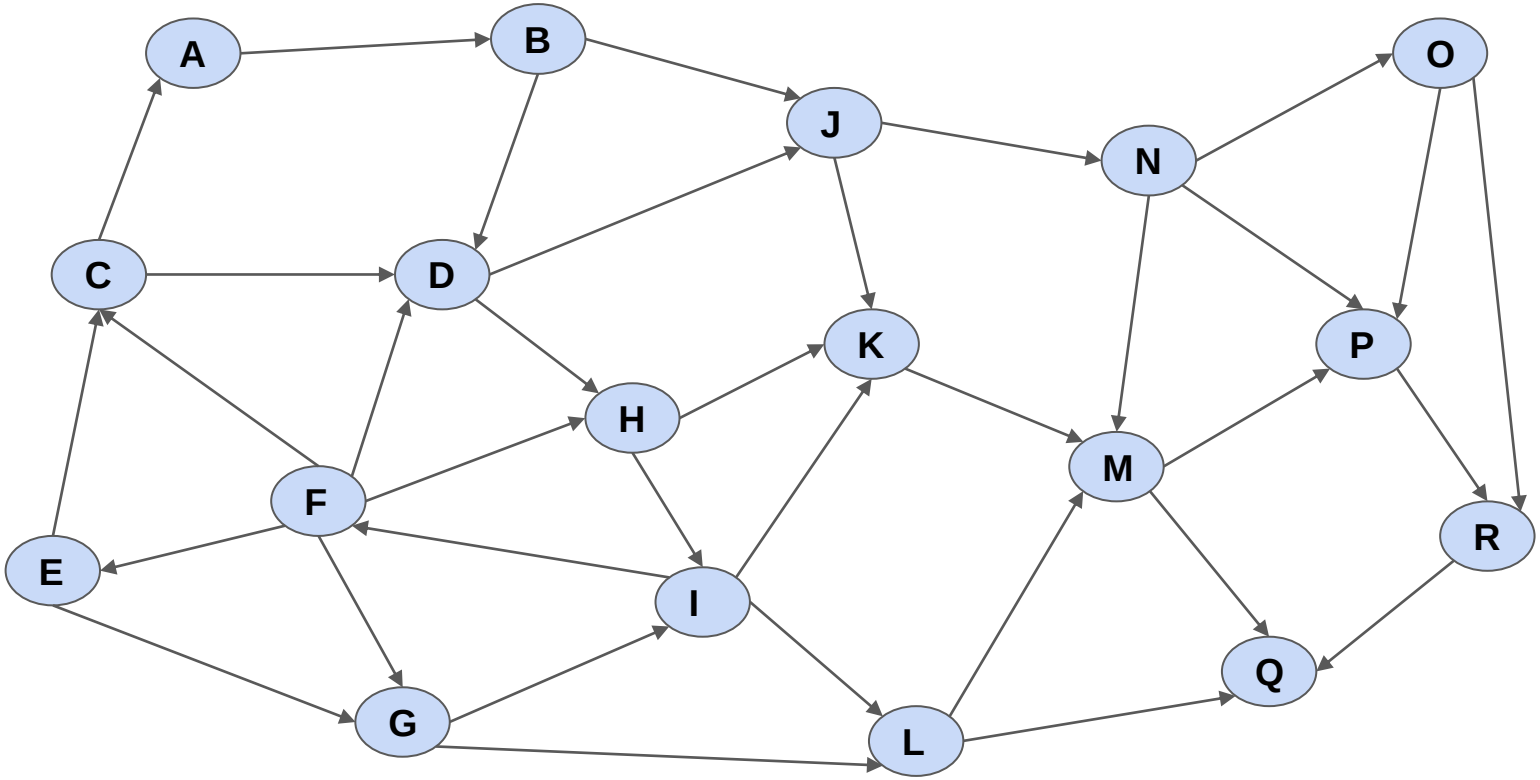
Each node has x,y coordinates

Edges have edge length



Sample Road Network

Dijsktra's Algorithm

1. Initialize a traditional priority queue (*PQ*)
2. Loop until (*PQ* is not empty or destination not closed)
   3. Extract-Min //descendant node closest to source
   4. Add extracted node to closed list
   5. Update current best path to other node, if a better path is found
6. End Loop

Claim: At this stage we have an optimal path to the closed node !

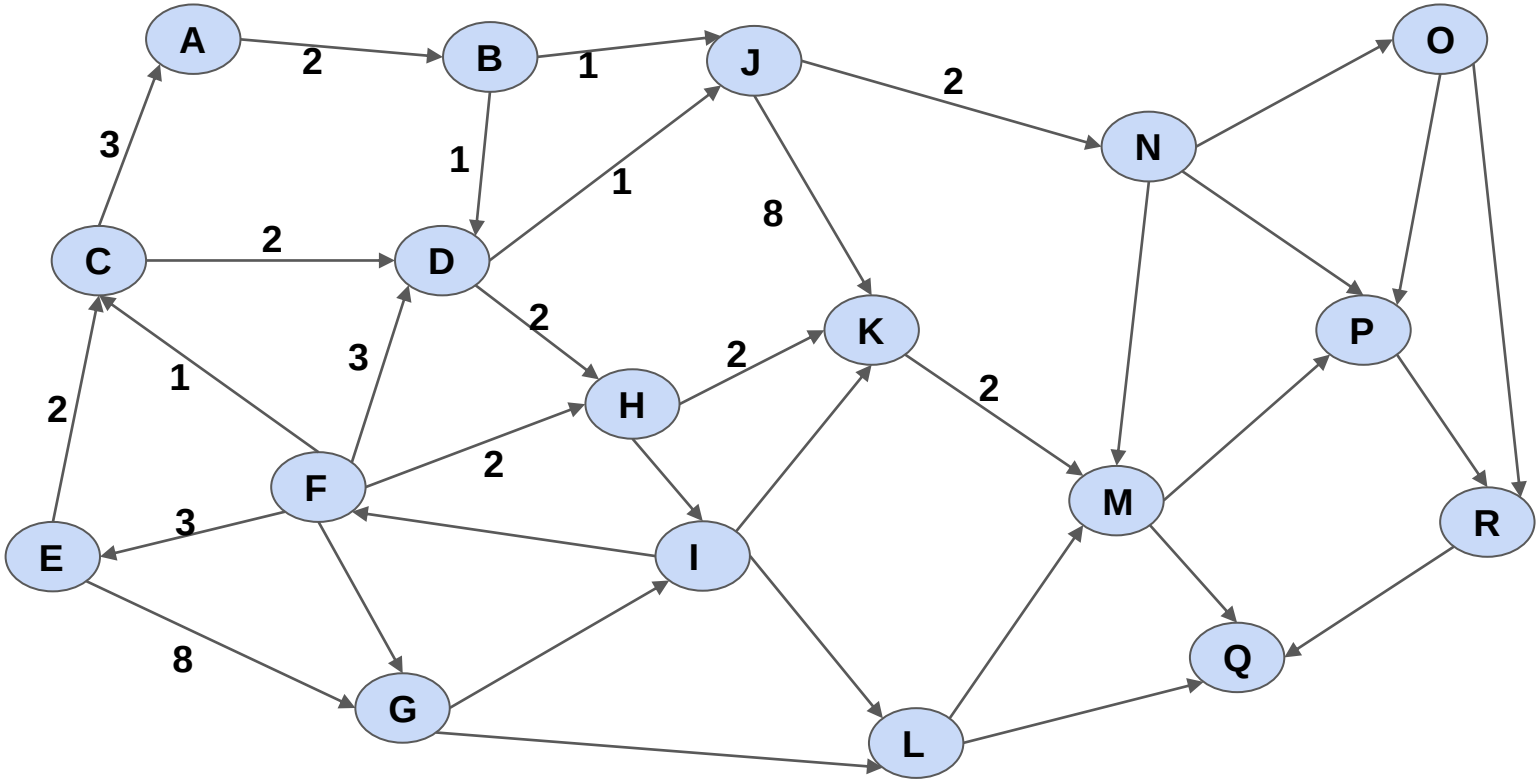
This is the central piece of correctness for Dijsktra's algorithm.

Open List:

A(0)

Closed List:

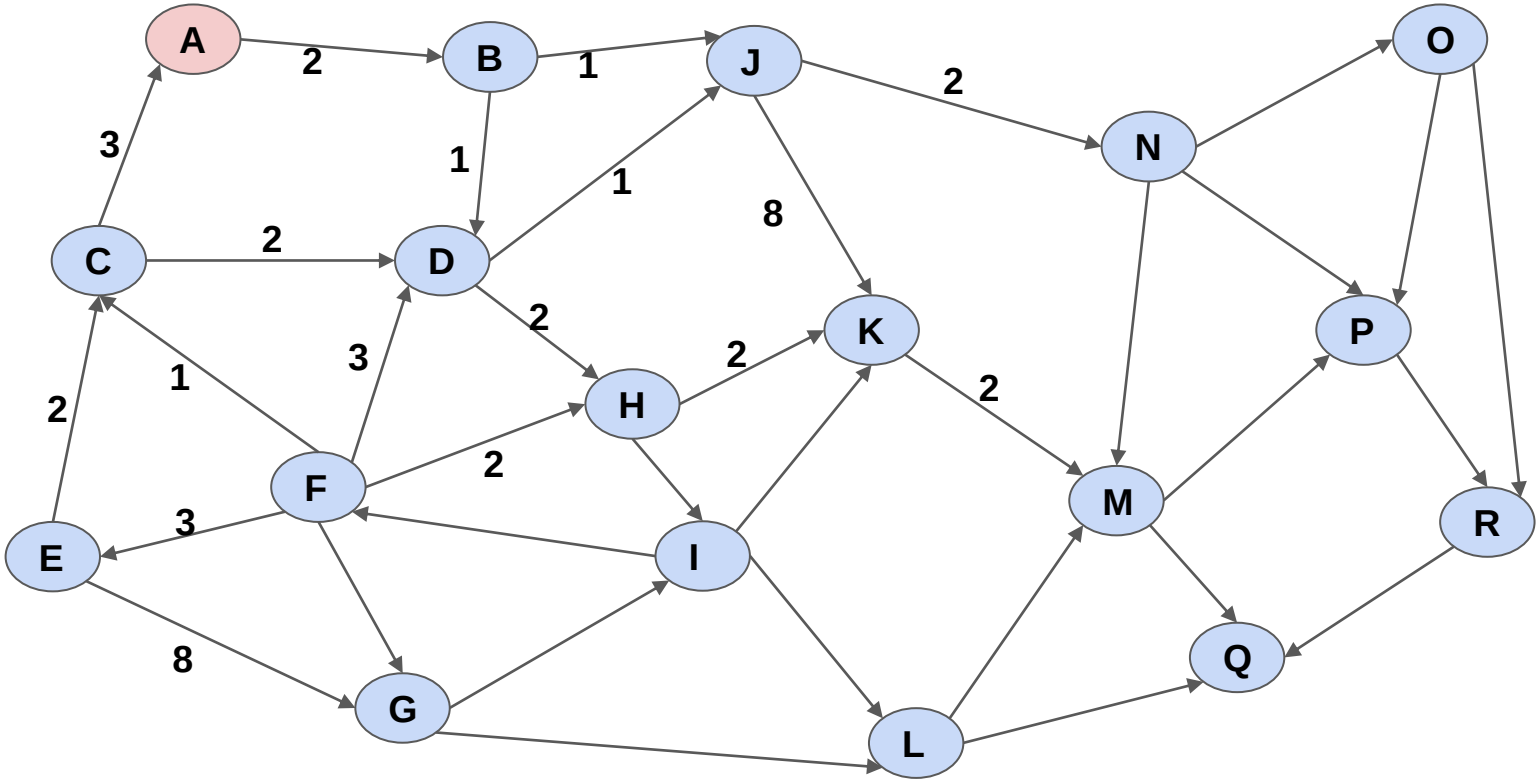
Source = A and Destination = K



Source = A and Destination = K

Open List:

B(2 ,A)



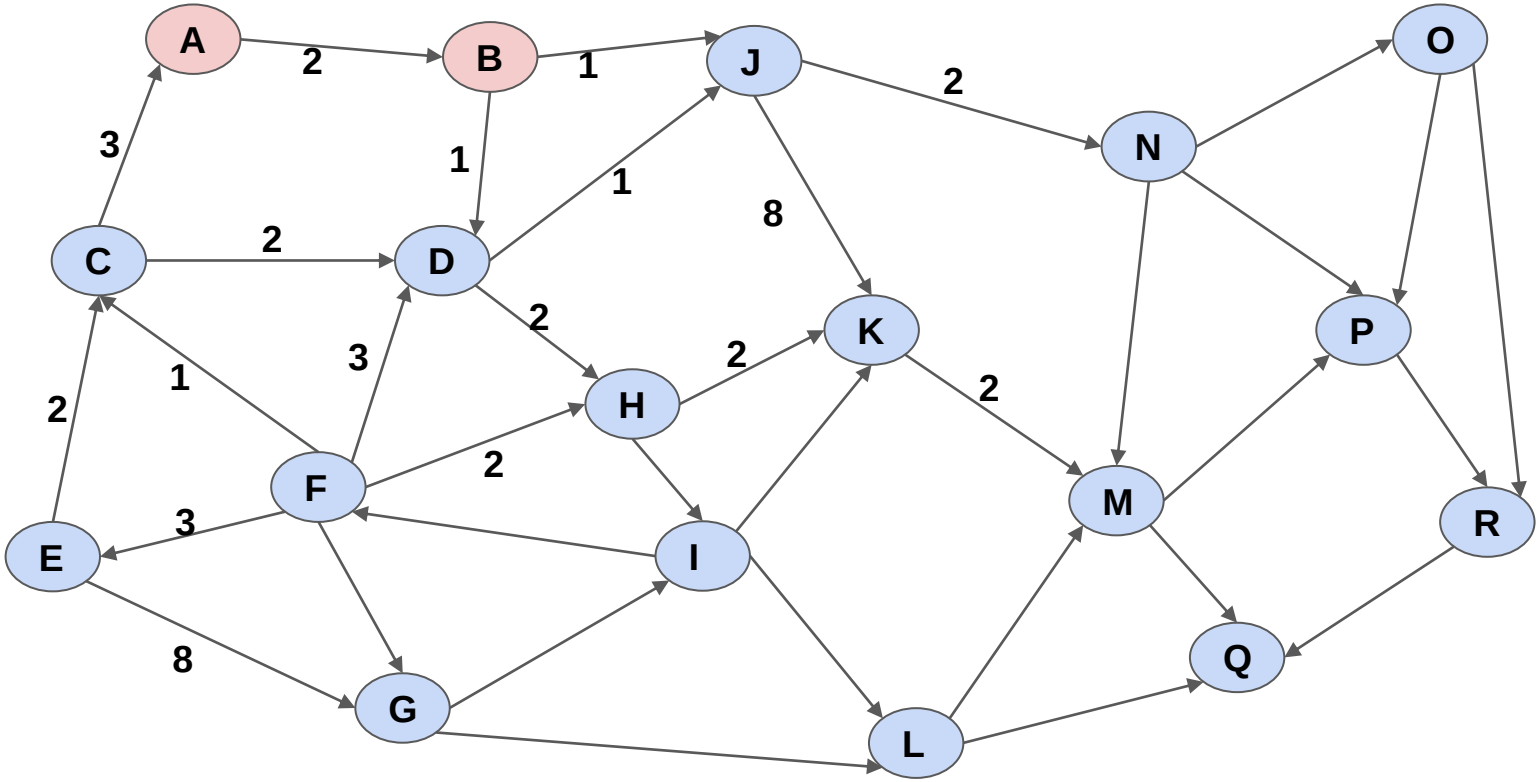
Closed List:

A(0)

Source = A and Destination = K

Open List:

J(3 ,B)
D(3,B)



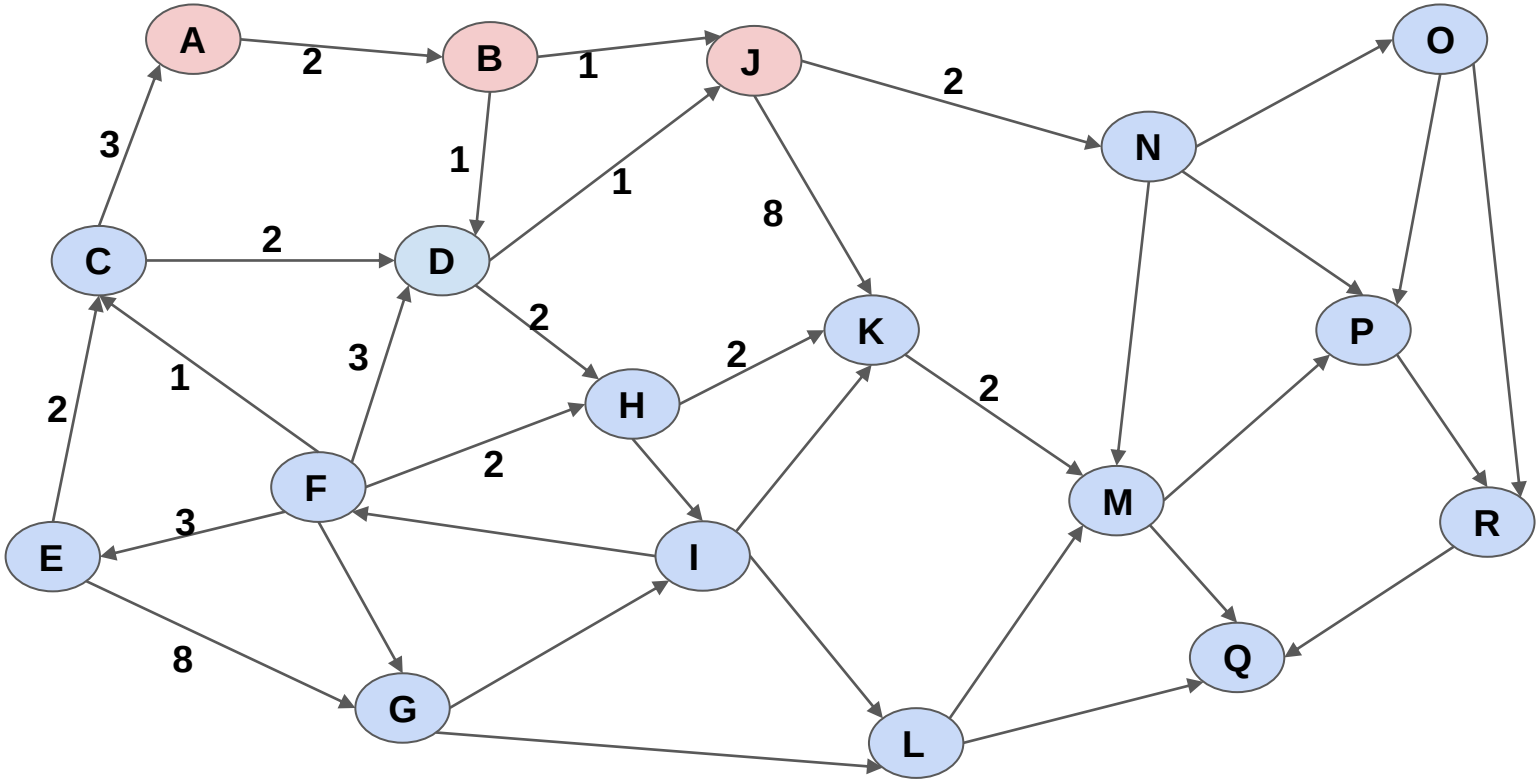
Closed List:

A(0), B(2,A)

Source = A and Destination = K

Open List:

K(11 ,J)
N(5,J)
D(3,B)



