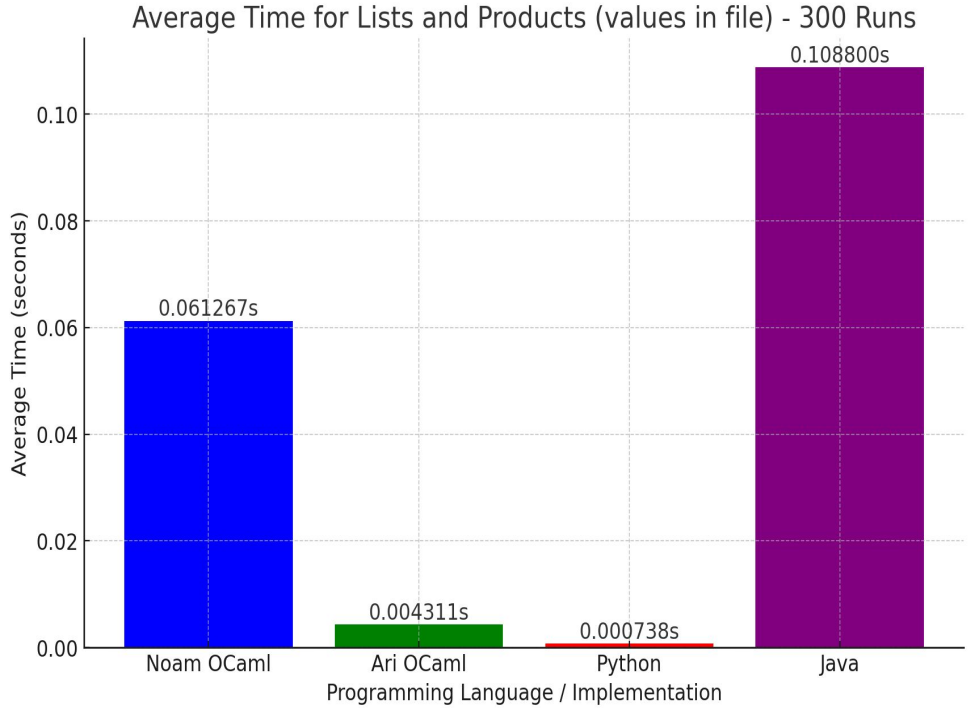

```
fun print (x) {
    _prim_print x;
}

fun curry (f) {
    fun curried (x) {
        fun curriedX (y) {
            f (x,y)
        }
        curriedX
    }
    curried
}

fun plus (x,y) {
    x + y
}

print (plus (10, 20));
let curry_plus = curry plus;
print (curry_plus 10 20);
```

דוגמא שנייה:




```
1   fun print (x) {
2       _prim_print x;
3   }
4
5   fun fibonacci (n) {
6       switch
7       | n == 0 -> { [0;] }
8       | n == 1 -> { [0; 1;] }
9       | 1 -> {
10          let prev = fibonacci (n-1);
11          prev @ [prev.(n-1) + prev.(n-2);]
12      }
13      end
14  }
15
16  print (fibonacci 30);
```

דוגמא 3:




```
fun print (x) {
    _prim_print x;
}

fun length (l) {
    _prim_len l
    _reg_out
}

fun tail (l) {
    _prim_tail l
    _reg_out
}

fun map (l,f) {
    if (length l == 0) {
        l
    }{
        [f (1.0);] @ (map ((tail l), f))
    }
}

fun square (n) {
    n * n
}
```

```
let arr = [0-3; 0-2; 0-1; 0; 1; 2; 3;];
print (map (arr, square));

fun fold_left (f,acc,l) {
    if (length l == 0) {
        acc
    }{
        fold_left (f, f (acc,1.0), tail l)
    }
}

fun sub (n,m) {
    n - m
}

let arr = [1; 2; 3; 4;];
print (fold_left (sub, 10, arr));

fun reverse (l) {
    if (length l == 0) {
        l
    }{
        reverse (tail l) @ [1.0;]
    }
}

print (reverse arr);

fun fst (a,b) {
    a
}

fun snd (a,b) {
    b
}

fun flip (p) {
    (snd p, fst p)
}

print (flip (10,20));
```

