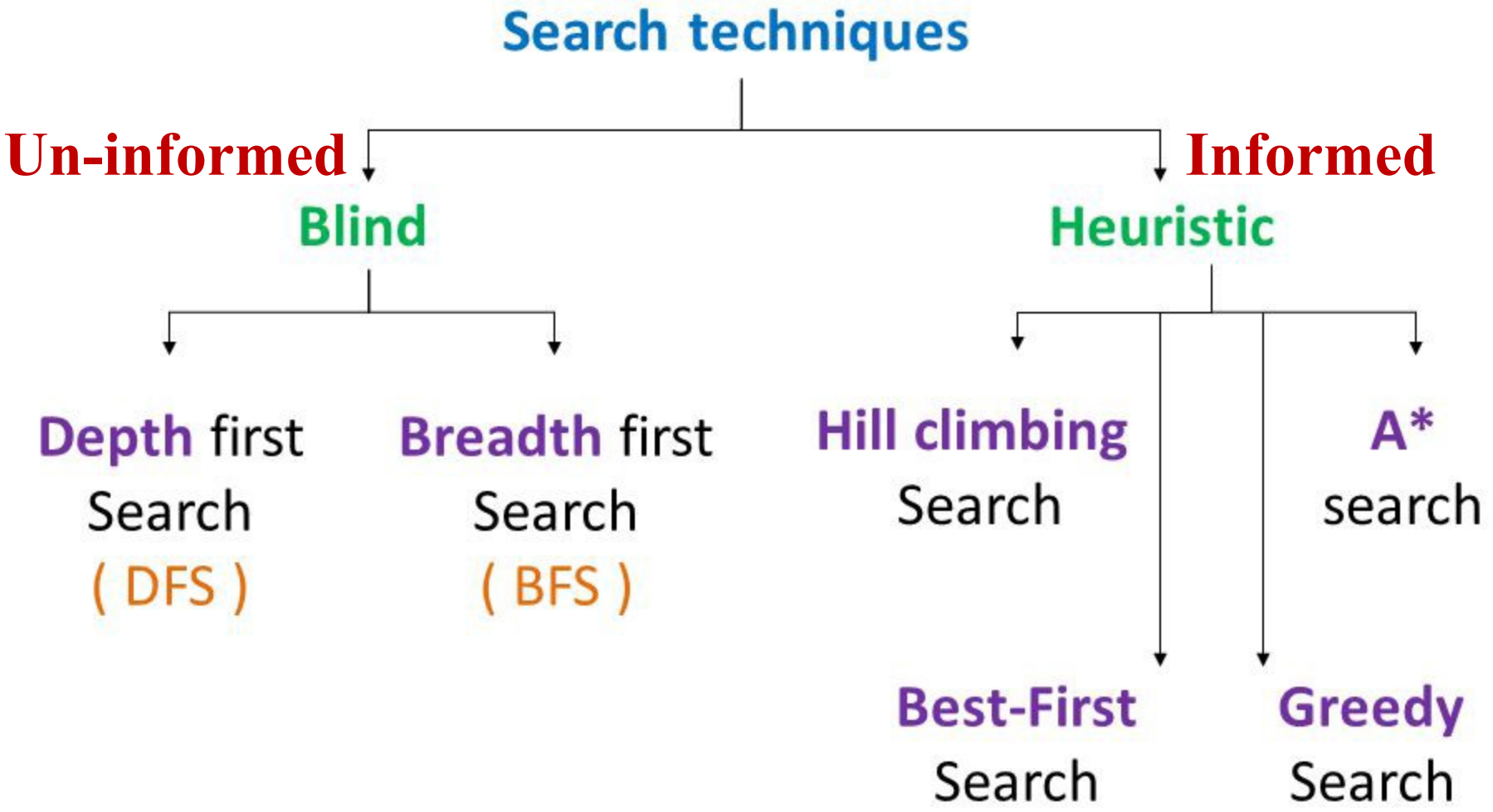
SEARCH TECHNIQUES



Informed search

- Generate-and-test
- Hill climbing
- Best-first search
- Problem reduction (A*/AO*)
- Constraint satisfaction
- Means-ends analysis

Generate-and-Test

Algorithm:

1. Generate a possible solution.
2. Test to see if this is actually a solution.
3. Quit if a solution has been found. Otherwise, return to step 1.

Advantages

- Complete
- Non Redundant

Disadvantage

- Acceptable for simple problems.
- Inefficient for problems with large space.

Hill Climbing Algorithm in AI

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/ value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbour has a higher value.
- A node of hill climbing algorithm has two components which are state and value.
- Hill Climbing is mostly used when a good heuristic is available.
- It does not backtrack the search space, as it does not remember the previous states.
- Types of Hill Climbing Algorithm:

❑**Simple hill Climbing:**

❑**Steepest-Ascent hill-climbing**

The **closest node** is chosen in a **simple hill climbing** algorithm while the node **closest to the solution** is identified and chosen in a **steepest ascent hill climbing algorithm**

Simple Hill Climbing

Simple hill climbing is the simplest way to implement a hill climbing algorithm. **It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state**. It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state.

Algorithm for Simple Hill Climbing:

Step 1: Evaluate the initial state, if it is goal state then return success and Stop.

Step 2: Loop Until a solution is found or there is no new operator left to apply.

Step 3: Select and apply an operator to the current state.

Step 4: Check new state:

- If it is goal state, then return success and quit.
- Else if it is better than the current state then assign new state as a current state.
- Else if not better than the current state, then return to step2.

Step 5: Exit.

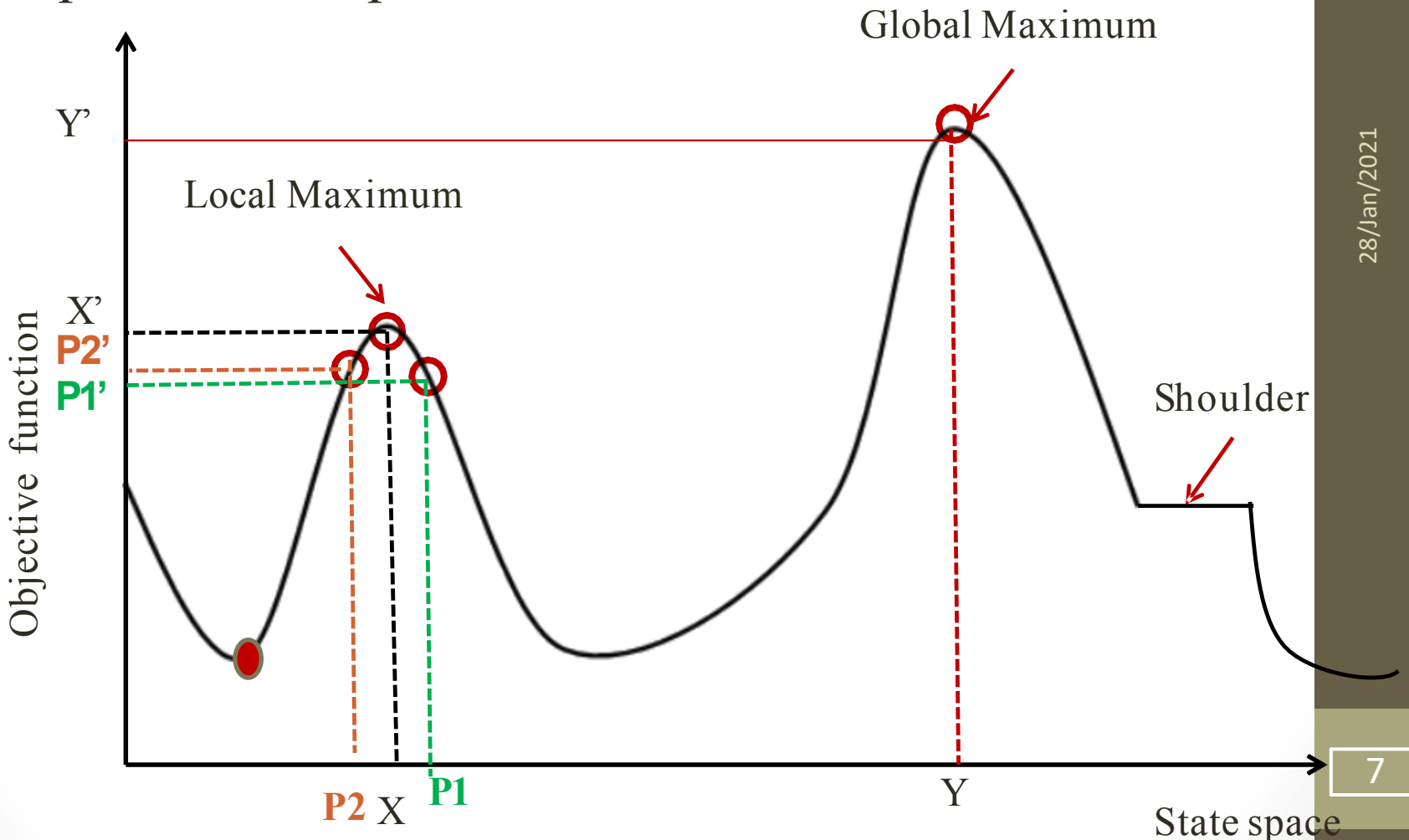
Steepest-Ascent Algorithm for Hill Climbing

1. Evaluate the initial state. If it is the goal state then return and quit. Otherwise continue with initial state as current state.
2. Loop until a solution is found or until there are no new operators left to be applied to the current state:
   a. Select operator that has not been applied to the current state and apply it to produce the new state.
   b. Evaluate the new state
      i. If it is the goal state, then return and quit
      ii. If it is not a goal state but it is better than the current state then make it the current state.
      iii. If it is not better than the current state then continue in the loop.

State-space Diagram for Hill Climbing:

Local Search Algorithm

State space landscape

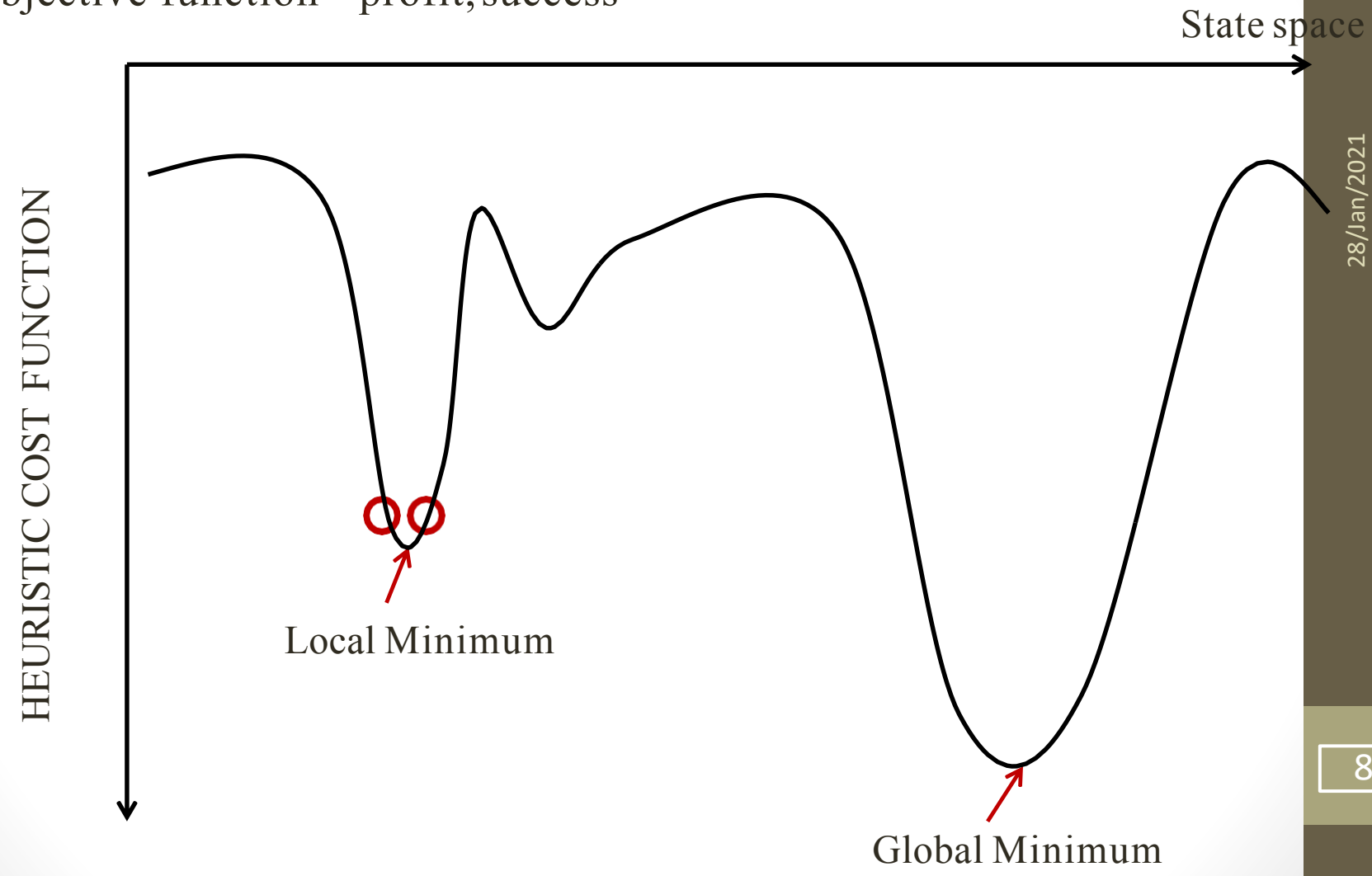


Maximization function is called Objective Function

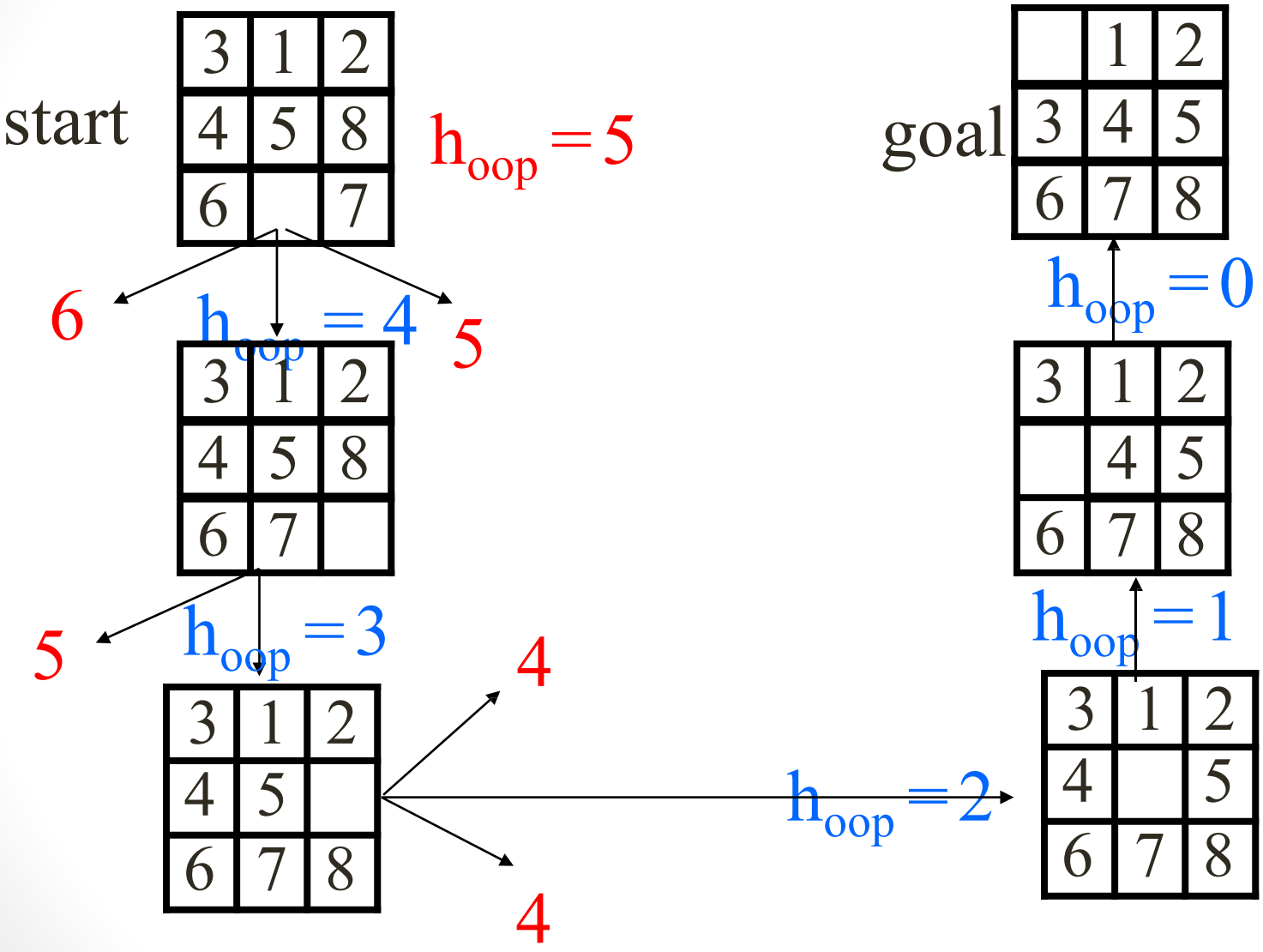
Minimization function is called Heuristic Cost function

