Module Name: Theory of Computation

Module Code: CIT3006

School: Computing and Information Technology

Lecturer: Khalilah Burrell Battick

Assignment: Group Project

Group Members:

Michael Johnson ID#2202369

Tashana Henry ID#1804274

Jordan Bogle ID#1704233

Kaydeen Walker ID#1902433

Due: April 4, 2025

PROJECT SYNOPSIS

This project explores the use of Deterministic Finite Automata (DFA) to detect significant biological patterns within DNA sequences, simulating a real-world application of finite automata in bioinformatics. DNA, composed of the nucleotide bases A, T, C, and G, encodes genetic information through codons, three-letter sequences that direct cellular processes. Some codons and motifs are biologically significant, such as the **start codon (ATG)**, **CAG repeats** associated with Huntington's disease and **GGT-GAT patterns** potentially linked to cancer mutations.

The objective of the project is to design a program that constructs and utilizes DFA to detect these patterns efficiently within a given DNA sequence. The DFA is built dynamically based on the input patterns, with well-defined states and transitions representing each nucleotide. The program processes the DNA sequence character-by-character, transitioning through DFA states and identifying when it reaches an accept state, thereby matching a pattern.

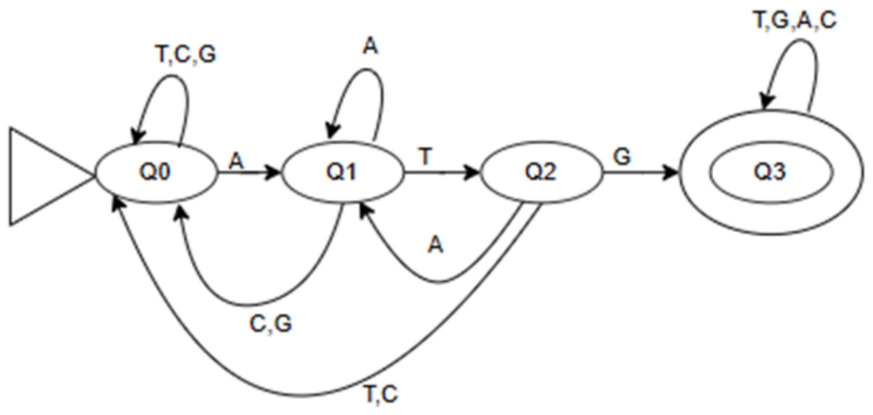
Key pattern detections include:

- **Start Codon Detection:** Identifies the presence of "ATG".
- **Huntington's Disease Gene Detection:** Searches for three consecutive repetitions of "CAG" following ATG.
- **Possible Cancer Mutation Detection:** Detects a "GGT" codon followed directly by "GAT".

If no significant patterns are found after locating ATG, the system outputs: "No significant patterns found." If ATG is not found at all, it reports: "Start codon not found."