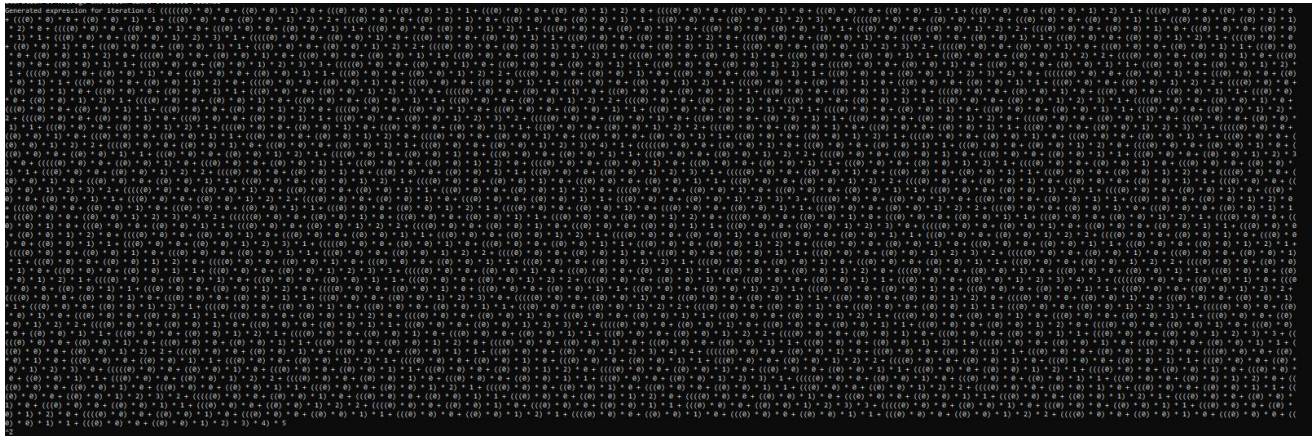
השוואות בין המודלים השונים בסדר קושי עולה

דוגמא ראשונה:

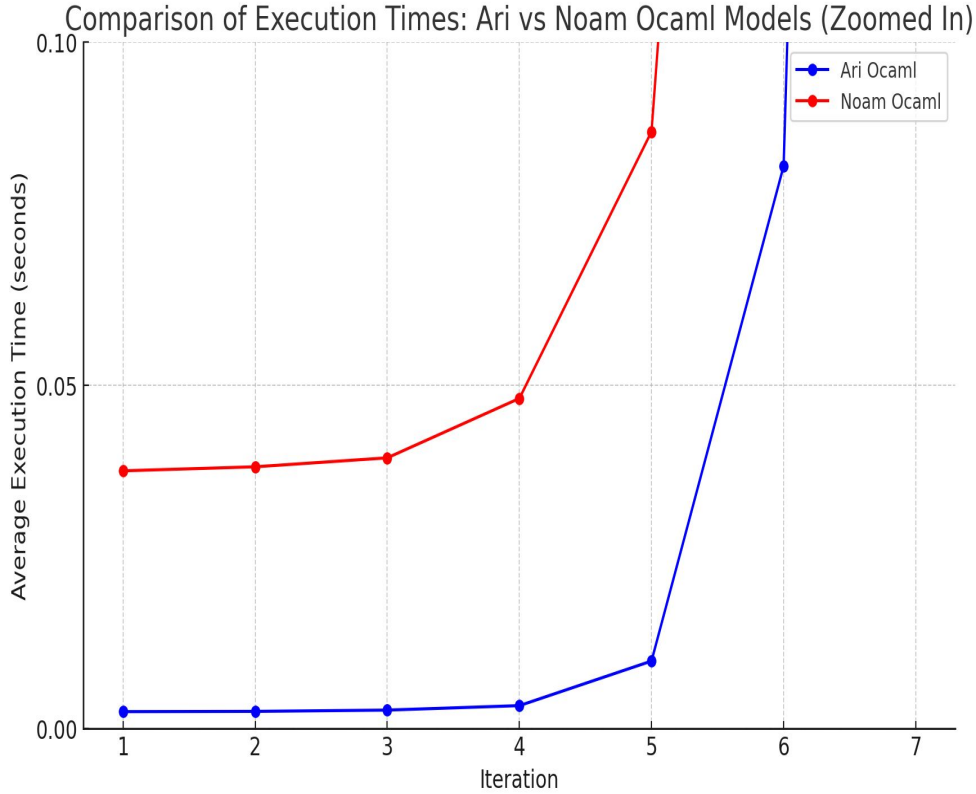
הקוד היוצר את הדוגמאות, depth,length מסמנים את העומק והרוחב של הביטוי מבחינת סוגריית ומספרים,
הבדיקה מתבצעת מ0,0 עד 7,7 (8,8 גורם לstack overflow במחשב)

