Search Techniques in AI

Prof. Amey D.S.Kerkar
Computer Engineering Department,
Don Bosco College of Engineering
Fatorda-Goa.

Control strategies

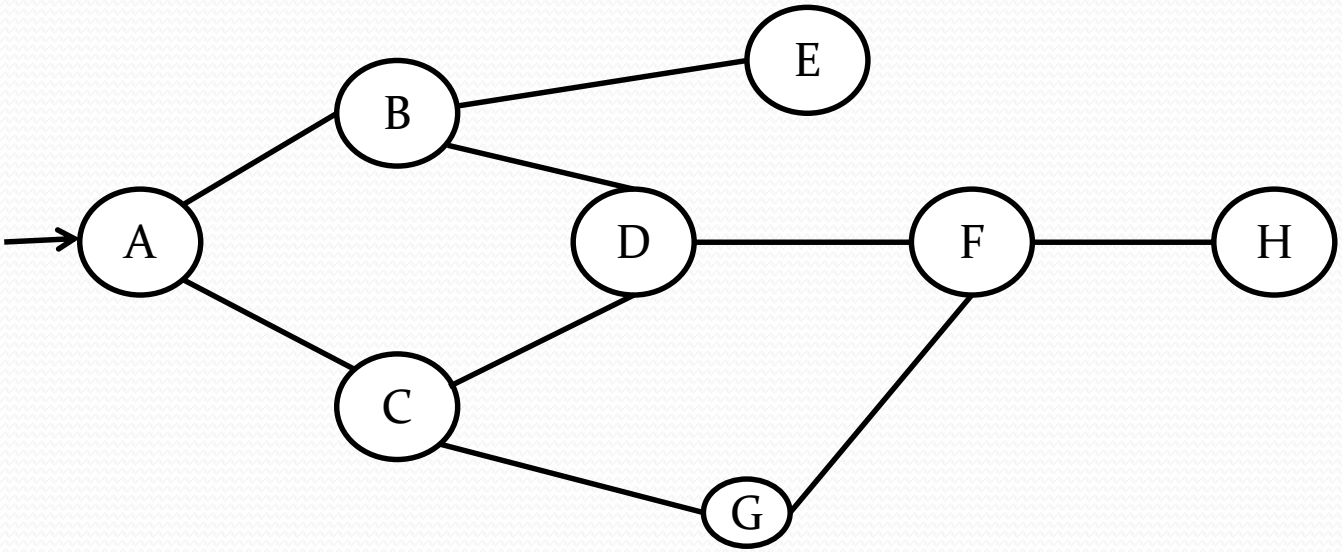
- Helps us decide which rule to apply next.
- What to do when there are more than 1 matching rules?

- Good control strategy should:
    1. cause motion
    2. Systematic

Control strategies are classified as:

1. Uninformed/blind search control strategy

- ✓ Do not have additional info about states beyond problem def.
- ✓ Total search space is looked for solution
- ✓ No info is used to determine preference of one child over other.
- ✓ Example: 1. Breadth First Search(BFS), Depth First Search(DFS), Depth Limited Search (DLS).



State Space without any extra information associated with each state

2. Informed/Directed Search Control Strategy

- ✓ Some info about problem space(heuristic) is used to compute preference among the children for exploration and expansion.
- ✓ Examples: 1. Best First Search, 2. Problem Decomposition, A*, Mean end Analysis
- ✓ Heuristic function:
- ✓ It maps each state to a numerical value which depicts goodness of a node.

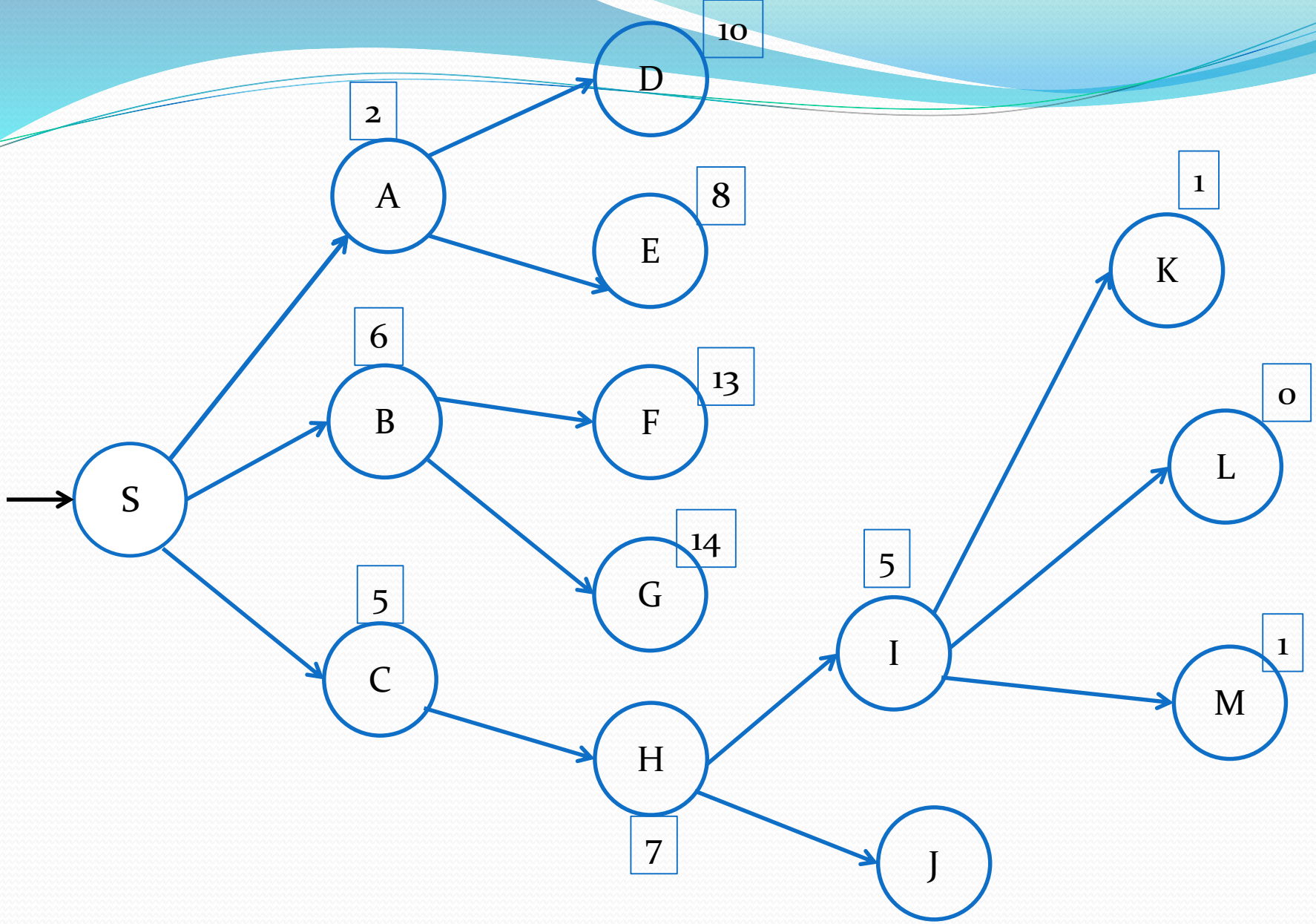
$$H(n)=value$$

Where ,

H() is a heuristic function and ‘n’ is the current state.

- Ex: in travelling salesperson problem heuristic value associated with each node(city) might reflect estimated distance of the current node from the goal node.

- The heuristic we use here is called HSLD Straight line Distance heuristic.

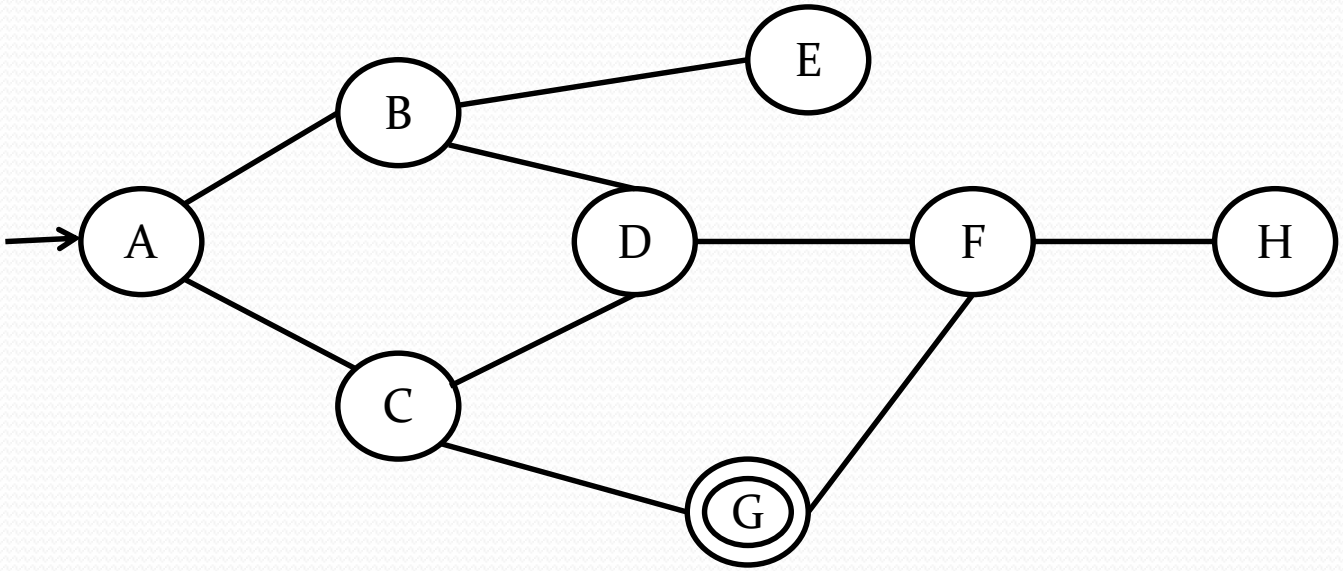


Example of the state space with heuristic values associated with each state

Breadth First Search (BFS)