Heuristic cost= distance, time, money spent

Objective function= profit, success



Hill climbing Example (minimizing h)



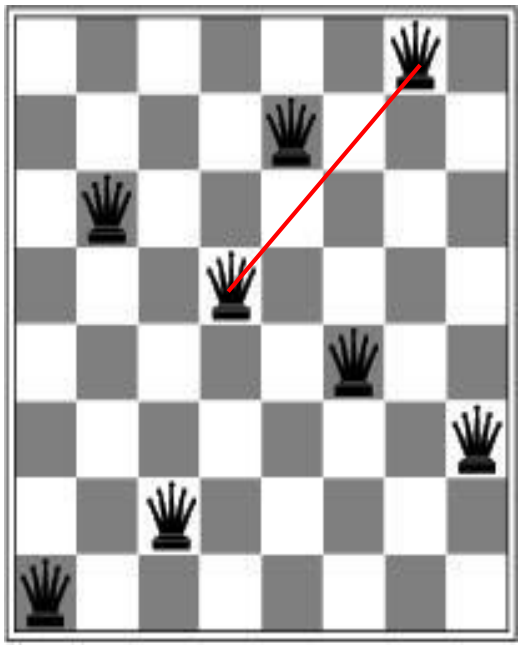
Hill-climbing Example: n-queens

- ***n*-queens problem: Put *n* queens on an *n* × *n* board with no two queens on the same row, column, or diagonal**
- ***Good heuristic:* *h* = number of pairs of queens that are attacking each other**

h=5 *h=3* *(for illustration)* *h=1*

Hill-climbing example: 8-queens

A state with *h*=17 and the *h*-value for each possible successor

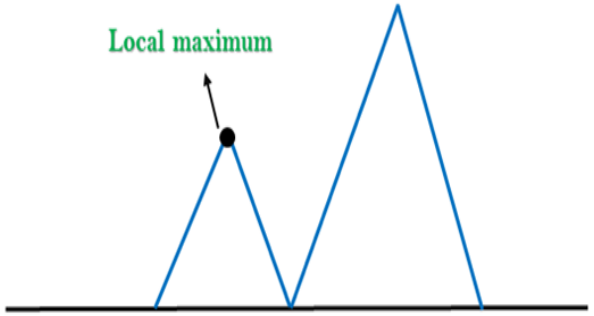


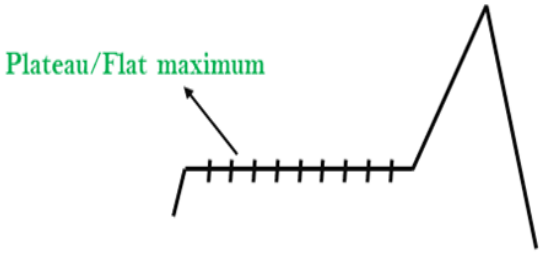
A local minimum of *h* in the 8-queens state space (*h*=1).

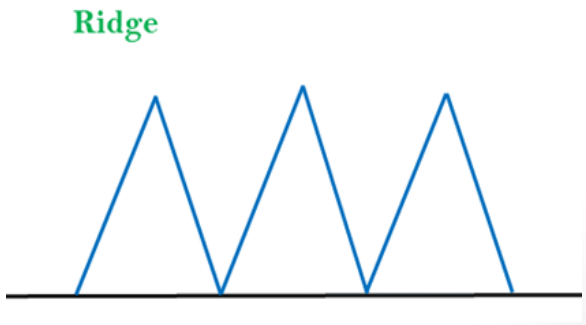
h = number of pairs of queens that are attacking each other

Problems with Hill Climbing

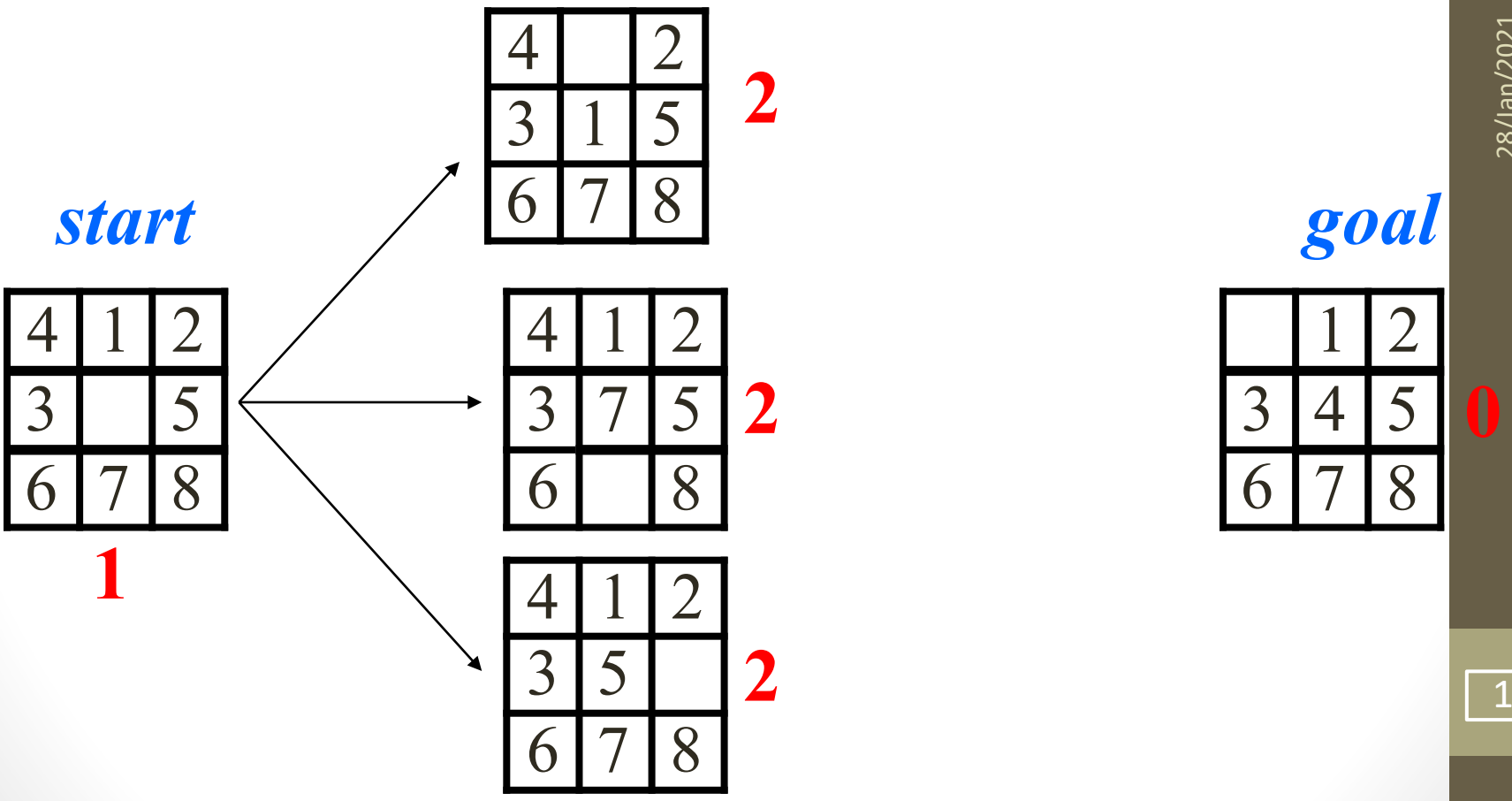
- Both simple and steepest hill climbing may fail to find solution because of the following.
- **Local Maximum:** A state that is better than all its neighbors but is not better than some other states farther away.
- **Plateau:** Is a flat area of the search space in which a whole set of neighboring states have the same value.
- **Ridge**: A special kind of local maximum. An area of the search space that is higher than surrounding areas and that itself has slope .







Example


start
4 1 2
3 5
6 7 8
1
4 2
3 1 5
6 7 8
2
4 1 2
3 7 5
6 8
2
4 1 2
3 5
6 7 8
2
goal
1 2
3 4 5
6 7 8
0

Advantages of Hill Climbing

It can be used in continuous as well as discrete domains.

Disadvantages of Hill Climbing

1. Not efficient method –not suitable to problems where the value of heuristic function drops off suddenly when solution may be in sight.
2. Local search method- gets caught up in local maxima/ minima.

Solution to Local Maxima problem

1. Backtracking to some earlier node and try different direction.
2. Simulated annealing

Global Search Techniques:

1. Best First Search(OR graph)

- Where not only the current branch of the search space but all the so far explored nodes/states in the search space are considered in determining the next best state/node.

2. A* Algorithm

- Which is improvised version of Best first Search.

3. Problem Reduction and And-Or Graphs.

- AO* Algorithm.

4. Constraint Satisfaction Problem (CSP)

- Graph Colouring Problem and Crypt Arithmetic Problems.

5. Mean End Analysis (MEA)

Best First Search

- Heuristic based search technique.
- Every node in the search space has an Evaluation function (heuristic function)associated with it.
- Evaluation function==heuristic cost function (in case of minimization problem) OR objective function(in case of maximization).
- Decision of which node to be expanded depends on value of evaluation function.
- Evaluation value= cost/distance of current node from goal node.
- For goal node evaluation function value=0

Best-first search (Differences from hill climbing)

- In hill climbing at each step one node is selected and all others are rejected and never considered again. While in Best-first search one node is selected and all others are kept around so that they may be revisited again.

- Best available state is selected even if it may have a value lower than the currently expanded state.
- Implementation of the best first search requires the following

1. **OPEN-** all those nodes that have been generated & have had heuristic function applied to them but have not yet been examined.
2. **CLOSED-** contains all nodes that have already been examined.

Best First Search Algorithm:

Step 1: Place the starting node into the OPEN list.

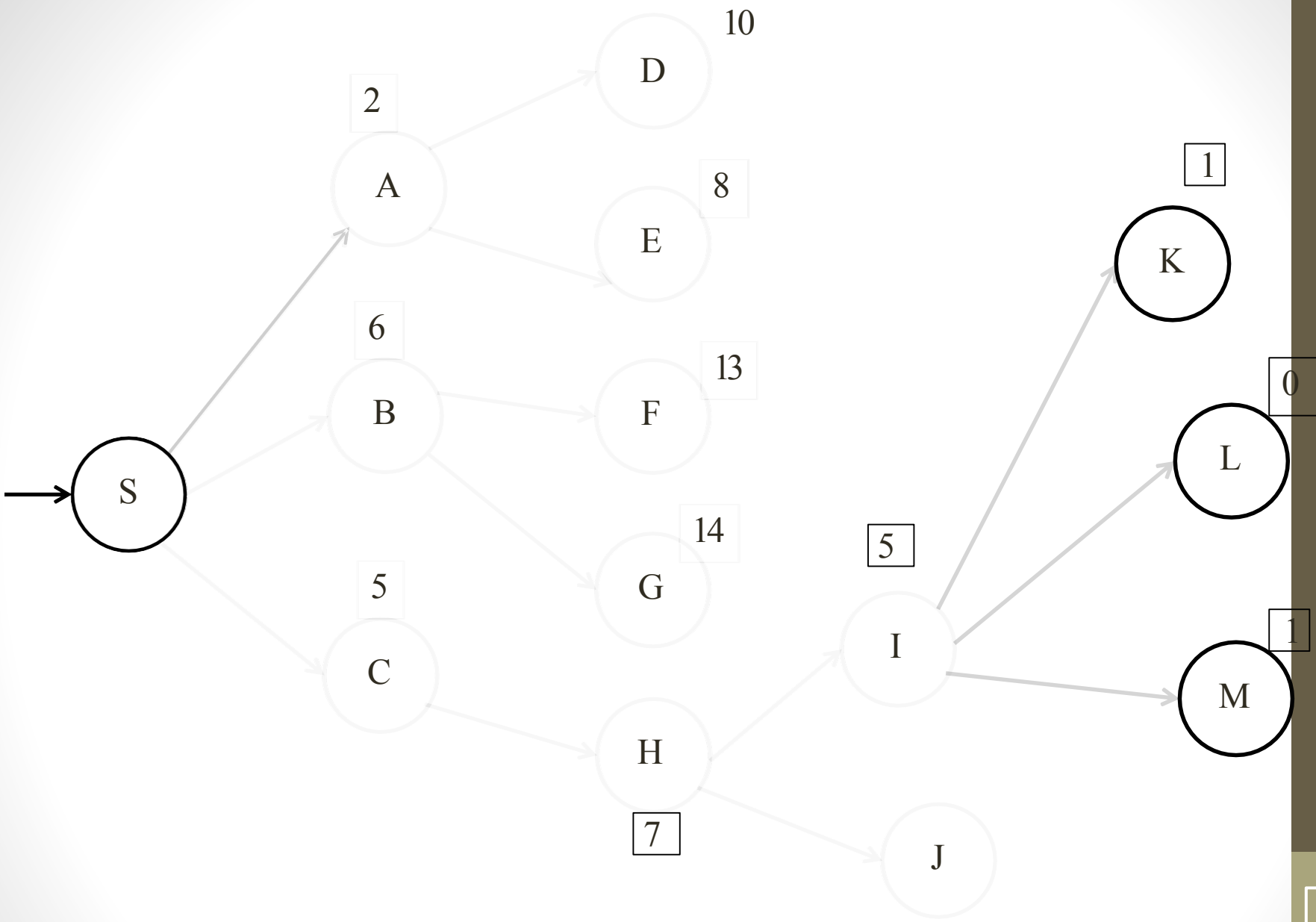
Step 2: If the OPEN list is empty, Stop and return failure.

Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.

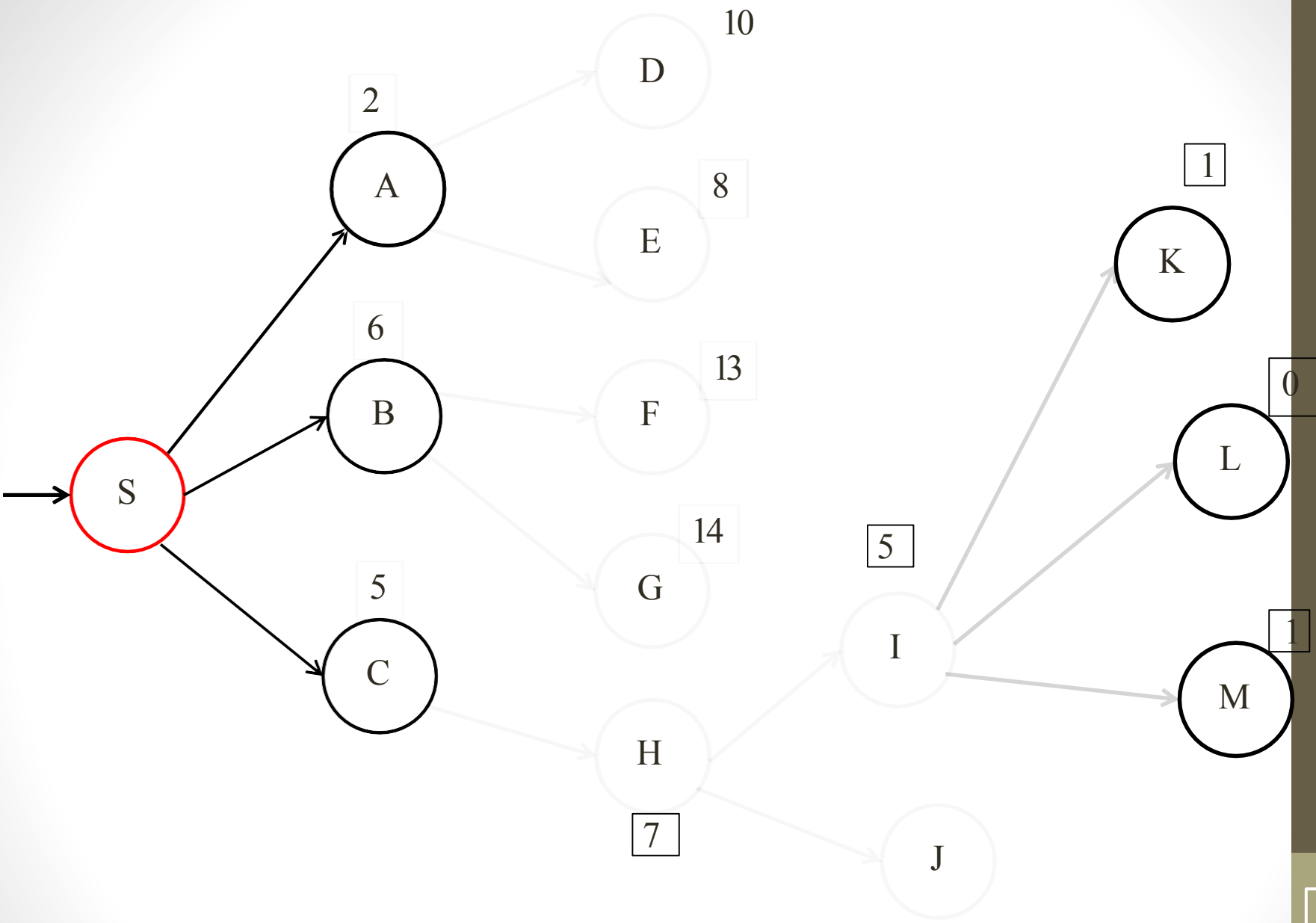
Step 4: Expand the node n, and generate the successors of node n.

Step 5: Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.

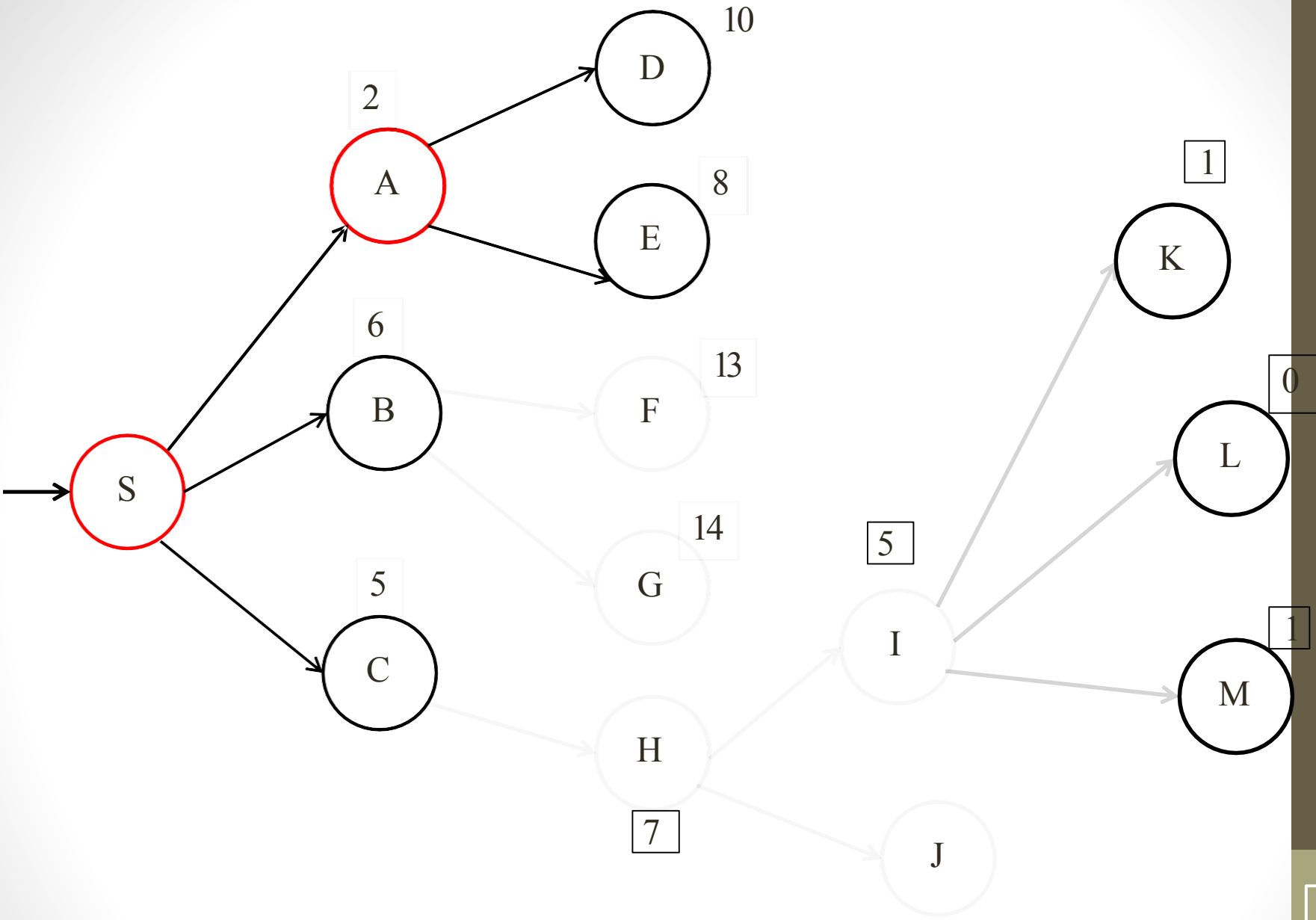
Step 6: For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.



