

Inputs：

the number of nodes 1\*N array

the number of edges 1\*N array

the matrix of the network N\*N matrix

the cost of each edge 1\*N array

the source of a path

the destination of a path

h: the maximum number of paths which can share the same link

Outputs:

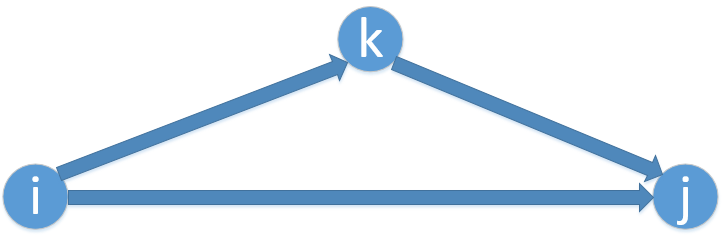
A graph showing the two shortest paths in different color

Primary path(N,N) : the shortest path of nodes N\*N matrix

Secondary path(N,N):the backup path which is edge-disjoint from primary path N\*N matrix

Cost sum: the cost of each path 1\*N array

Objective Function:

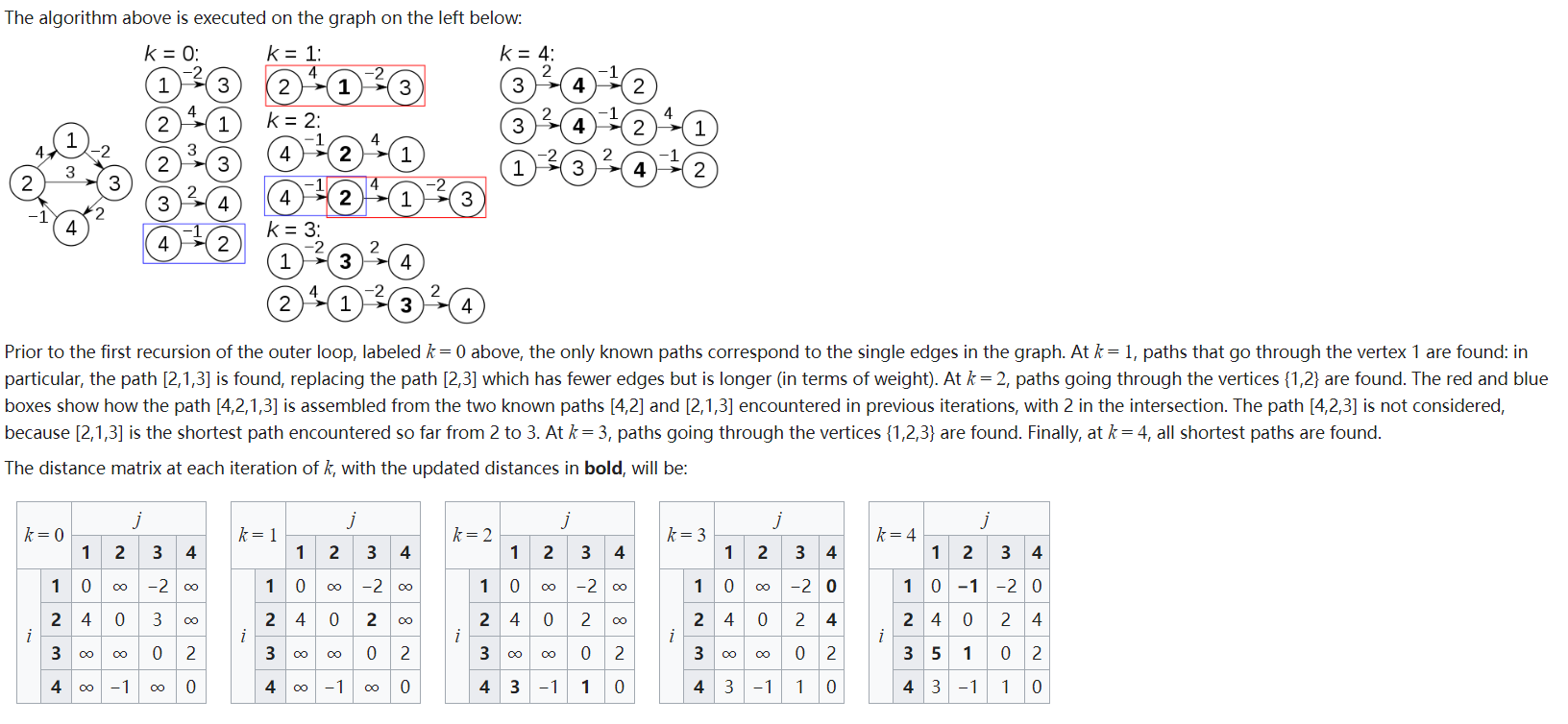


= cost（i，j）                                                        when k=0；

= min(, +)             when k>0；

Constrain:

at most h primary paths can share the same link, every time we use this link as primary path, h is decreased by 1 until it goes to 0, this link can not be used again, we delete this edge from G(V,E).

Algorithm:

let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)

for each edge (u, v) do

dist[u][v] ← w(u, v) // The weight of the edge (u, v)

for each vertex v do

dist[v][v] ← 0

for k from 1 to |V|

for i from 1 to |V|

for j from 1 to |V|

if dist[i][j] > dist[i][k] + dist[k][j]

dist[i][j] ← dist[i][k] + dist[k][j]

end if