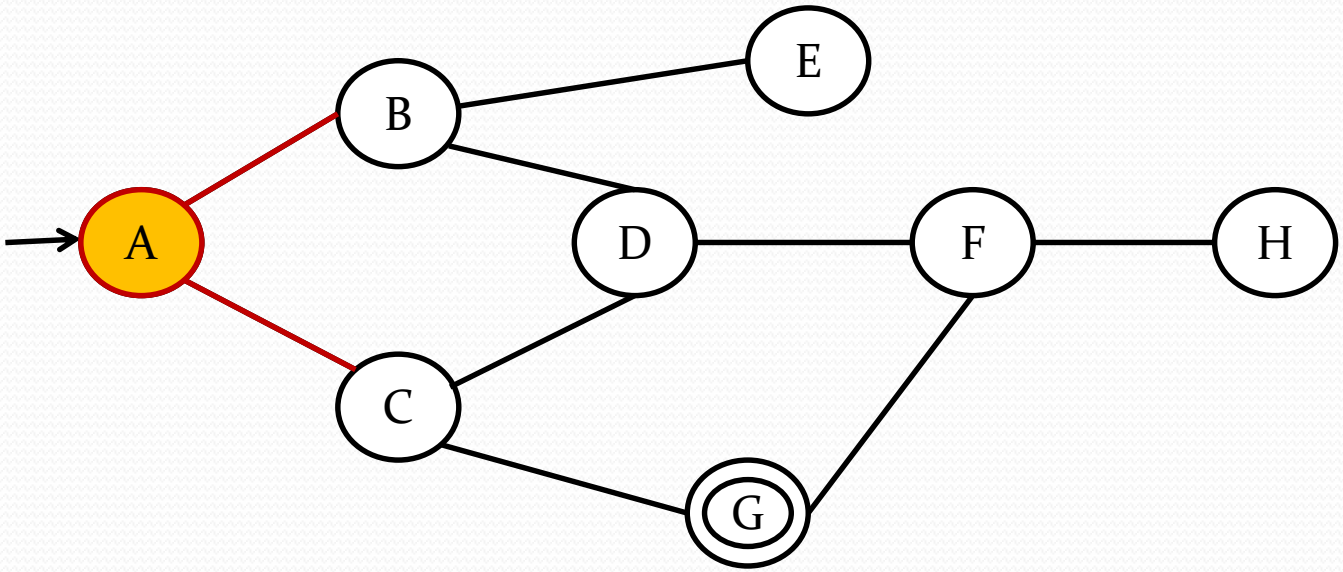
- Algorithm:
- 1. Create a variable NODE_LIST and set it to initial state.
- 2.Until a Goal State is found or NODE_LIST is empty:
  - A) Remove the first element from NODE_LIST amd call it as ‘E’. If the node list was empty then Quit.
  - B) For each way that each rule can match the state described in ‘E’ do:
    - i) Apply the rule to generate the new state
    - Ii) If the new state is a goal state, quit and return this state.
    - Iii) otherwise add the new state at the end of NODE_LIST.

- Consider the following State Space to be searched:

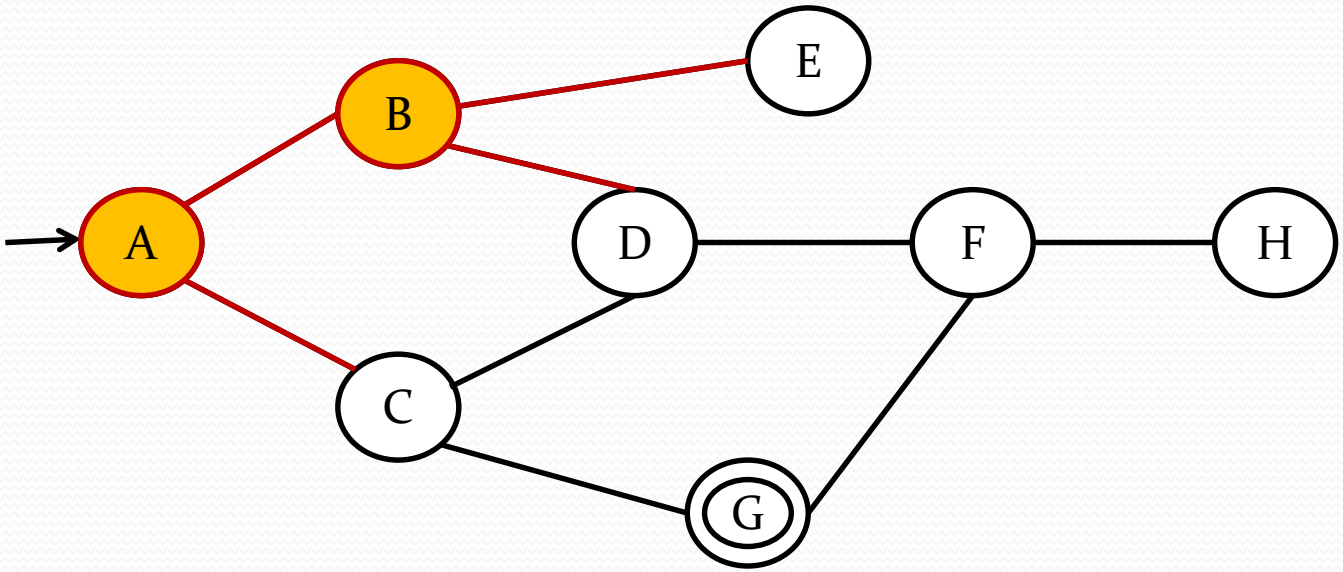


Let A be the start state and G be the final or goal state to be searched.

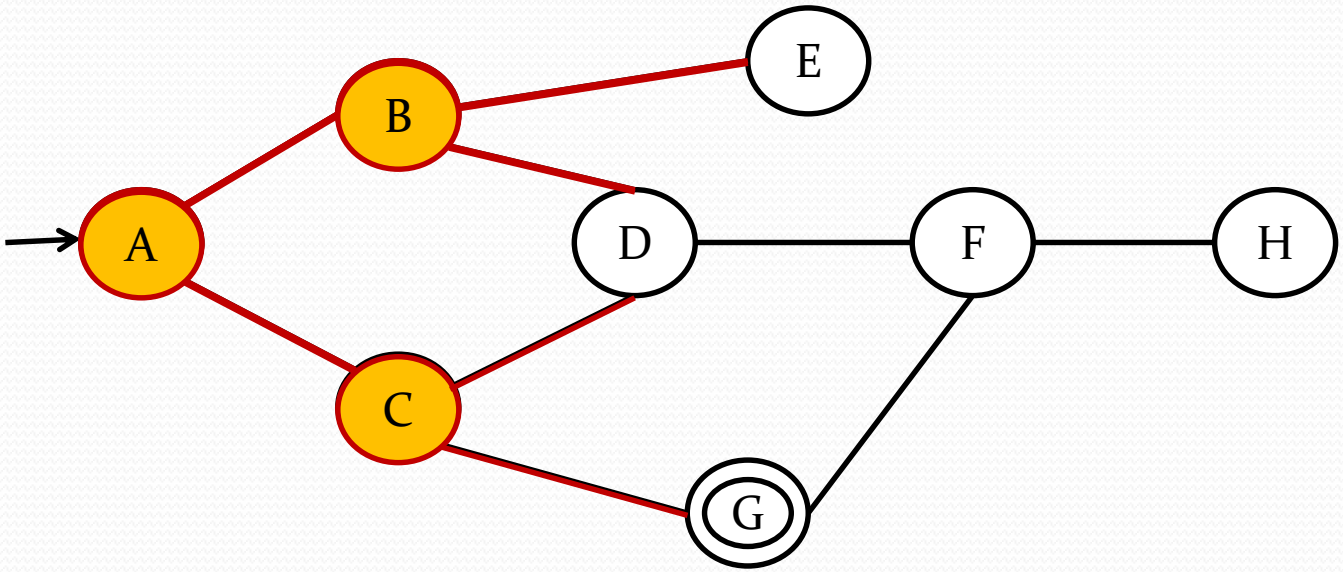
NODE_LIST={A} A is not goal node it is expanded .