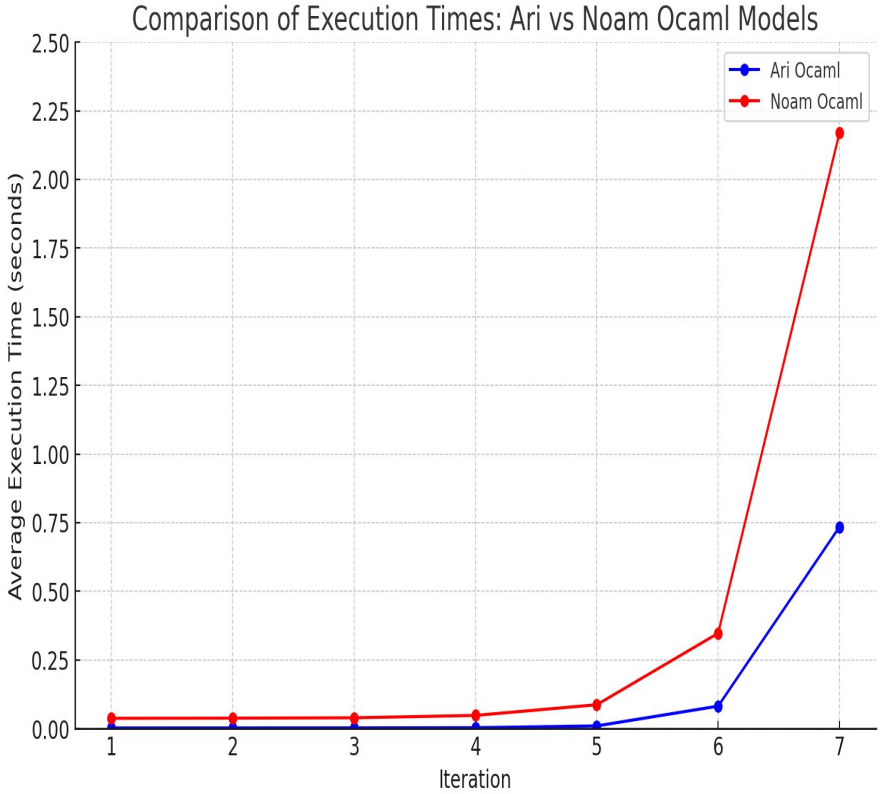
```
def generate_expression(depth, length):
    if depth == 0:
        return str(length)  # Base case: return a number representing the length when depth is 0

    string = ""
    for i in range(length):
        if depth > 0:
            # Recursively increase the depth with parentheses
            string += "(" + generate_expression(depth - 1, length - 1) + ")"
        string += f" * {i}"  # Add the multiplication operator between terms

        if i < length - 1:
            string += " + "  # Add the plus operator between terms except after the last one

    # Adding the final number to close the expression
    return string
```