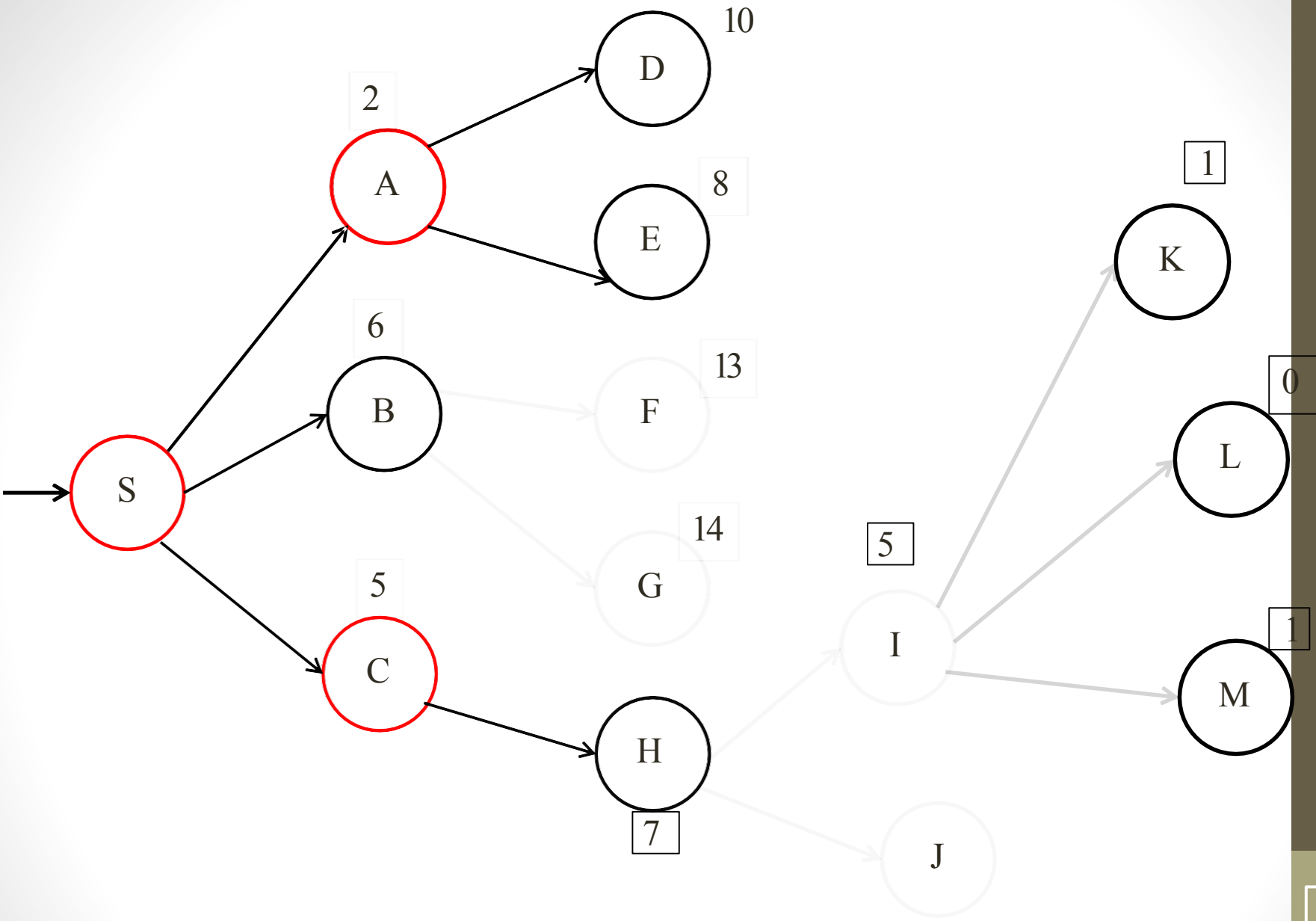
OPEN={ S}
CLOSED={}



OPEN={ A(2),C(5),B(6)}
CLOSED={S}

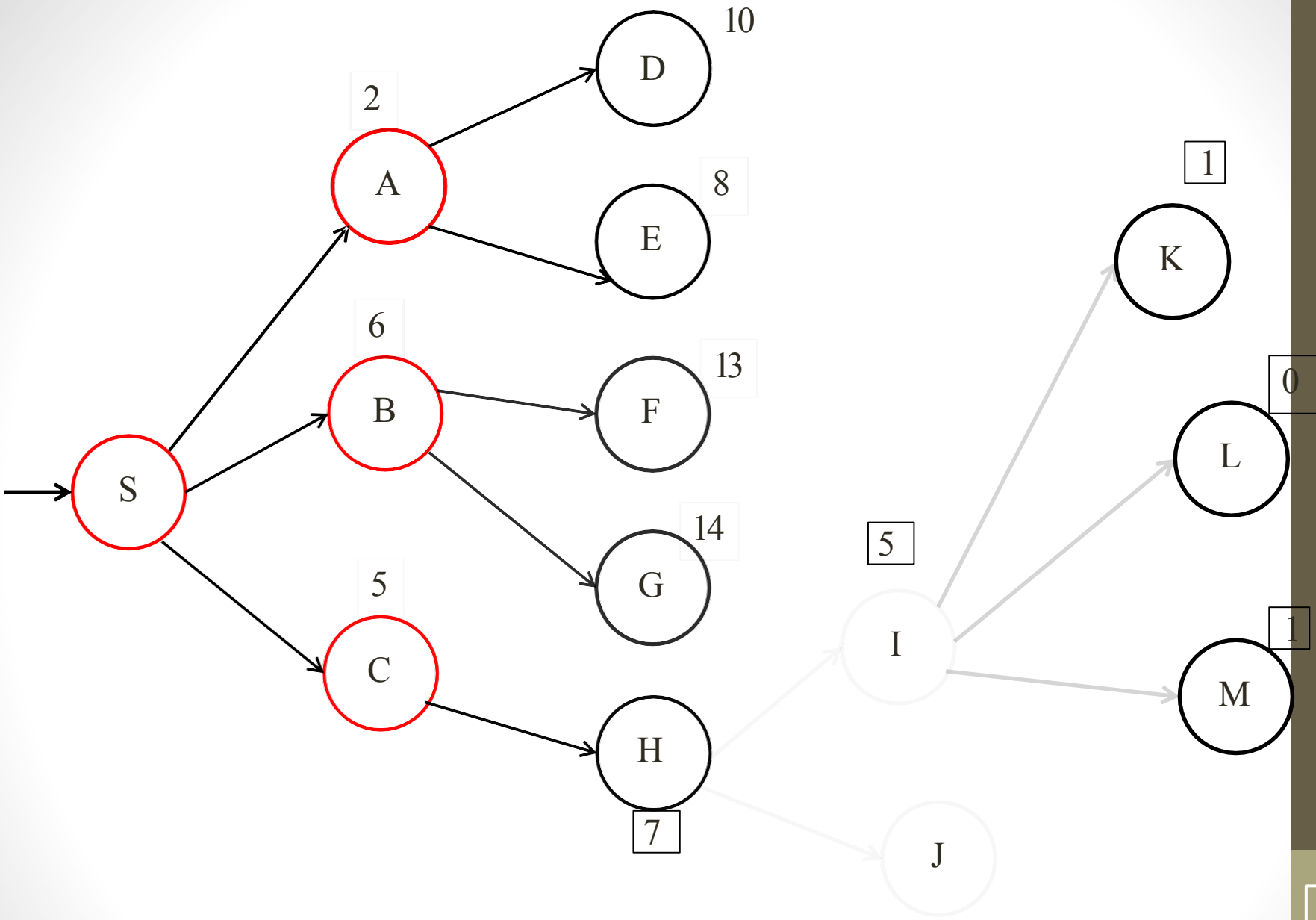


OPEN={ C(5), B(6), **E(8), D(10)**}
CLOSED={S,**A**}

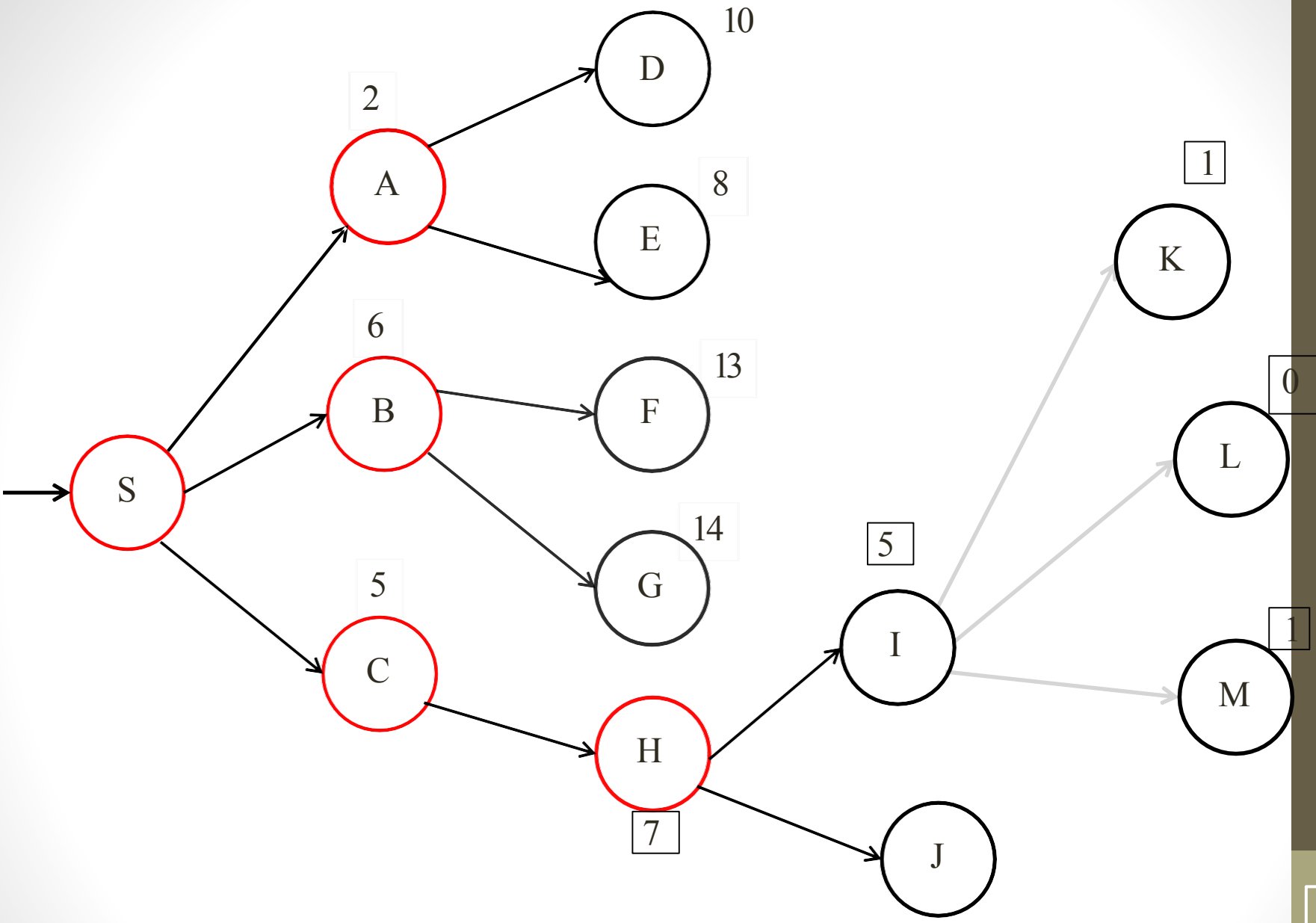


OPEN={ B(6), H(7), E(8), D(10)}
CLOSED={S,A, C}

6

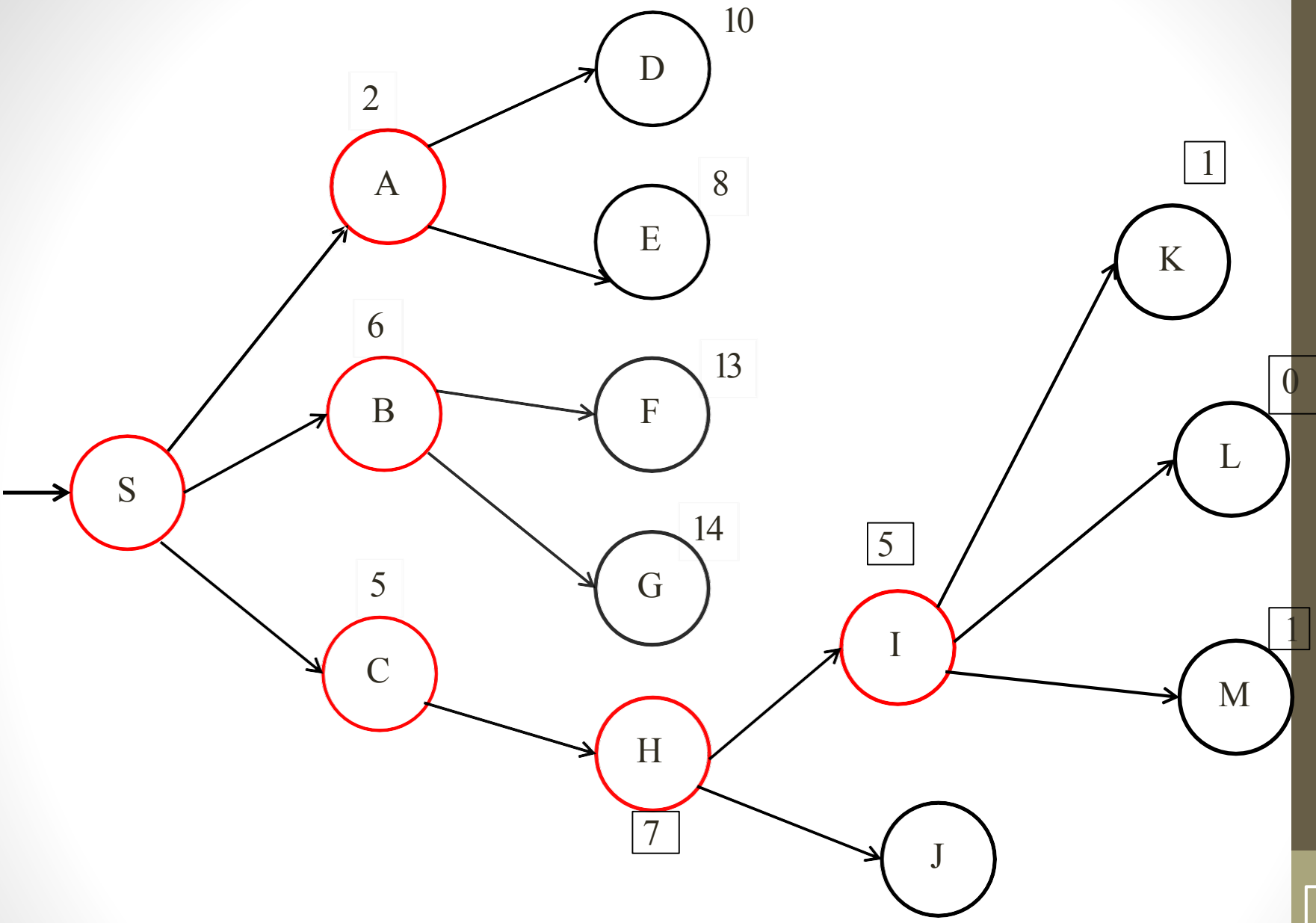


OPEN={H(7),E(8), D(10), **F(13),G(14)**}
CLOSED={S,A, C, **B**}

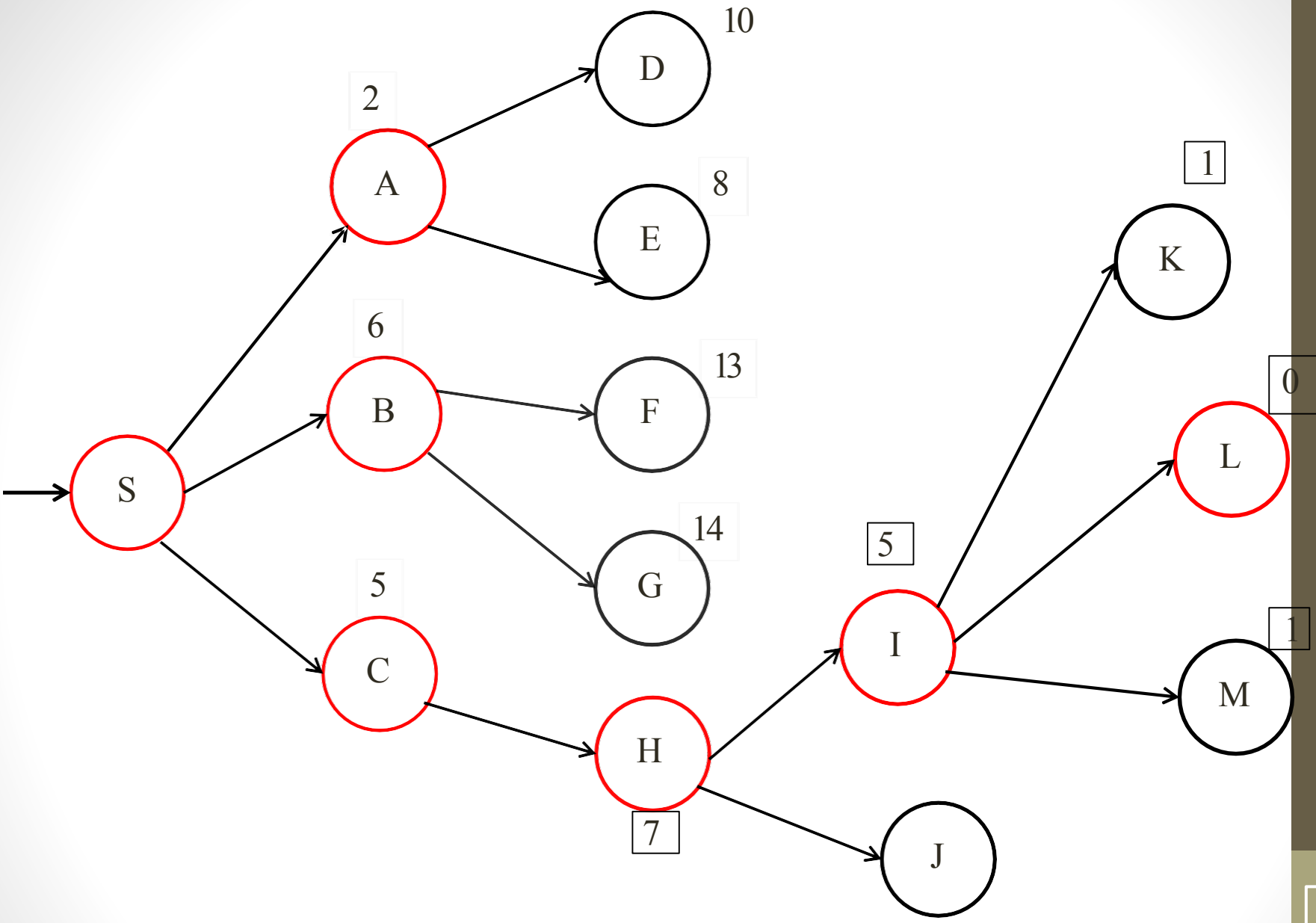


OPEN={**I(5), J(6),**H(7),E(8), D(10), F(13),G(14)}
CLOSED={S,A, C, B,**H**}

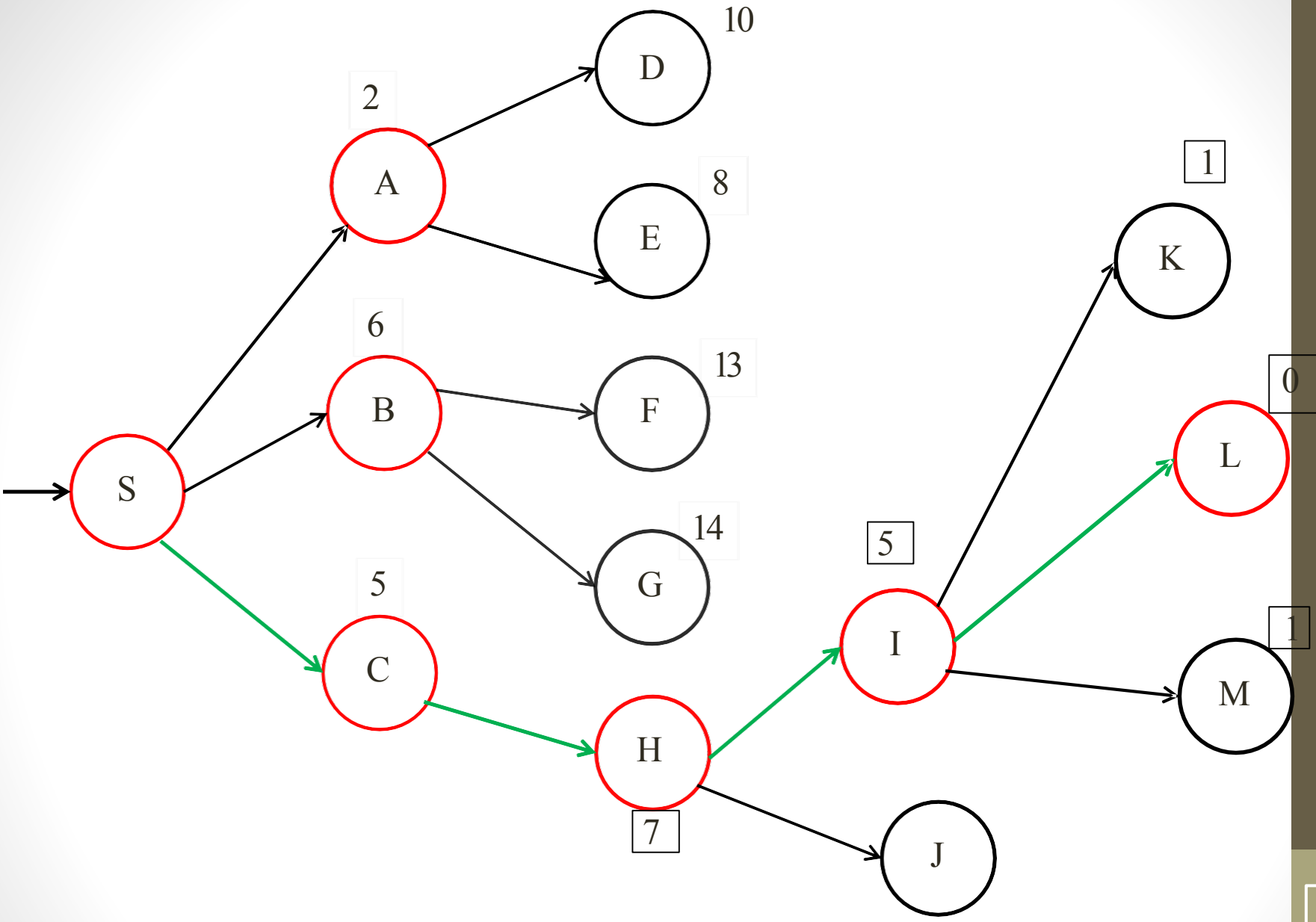
6



OPEN={**L(0),K(1),M(1),**I(5), J(6),H(7),E(8), D(10), F(13),G(14)6}
CLOSED={S,A, C, B,H,**I**}



OPEN={K(1),M(1),I(5), J(6),H(7),E(8), D(10), F(13),G(14)} 6
CLOSED={S,A, C, B,H,I,L}



OPEN={K(1),M(1),I(5), J(6),H(7),E(8), D(10), F(13),G(14)} 6

CLOSED={S,A, C, B,H,I,L} **GOAL NODE FOUND!!**

In Best First Search we jump all around in the search graph to identify the nodes with minimal evaluation function value.