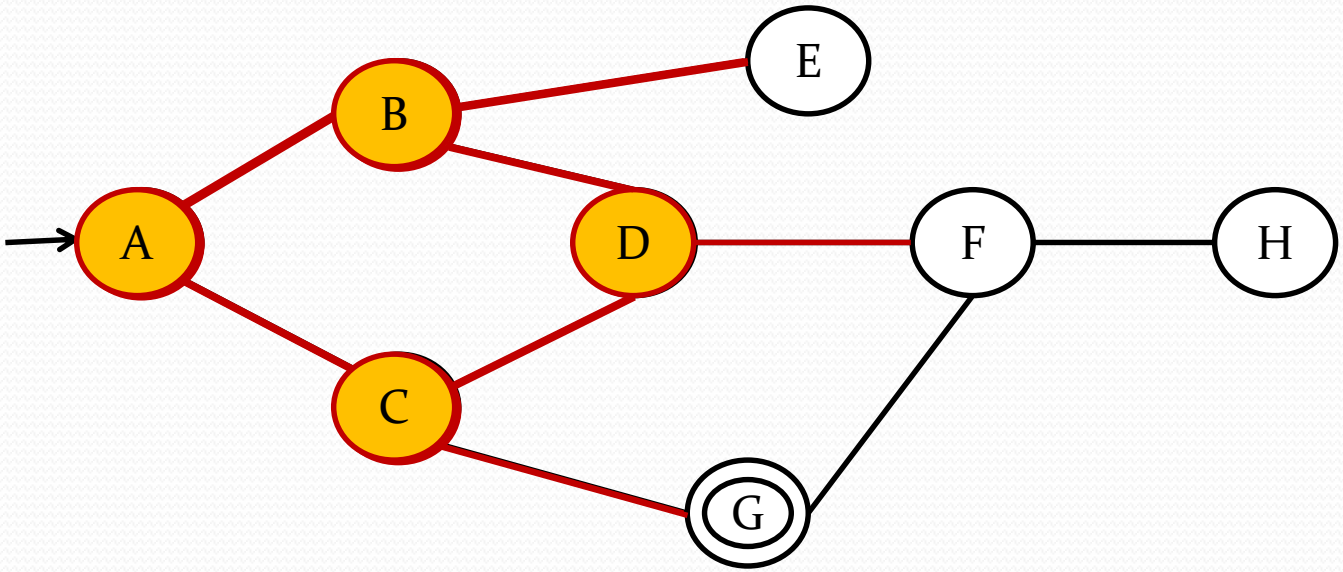
NODE_LIST={**B,C**}



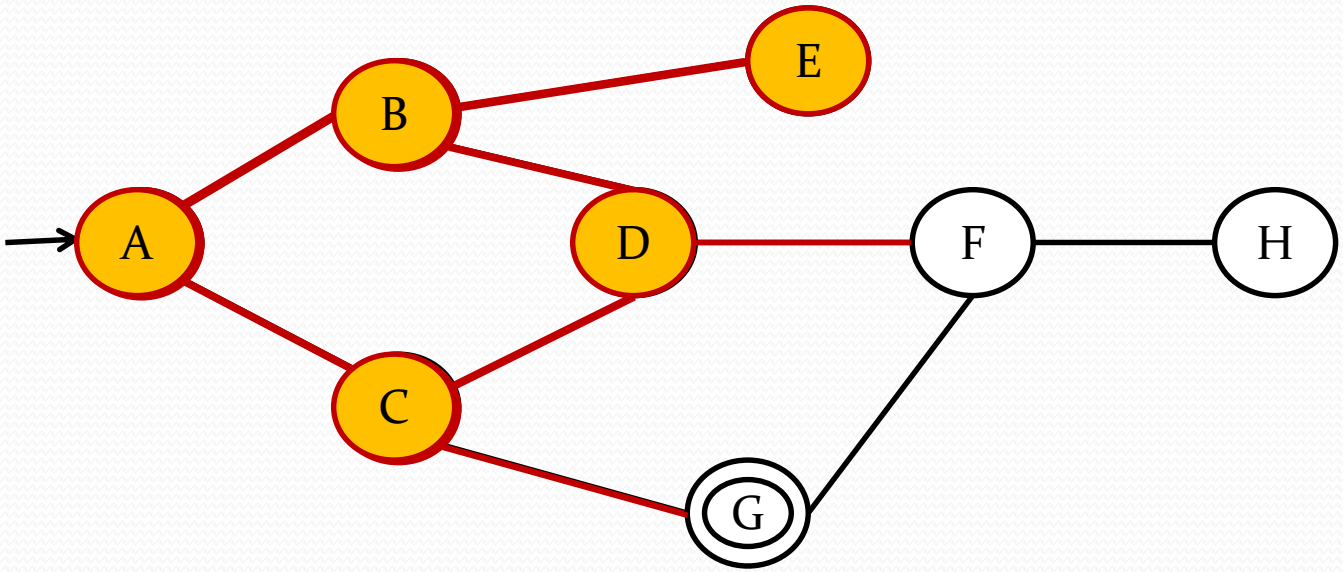
NODE_LIST={C,**D**,**E**}



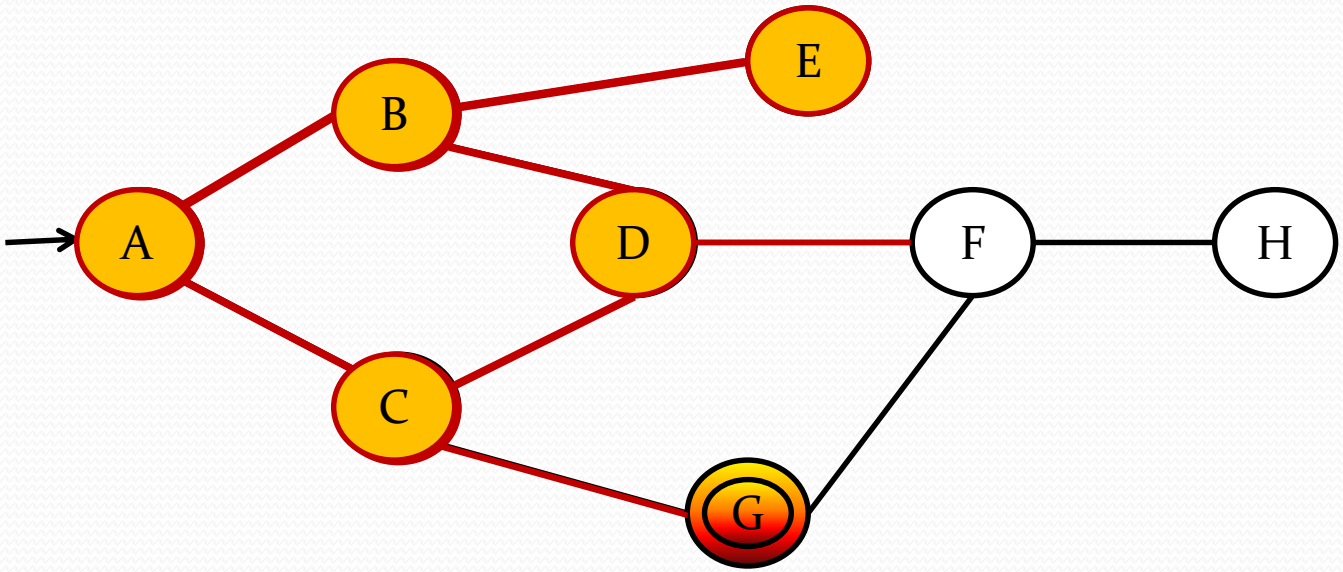
NODE_LIST={D,E,G}



NODE_LIST={E,G,F}

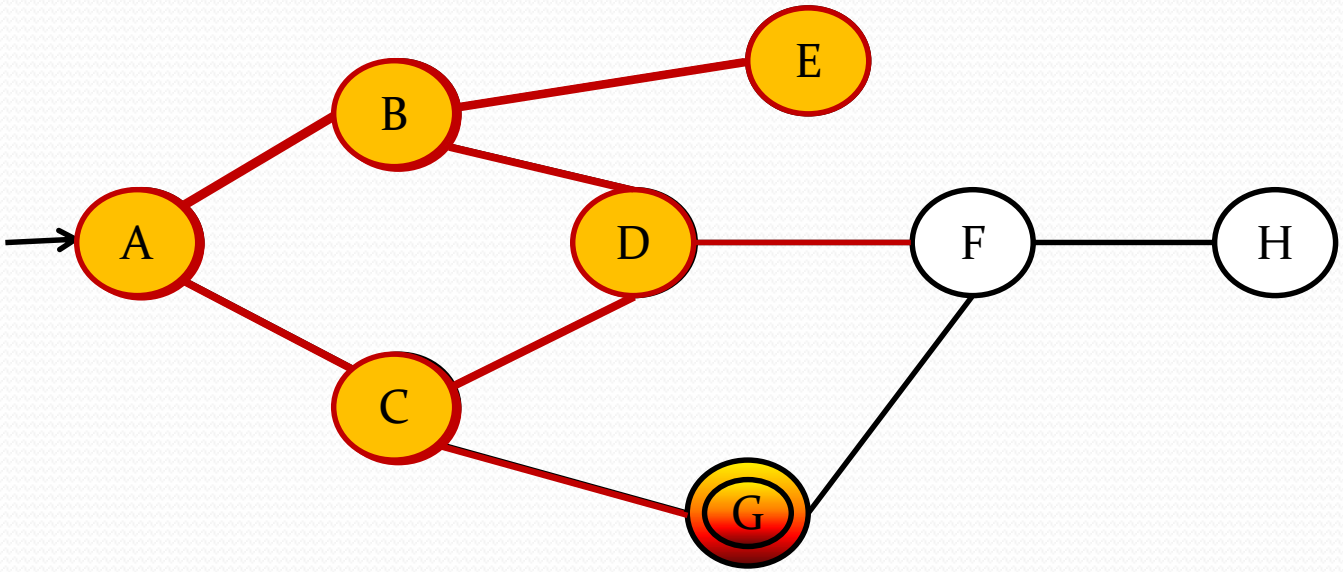


NODE_LIST={G,F}



NODE_LIST={**G**,F}

GOAL NODE FOUND!!



NODE_LIST={G,F}

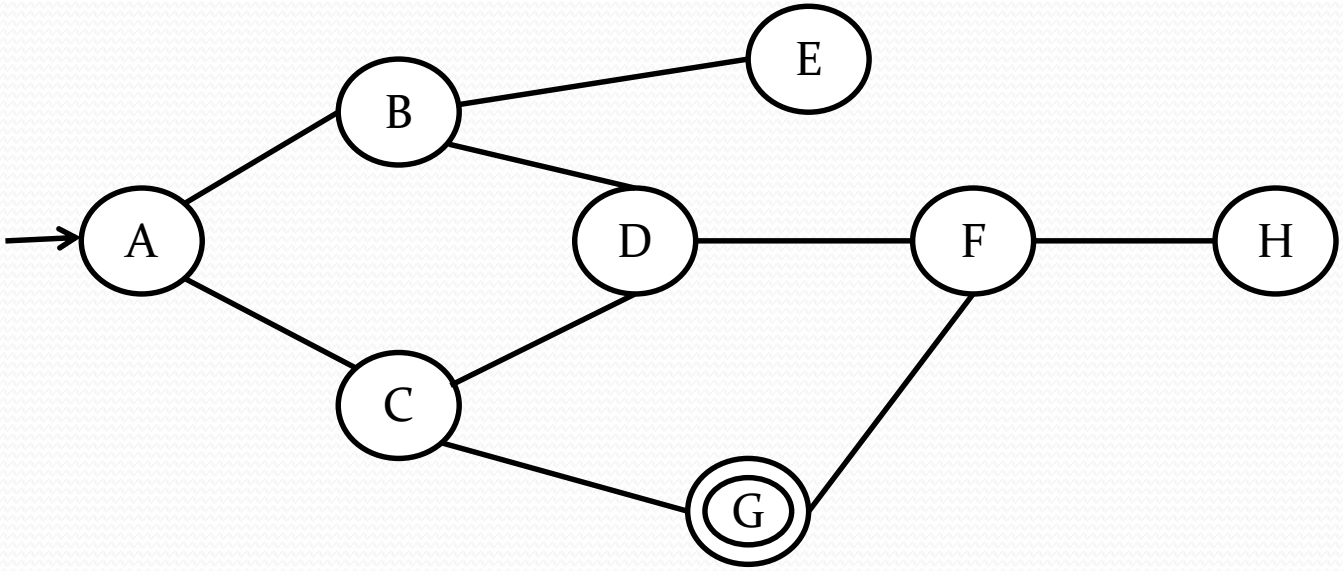
TRAVERSAL ORDER: A-B-C-D-E-G

Depth First Search

Algorithm:

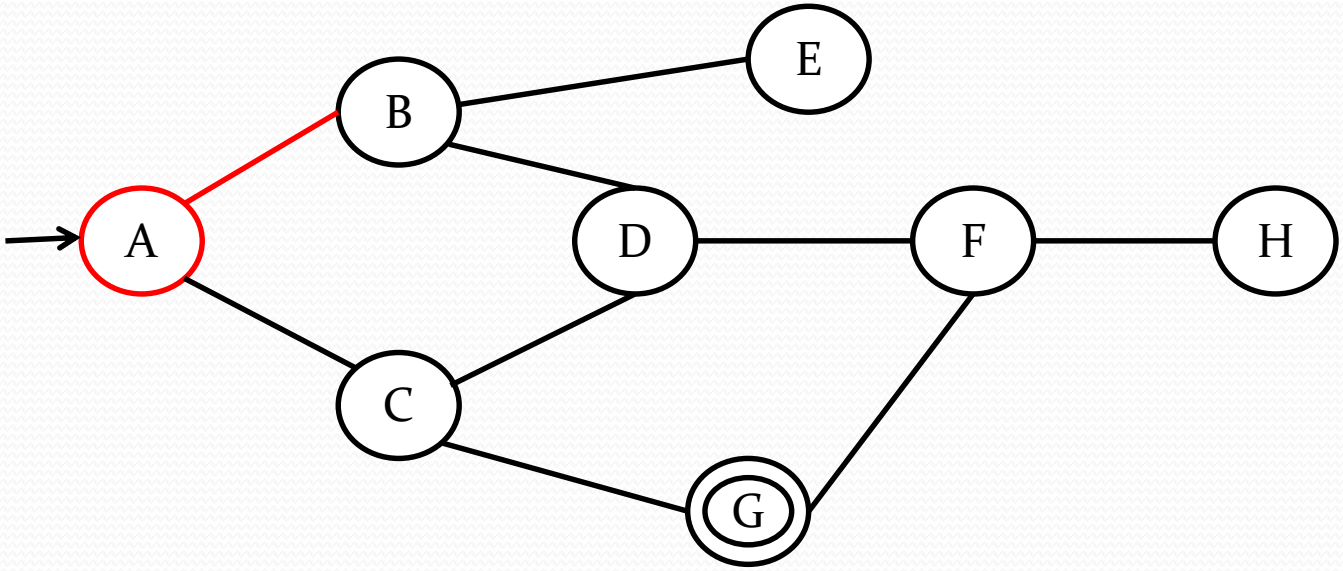
1) If initial state is a goal state, quit and return success.
2) Otherwise do the following until success or failure is reported:
   a. Generate successor ‘E’ of the initial state. If there are no more successors signal failure.
   b. Call Depth-First-Search with ‘E’ as he start state. If there are no more successors then , signal failure.
   c. If success is obtained, return success, otherwise continue in this loop.

