Appendix: Computing all paths in Preprint

9.1 Semiring

A *path* is a walk in a graph with all its vertices different.

A and B set of paths on network.

$A + B = A \cup B$

$A \cdot B = \{a \bullet b : a \in A, b \in B\}$

$$a \bullet b = \begin{cases} a \circ b & last(a) = first(b) \wedge set(a) \cap set(bf(b)) = \emptyset \\ \text{nothing} & \text{otherwise} \end{cases}$$

$\circ$ is the operation of *concatenation* of paths.

$\emptyset \cdot A = A \cdot \emptyset = \emptyset$

Kleene, Warshall, Floyd and Roy are contributed to the development of the procedure which final form was given by Fletcher.

$\mathbf{C}_0 := \mathbf{W}$;
for $k := 1$ **to** n **do begin**
  for $i := 1$ **to** n **do for** $j := 1$ **to** n **do**
    $c_k[i,j] := c_{k-1}[i,j] + c_{k-1}[i,k] \cdot (c_{k-1}[k,k])^\star \cdot c_{k-1}[k,j]$;
  $c_k[k,k] := 1 + c_k[k,k]$;
end;
$\mathbf{W}^\star := \mathbf{C}_n$;

If we delete the statement $c_k[k,k] := 1 + c_k[k,k]$ we obtain the algorithm for computing the strict closure $\overline{\mathbf{W}}$.

We have an idempotent ($A + A = A$) semiring. The unit for $+$ is the empty set $\emptyset$. The unit for $\cdot$ is $1 = \{[v] : v \in V\}$.

Let

$$A = c_{k-1}[k,k] = \{a \in Path : first(a) = last(a) = k\} = \{[k]\}$$

Therefore

$$A^* = 1 + A + A^2 + A^3 + A^4 + \cdots = 1 + \{[k]\} + \{[k]\} + \{[k]\} + \{[k]\} \cdots = 1$$

Since the semiring is idempotent the Fletcher's algorithm can be performed in place – we can omit indices in c_k.

9.2 Python

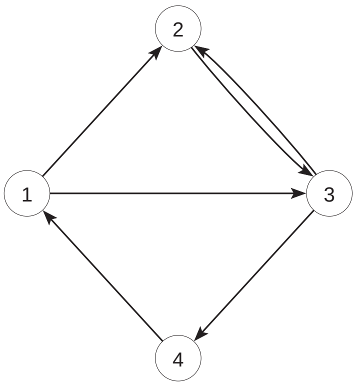
```
def times(A,B):
  C = []
  if (A == [])|(B == []): return(C)
  for a in A:
    for b in B:
      la = a[len(a)-1]; fb = b[0]
      if la == fb:
        if set(a) & set(b[1:]) == set(): C.append(a+b[1:])
  return(C)

def closure(R):
```

```
  n = len(R); C = R
  for k in range(n):
    for u in range(n):
      for v in range(n):
        C[u][v] = C[u][v] + times(C[u][k],C[k][v])
  return(C)

def output(R):
  n = len(R)
  for u in range(n):
    for v in range(n):
      print(u+1,v+1,R[u][v])

...
   r = [ [[] for j in range(stcNver)] for i in range(stcNver)]
   while True:
     line = stc.readline()
     if not line: break
     row = list(filter(lambda s: s not in [’’], line.split(’ ’)))
     u = eval(row[0]); v = eval(row[1])
     r[u-1][v-1] = [[u,v]]
...
```




```
R = [ [ []     , [[1,2]], [[1,3]], []      ],
      [ []     , []     , [[2,3]], []      ],
      [ []     , [[3,2]], []     , [[3,4]] ],
      [ [[4,1]], []     , []     , []      ] ]
I = [[1], [2], [3], [4]]

C = closure(R)
output(C)

1 1 []
1 2 [[1, 2], [1, 3, 2]]
1 3 [[1, 3], [1, 2, 3]]
1 4 [[1, 3, 4], [1, 2, 3, 4]]
2 1 [[2, 3, 4, 1]]
2 2 []
2 3 [[2, 3]]
2 4 [[2, 3, 4]]
3 1 [[3, 4, 1]]
3 2 [[3, 2], [3, 4, 1, 2]]
3 3 []
3 4 [[3, 4]]
4 1 [[4, 1]]
4 2 [[4, 1, 2], [4, 1, 3, 2]]
4 3 [[4, 1, 3], [4, 1, 2, 3]]
4 4 []
```