Gives faster solutions but still no guarantee.

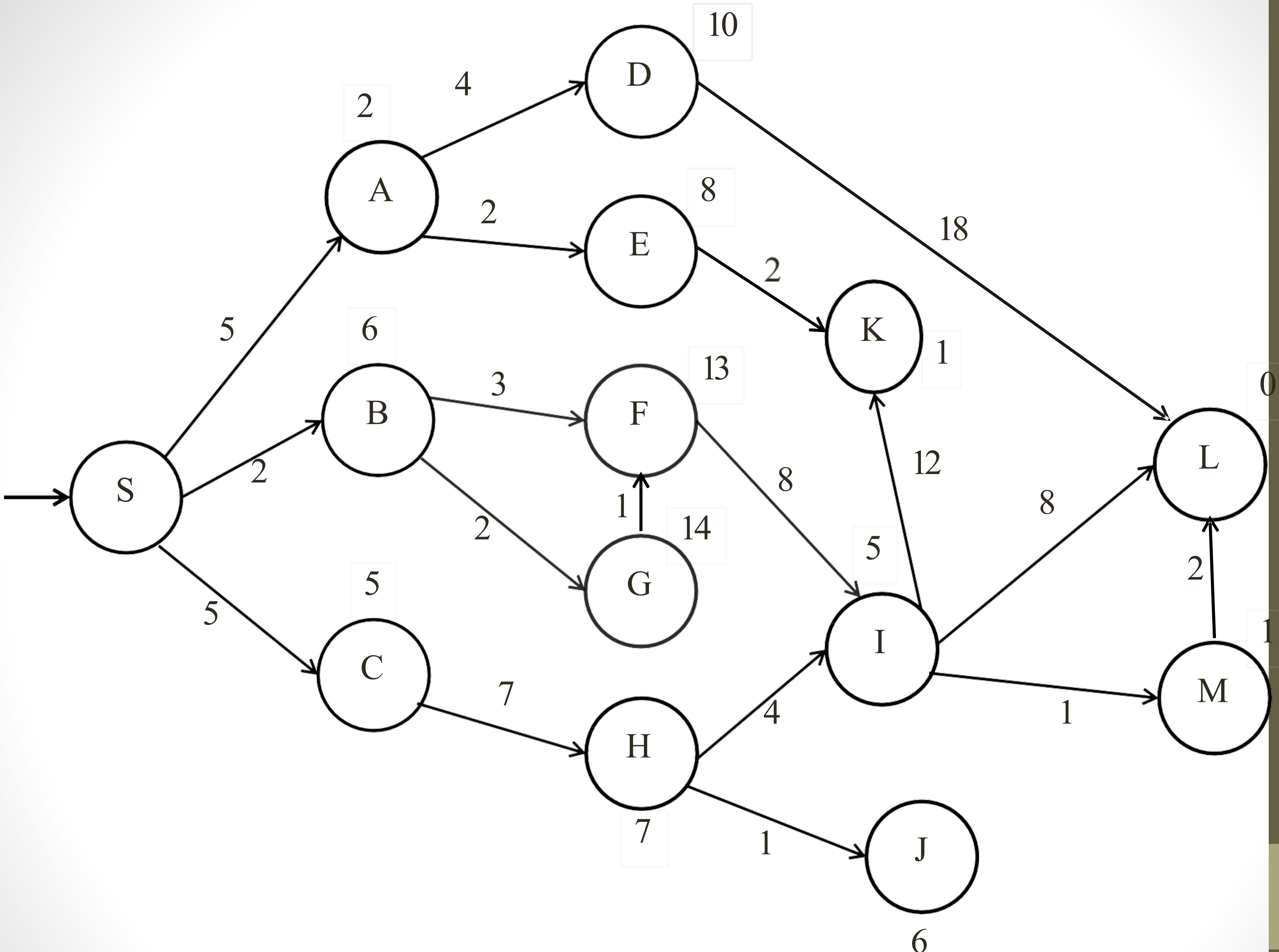
A* Algorithm

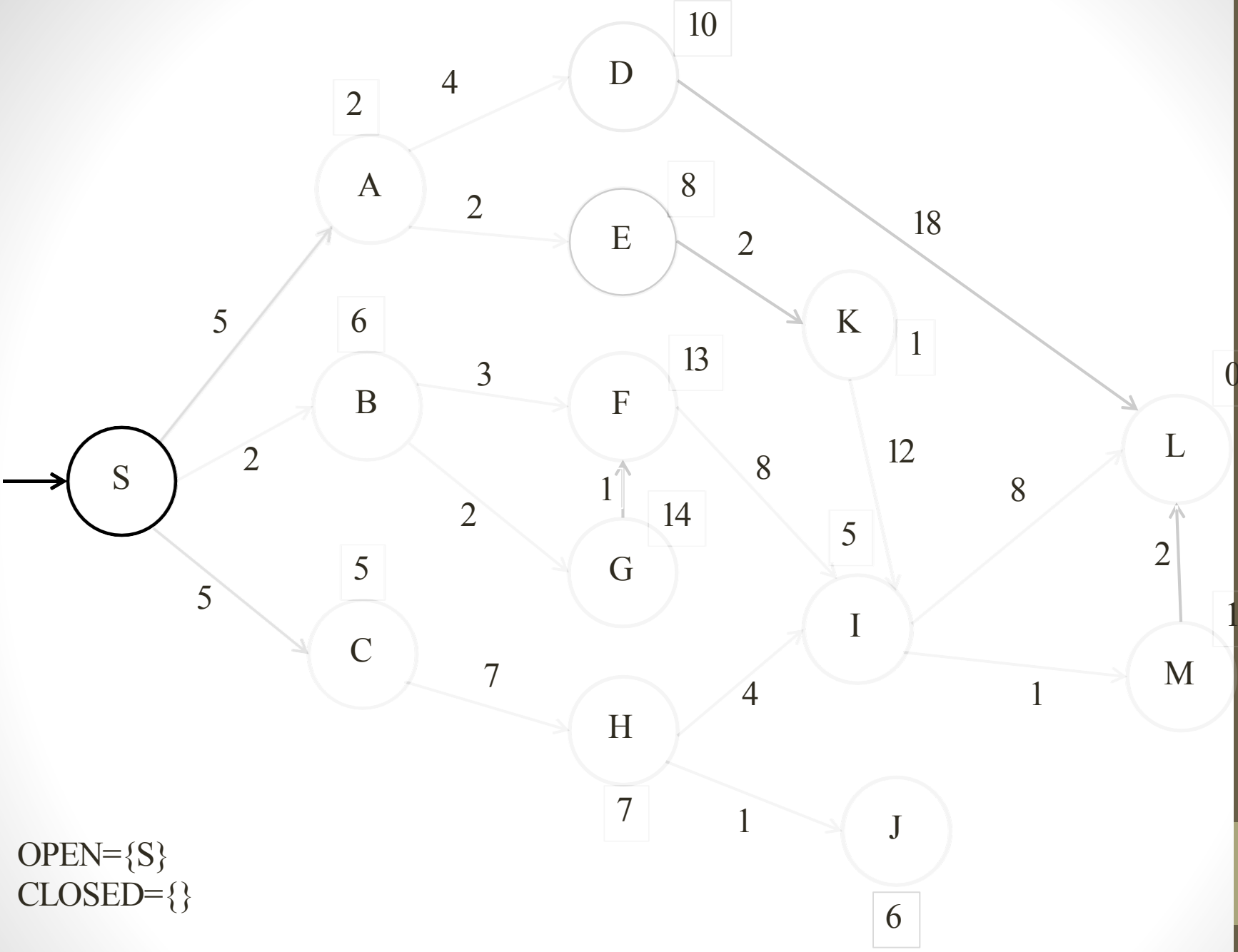
- A* algorithm is similar to Best first Search.
- Only difference: Best First Search takes h(n) as evaluation function/heuristic value.
- In Best first search h(n)= estimated cost of current node 'n' from the goal node.

- A* takes the evaluation function as:
- F(n)=g(n)+h(n)
- where,

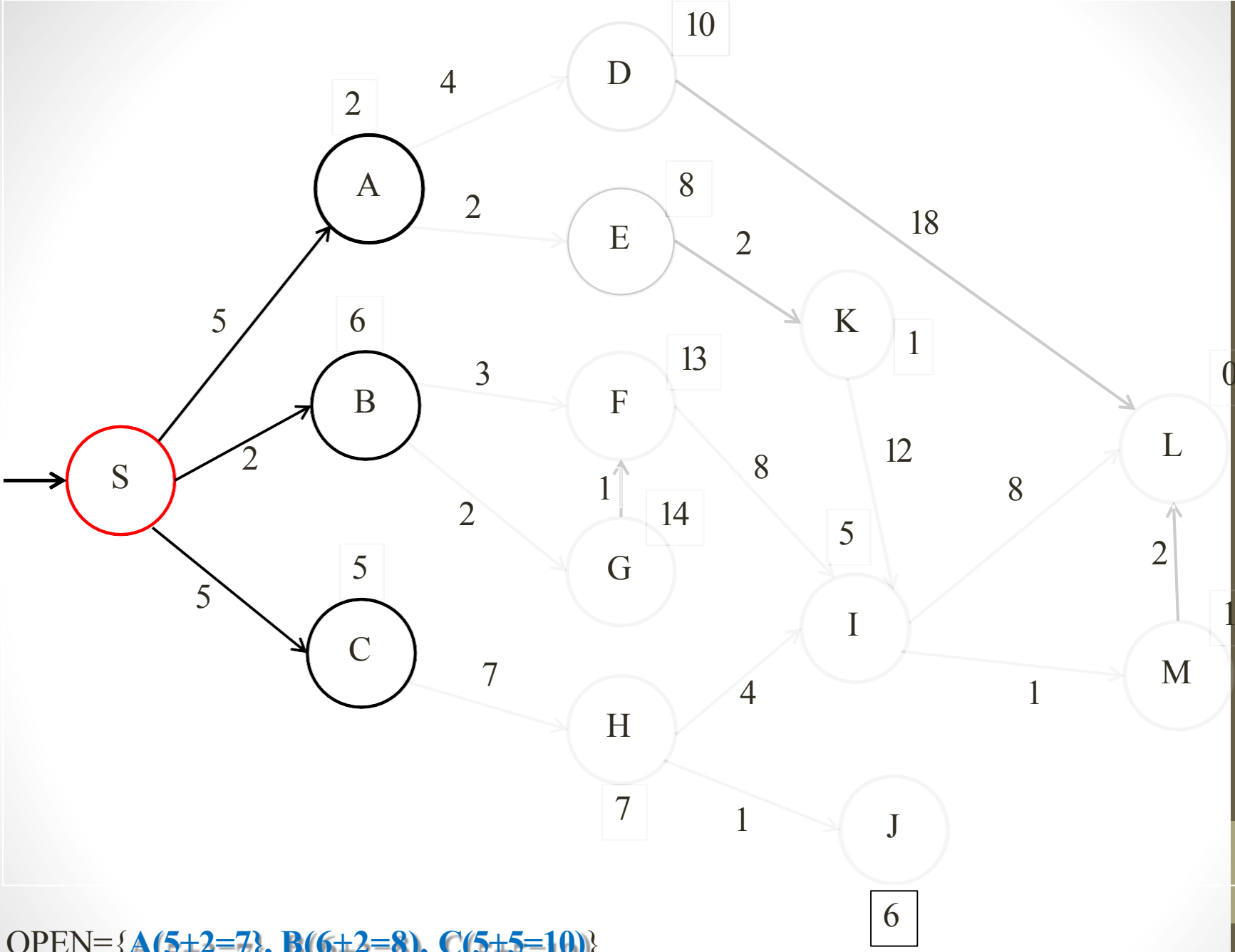
g(n)= cost(or distance) of the current node from **start state.**

h(n)= estimated cost of current node from goal node.