Consider the following Search Space:



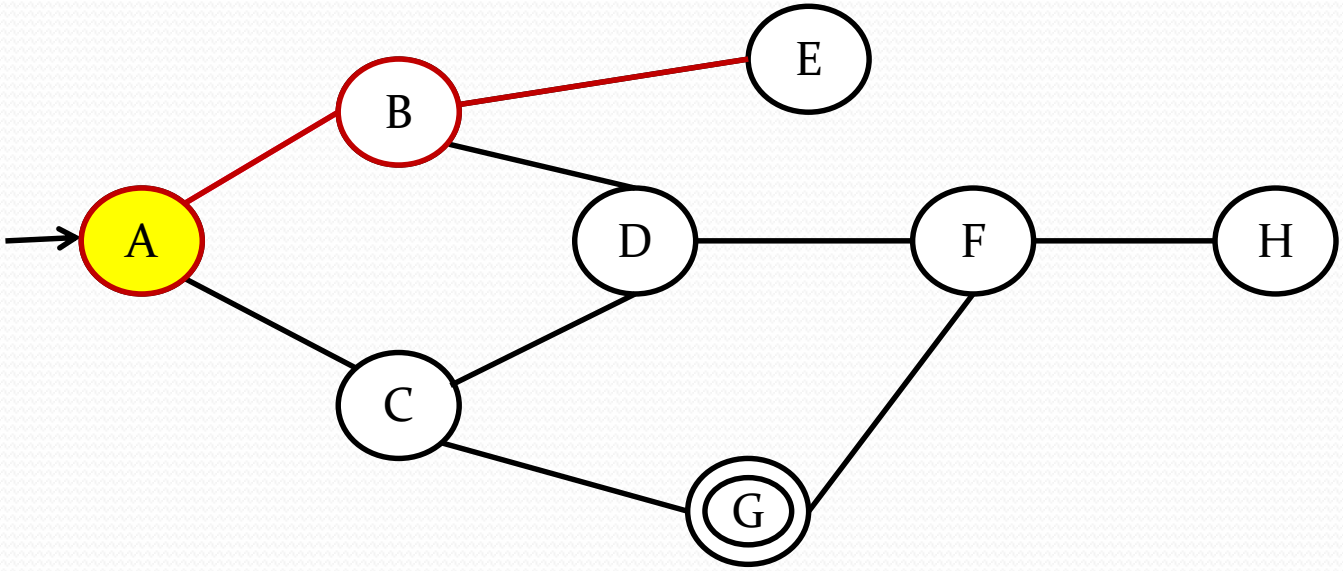
DFS(A)

Consider the following Search Space:



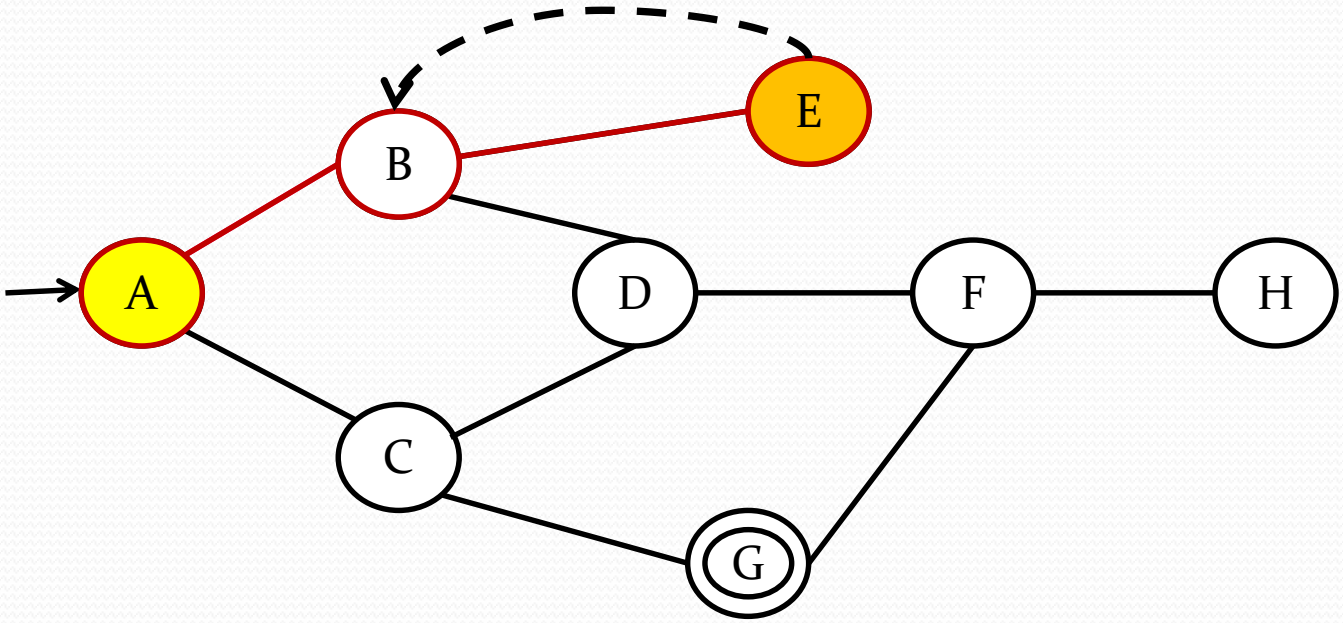
DFS(A)

Consider the following Search Space:



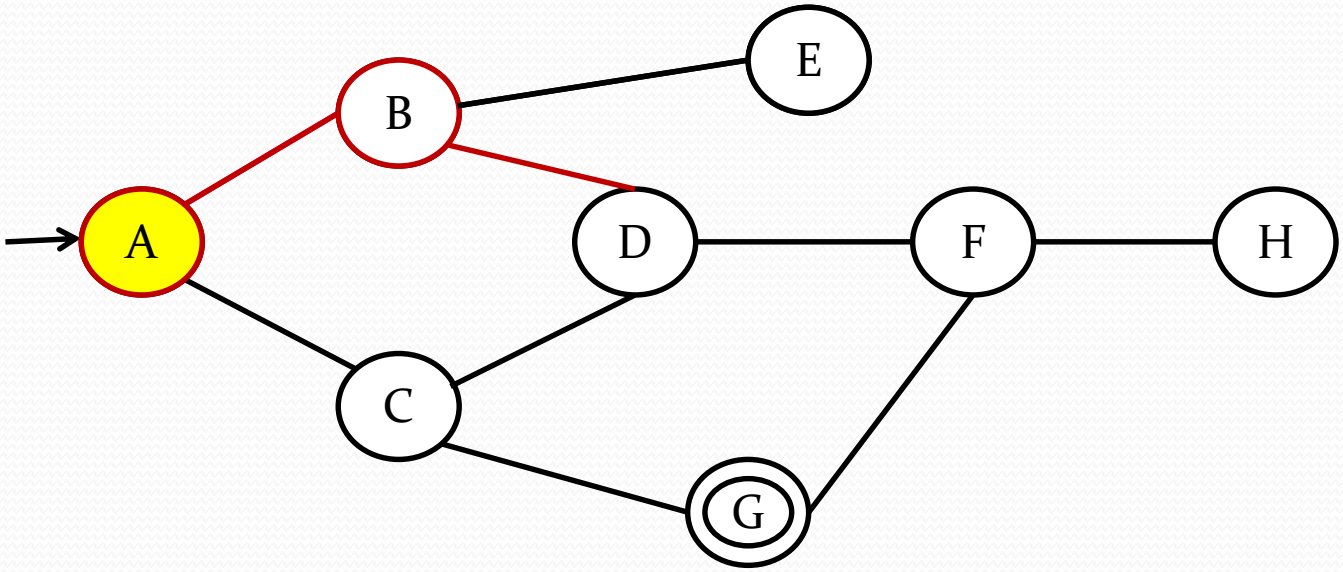
DFS(B)

Consider the following Search Space:



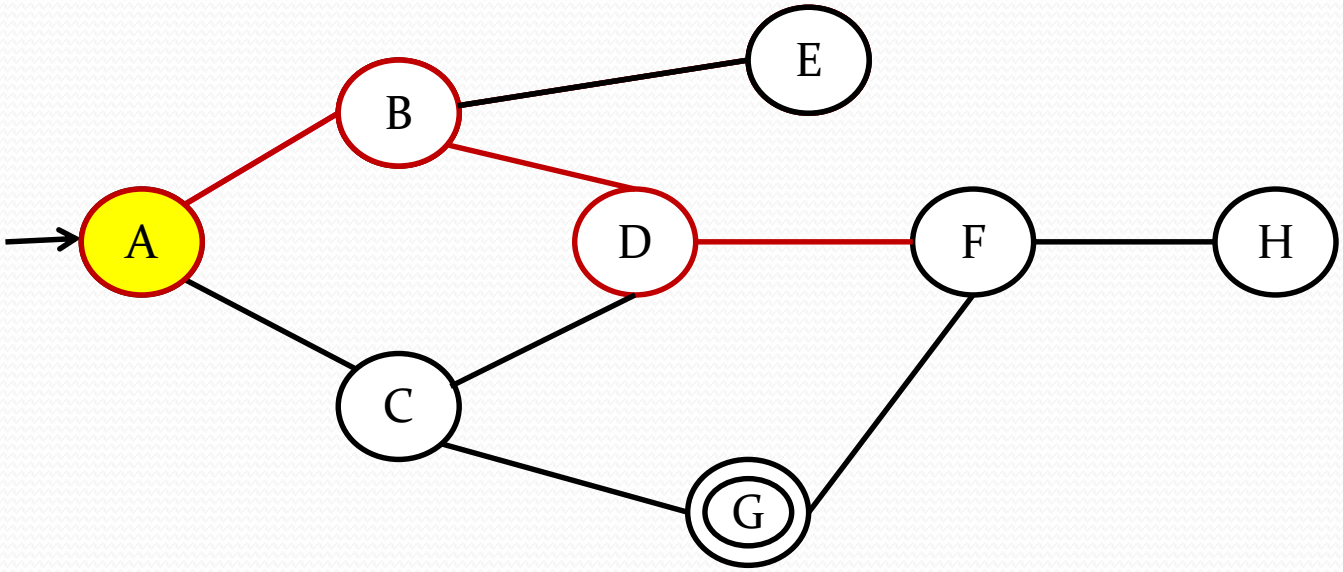
DFS(E)

Consider the following Search Space:



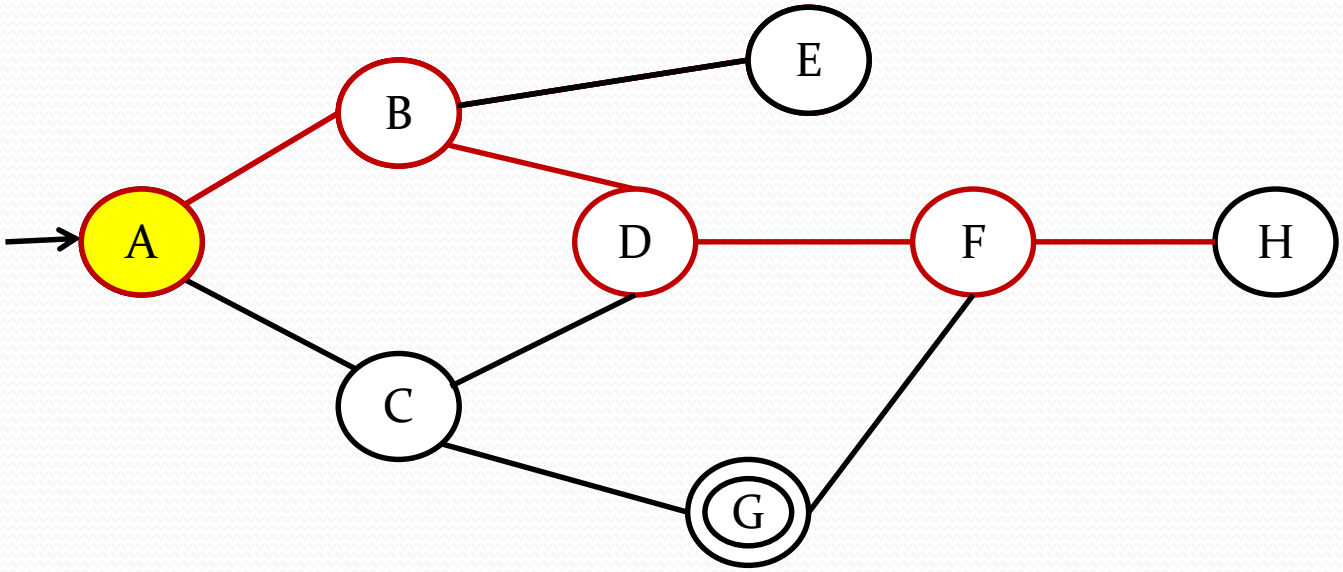
DFS(B)

Consider the following Search Space:



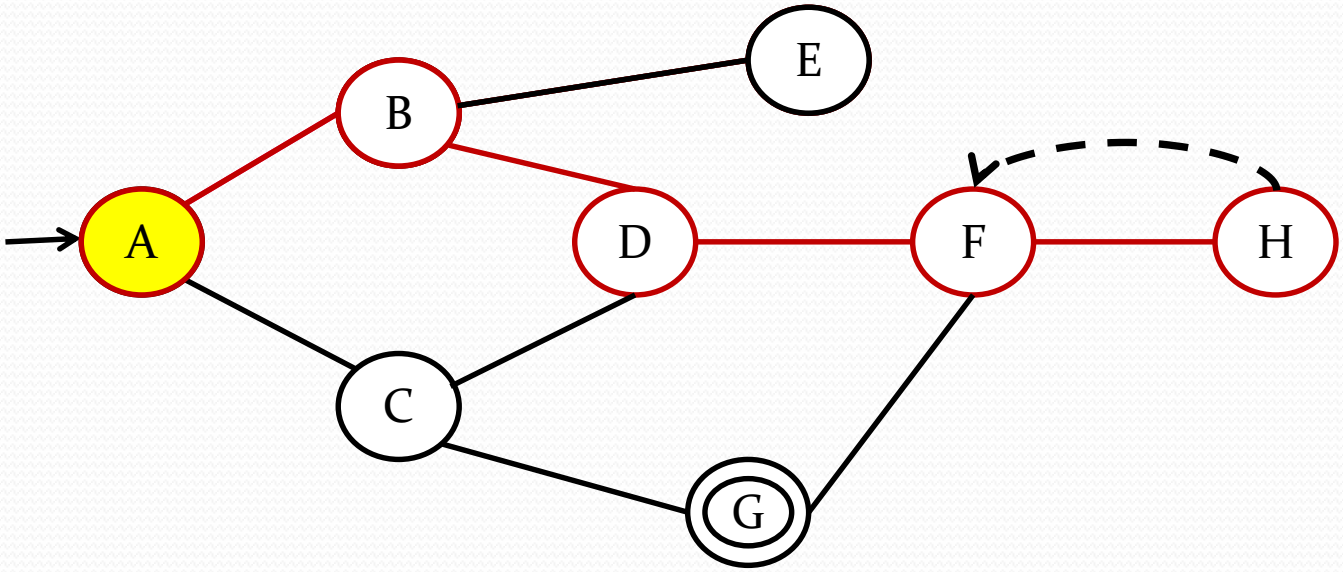
DFS(D)

Consider the following Search Space:



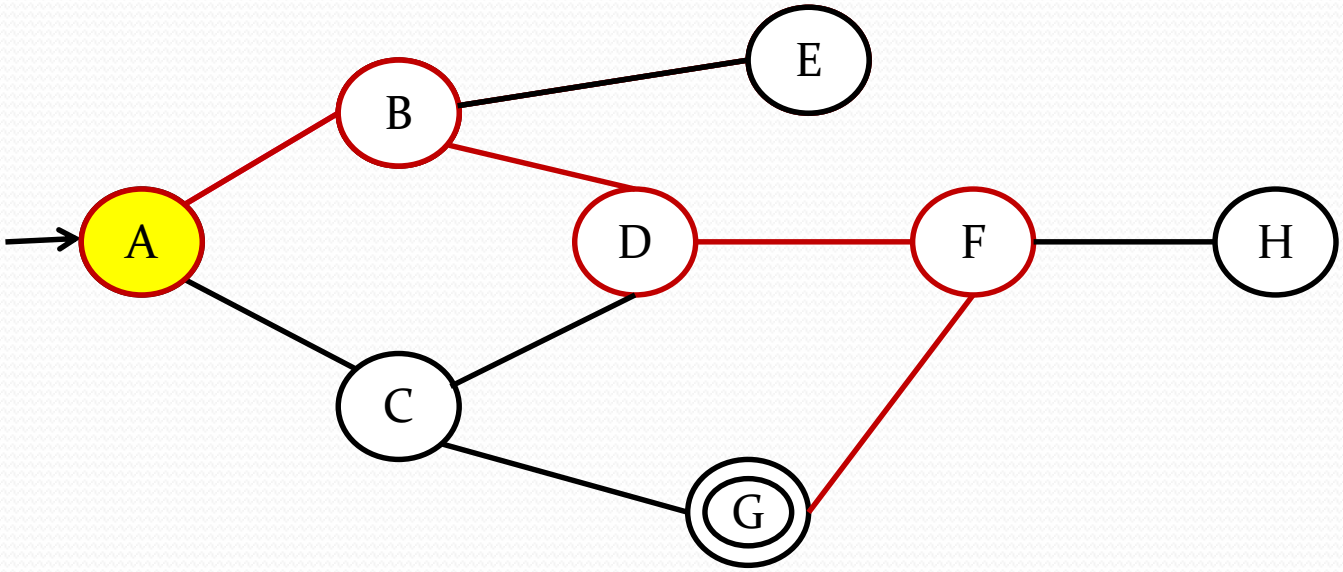
DFS(F)

Consider the following Search Space:



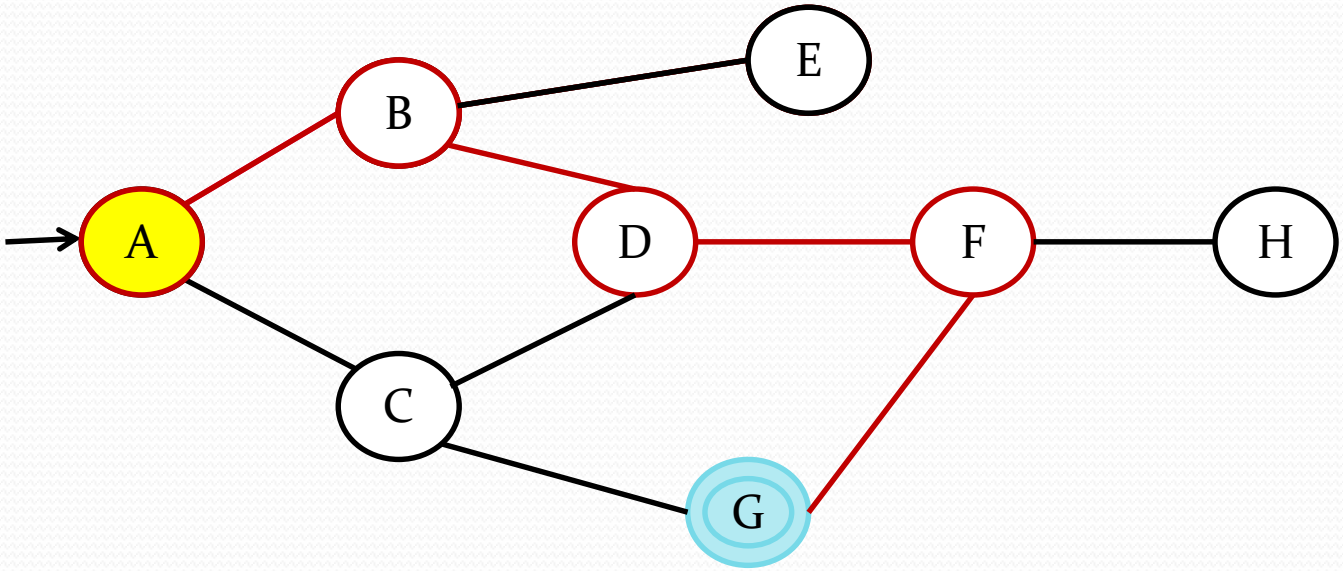
DFS(H)

Consider the following Search Space:



DFS(F)

Consider the following Search Space:



DFS(G)

GOAL NODE FOUND!!

Advantages of BFS:

1. BFS is a systematic search strategy- all nodes at level n are considered before going to n+1 th level.
2. If any solution exists then BFS guarentees to find it.
3. If there are many solutions , BFS will always find the shortest path solution.
4. Never gets trapped exploring a blind alley

Disadvantages of BFS:

1. All nodes are to be generated at any level. So even unwanted nodes are to be remembered. Memory wastage.
2. Time and space complexity is exponential type- Hurdle.