EXPLANATION OF THE DFA DESIGN

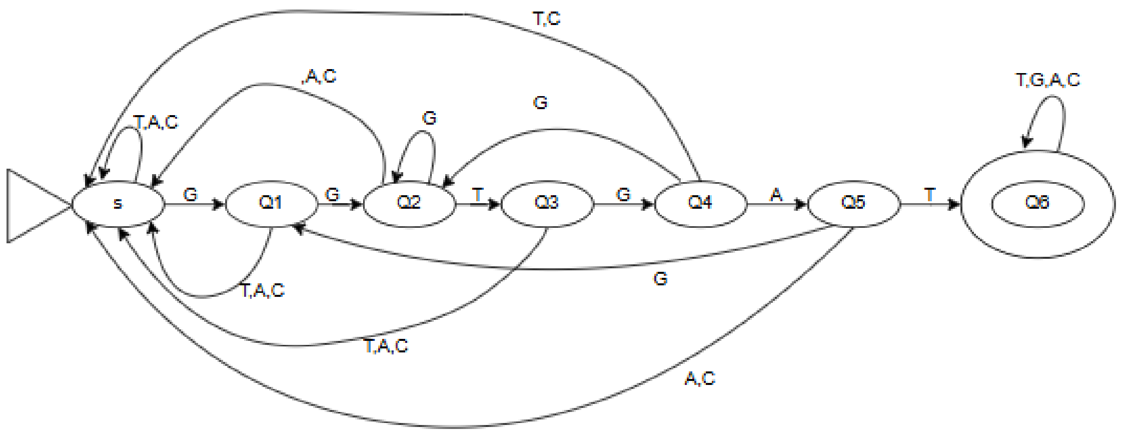
DFA for start codon



This DFA detects the start codon (ATG) in a DNA sequence by processing each nucleotide sequentially. It starts in Q0 and transitions based on input characters. When an 'A' is encountered, it moves to Q1, as this is the first letter of "ATG". If the next character is 'T', it transitions to Q2, and upon receiving 'G', it reaches the accepting state Q3, confirming the presence of "ATG".

If an incorrect nucleotide appears at Q1 or Q2, the DFA resets appropriately. However, instead of always going back to Q0, some transitions return to Q1 when an 'A' appears. This optimization prevents unnecessary resets—since 'A' is a valid start for "ATG", staying in Q1 allows the DFA to immediately check for a following 'T' without starting over completely. Once in Q3, the DFA remains in the accepting state regardless of further input, ensuring efficient O(n) detection of "ATG" in long DNA sequences.

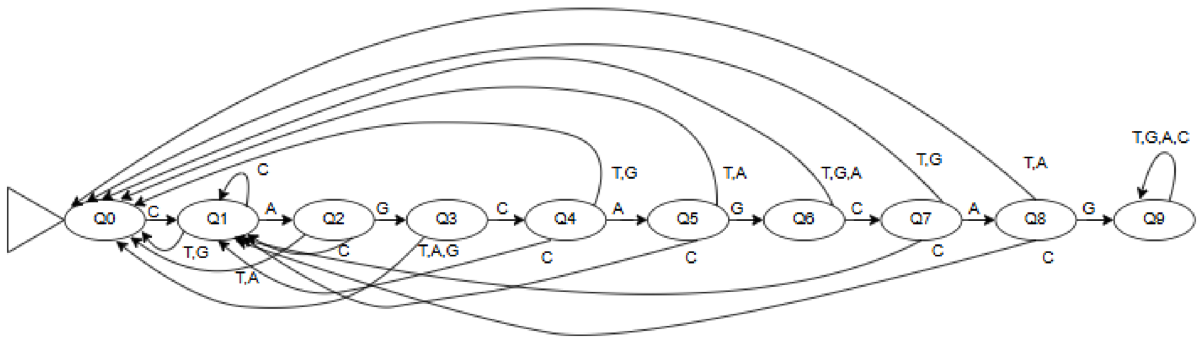
DFA for cancer detection



This DFA is designed to detect the DNA sequence of cancer mutations (GGT GAT). It starts at state S and transitions through Q1 to Q6 as it matches each character in "GGT GAT". When a 'G' is encountered, the DFA moves to Q1, then to Q2 on another 'G', Q3 on 'T', Q4 on 'G', Q5 on 'A', and finally reaches the accepting state Q6 upon detecting 'T', confirming the presence of "GGT GAT".