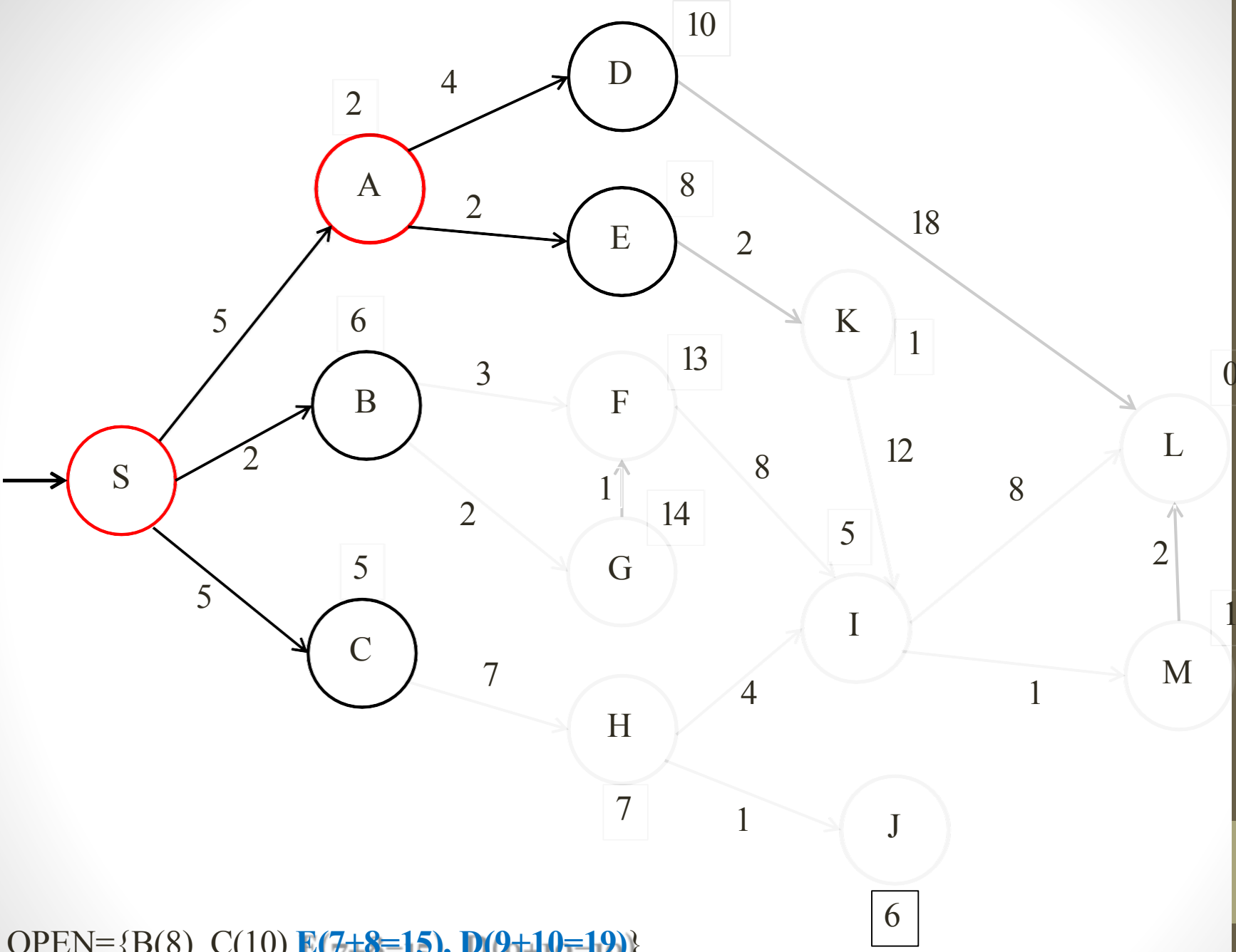

10
D
4
2
A
2
8
E
18
2
5
6
K
1
13
3
B
F
0
12
S
2
8
L
8
2
1
14
G
5
2
5
5
I
1
C
M
7
4
1
H
7
1
J
6



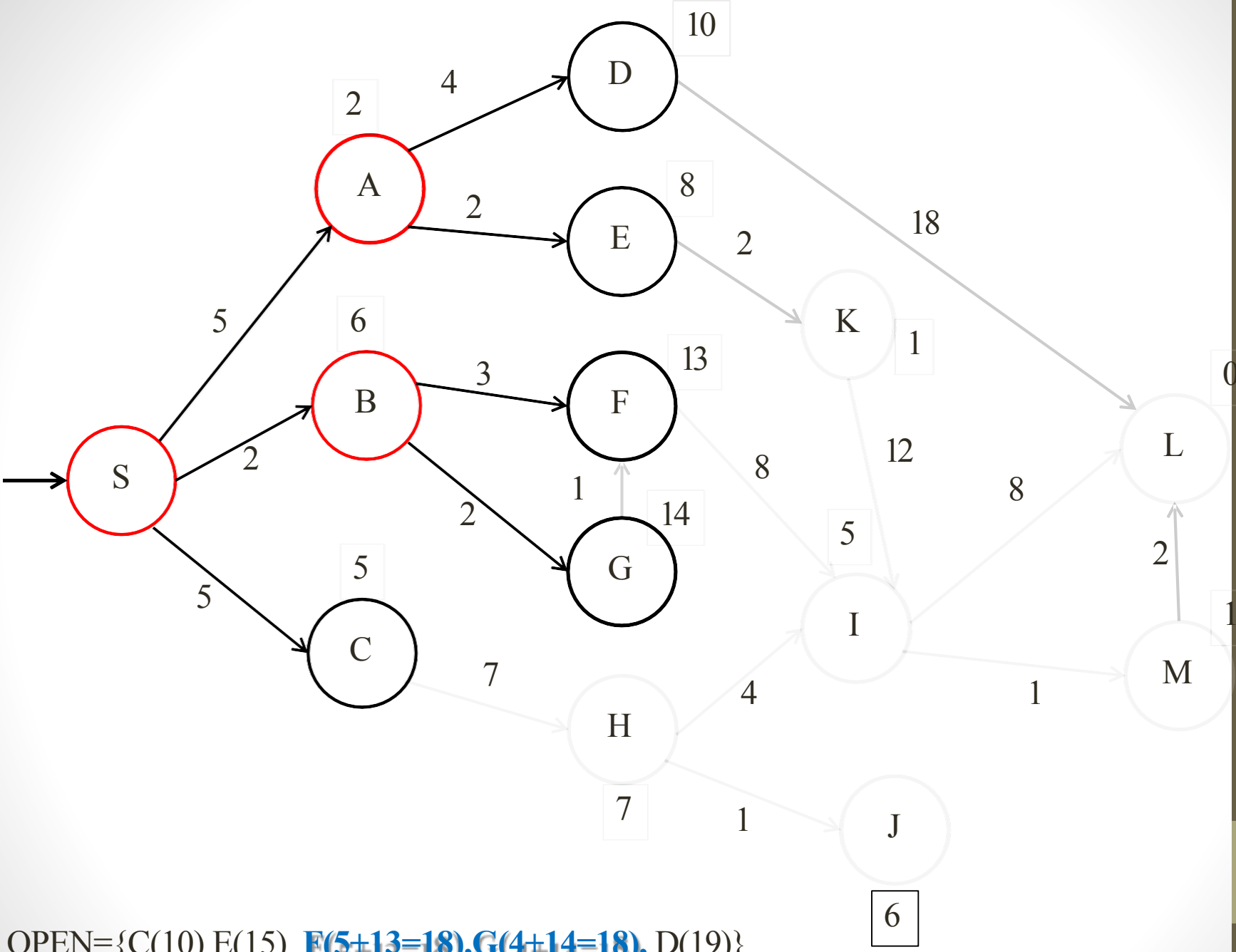
OPEN={S}
CLOSED={}



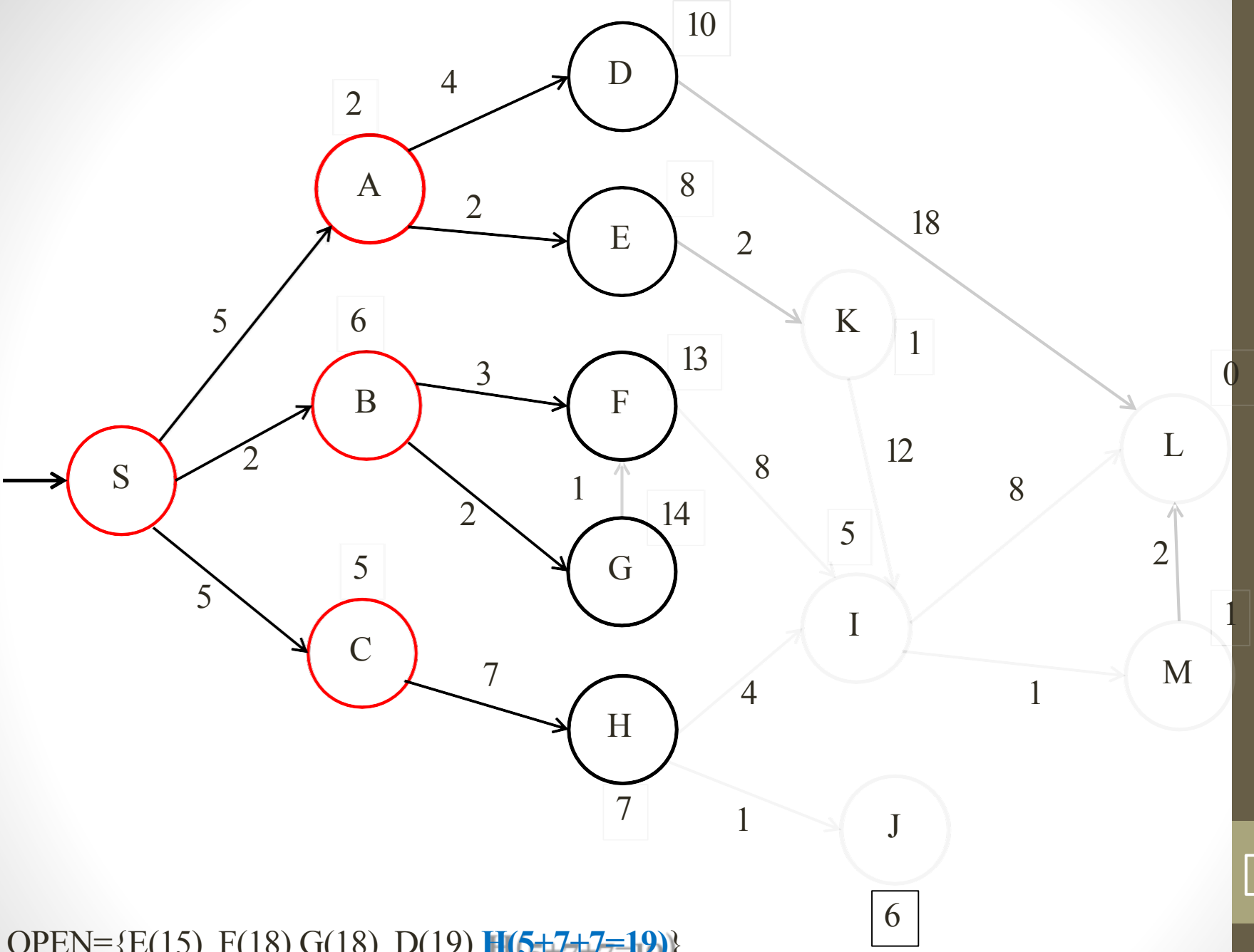
OPEN={A(5+2=7}, B(6+2=8), C(5+5=10)}
CLOSED={S}



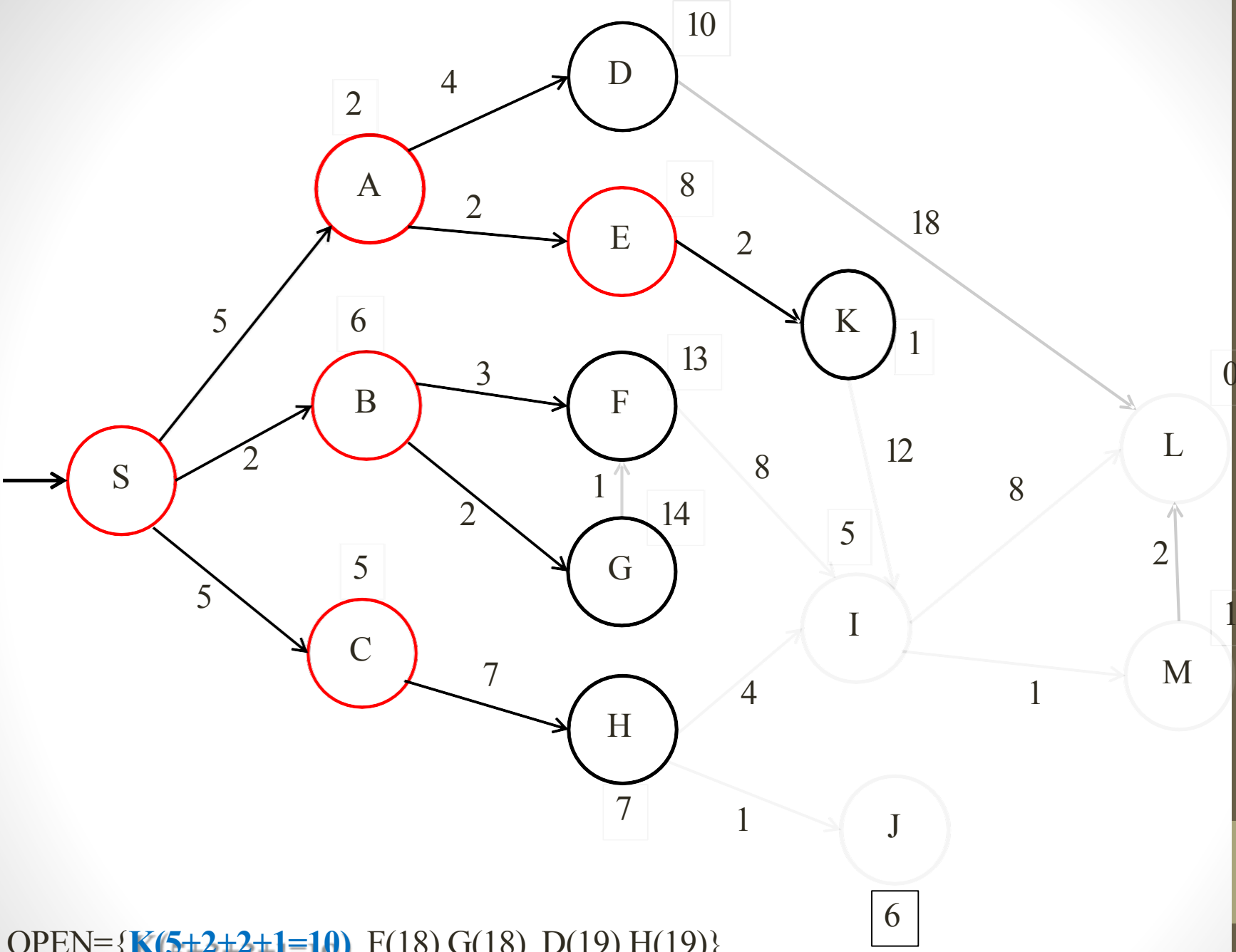
OPEN={B(8), C(10),E(7+8=15), D(9+10=19)}
CLOSED={S,A}