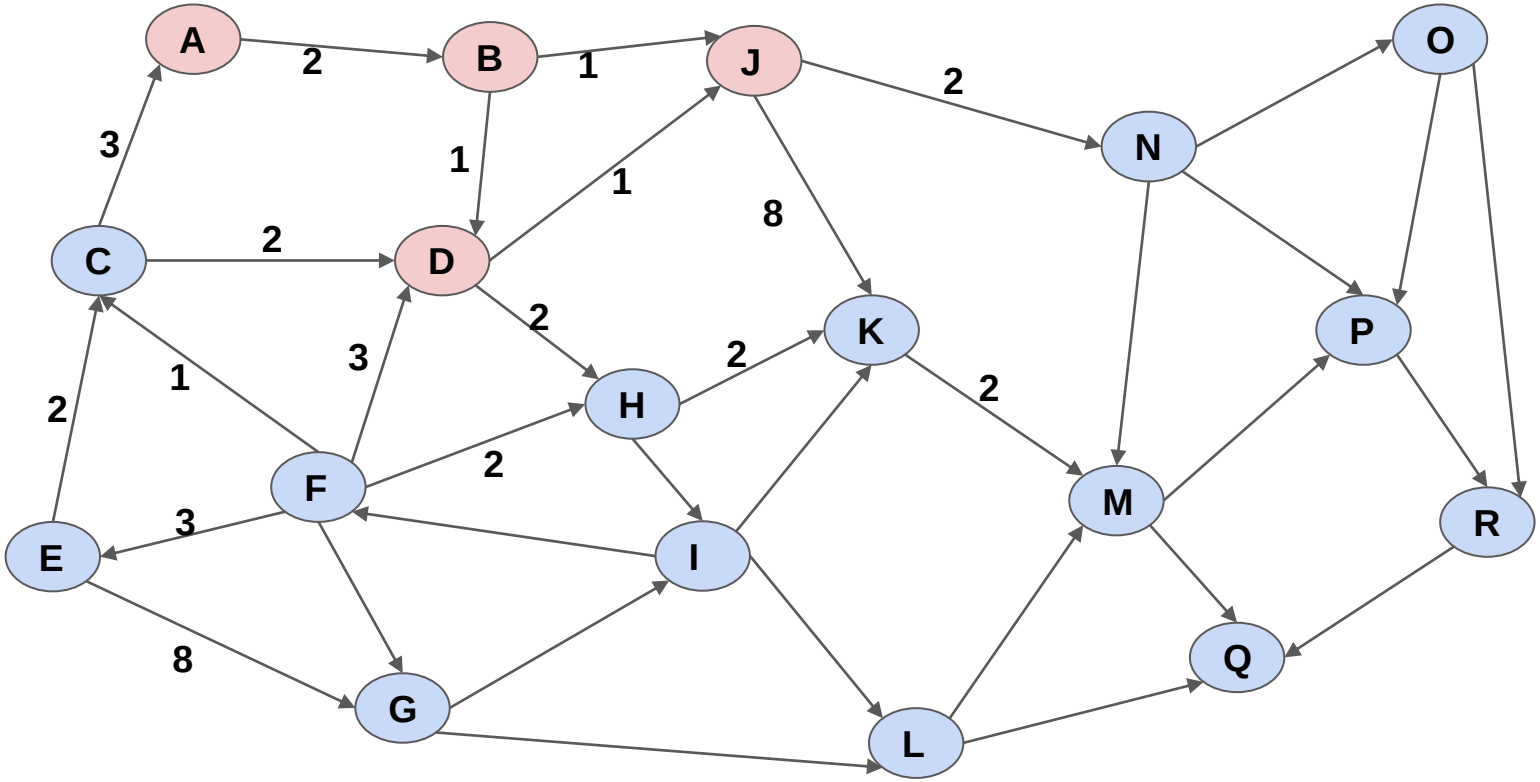
Closed List:

A(0), B(2,A)
J(3,B)

Source = A and Destination = K

Open List:

K(11 ,J)
N(5,J)
H(5,D)



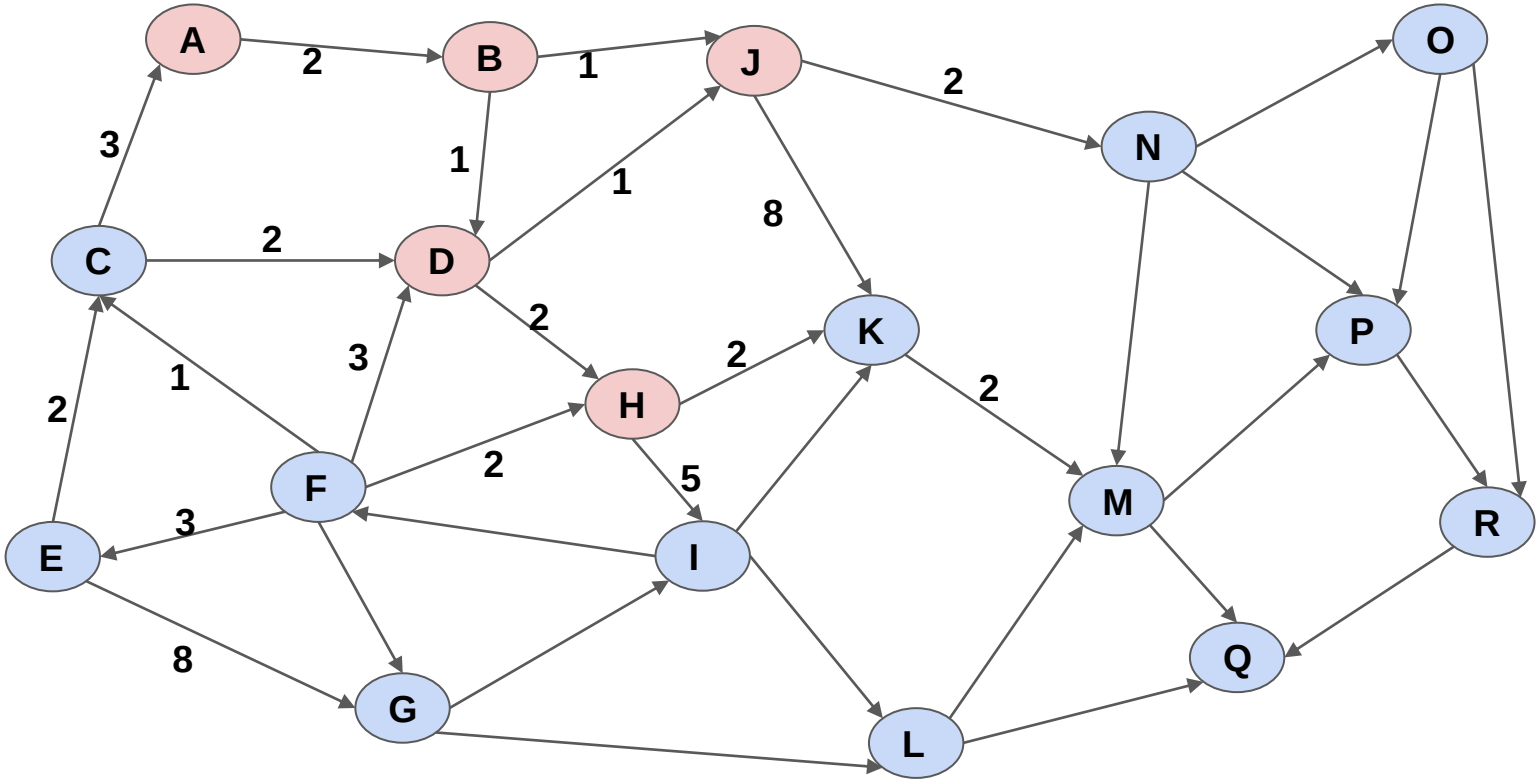
Closed List:

A(0), B(2,A)
J(3,B) D(3,B)

Source = A and Destination = K

Open List:

K(7 ,H)
N(5,J)
I(10,H)



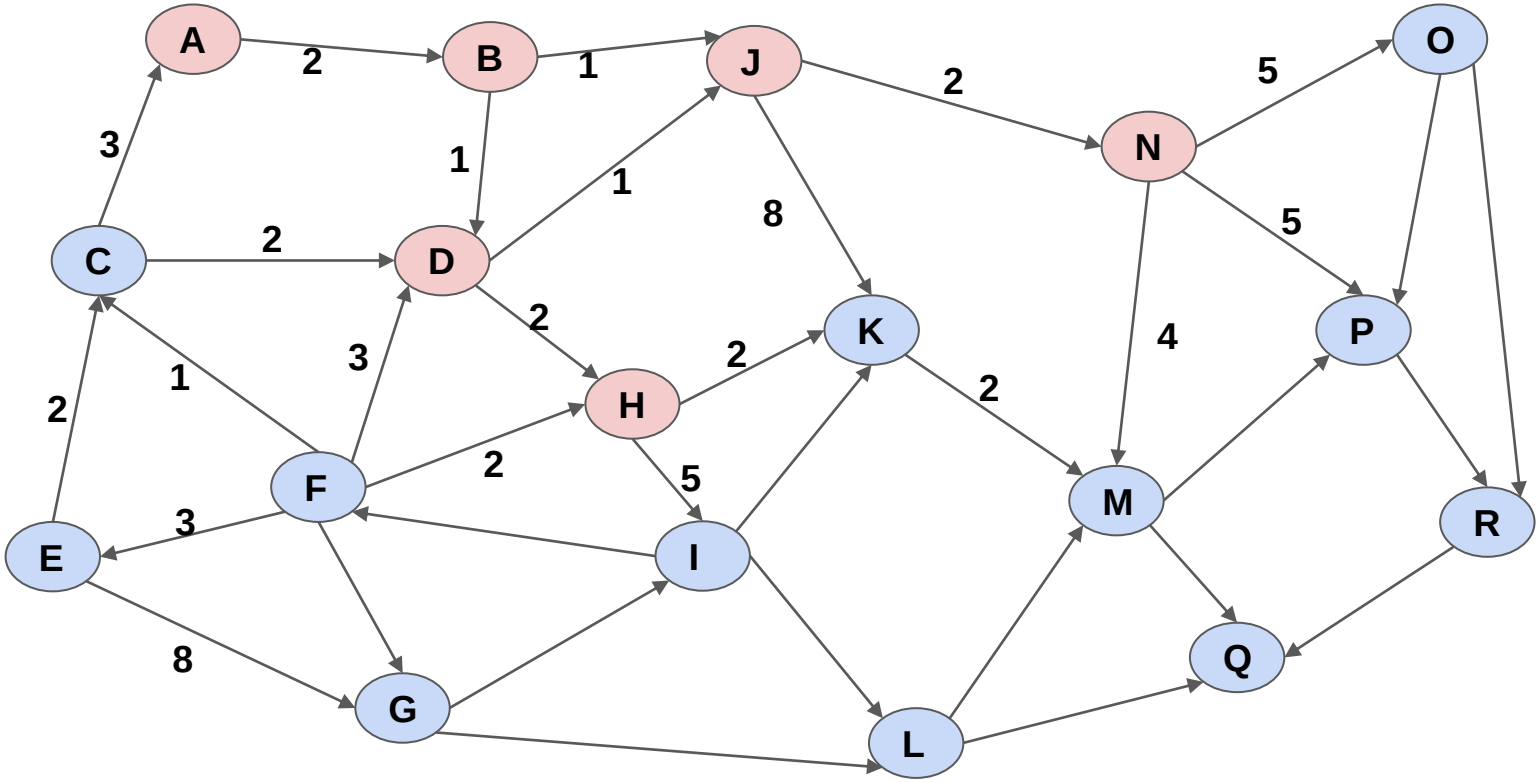
Closed List:

A(0), B(2,A)
J(3,B) D(3,B)
H(5,D)

Source = A and Destination = K

Open List:

K(7 ,H)
O(10,N)
P(10,N)
M(9,N)
I(10,H)



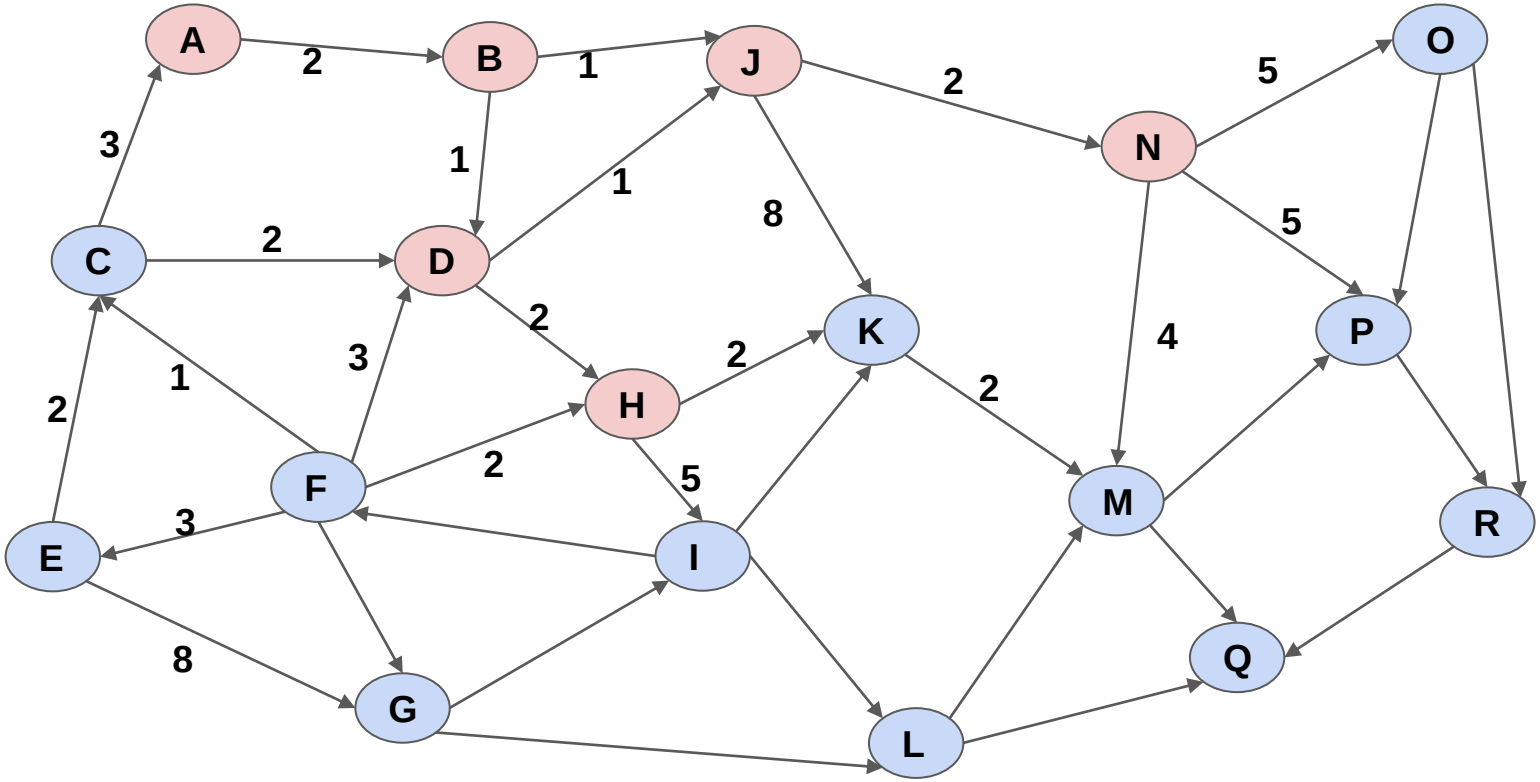
Closed List:

A(0), B(2,A)
J(3,B) D(3,B)
H(5,D), N(5,J)

Source = A and Destination = K

Open List:

O(10,N)
P(10,N)
M(9,N)
I(10,H)



Closed List:

A(0), B(2,A)
J(3,B) D(3,B)
H(5,D), N(5,J)
K(7,H)

Algorithm Terminates as destination node is closed

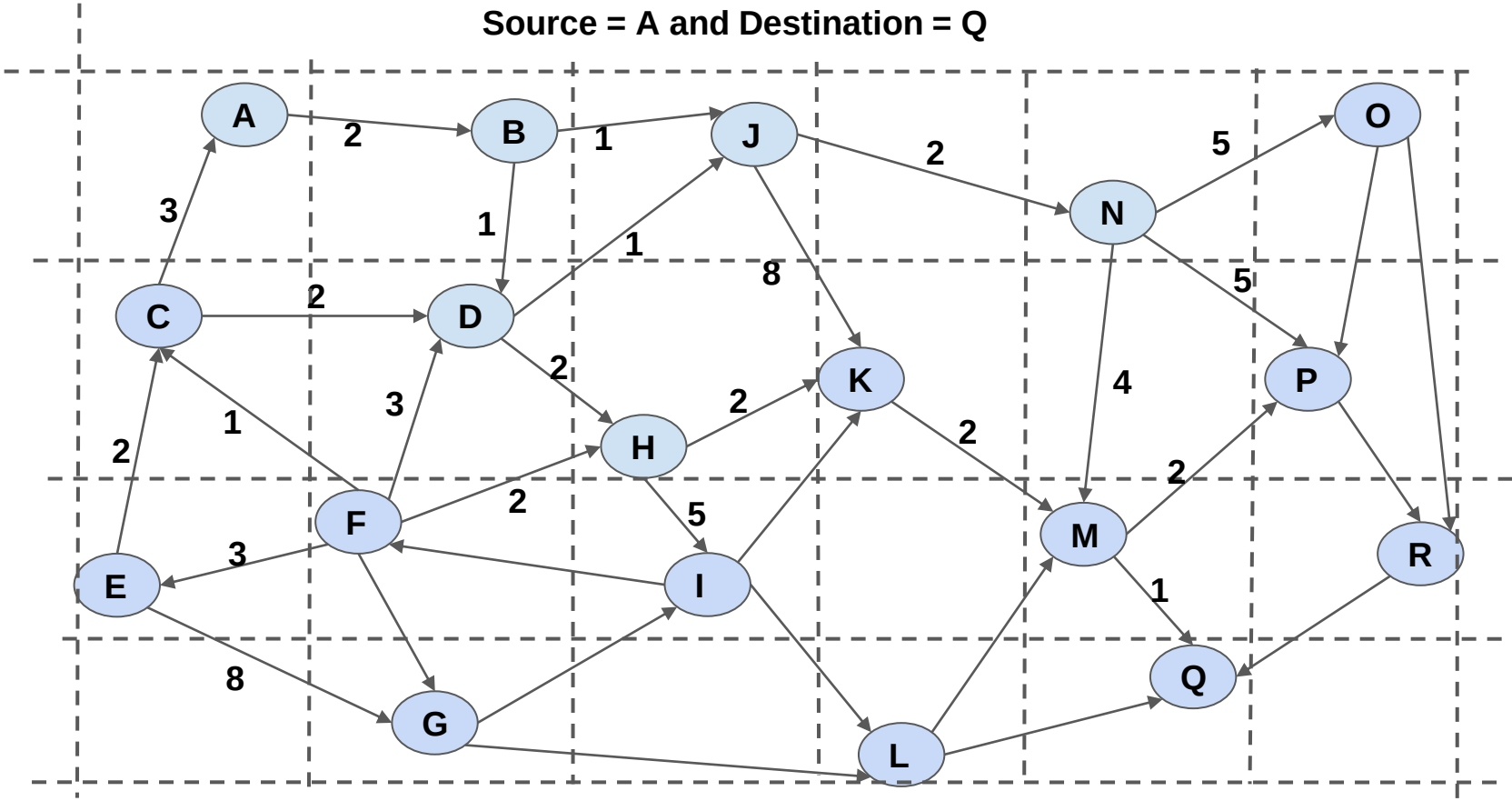
Dijkstra's Algorithm on Partitioned Graph

Open List:

A(0)

Closed List:

