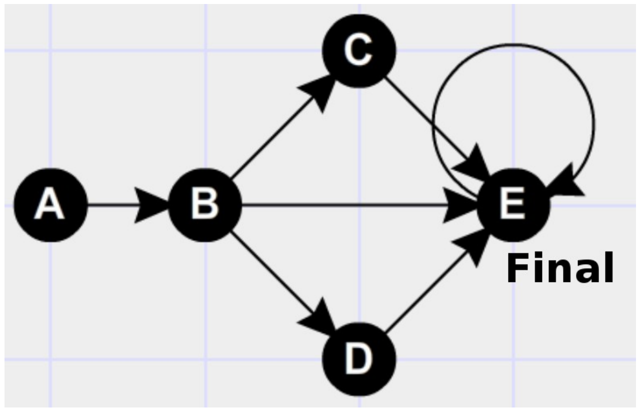
Regular Expressions

George Miller

Big Picture

AB(C|D)?E+



A AB BAC CWWHHA
BAACDEEEABCDD BBA
PDJKWONRLABKJDFKPWJ
ABCEEEEE
....

Searcher
(DFS)

AB
ABC
AB
ABCEEEEE
...

Syntax

Good Characters

| [ ] { } ( ) + * ? - , \ .   alphanumerics

Matching Parenthesis

([{]})   (([)]   {[(])()}

Legal ranges

{1} {2,200} [d-e]   [(as)-] {,2}

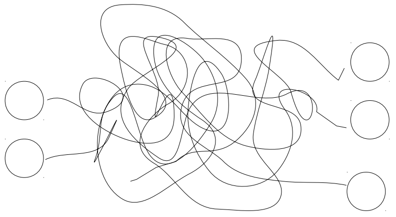
Legal Positions

x|y|z   * + ? {}   *abc   {1,2}?xy

Graph

○ Node matches list of character codes or singular character code (ord() and chr() help to convert)

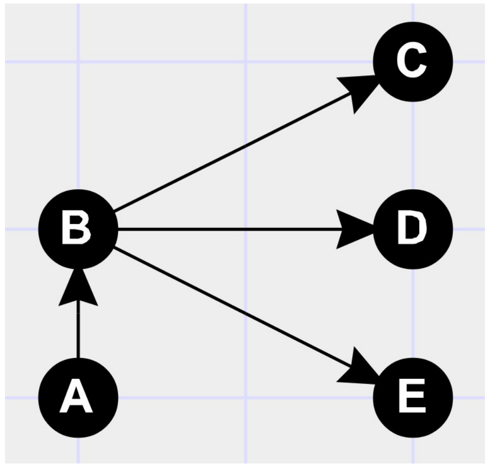
Input Nodes



Output Nodes

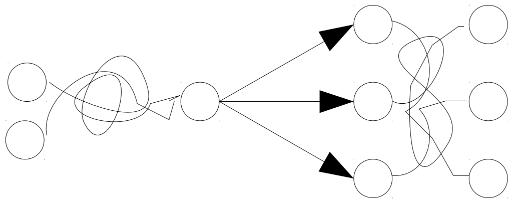
Example Graph:
A?B(C|D|E)