דוגמא לאיך נראה הקוד עם הערכים 5,5:



ההשוואת לדוגמא הקודמת:





דוגמא שנייה:

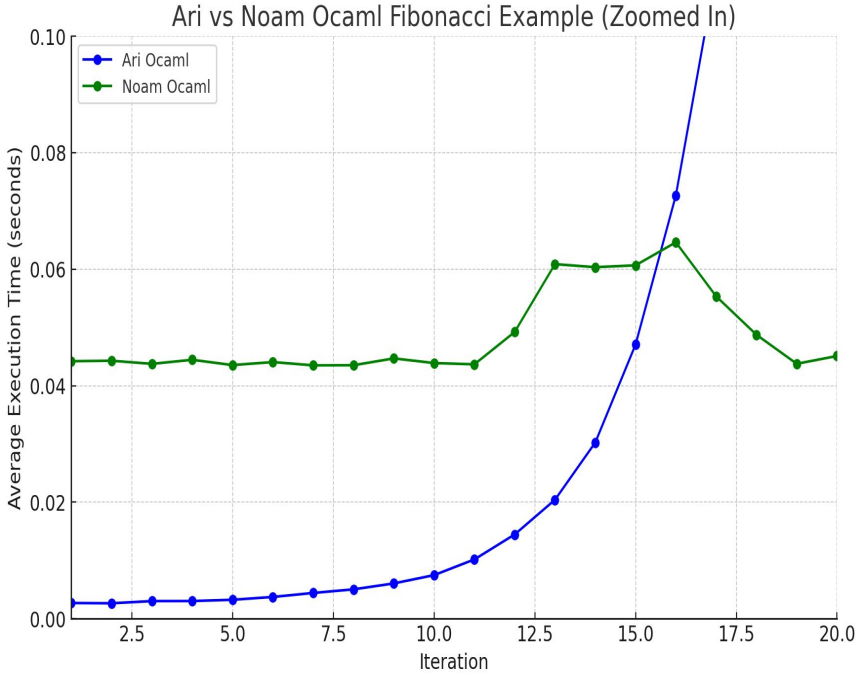
הקוד היוצר את הדוגמאות, עוברים בלולאה מ1 עד 20, ערך זה הוא הערך שעבורו תחושב סדרת הפיבונצ'י, השוני בין מקרה זה למקרה מהדוגמאות הקודמות הינו שזה פיבונאצ'י
לא לינארי