Source = A and Destination = Q



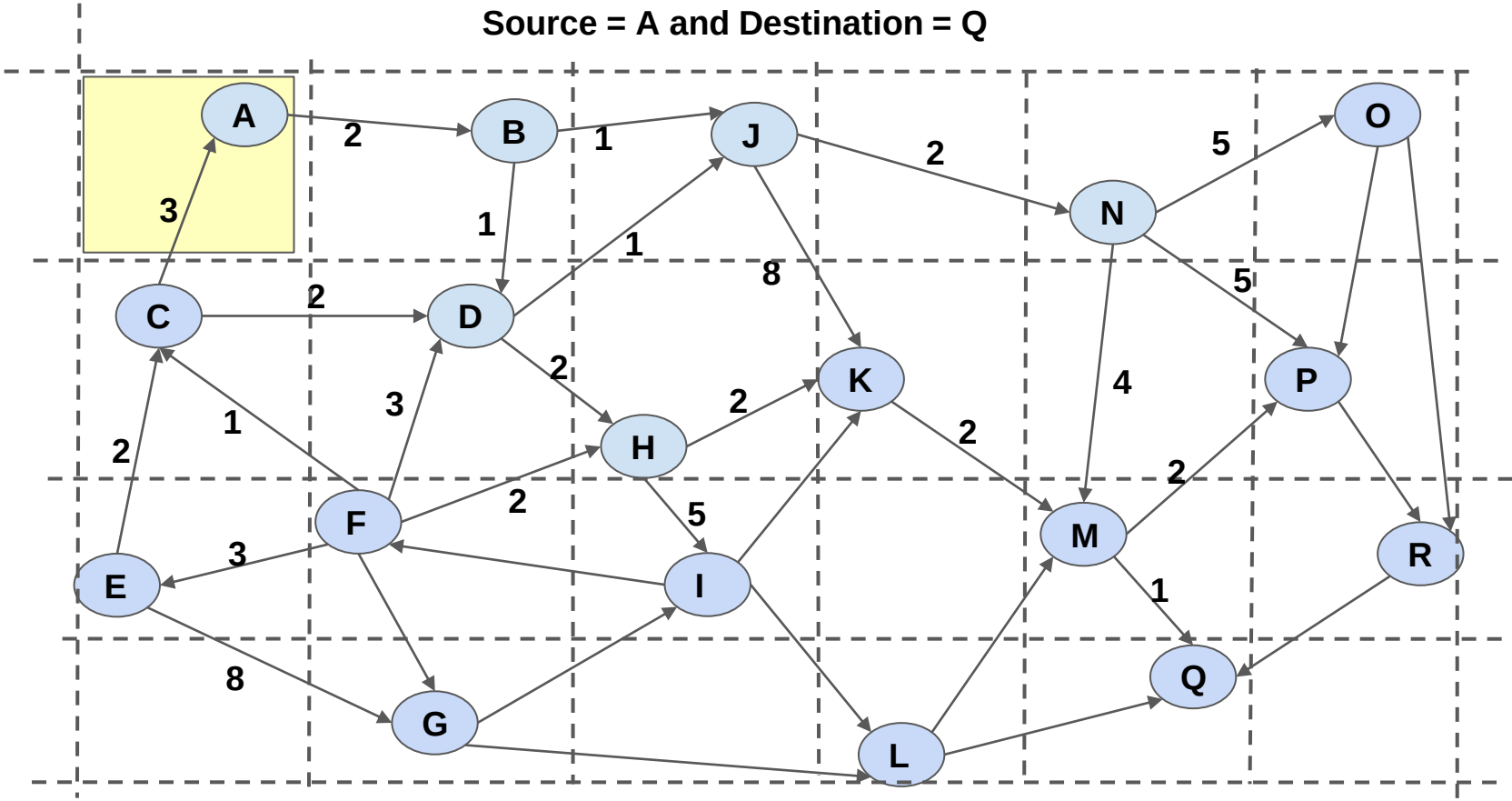
Shaded pages are already in Main memory

Open List:

A(0)

Closed List:

Source = A and Destination = Q



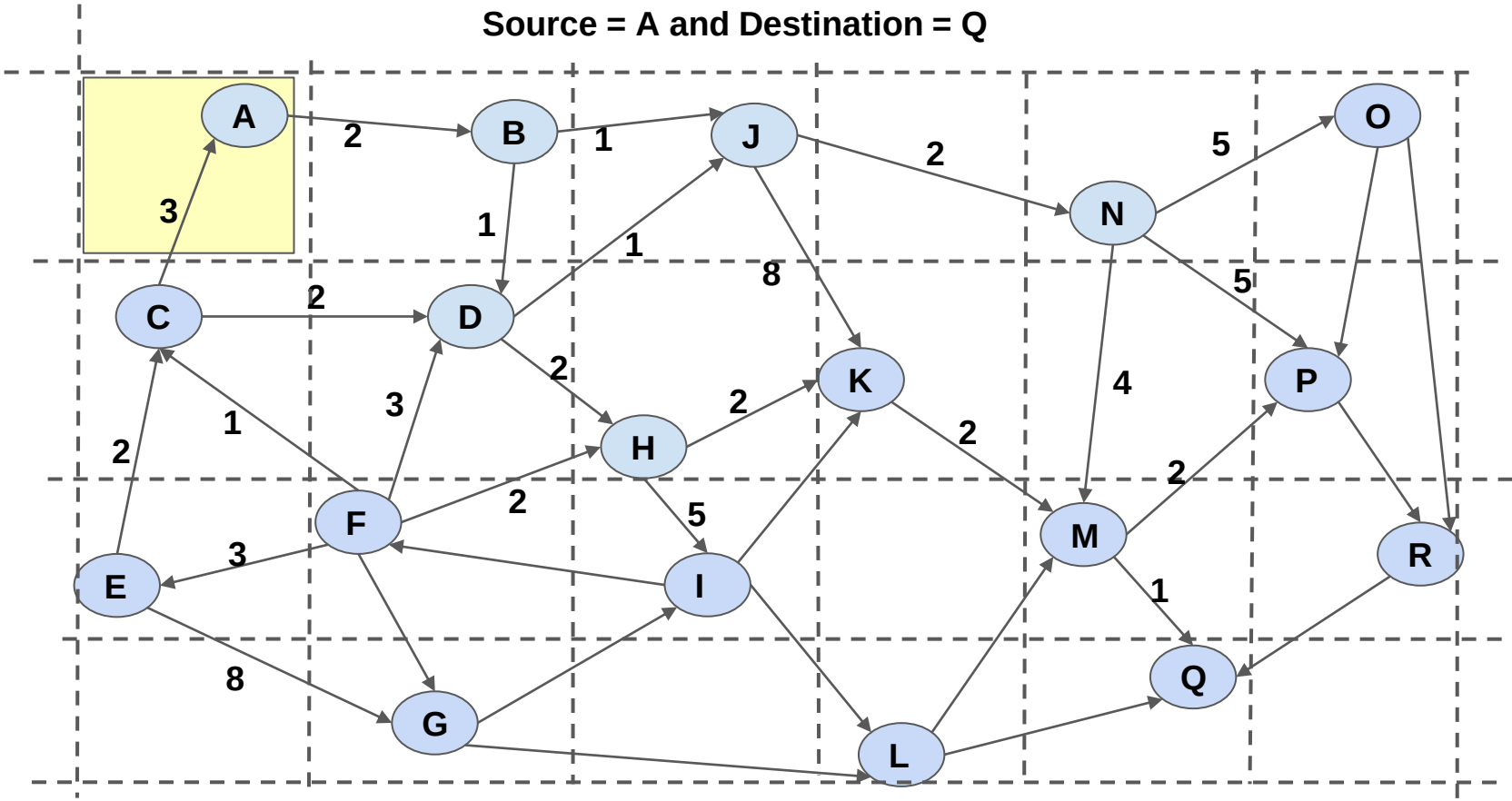
Shaded pages are already in Main memory

Open List:

B(2,A)

Closed List:

A(0)

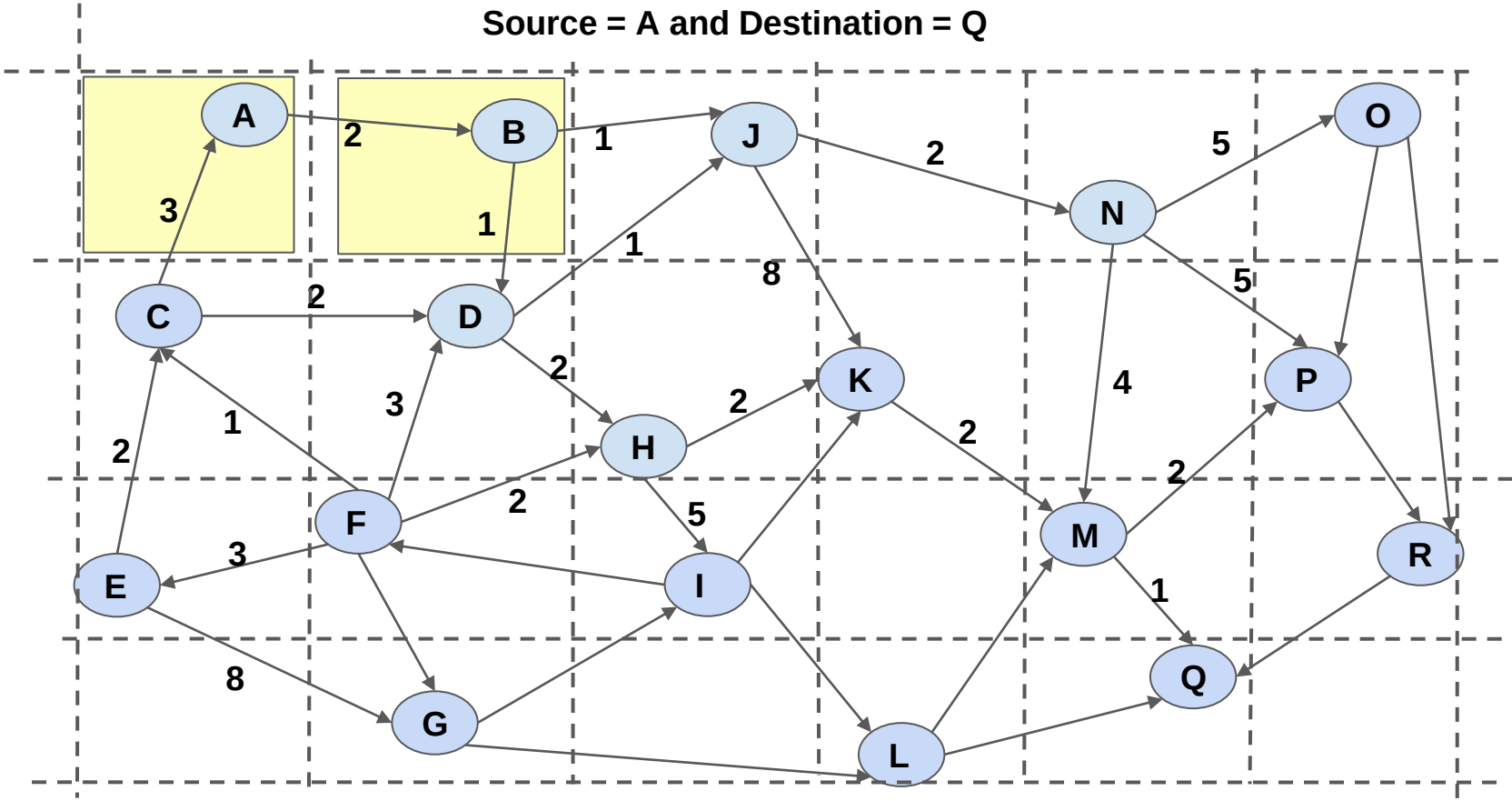


Shaded pages are already in Main memory

Source = A and Destination = Q

Open List:

J(3,B), D(3,B)



Closed List:

A(0), B(2,A)

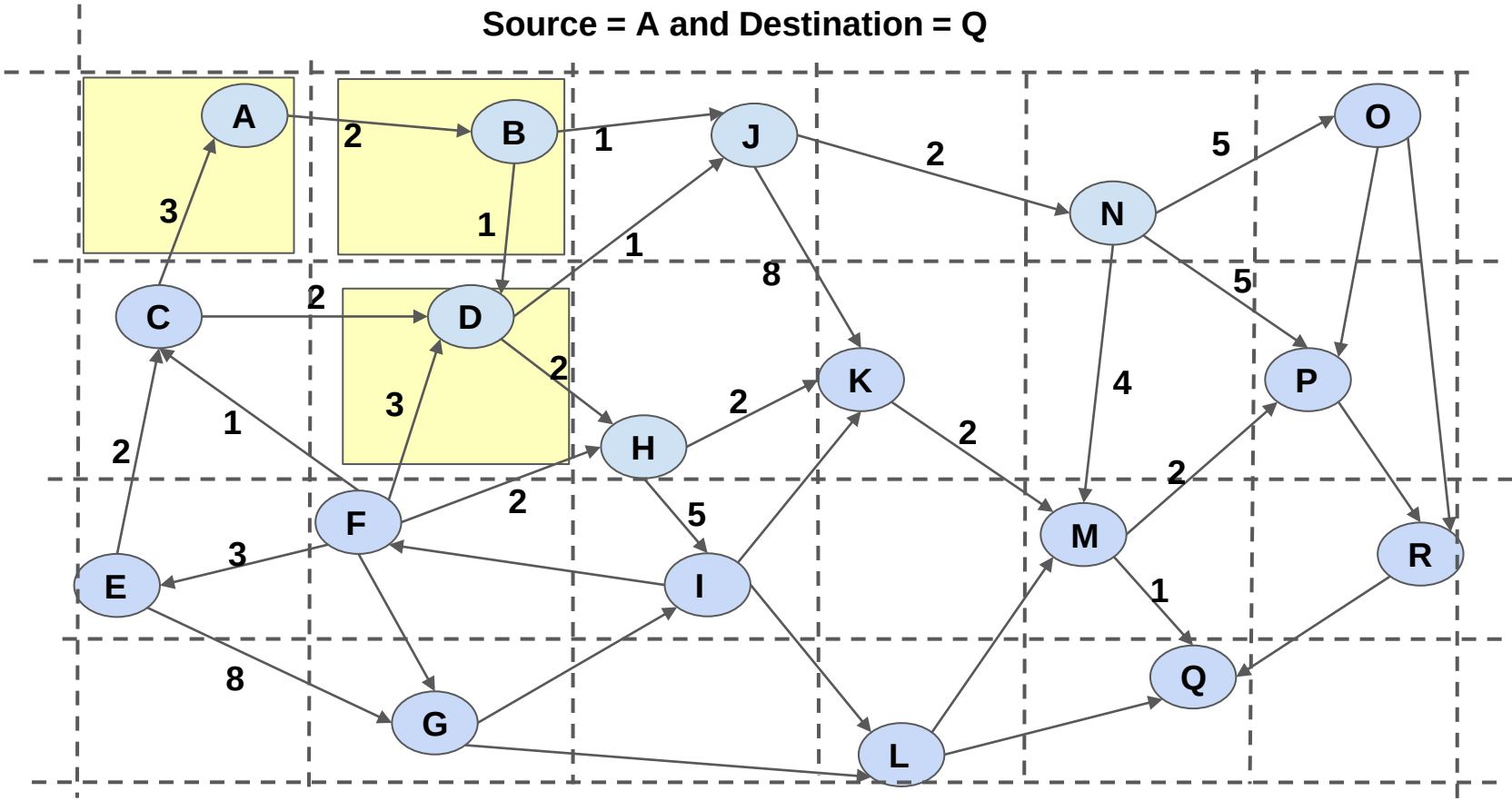
Shaded pages are already in Main memory

Open List:

J(3,B),
H(5,D)

Closed List:

A(0),
B(2,A),
D(3,B)



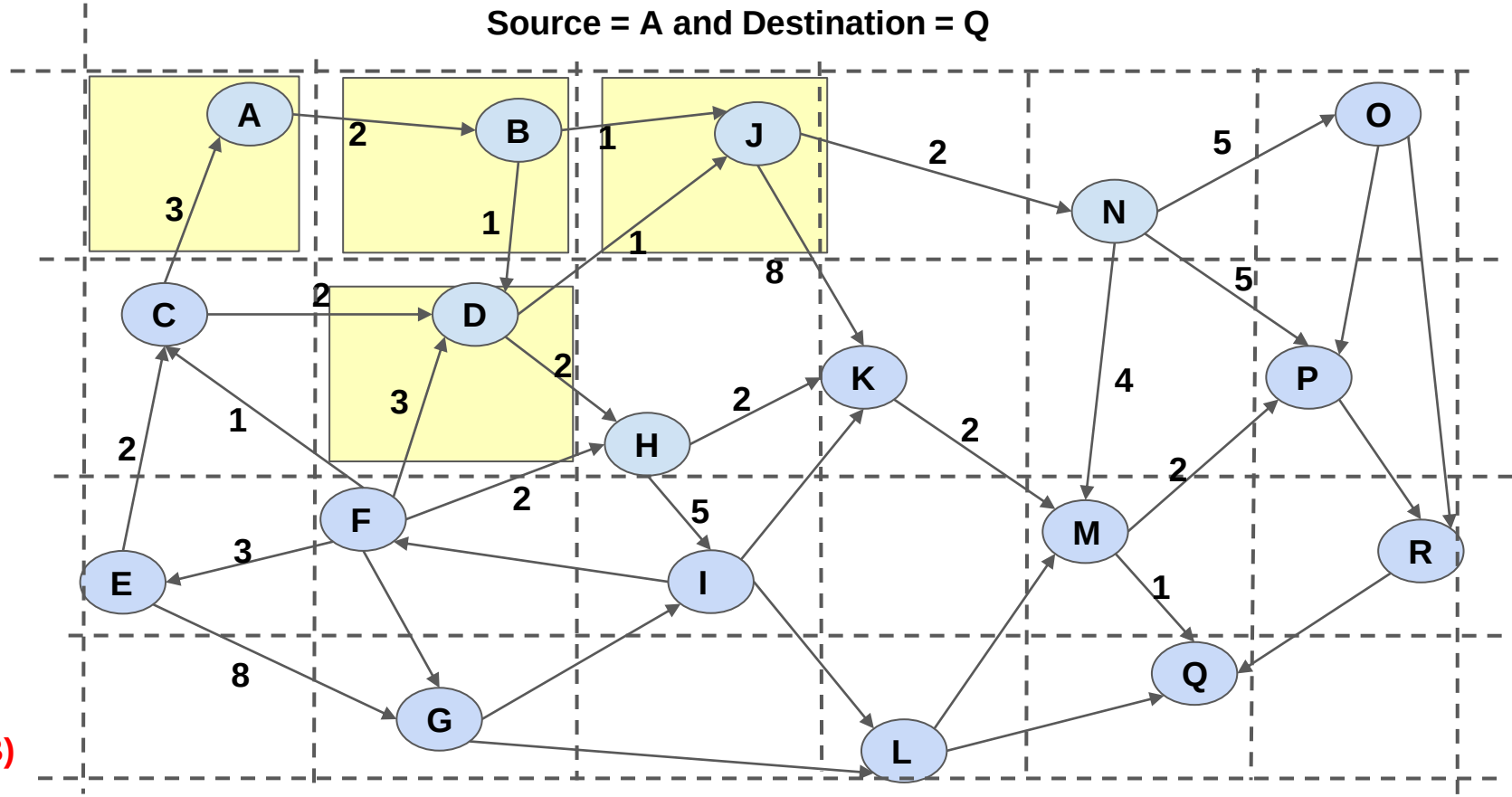
Shaded pages are already in Main memory

Open List:

H(5,D)
K(11,J)
N(5,J)

Closed List:

A(0), B(2,A),
D(3,B), J(3,B)