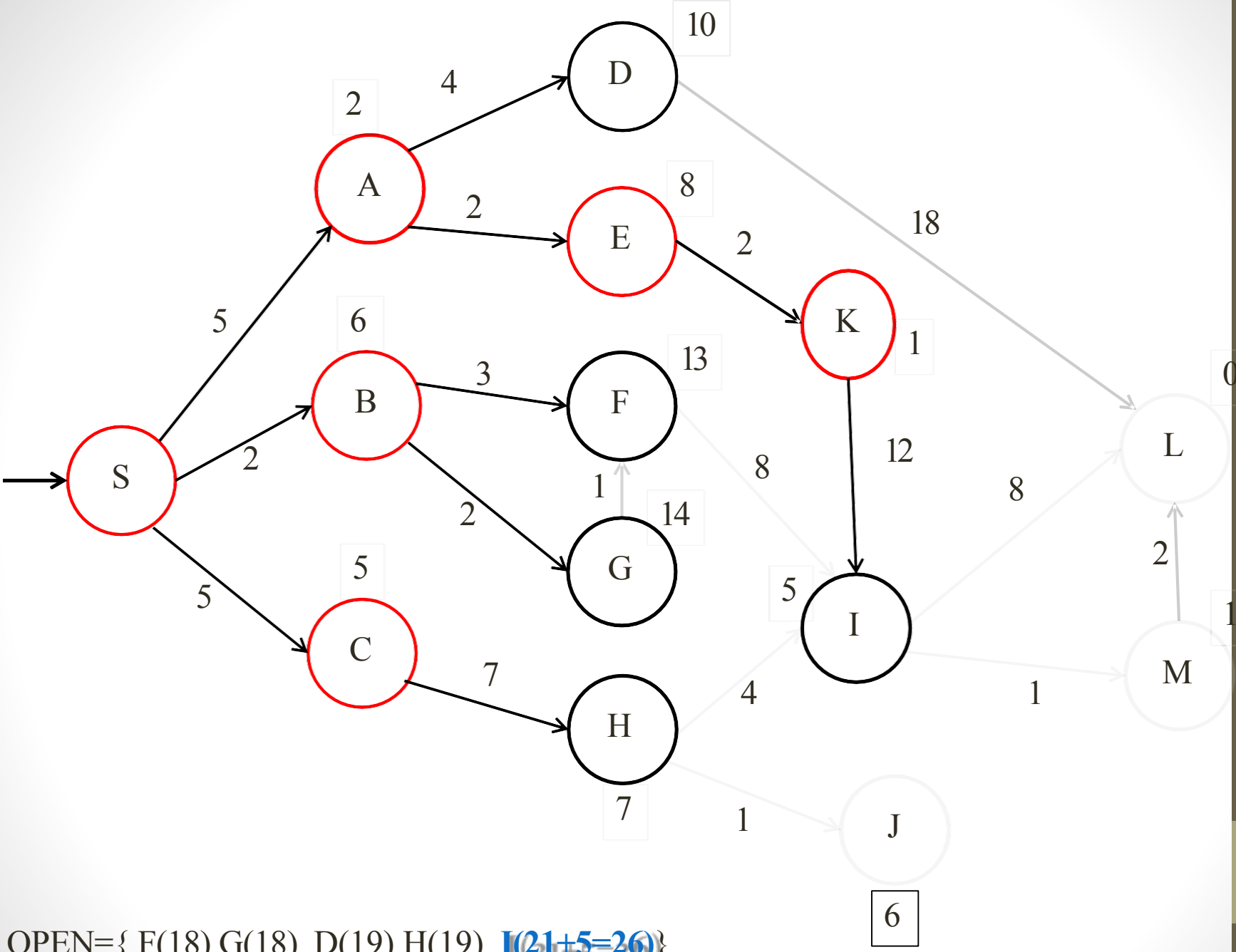
OPEN={C(10),E(15), **F(5+13=18),G(4+14=18),** D(19)}
CLOSED={S,A,**B**}



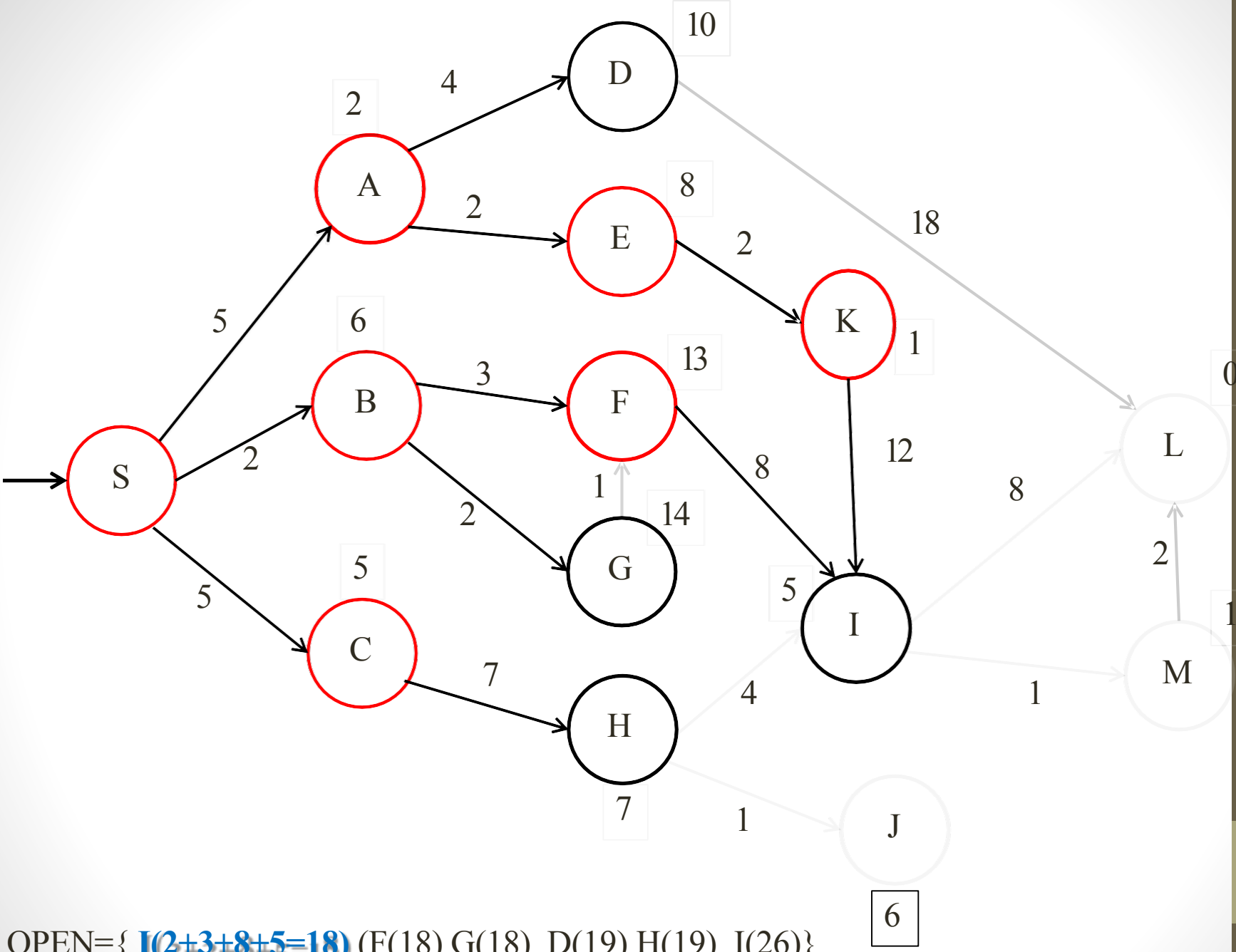
OPEN={E(15), F(18),G(18), D(19),H(5+7+7=19)}
CLOSED={S,A,B,C}



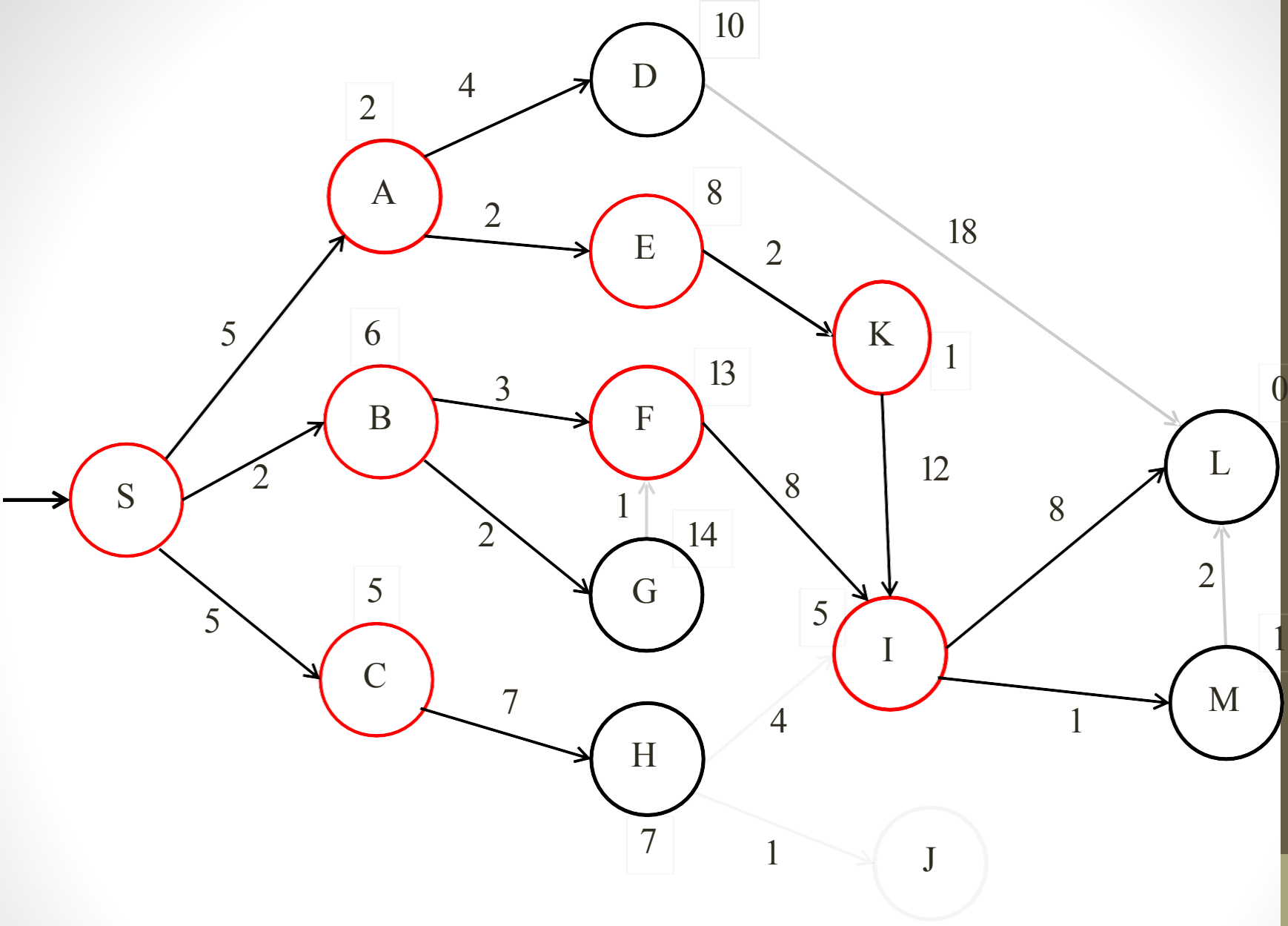
OPEN={K(5+2+2+1=10), F(18),G(18), D(19),H(19)}
CLOSED={S,A,B,C,E}



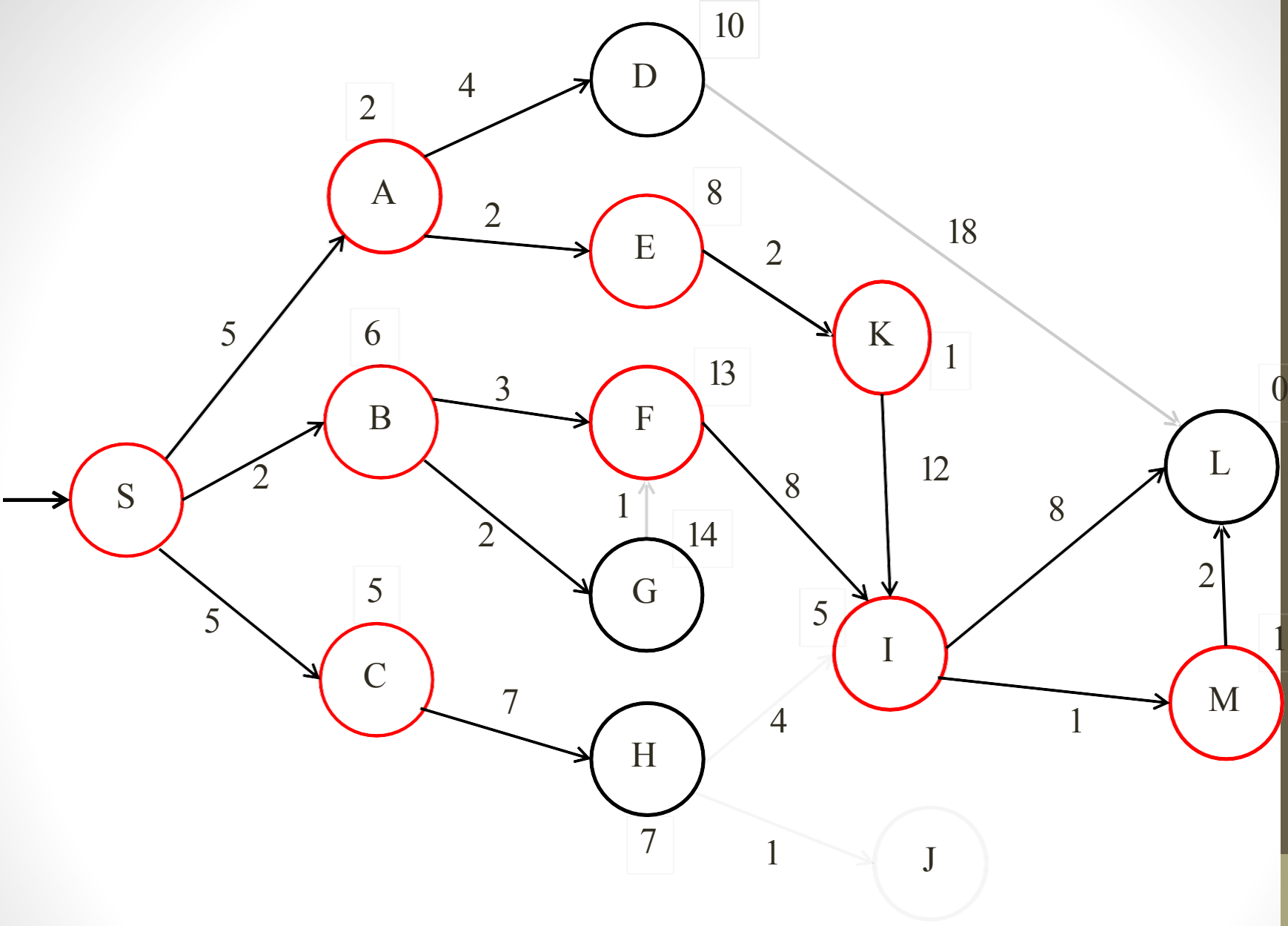
OPEN={ F(18),G(18), D(19),H(19), I(21+5=26)}
CLOSED={S,A,B,C,E,K}



