Project Description

First, we get lines of right-linear-grammar, delete useless states and make map of left side variables to list of right-side expression from that variable.

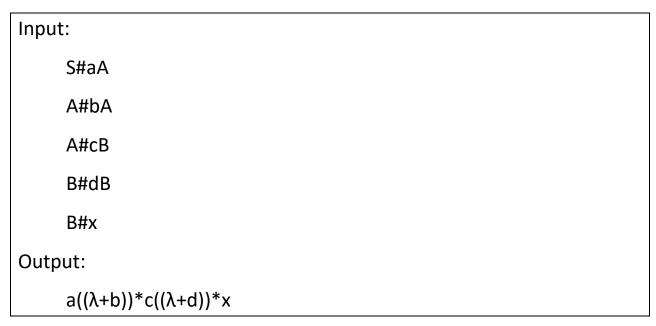
Then, we build a simple NFA class with 1 attribute: List of Transitions with the format of "Strt_State1->terminals->Dst_State"

After that, we convert NFA to Adjacency Matrix with default value of **NULL** and with matrix main diagonal of **landa**. Then we add transitions to matrix.

Then we apply the Recursive Transitive Closure Method (explained in link below) to our Adjacency Matrix to extract Regex from it.

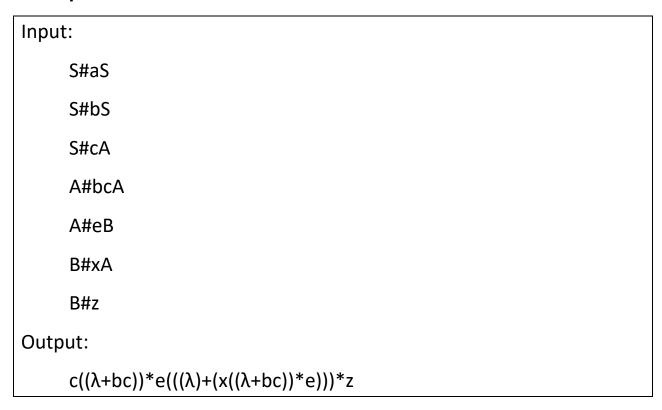
https://cs.stackexchange.com/questions/2016/how-to-convert-finite-automata-to-regular-expressions

Example1:



Project Description

Example2:



Example3:

```
Input:
    S#aA
    S#bS
    A#bT (will be deleted)
    A#ddB
    B#xA
    B#y
    A#z
Output:
    ((az)+(add(((λ)+(xdd)))*((y)+(xz))))
```