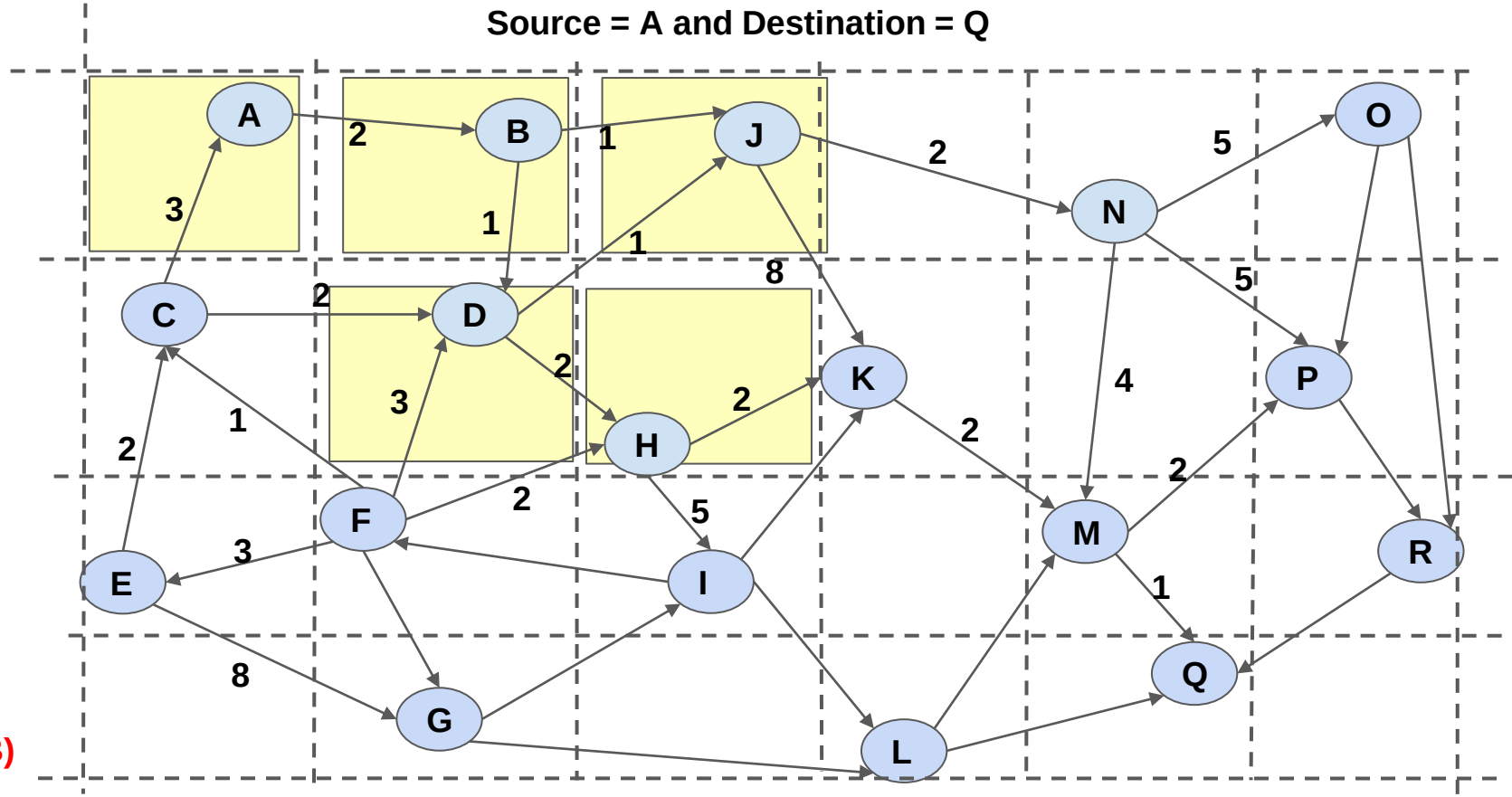


Shaded pages are already in Main memory

Source = A and Destination = Q

Open List:

I(10,H)
K(7,H)
N(5,J)



Closed List:

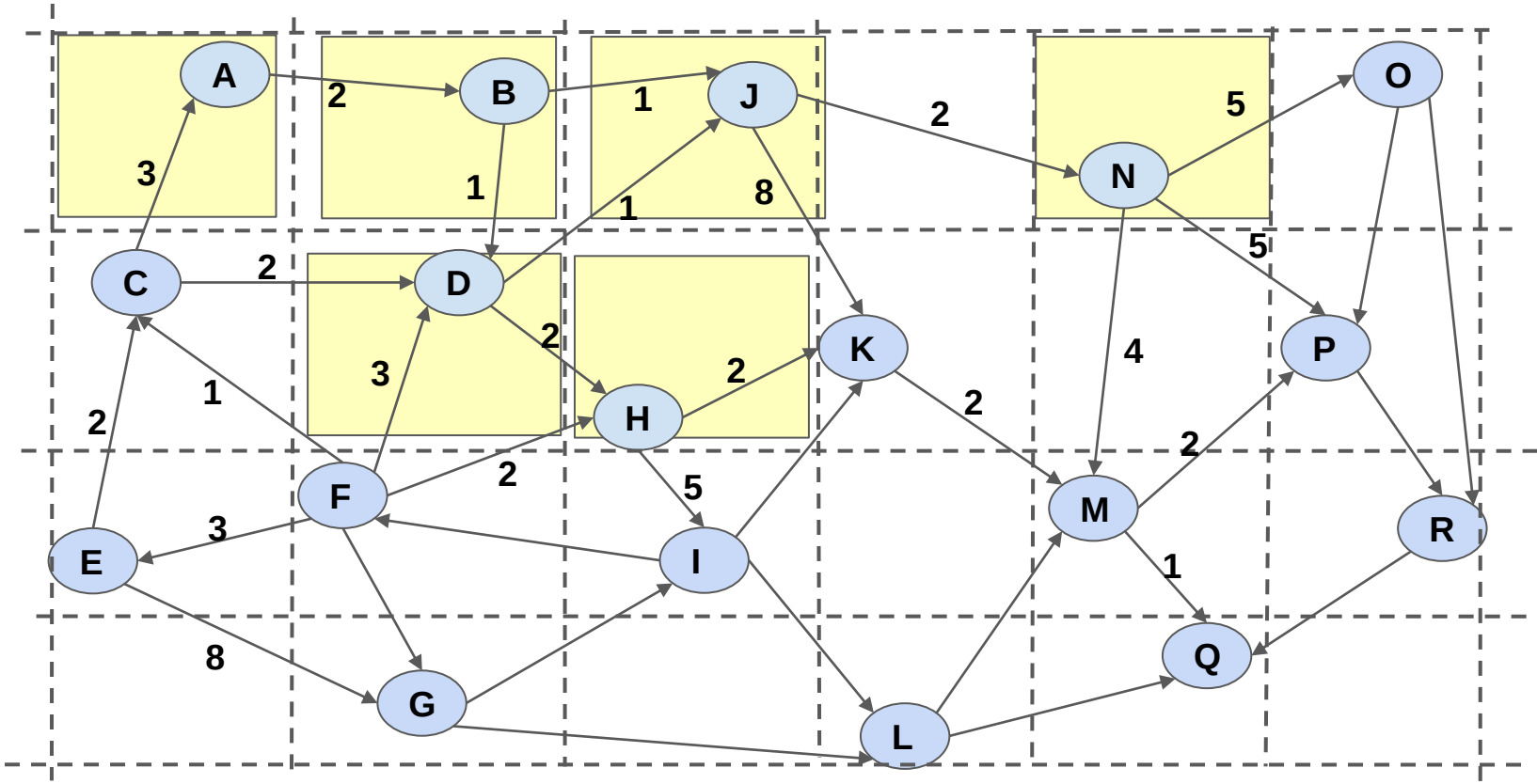
A(0), B(2,A),
D(3,B), J(3,B)
H(5,D)

Shaded pages are already in Main memory

Source = A and Destination = Q

Open List:

I(10,H)
K(7,H)
O(10,N)
P(10,N)
M(9,N)



Closed List:

A(0), B(2,A),
D(3,B), J(3,B)
H(5,D),N(5,J)

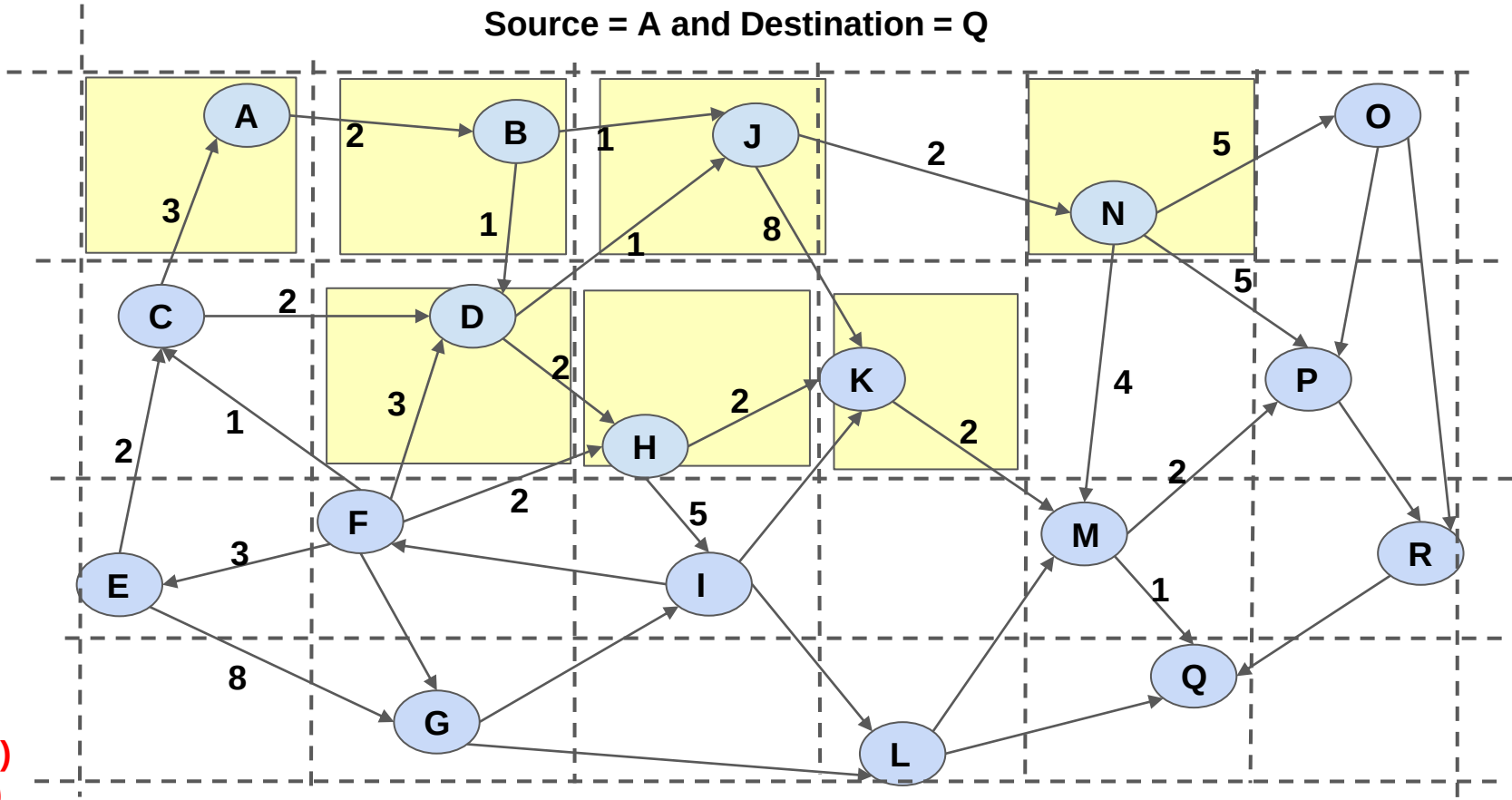
Shaded pages are already in Main memory

Open List:

I(10,H)
O(10,N)
P(10,N)
M(9,N)

Closed List:

A(0), B(2,A),
D(3,B), J(3,B)
H(5,D),N(5,J),
K(7,H)



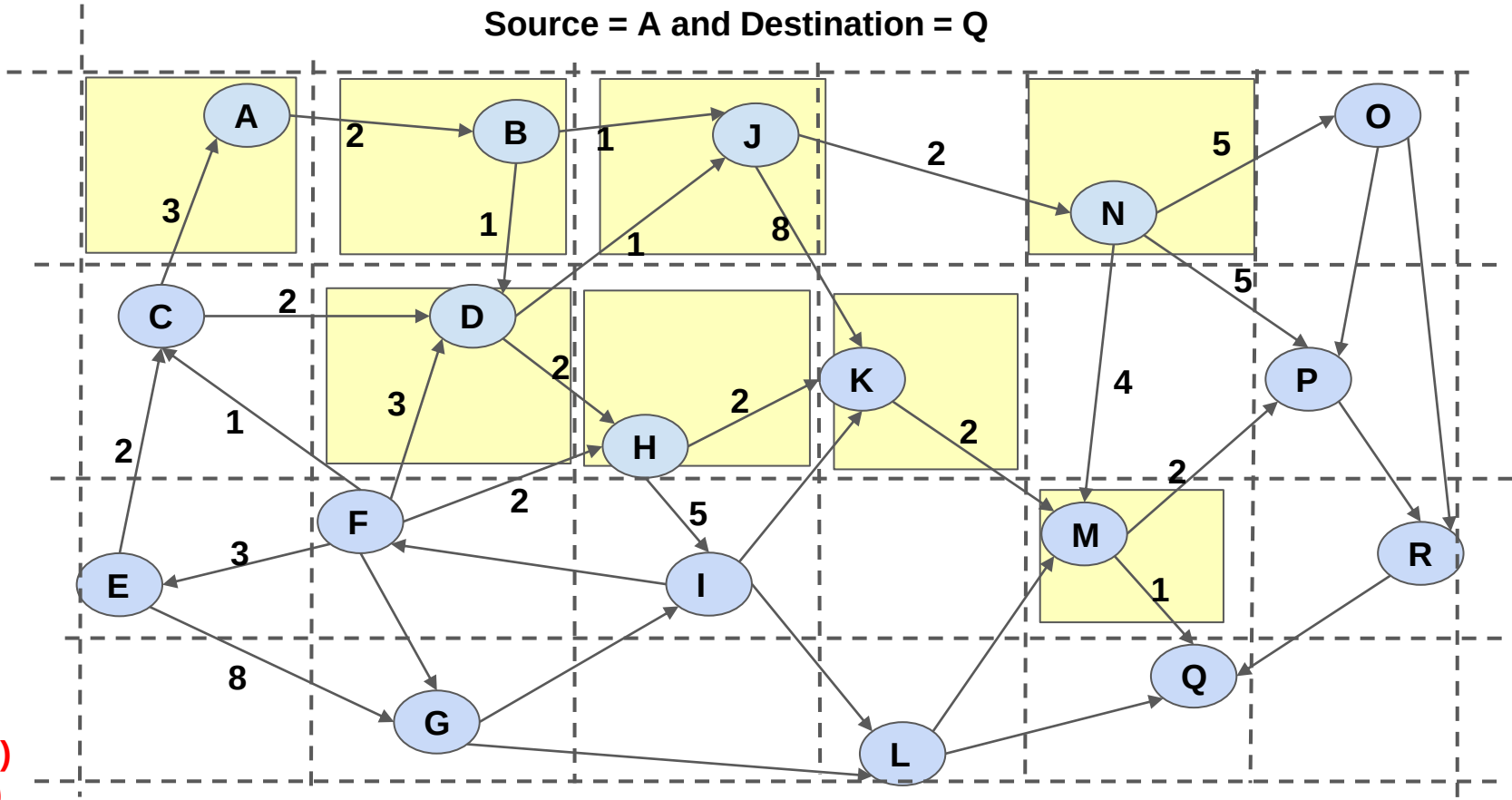
Shaded pages are already in
Main memory

Open List:

I(10,H)
O(10,N)
P(10,N)
Q(10,M)

Closed List:

A(0), B(2,A),
D(3,B), J(3,B)
H(5,D),N(5,J),
K(7,H), M(9,N)

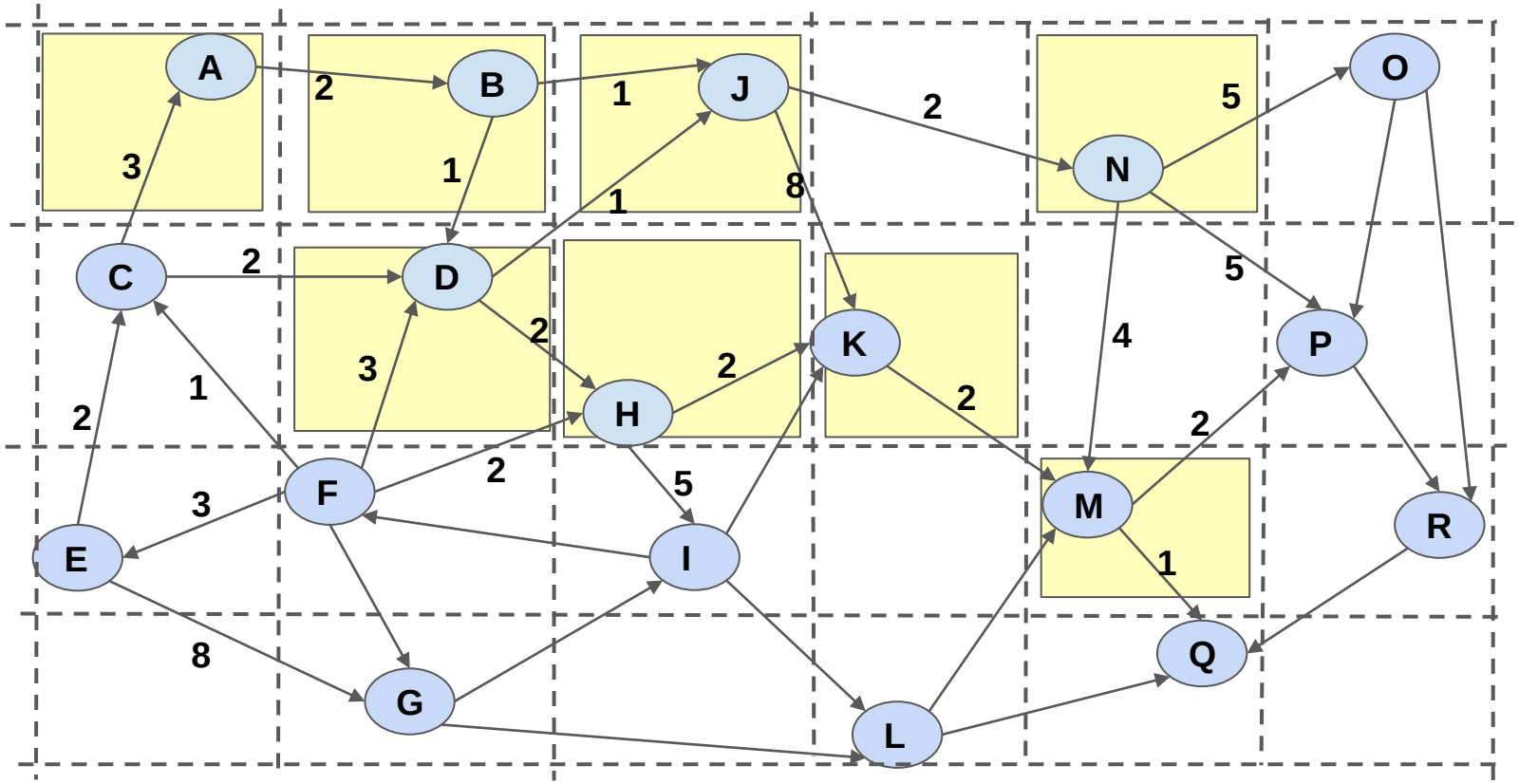


Shaded pages are already in
Main memory

Source = A and Destination = Q

Open List:

I(10,H)
O(10,N)
P(10,N)



Closed List:

A(0), B(2,A),
D(3,B), J(3,B)
H(5,D),N(5,J),
K(7,H), M(9,N)
Q(10,M)

Destination node closed,
algorithm terminates

Dijsktra's Algorithm on Partitions

Input: src node, dest node

1. Initialize a traditional priority queue (*PQ*) with src.
2. Initialize graph-seen to NULL
3. Loop until (*PQ* is not empty or dest node not closed)
   3.1. X ← result of Extract-Min
   3.2. Cid ← cell id of node X
   3.3. If graph of Cid not already present
      Retrieve the "disk block(s)" of Cid and update the graph-seen
   3.4. Add X to closed list
   3.5. Update current best path to other nodes in open list (if a better path is found) by exploring the outgoing edges of X.
4. End Loop