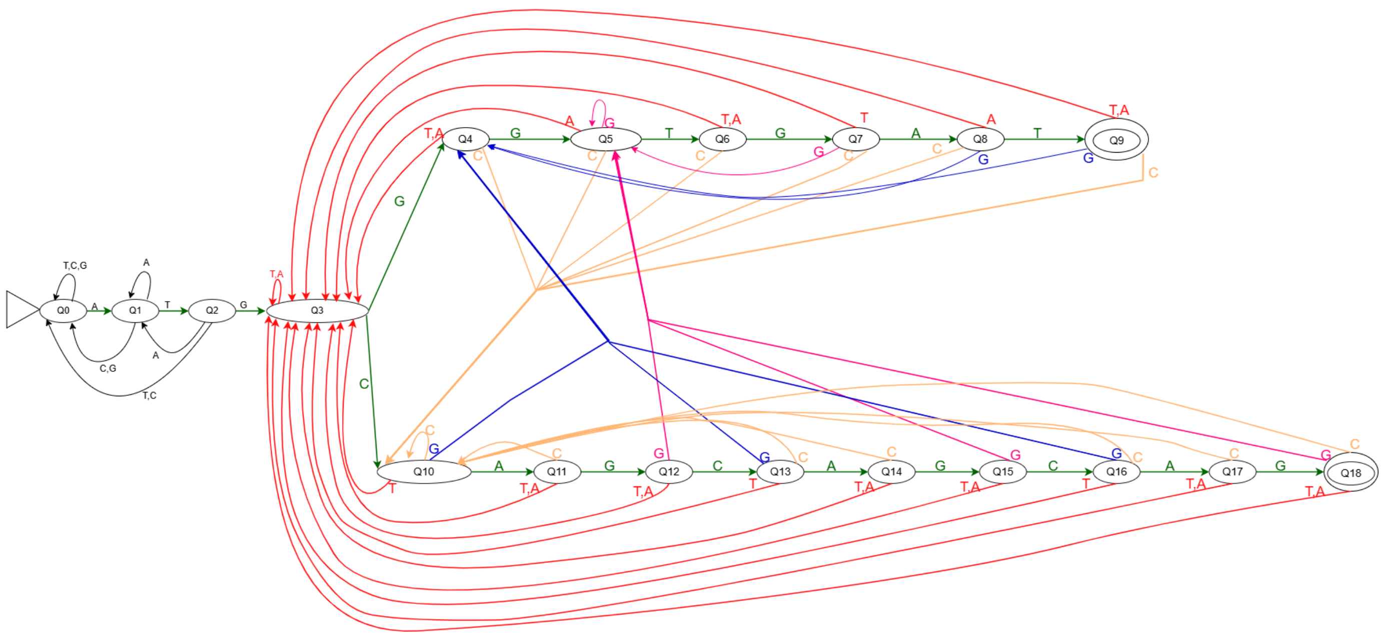
The DFA efficiently handles incorrect inputs by either resetting to an earlier state—if part of the sequence is still valid—or returning to S to restart detection. Notably, when an unexpected 'G' appears, the DFA may transition to a relevant state instead of fully resetting, allowing partial matches to continue. Once Q6 is reached, the DFA remains in the accepting state, this has a O(n) time complexity

DFA for Huntington’s detection



This DFA is designed to detect Huntington's disease mutation (CAG CAG CAG). It starts at state Q0 and transitions through Q1 to Q9, matching each character in "CAG CAG CAG". The DFA moves forward on the correct input and either resets completely or returns to an earlier state if part of the sequence is still valid. It always returns to Q0 on a 'T', as this nucleotide is not part of the sequence. If an unexpected 'C' appears, the DFA transitions back to Q1, while incorrect occurrences of 'G' or 'A' result in transitions to the start state.

A Combination of DFA’s



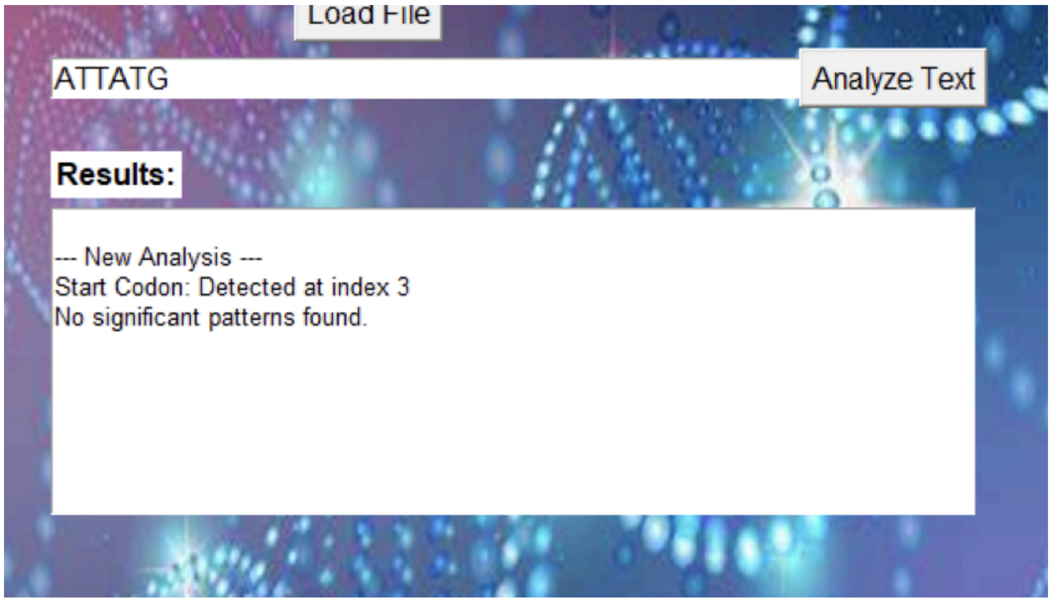
Key	
Green line	ideal transition in the sequence
Red lines	Goes to q3
Blue lines	Carrys G to q4
Pink lines	Carrys G to q5
Gold lines	Carrys C to Q10

