CSE 330 Project

Program Status

- Concatenation : `ab`
- Union : `a+b`
- Kleene Start : `a*`
- Parentesis based mixed Operators : `(l*o)+(o*)`

Bugs

- No Bugs currently discovered.
- variety of test cases tried.
- If improper regex string is an input then the program might fail.
- Code works properly but will require more refactoring as there are a lot of unused methods and redundant methods.

Process Flow

- Parse regular expression string to readable regex string with missing operators such as concatenation
- Parse the preprocessed Regex to postfix notation using Shunting Yard Algorithm.
- Evaluate the postfix string with an NFA class that creates resultant NFA's based on Thompson's construction.
- Convert the final resultant NFA from postfix evaluation to a DFA Using Subset Construction
- Create a node graph from the resultant DFA
- Evaluate the text in the files by traversing the graph using recursion and print the matching values when a final state is reached in the graph.

Data Structures

- `class NFA` : Contains Methods to create an NFA and a DFA from any NFA.
- `struct transition` : stucture that represents the transiton between the nodes of the automaton. It contains the starting and ending edge and the symbol of transition.
- `vector<transtion>` : Vector containing the infomation needed to construct the node graph of the Automaton.
- `vector< vector<trans> > dfa_node_graph` : actual graph used for Traversing the Tree.
- `struct matched_symbol` : The structure that stores the matched tokens and postion of matched tokens from the text on which the search takes place.

Algorithms

- `string changeRegexOperators()`: Regex Preprocess : adds the missing concatenation symbol to convert to postfix.
- `string convertRegexToPostfix()`: Shuntting Yard Algorithm : To convert infix regex into postfix notation
- `NFA postFixNFABuilder()`: Post Fix Evaluation with Thompson Construction : A Method to build the final NFA from a postfix regular expression

- `void convert_to_dfa()::NFA`: Subset Constuction : Algorithm to convert the NFA to DFA. Uses `set<int> epsilon_closure ::NFA` and `set<int> move::NFA` to find the resulting DFA states.
- `vector<matched_symbol> traverse_dfa_graph()::NFA`: DFS with Recurrsion : DFA Node graph explored with recursion. No backtracking supported in the algorithm as DFA's are deterministic to once a path is chosen in a graph there is no point in back tracking. String matching takes place here. a `vector<matched_symbol>` holds the tokens that got matched.

Interactions and Citations

- When I first approached the problem I tried to solve it using normal string matching but the solution was not the most optimal solution. After consulting William Sengir I realised that ordinary string matching with KMP algorithm and finding the largest word in the string is a half baked solution.
- Looking for more solutions and desperately trying to avoid using NFA's and DFA's I spoke to more class mates such as Craig Ignatowski who suggested me solutions like grep in linux. But Even though the thoughtprocess was good the solution was not complete.
- Finally after reading a lot online and in Books *Beautiful Code* and *The Practice of Programming by Brian W. Kernighan* I realised that the most optimal solution will come by using NFA' and DFA's.

REFERENCES

1. C++ Best Practices
2. Ways to Create String Arrays
3. I/O with Files C++
4. Array Vs Vectors In C++
5. How Arrays Are Treated Within Functions
6. Passing a Vector to Function in C++
7. Using Iterators with Vectors
8. Using Cont Iterators When iterating Through Constant Arrays.
9. Shunting Yard Algorithm Psuedo Code
10. Using :: in C++
11. Stacks in C++
12. Queues in C++
13. Printing Vectors in C++
14. Reference in C++
15. Pointers Vs References in C++
16. KMP Search Algorithm
17. Pointers vs. References in C++
18. Building NFAs with Thompson Construction
19. Building Regex Machine with NFAs
20. Subset Construction For Converting DFA to NFA
21. Regex Engine In Python For Thompson Construction
22. Thompson's Construction With C++
23. The Practice of Programming by *Brian W. Kernighan*
24. Beautiful Code *(O'Reilly)*
25. Subset Construction From NFA

Author

- Valay Dave