Reporte.

Solución planteada:

Se propuso una solución para validar cadenas de entrada en un autómata utilizando el lenguaje de programación Racket. La solución se basa en la simulación del autómata, realizando transiciones de estado de acuerdo con la descripción formal proporcionada y verificando si cada cadena de entrada es aceptada o no.

Algoritmos implementados:

1. Funciones de búsqueda de transiciones:

- encontrarTransiciones: Encuentra las transiciones que comienzan desde un estado dado.

- encontrarSimbolos: Encuentra las transiciones que contienen símbolos de una lista dada.

- buscarTransiciones: Encuentra las transiciones alcanzables desde un estado inicial con símbolos de una lista dada.

2. Funciones de validación:

- mover: Realiza transiciones de estado basadas en una transición dada y la lista de entrada.

- verificar: Verifica si una lista de entrada es aceptada por el autómata.

- resultadoEntrada: Obtiene el resultado de verificar múltiples listas de entrada contra el autómata.

- validate: Función principal que valida las listas de entrada contra el autómata.

Complejidad del algoritmo:

La complejidad del algoritmo propuesto está dominada por el número de símbolos en la cadena de entrada y el número de transiciones en el autómata. Suponiendo que el número de estados y el tamaño del alfabeto son constantes, la complejidad sería O(n), donde n es la longitud de la cadena de entrada.

Conclusión.

En conclusión, la solución propuesta ofrece una manera eficiente y modular de validar cadenas en un autómata utilizando el lenguaje de programación Racket. La implementación de algoritmos específicos para buscar transiciones y validar las entradas garantiza un proceso claro y efectivo para verificar la aceptación de las cadenas por parte del autómata. Además, la complejidad del algoritmo asegura un rendimiento aceptable incluso para entradas grandes.