Advantages of DFS:

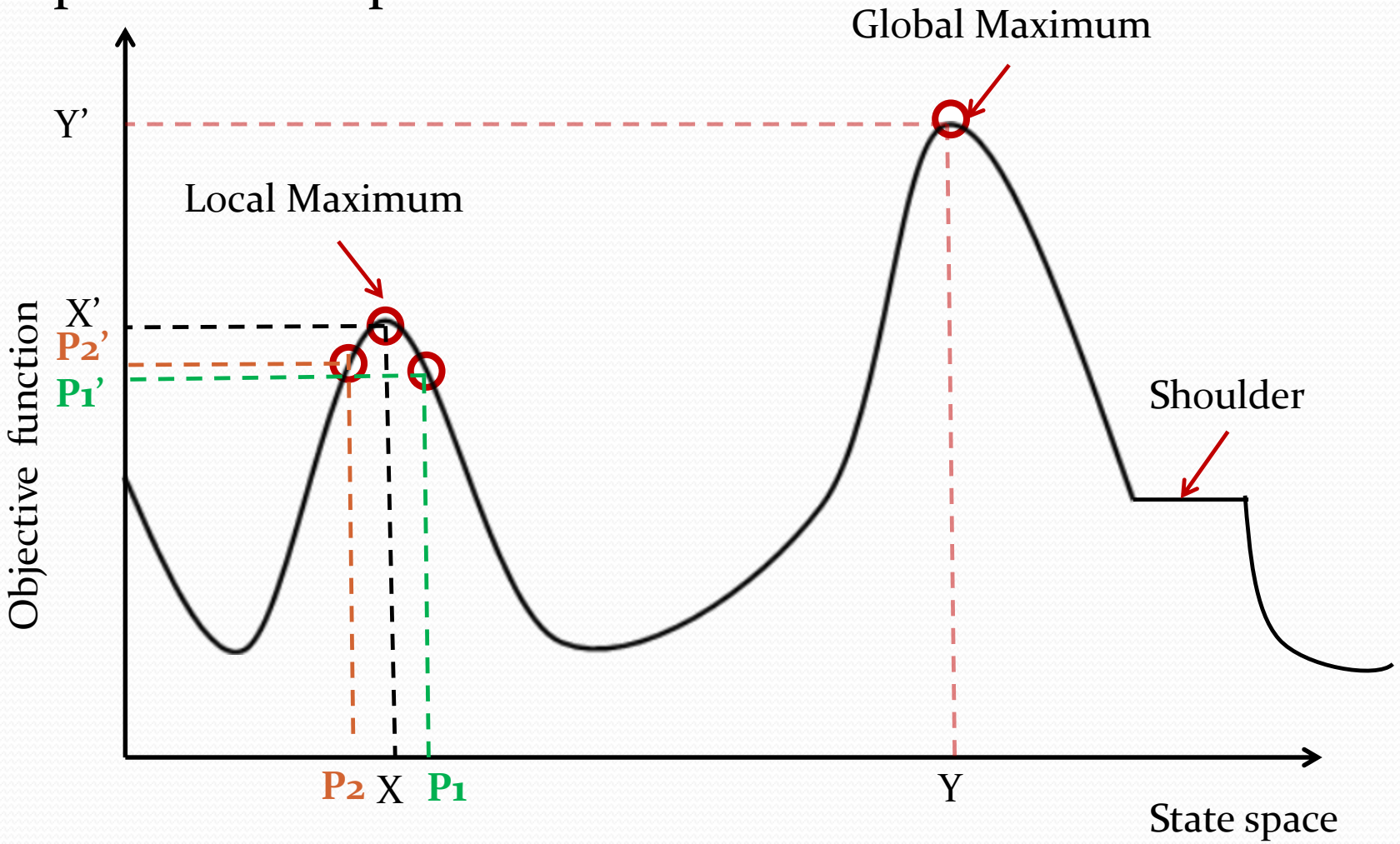
1. Memory requirements in DFS are less compared to BFS as only nodes on the current path are stored.
2. DFS may find a solution without examining much of the search space of all.

Disadvantages of BFS:

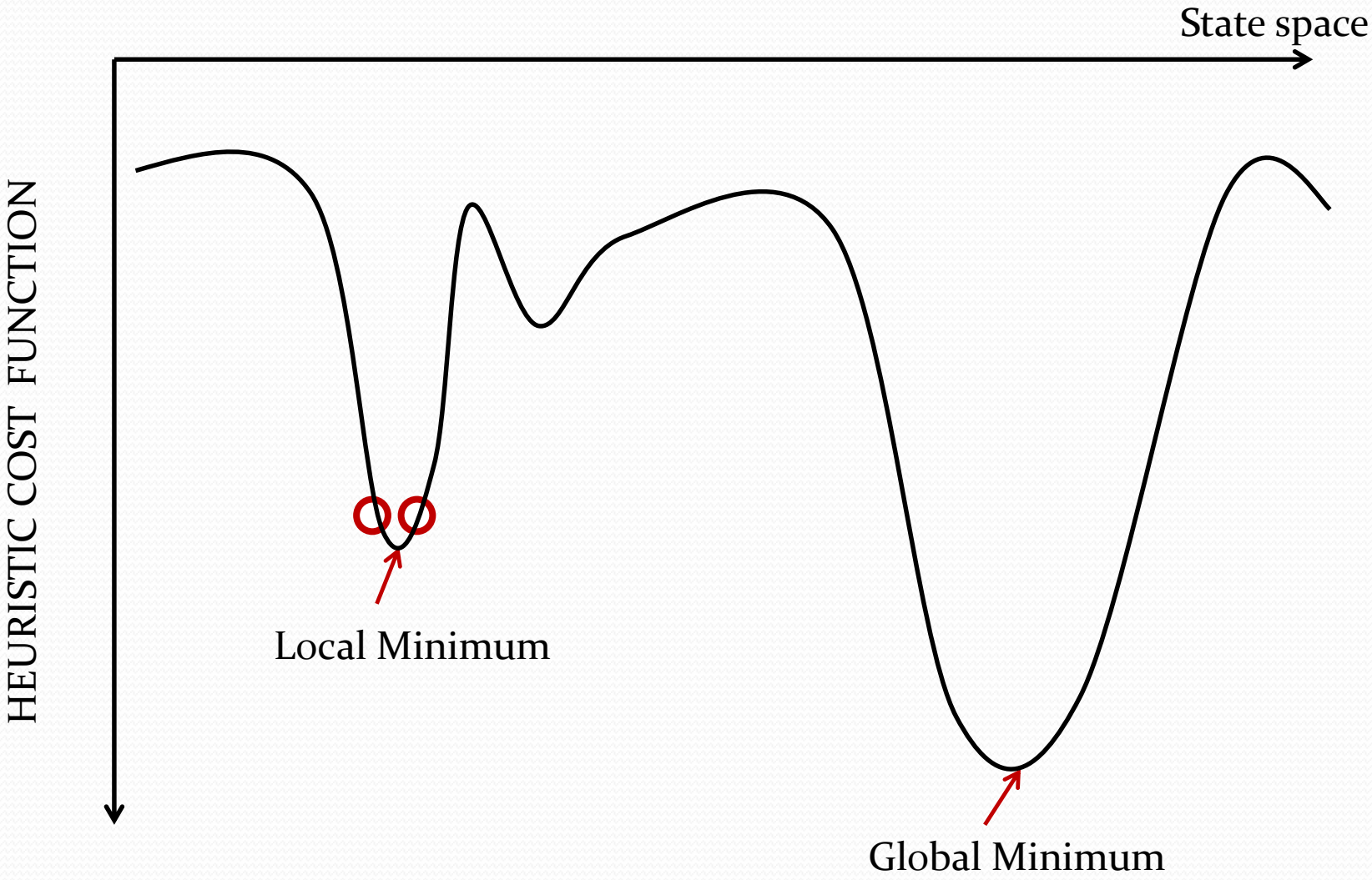
1. This search can go on deeper and deeper into the search space and thus can get lost. This is referred to as **blind alley**.

Hill Climbing Algorithm

- Local Search Algorithm
- State space landscape



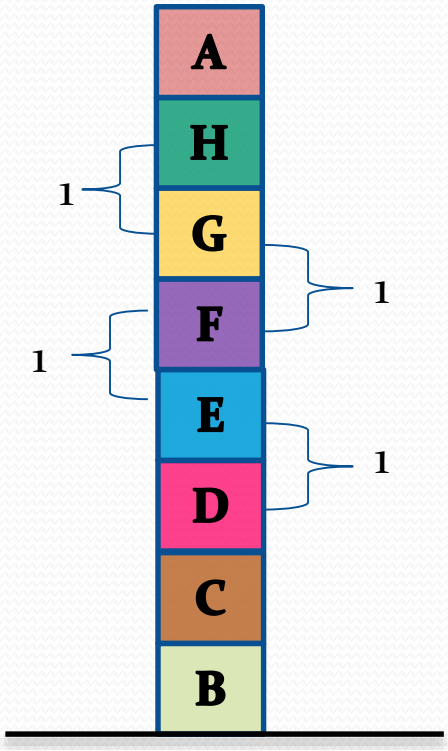
- Maximization function is called Objective Function
- Minimization function is called Heuristic Cost function
- Heuristic cost= distance, time, money spent
- Objective function= profit, success



Algorithm for Hill Climbing

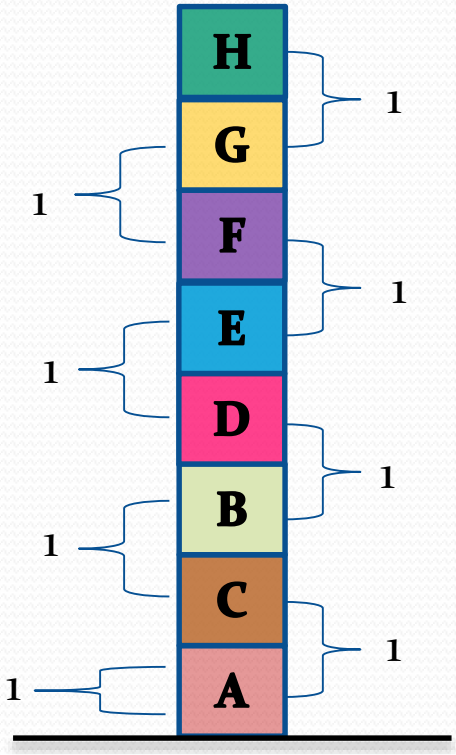
1. Evaluate the initial state. If it is the goal state then return and quit. Otherwise continue with initial state as current state.
2. Loop until a solution is found or until there are no new operators left to be applied to the current state:
    a. Select operator that has not been applied to the current 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.

- Example block world problem:



INITIAL STATE (P)

H(P)=4

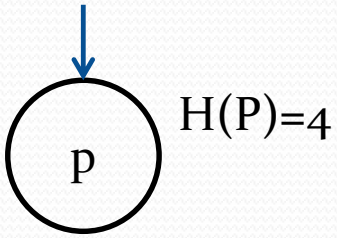


GOAL STATE (Z)

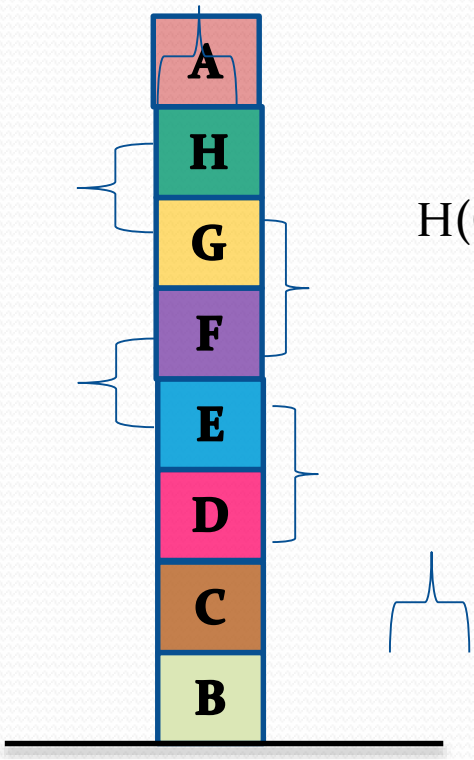
H(Z)=8

HEURISTIC FUNCTION- add one point for every block which is resting on the thing that it must be resting on.