Input Nodes



Output Nodes

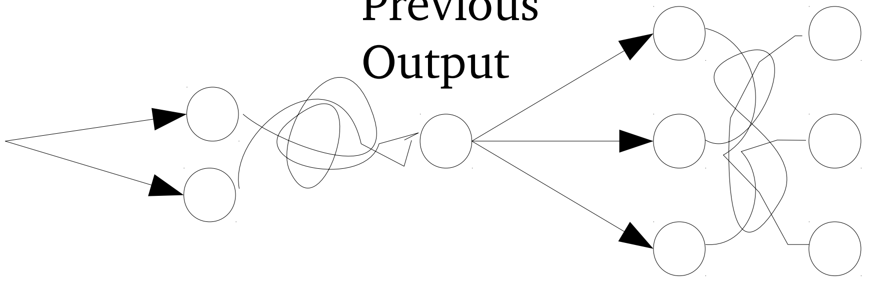
Input Nodes



Output Nodes

Useful Structure: …

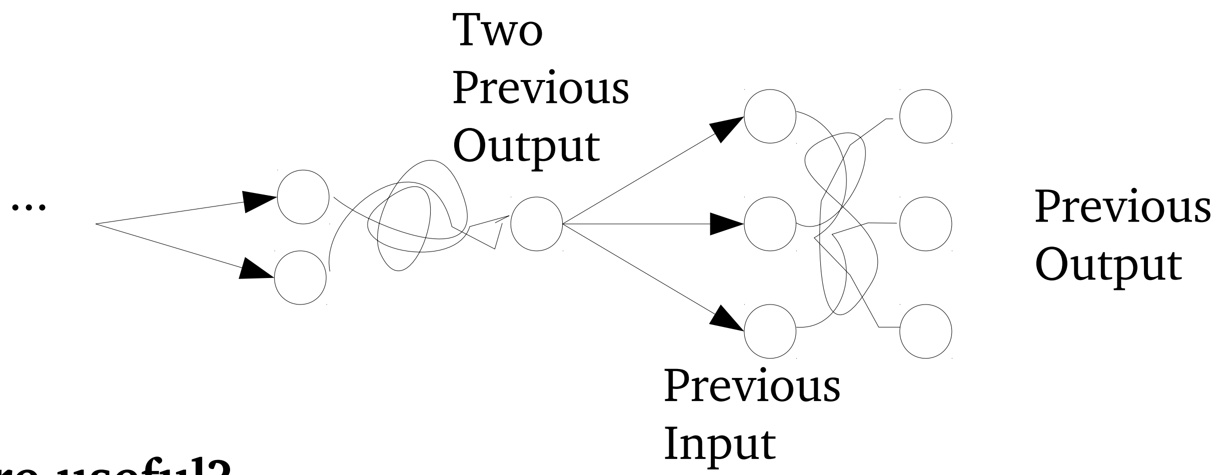
Two Previous Output



Previous Output

Previous Input

Graph

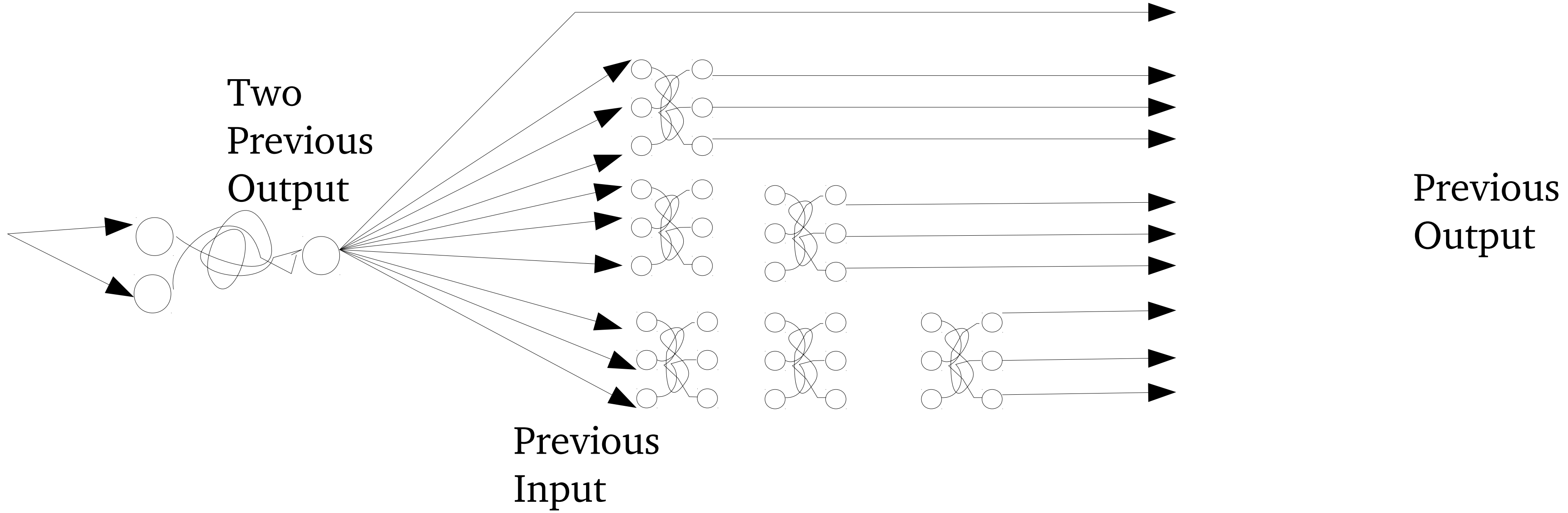


Why is this structure useful?

+ Connect previous output to previous input to allow multiple traverses of the previous Graph

* Do above, but also add nodes from two previous output to previous output to allow skipping