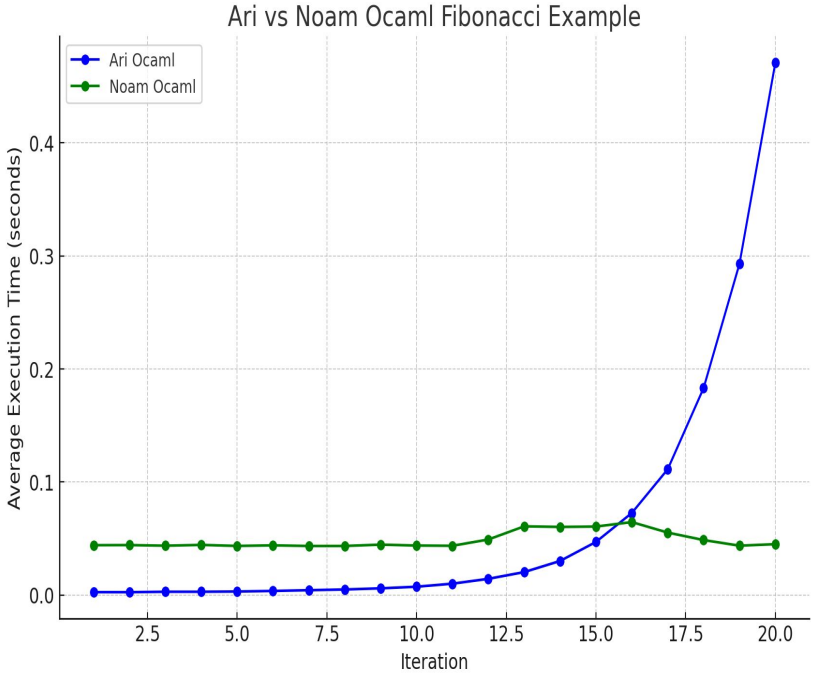
```python
import os

# Function to generate the OCaml Fibonacci code with changing `num`
def generate_fib_code(num):
    return f"""
fun print (x) {{
  _prim_print x
}}

fun fib (n) {{
  switch
  | n <= 1 -> {{ 1 }}
  | 1 -> {{ fib (n-1) + fib (n-2) }}
  end
}}

print (fib {num})
"""
# Function to write the OCaml code to a file
def write_fib_code_to_file(num, iteration):
    filename = f"fib_expression_{iteration}.ml"
    expression = generate_fib_code(num)
    # Print the expression for debugging purposes
    print(f"Generated Fibonacci expression for iteration {iteration}: {expression}")
    with open(filename, "w") as f:
        f.write(expression)
    return filename

# Main loop for 7 iterations
for i in range(1, 31):
    num = i  # You can adjust how `num` changes according to the iteration

    # Generate and save the Fibonacci expression
    expression_file = write_fib_code_to_file(num, i)

    # Run the `run_300_times_with_time` script with the generated expression
    os.system(f"./run_300_times_with_time {expression_file}")

    # Read the execution times and print the average only
    with open("execution_times.txt", "r") as f:
        times = [float(line.strip()) for line in f.readlines() if line.strip()]
        average_time = sum(times) / len(times) if times else 0
        print(f"Iteration {i}: Average execution time: {average_time:.6f} seconds")
```

דוגמא כיצד הקוד נראה עם הערך 10:

```
fun print (x) {
  _prim_print x
}

fun fib (n) {
  switch
  | n <= 1-> { 1 }
  | 1 -> { fib (n-1) + fib (n-2) }
  end
}

print (fib 10);
```

ההשוואות לדוגמא הקודמת:





רפלקציה של ארי:

I found this project a profound experience which I enjoyed thoroughly. Being able to fully dissect an original idea was something I had not been able to fully do before. Since I was placed in a position where people were depending on me to finish my work, I was able to do more work quickly, and I enjoyed it more.

I enjoyed the process of coming up with our algorithm. I enjoyed watching it evolve over time, from something basic which could handle only numerical expressions, to something which could interpret and parse an entire programming language.

Working with another person was also quite enlightening. My experience with working in a group is somewhat limited, and the opportunity to work in a group was valuable. In many of my previous experiences with group projects, it was hard to work with my partner, but Noam was great and we worked well together.

רפלקציה של נועם:

I found this project was a deeply rewarding experience that I thoroughly enjoyed. Having the opportunity to fully break down an original idea and watch it grow was something I hadn’t done to this extent before. Knowing that Ari and I were relying on each other to meet our deadlines helped me stay focused, and the sense of teamwork made the process even more engaging.

I particularly enjoyed developing our algorithm. It started off as a simple algorithm but over time, we refined it into something that could interpret and parse a complete programming language, furthermore i really enjoyed creating and make the comparison between the models and seeing how well our new algorithm performs.

Collaborating with Ari was a great experience. I haven’t had a lot of opportunities to work in pairs on technical projects in this scale, but this time, the process was smooth and productive. One of the highlights was meeting in person to discuss about the project. It added a layer of creativity and allowed us to improve out ideas in real-time. In previous group projects, collaboration was sometimes challenging, but Ari and I worked well together and communicated effectively throughout.