חישוב ביולוגי – פרויקט
שאלה 1

מגישים :

אביתר כהן 205913858
יונתן טואף 324170497

שלב א' :

יצרנו מחלקה בשם GenNetwork אשר מכילה את המשתנים הבאים:

num_of_open_activators - מספר אקטיבטורים פעילים (0\1\2)

num_of_open_repressor – מספר רפרסורים פעילים (0\1\2)

gen_id – מציין את הרשת המתוארת (מתוך 9 רשתות נתונות)

```
class GenNetwork:
    num_of_open_activators: int
    num_of_open_repressor: int
    gen_id: str
```

שלב ב':

יצרנו את כל הרשתות האפשריות המתקבלות מהמצבים השונים של הגנים האקטיבטורים והרפרסורים.

```
def find_all_possible_networks() -> List[GenNetwork]:
    all_networks = []
    gen_id = 1

    for num_of_open_repressor in [0, 1, 2]:
        for num_of_open_activators in [0, 1, 2]:
            all_networks.append(
                GenNetwork(
                    num_of_open_activators, num_of_open_repressor, str(gen_id)
                )
            )
            gen_id += 1

    return all_networks
```

שלב ג':

נחשב את כל הפלטים האפשריים השונים (כל הקומבינציות האפשריות) מהרשתות שיצרנו קודם.

```
def generate_output_combinations(
    networks: List[GenNetwork],
) -> Dict[str, Dict[str, int]]:
    num_networks = len(networks)

    all_combinations = list(product([0, 1], repeat=num_networks))

    output_dict = {}

    for idx, combination in enumerate(all_combinations):
        func_key = f"func{idx+1}"
        output_dict[func_key] = {}
        for network, output in zip(networks, combination):
            output_dict[func_key][network.gen_id] = output

    return output_dict
```

שלב ד:

בדיקת הפונקציות המונוטוניות מכלל הפונקציות שהתקבלו.

```python
def is_monotonic(networks: List[GenNetwork], combination: Dict[str, int]) -> bool:
    # if all outputs are the same, return False
    if len(set(combination.values())) == 1:
        return False

    for net1 in networks:
        for net2 in networks:
            if net1 == net2:
                continue

            output1 = combination[net1.gen_id]
            output2 = combination[net2.gen_id]

            # Check monotonicity condition for activators
            if (
                net1.num_of_open_activators >= net2.num_of_open_activators
                and net1.num_of_open_repressor <= net2.num_of_open_repressor
            ):
                if (
                    output1 < output2
                ):  # Increasing activators should not decrease the output
                    return False

            # Check monotonicity condition for repressors
            if (
                net1.num_of_open_activators <= net2.num_of_open_activators
                and net1.num_of_open_repressor >= net2.num_of_open_repressor
            ):
                if (
                    output1 > output2
                ):  # Increasing repressors should not increase the output
                    return False

    return True
```

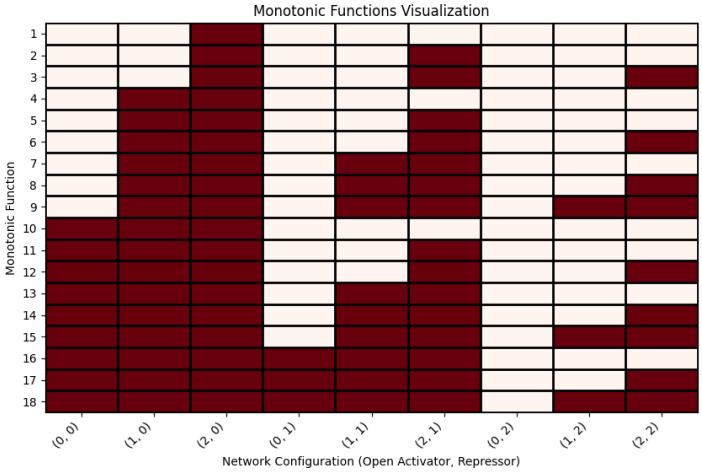
```python
def filter_monotonic_functions(
    networks: List[GenNetwork], output_dict: Dict[str, Dict[str, int]]
) -> Dict[str, Dict[str, int]]:
    monotonic_dict = {}

    for func_key, combination in output_dict.items():
        if is_monotonic(networks, combination):
            monotonic_dict[func_key] = combination

    return monotonic_dict
```

שלב ה':

הצגת הפונקציות המונוטוניות שהתקבלו.



שלב ו:

יצירת 'טסט' לבחינת בפונקציות המונוטוניות.

```python
def test_monotonic_functions():
    # arrange
    expected_monotonic_functions = EXPECTED_MONOTONIC_FUNCTIONS
    expected_num_monotonic_functions = EXPECTED_NUM_MONOTONIC_FUNCTIONS

    # act
    gen_networks = find_all_possible_networks()
    output_combinations = generate_output_combinations(gen_networks)
    monotonic_functions = filter_monotonic_functions(gen_networks, output_combinations)

    # assert
    monotonic_functions_list = [
        tuple(val.values()) for val in monotonic_functions.values()
    ]
    assert expected_num_monotonic_functions == len(monotonic_functions_list)

    for func in expected_monotonic_functions:
        assert func in monotonic_functions_list

    print("All tests passed successfully!")
```

הפלט המצופה (נעשה שימוש ע"י פונקציית הטסט).

```
EXPECTED_MONOTONIC_FUNCTIONS = [
    (0, 0, 1, 0, 0, 0, 0, 0, 0),
    (0, 1, 1, 0, 0, 0, 0, 0, 0),
    (0, 0, 1, 0, 0, 1, 0, 0, 0),
    (0, 1, 1, 0, 0, 1, 0, 0, 0),
    (0, 0, 1, 0, 0, 1, 0, 0, 1),
    (0, 1, 1, 0, 0, 1, 0, 0, 1),
    (0, 1, 1, 0, 1, 1, 0, 0, 0),
    (0, 1, 1, 0, 1, 1, 0, 0, 1),
    (0, 1, 1, 0, 1, 1, 0, 1, 1),
    (1, 1, 1, 0, 0, 0, 0, 0, 0),
    (1, 1, 1, 0, 0, 1, 0, 0, 0),
    (1, 1, 1, 0, 1, 1, 0, 0, 0),
    (1, 1, 1, 1, 1, 1, 0, 0, 0),
    (1, 1, 1, 0, 0, 1, 0, 0, 1),
    (1, 1, 1, 0, 1, 1, 0, 0, 1),
    (1, 1, 1, 1, 1, 1, 0, 0, 1),
    (1, 1, 1, 0, 1, 1, 0, 1, 1),
    (1, 1, 1, 1, 1, 1, 0, 1, 1),
]

EXPECTED_NUM_MONOTONIC_FUNCTIONS = len(EXPECTED_MONOTONIC_FUNCTIONS)
```