This is the combined approach of all three DFAs, designed for efficiency. Searching for both cancer and Huntington's disease simultaneously is more effective than doing so independently, as the DFA can transition between the two sequences. For instance, when the start codon is detected and "CAG" is found, the DFA begins the Huntington's sequence, progressing to state Q12. However, if another "G" is detected, the sequence becomes "CAGG," triggering the switch to cancer detection Q6, as cancer mutations start with two consecutive G's, while Huntington's disease does not have any sequence with two G's. Similarly, if a "C" is detected anywhere in the cancer sequence, the DFA switches to Huntington's disease detection, as cancer mutations do not contain a "C." This dynamic switching optimizes the detection process for both conditions.

The time complexity remains O(n). A key is provided to aid in reading the DFA

TEST CASES

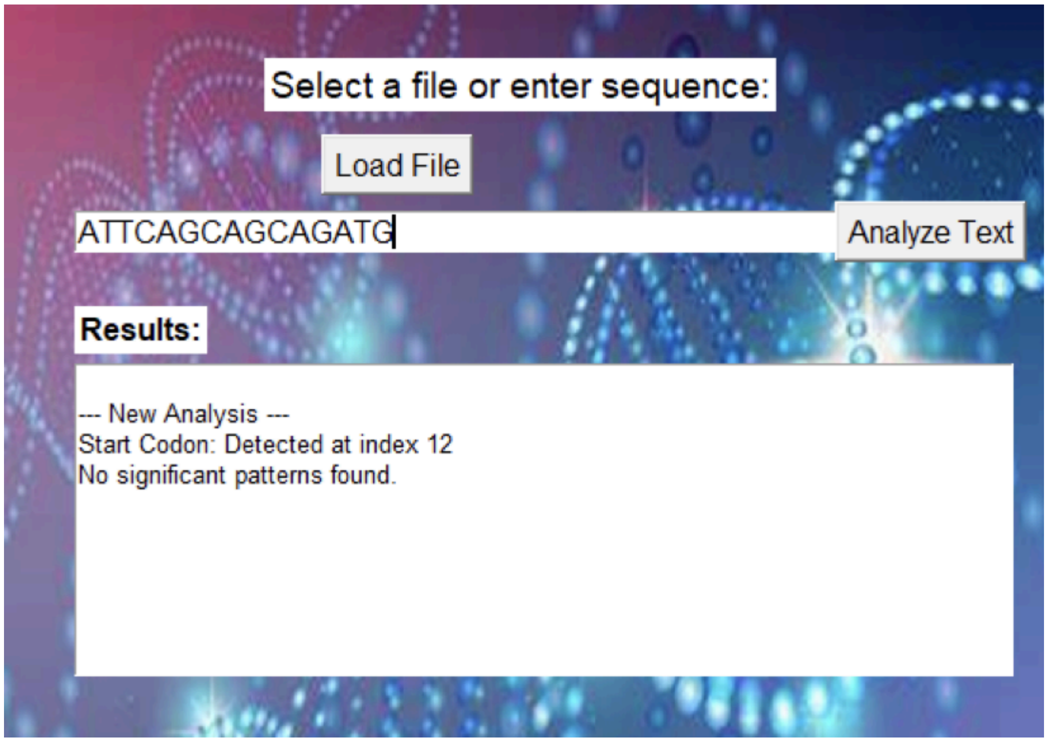
ATG with leading characters:



ATTATG

```
--- New Analysis ---
Start Codon: Detected at index 3
No significant patterns found.
```

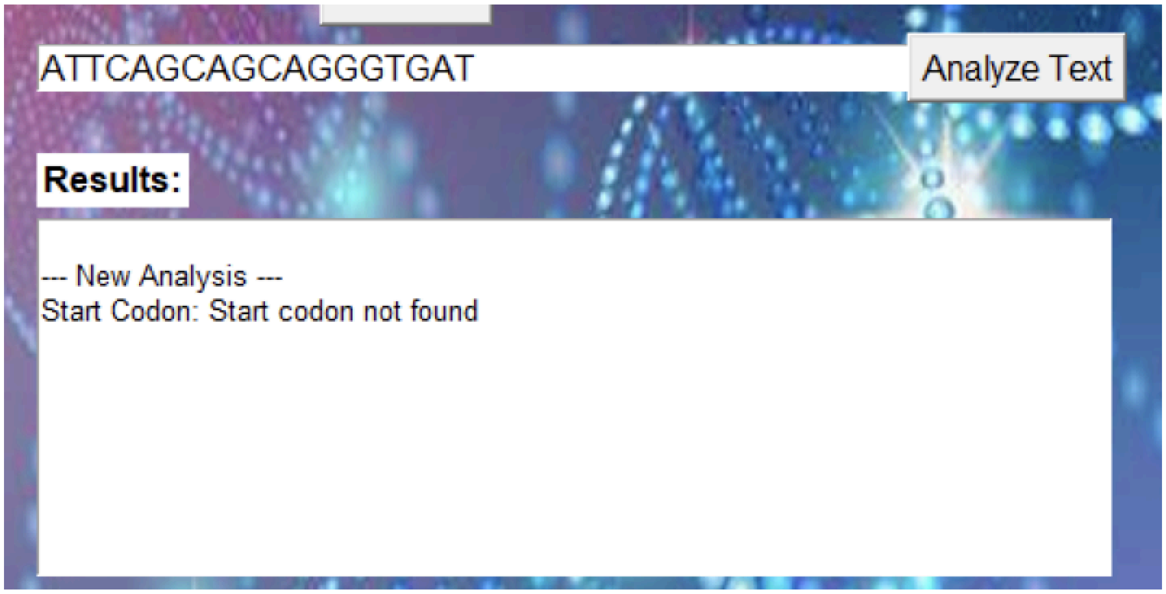
ATG after Huntington’s mutation sequence:



ATTCAGCAGCAGATG

```
--- New Analysis ---
Start Codon: Detected at index 12
No significant patterns found.
```