- The tree generated till now is:



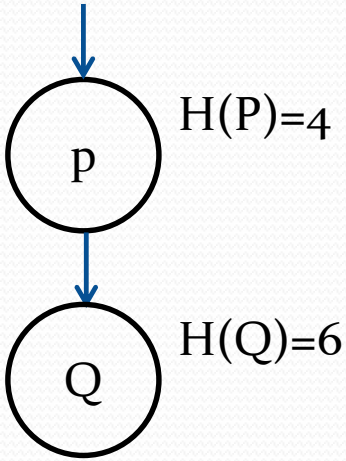
- From initial state most natural move is:



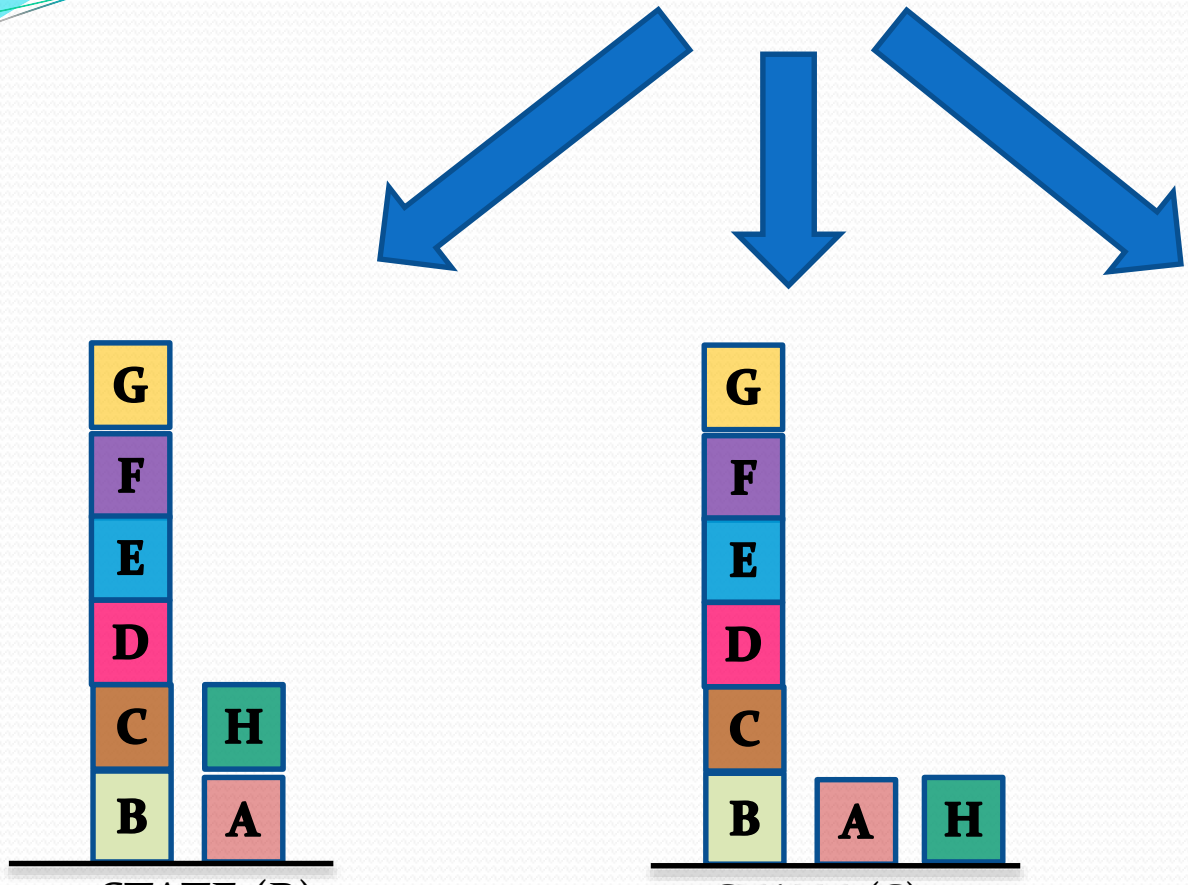
H(Q)= 6

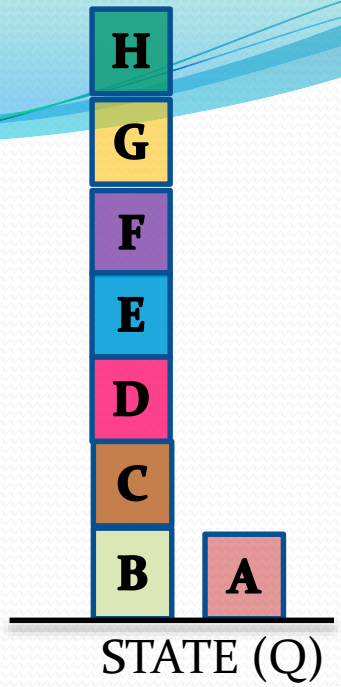
STATE (Q)

The tree Generated till Now is:

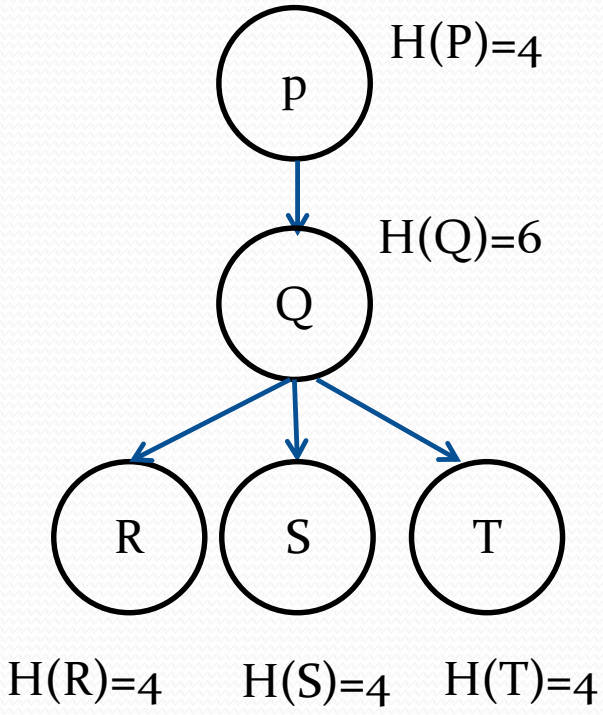


Now from Q there are three states possible which are as follows:





Thus we have the tree until now as:



We see that from state Q
When nodes R,S and T are
Generated, their value of
Heuristic is less compared
To heuristic value of Q itself.

Hill Climbing Algorithm
Has reached local maxima

Whereas in reality the actual
Goal node has heuristic
value of 8.

THIS IS A DISADVANTAGE OF LOCAL SEARCH ALGORITHM!!

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.

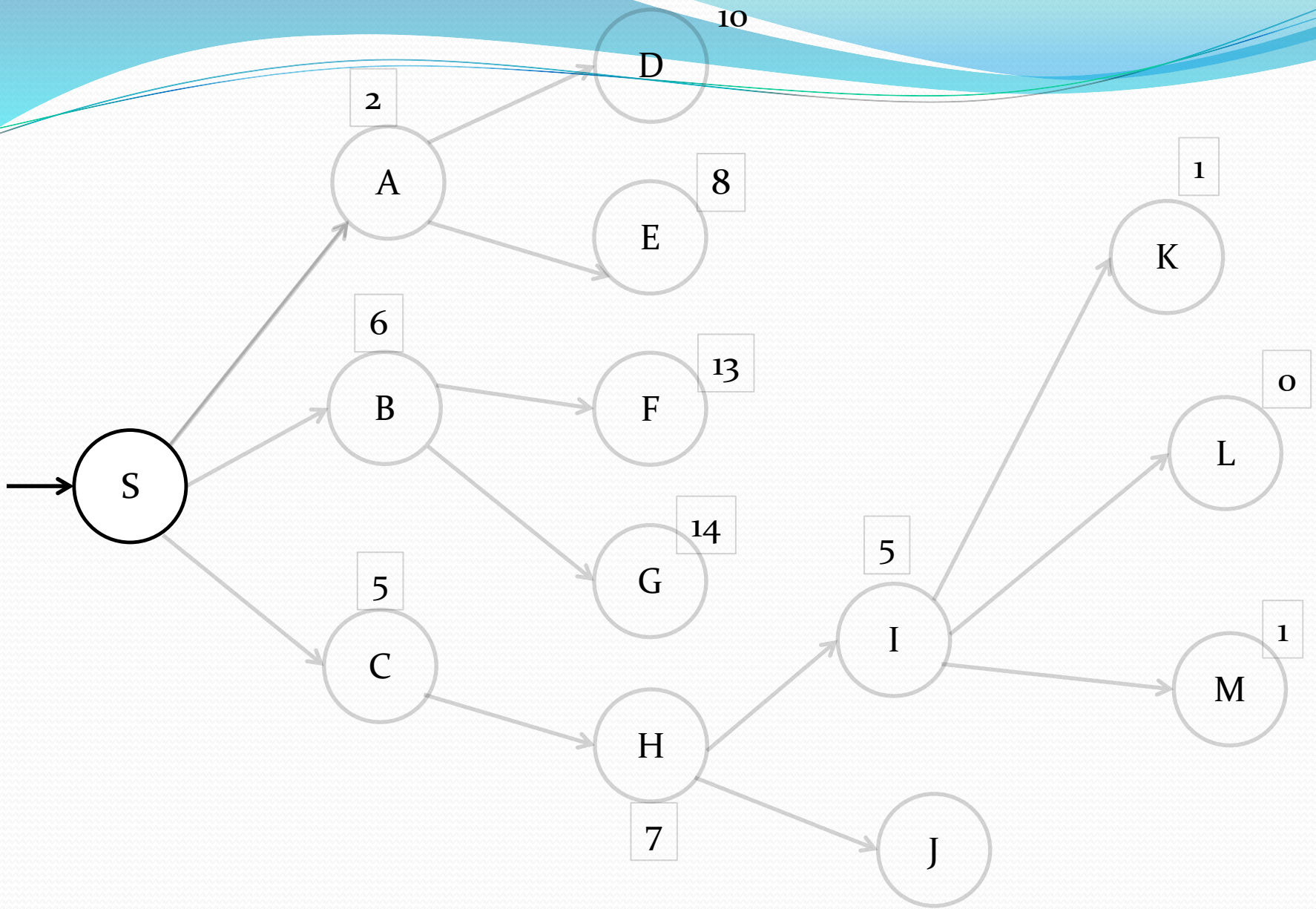
5. Mean End Analysis (MEA)