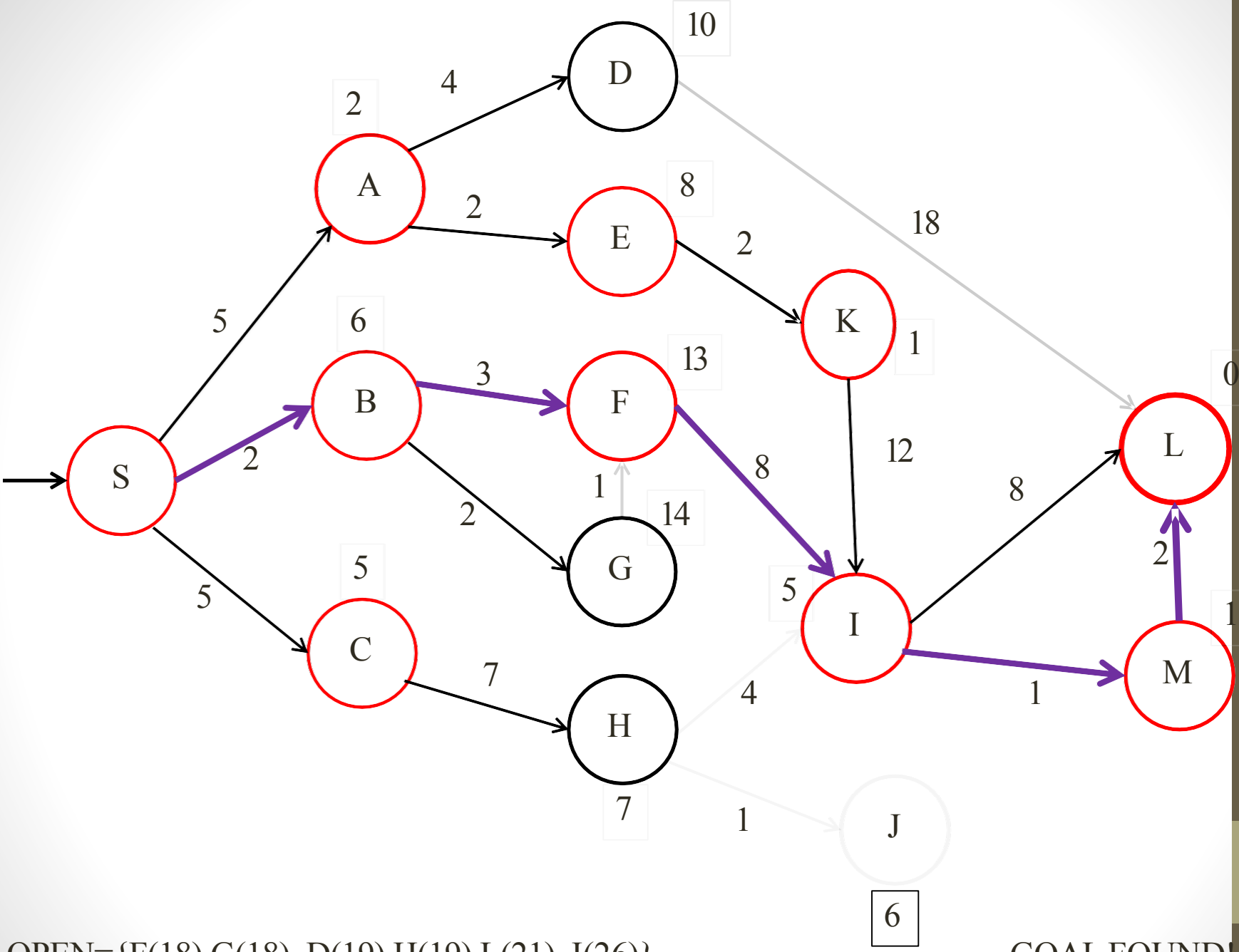
OPEN={ I(2+3+8+5=18),(F(18),G(18), D(19),H(19), I(26)}
CLOSED={S,A,B,C,E,K,F}



OPEN={ M(2+3+8+1+1=15),F(18),G(18), D(19),H(19),L(21+0=21) ,I(26)}
CLOSED={S,A,B,C,E,K,F,I}



OPEN={ L(2+3+8+1+2+0=16), F(18),G(18), D(19),H(19),L(21) ,I(26)}
CLOSED={S,A,B,C,E,K,F,I,M}

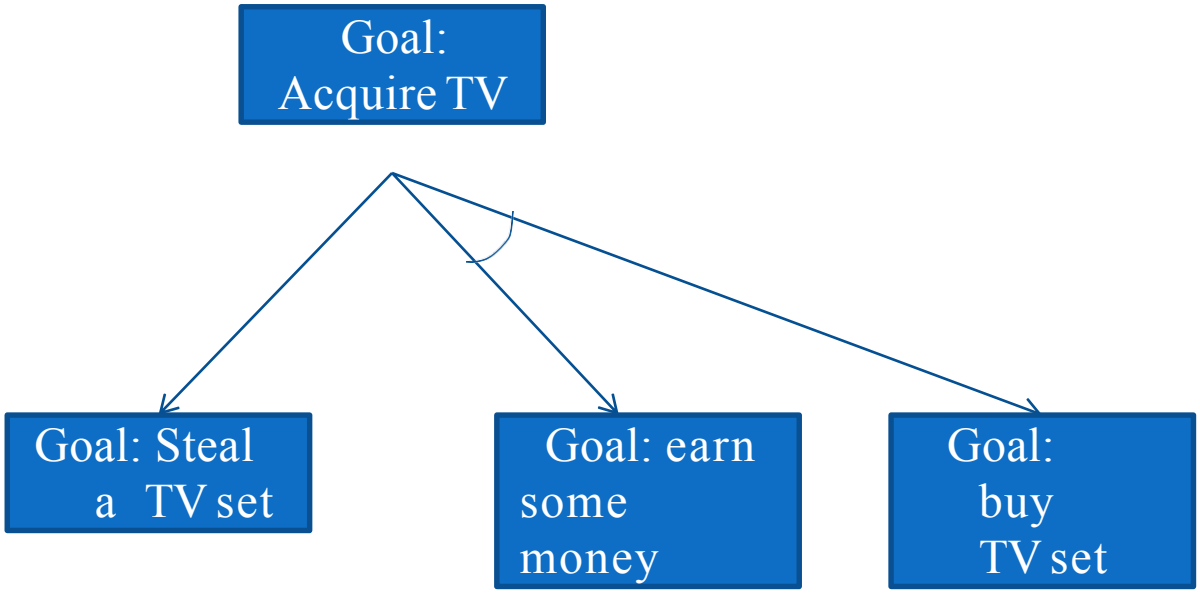


OPEN={F(18),G(18), D(19),H(19),L(21) ,I(26)}

CLOSED={S,A,B,C,E,K,F,I,M,L}

GOAL FOUND!!

Problem Reduction and Decomposition (AND-OR Graphs)



Constraint Satisfaction Problem

Each state contains:

Variables X1,X2,X3,……Xn

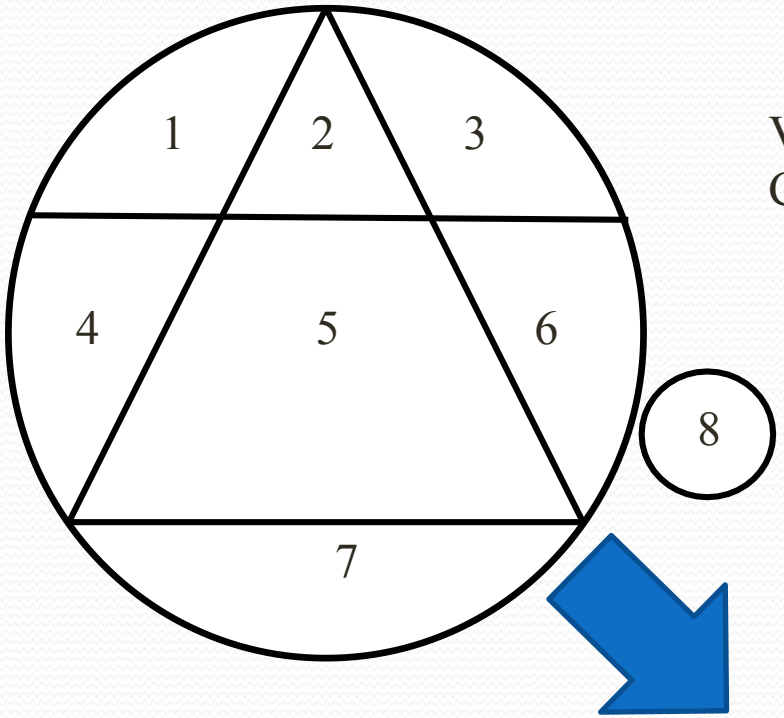
Constraints C1,C2,…..Cn

Variables have to be assigned with values V1,V2,…..Vn

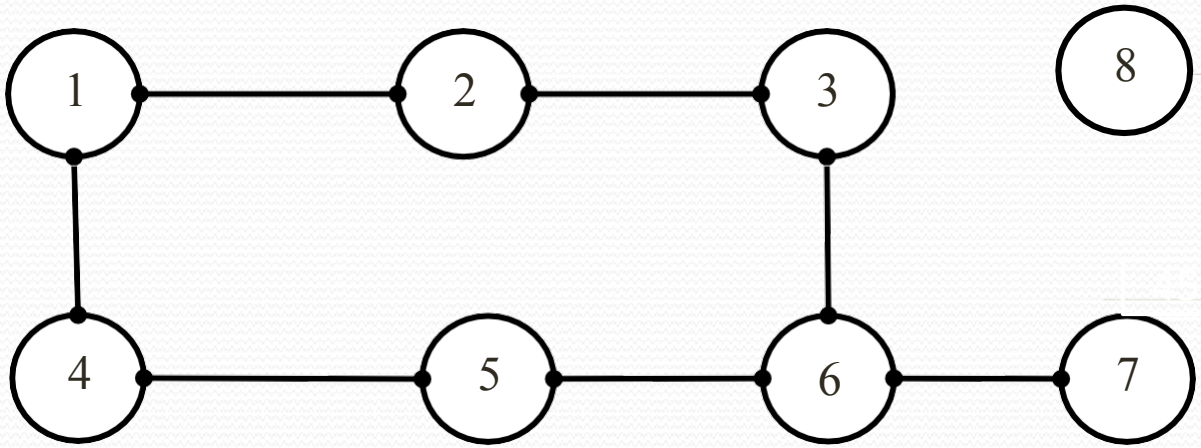
Such that none of the constraints are violated.

Goal state- one in which all variables are assigned resp. values and those values do not violate any constraint

Example graph colouring problem



Variables: X1,X2,…….X7
Constraints: {red, green, blue}

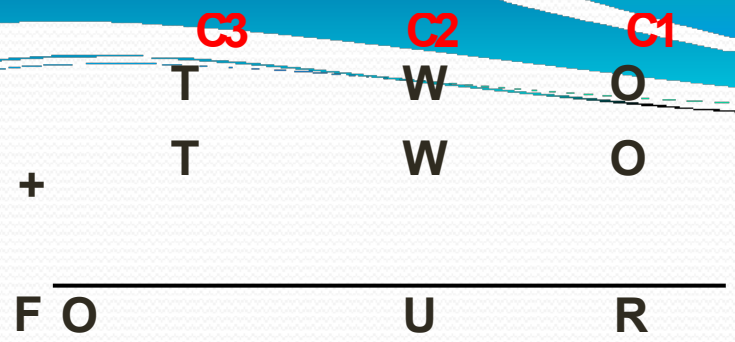


Crypt Arithmetic

Constraints:

1. Variables: can take values from 0-9
2. No two variables should take same value
3. The values should be selected such a way that it should comply with arithmetic properties.

	T	W	O
+	T	W	O
F	O	U	R



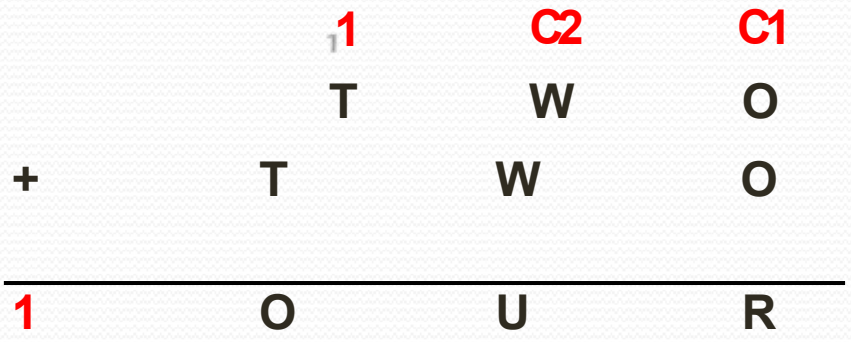
STEP 1:

C3 =1 since 2 single digit number s plus a carry cannot be more than 19 thus,

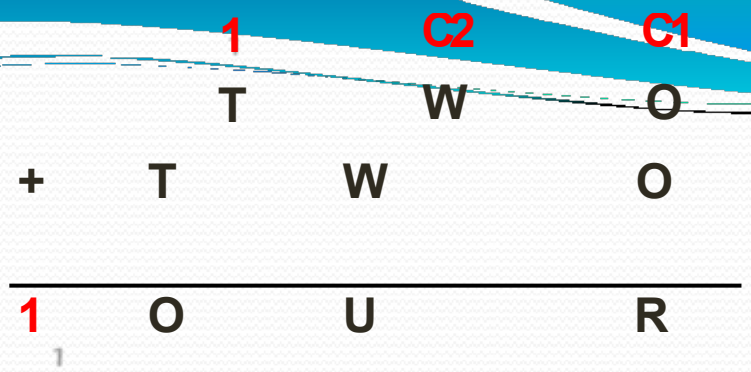
C3=1

F=1

Thus,



28/Jan/2021



STEP 2: **T+T+C2 > 9** because only then it can generate carry. C2 can be **0 or 1**, depending on: if previous column is generating carry or not.

Assume: C2=1 **Then, 2T+1>9 So, 2T>8 hence T>4**

Thus, T can take value from 4,5,6,…9

Assume: T=5

STEP 3:

	1	1	C1
	5	W	O
+	5	W	O
1	O	U	R

BUT , if T=5 , **T+T+C2=11 which means O=1 !!! CONSTRAINT VIOLATED as F=1.**

GOBACK TO STEP 2 AND ASSUME DIFFERENT VALUE FOR T

We know , T can take value from **5**,6,…9

Assume: T=6

STEP 3:

	1	1	C1
	5	W	O
+	5	W	O
1	O	U	R

BUT , if T=5 , **T+T+C2=11 which means O=1 !!! CONSTRAINT VIOLATED as F=1.**

GOBACK TO STEP 2 AND ASSUME DIFFERENT VALUE FOR T

We know , T can take value from **5**,6,…9

Assume: T=6

	1	C2	C1	
	6	W	O	
+	6	W	O	
1	O	U	R	

STEP 4: T+T+C2 > 13 so,

O=3 Accepted till now

	1	C2	C1	
	6	W	3	
+	6	W	3	
1	3	U	R	

O+O =R so, Since O=3, R=6 !!! VIOLATION as T=6
Hence T=6 cant generate Solution.

STEP 5: Assume: T=7

	1	C2	C1
	7	W	O
+	7	W	O
1	O	U	R

T+T+C2= 7+7+1=15 Thus, O=5

	1	C2	C1
	7	W	5
+	7	W	5
1	5	U	R

O+O =R so, Since O=5, R= 0 and C1= 1

O=5 R=0 C1=1

	1	C2	1
	7	W	5
+	7	W	5
1	5	U	0

STEP 6: We have middle Column left i.e.

W+W+C1=U

Since C1 =1 W+W must be >9 [to generate carry]

W>=5 To generate carry C2

W can take values 5,6,7,…9

STEP 7: Assume: W=5

Since W+W+C1=U if W=5 then,

5+5+1= 11 **Thus U=1 !!! VIOLATION as F=1 thus W Cannot be 5 Repeat step 7.**

STEP 8: Assume: W=6

Since W+W+C1=U if W=6 then,

6+6+1= 13 **Thus U=3 which is Accepted**

U=3

THUS AT THIS STATE SINCE ALL THE VARIABLES HAVE BEEN ASSIGNED VALUES WHICH COMPLY WITH CONSTRAINTS GIVEN, WE HAVE REACHED FINAL STATE!!

	1 1	1		1
	7	6		5
+	7	6		5
1	5	3		0

C R O S S

R O A D S

D A N G E R