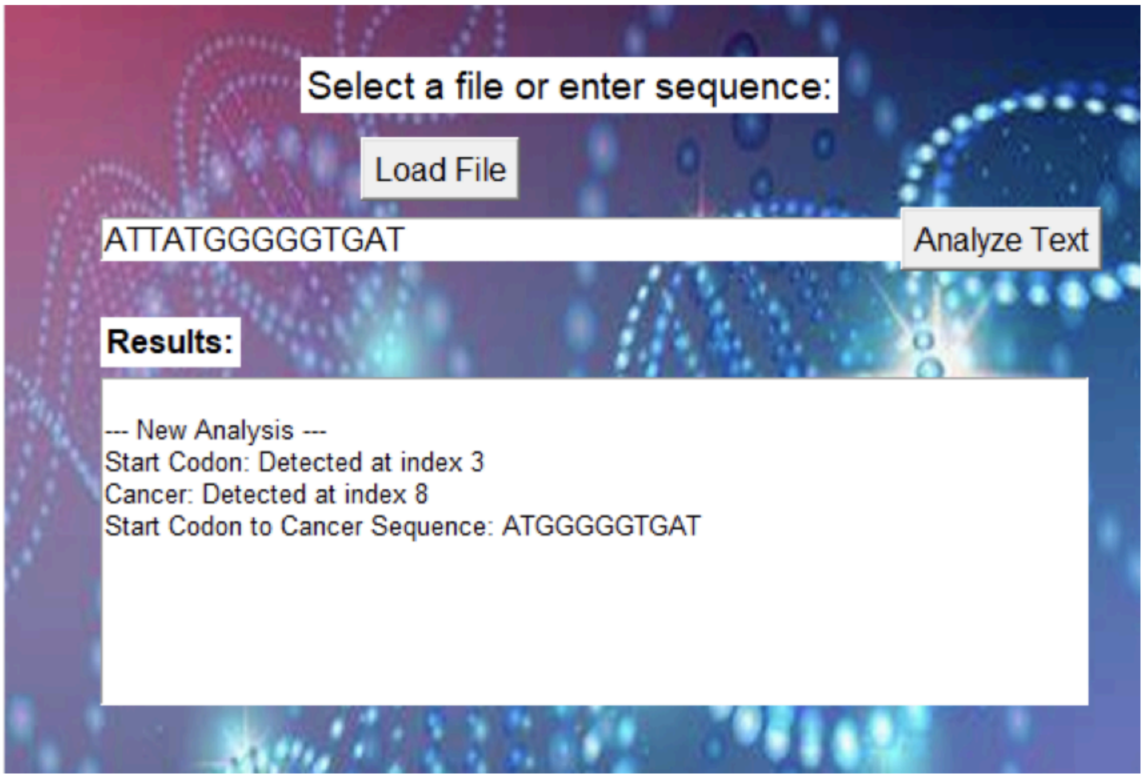
Huntingtons without start codon (No start codon)



```
ATTCAGCAGCAGGGTGAT
```

```
--- New Analysis ---
Start Codon: Start codon not found
```

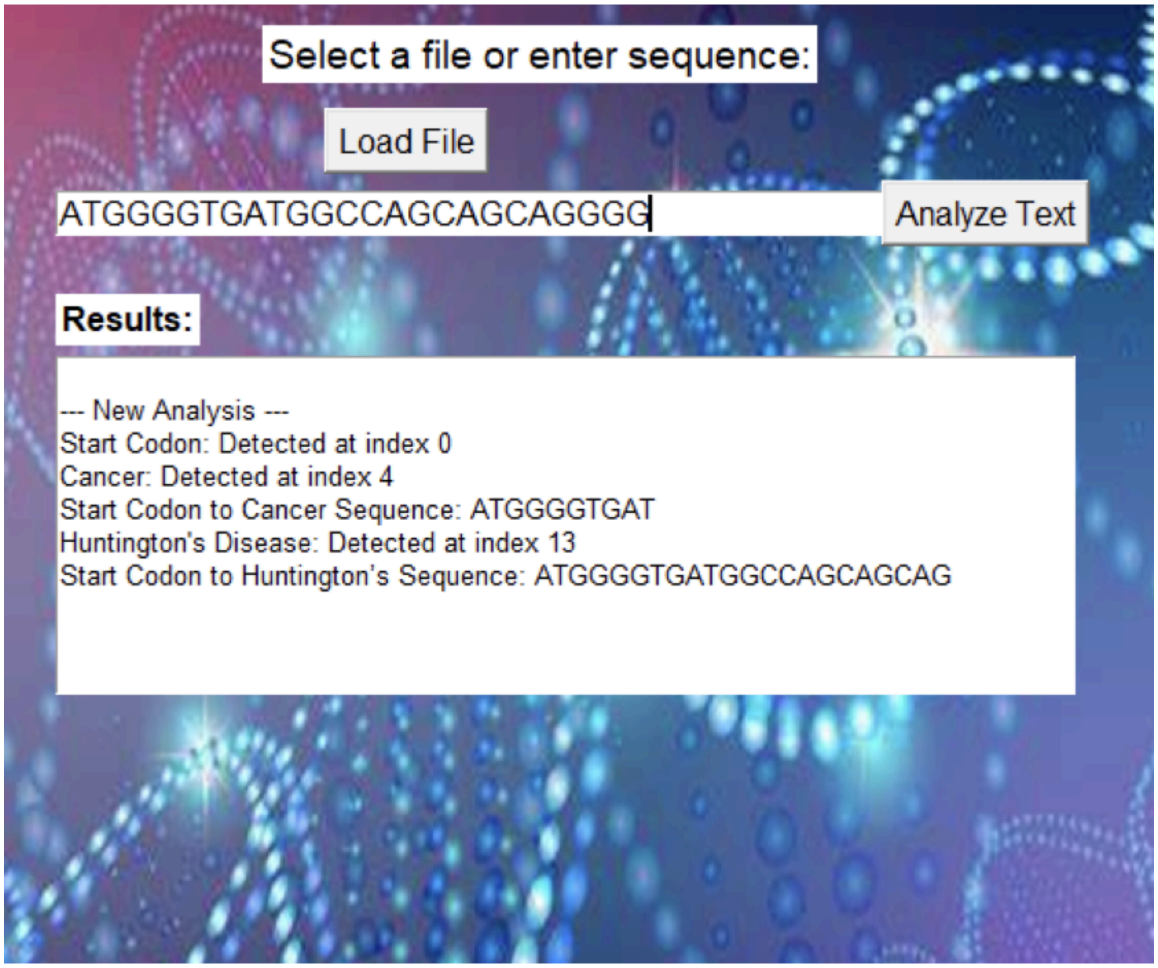
Cancer detection



```
ATTATGGGGGTGAT
```

```
--- New Analysis ---
Start Codon: Detected at index 3
Cancer: Detected at index 8
Start Codon to Cancer Sequence: ATGGGGGTGAT
```