A?B(C|D|E){0, 3} Shown Below



Searcher

Traverses Graph in DFS manner

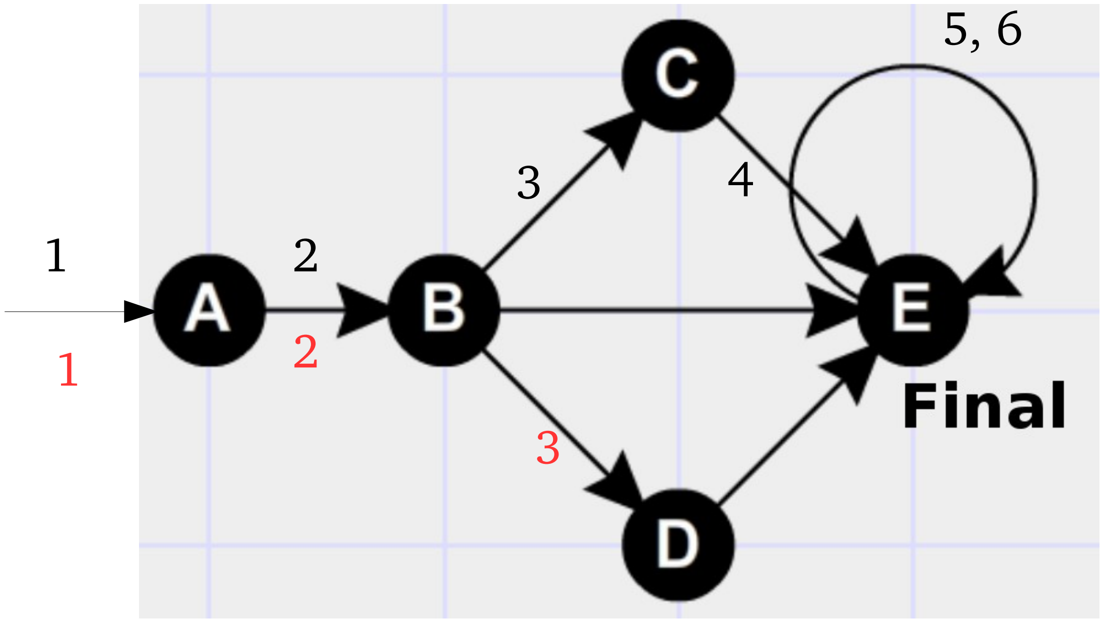
as deep as possible to get the most specific match

If connection is made from input node to output node, we have a match

Initiates search at every index in given text, skips indexes inside last found instance

1	2	3	4	5	6
A	B	C	E	E	E

1	2	3	4	5	6
A	B	D	C	E	E



Demo

https://github.com/george-miller/regex