Huntingtons and cancer detected



```
--- New Analysis ---
Start Codon: Detected at index 0
Cancer: Detected at index 4
Start Codon to Cancer Sequence: ATGGGGTGAT
Huntington's Disease: Detected at index 13
Start Codon to Huntington's Sequence: ATGGGGTGATGGCCAGCAGCAG
```

PERFORMANCE ANALYSIS

This program uses a Deterministic Finite Automaton (DFA) to detect genetic markers such as the start codon, cancer sequences, and Huntington's sequences within a DNA strand.

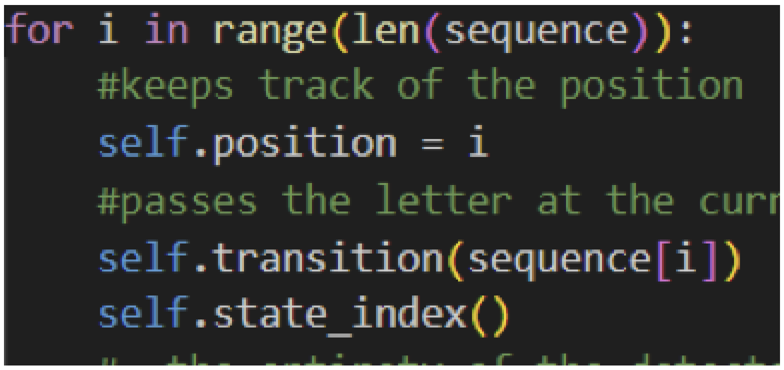
The DFA is implemented using a transition table (dictionary) where each state defines the next possible states based on input characters. The DNA sequence is processed using a single for loop that examines each character once, from left to right. For each character:

- A state transition occurs using a constant-time dictionary lookup (O(1))
- The program checks if a known sequence has been detected using a few conditional statements (also O(1))

Since the loop runs once over the entire sequence, and every operation inside the loop takes constant time, the total time complexity of the algorithm is O(n) — where *n* is the length of the input DNA sequence.

This is significantly more efficient than traditional string-matching algorithms that may check every possible substring, resulting in O(n × m) time per pattern (where *m* is the length of the pattern).

Image showing DFA Traversal Loop (O(n))



```
for i in range(len(sequence)):
    #keeps track of the position
    self.position = i
    #passes the letter at the curr
    self.transition(sequence[i])
    self.state_index()
```

This loop processes each character in the DNA sequence once. Inside the loop:

- transition() checks the next DFA state based on the character.
- state_index() checks if any genetic marker has been detected.

Since this loop only runs once over the sequence, and each step is constant time, the total runtime is **O(n)**.