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

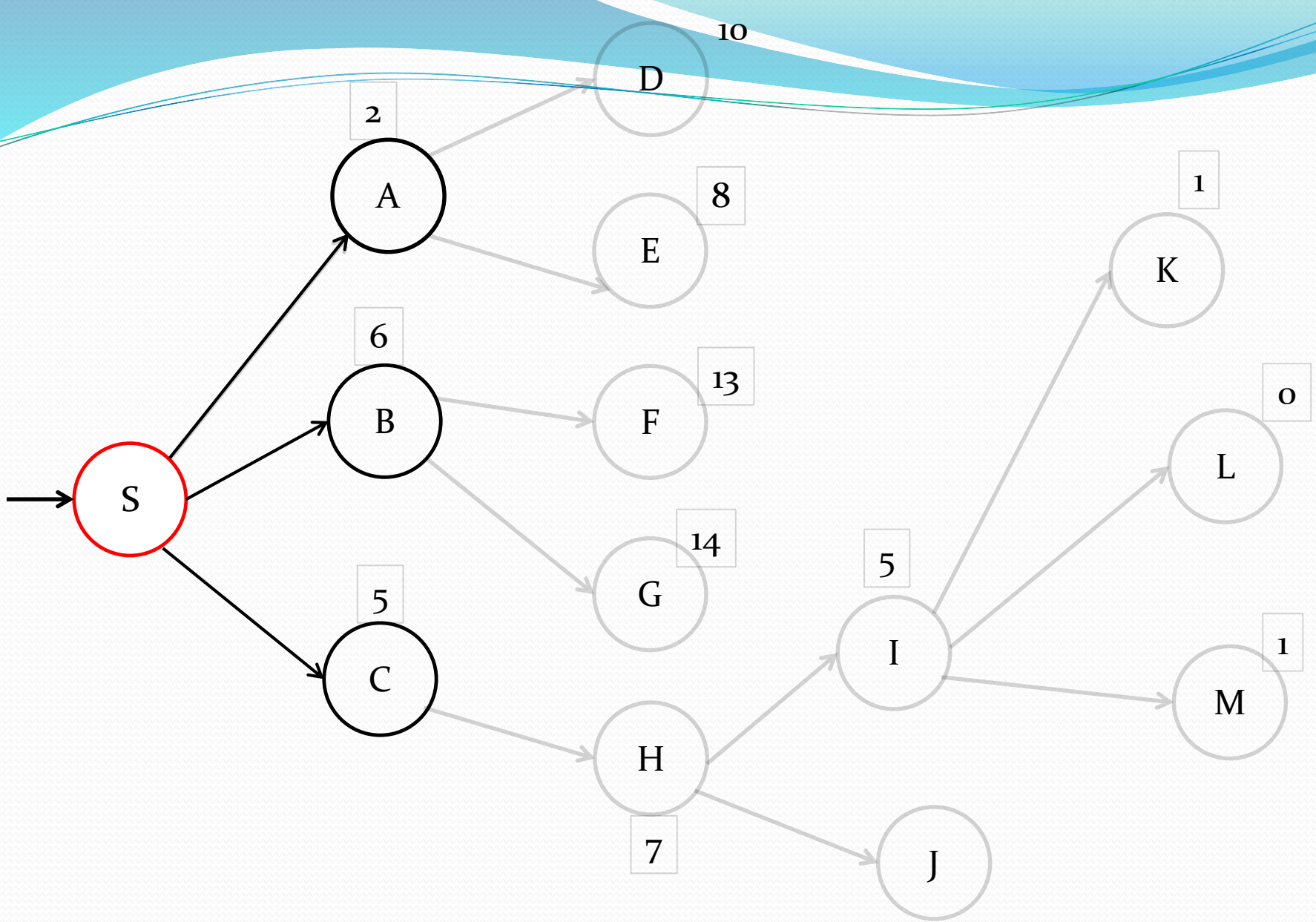
- Algorithm:
- Uses 2 lists:

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.

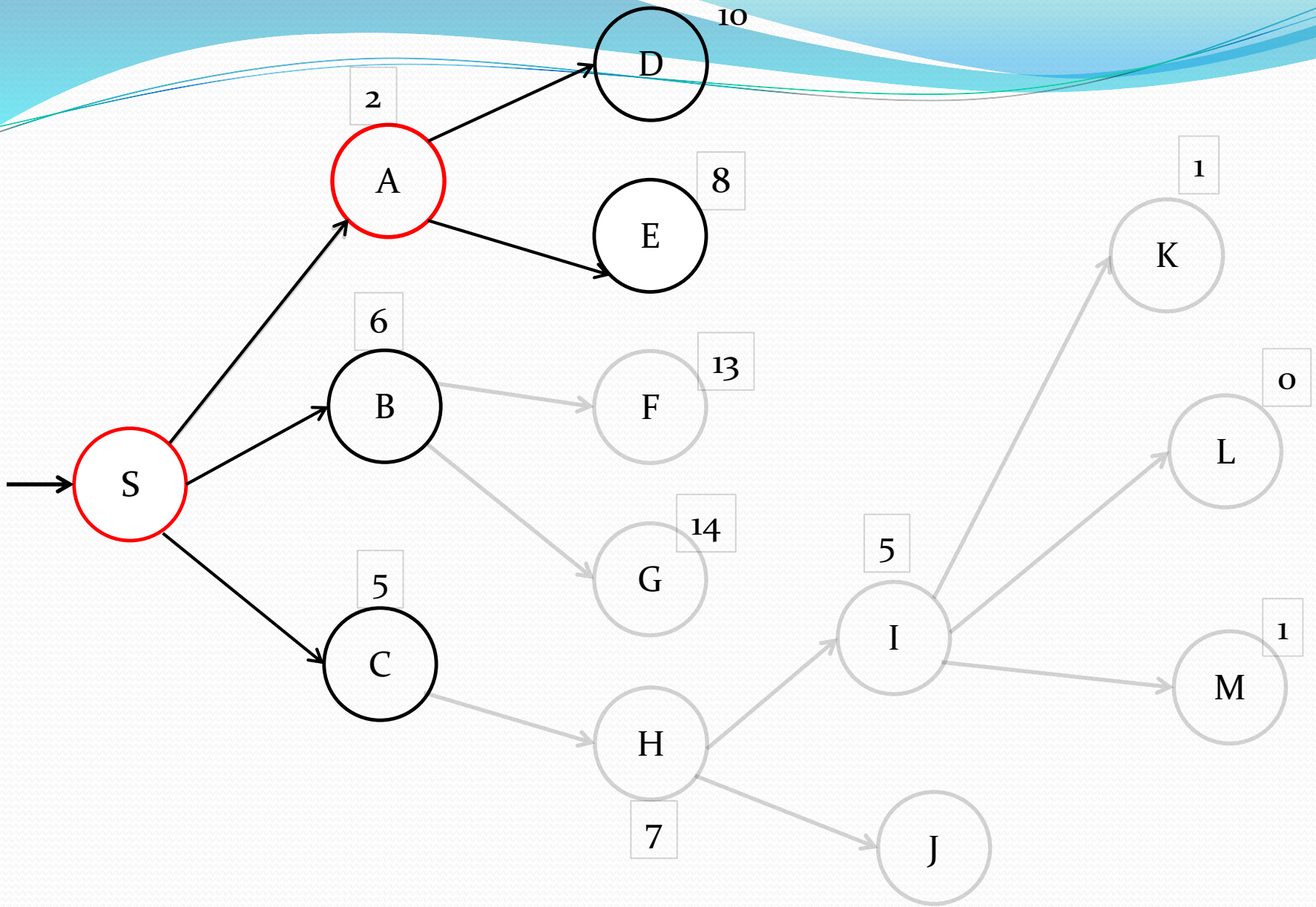


OPEN={ **S**}
CLOSED={}

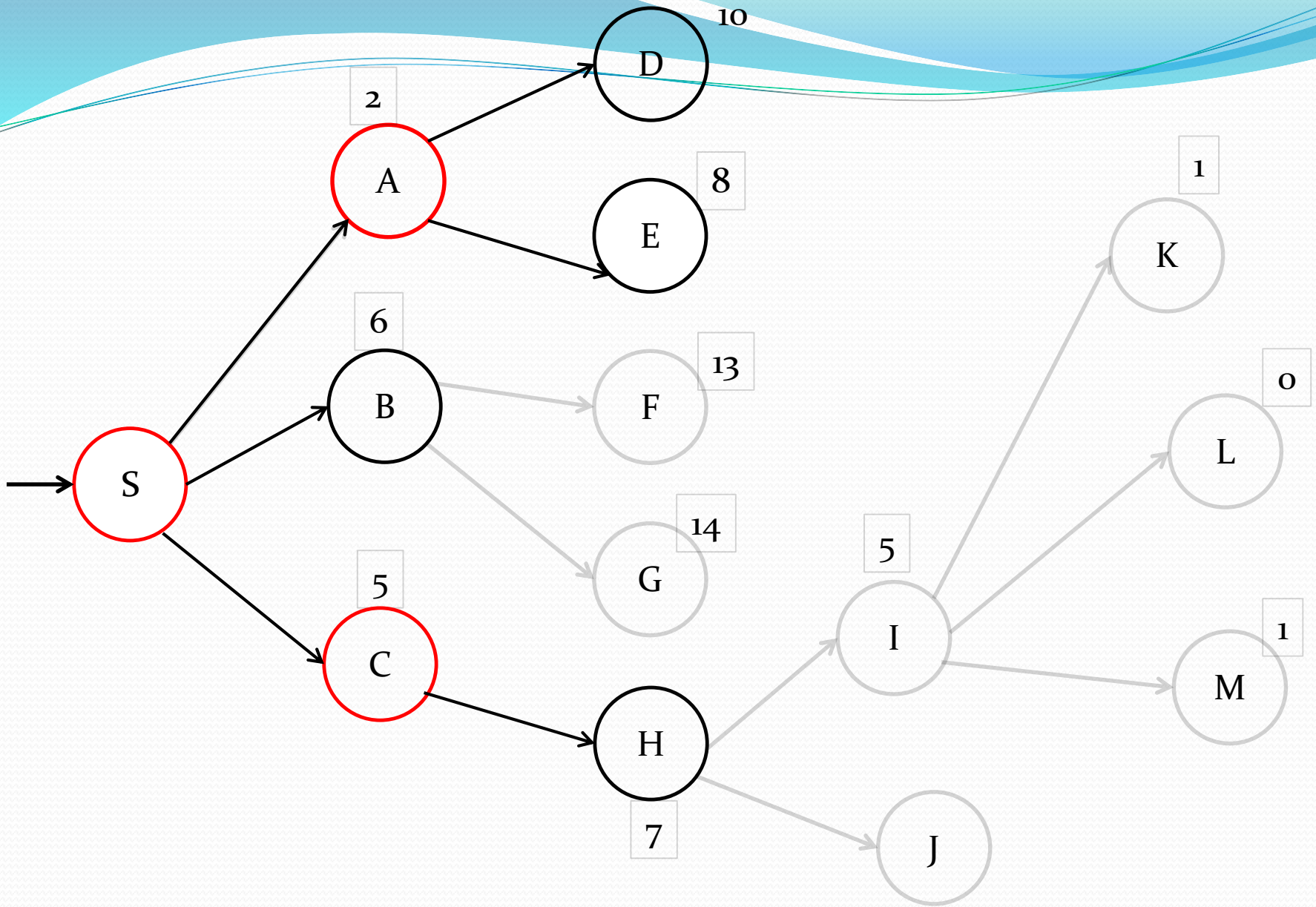
6



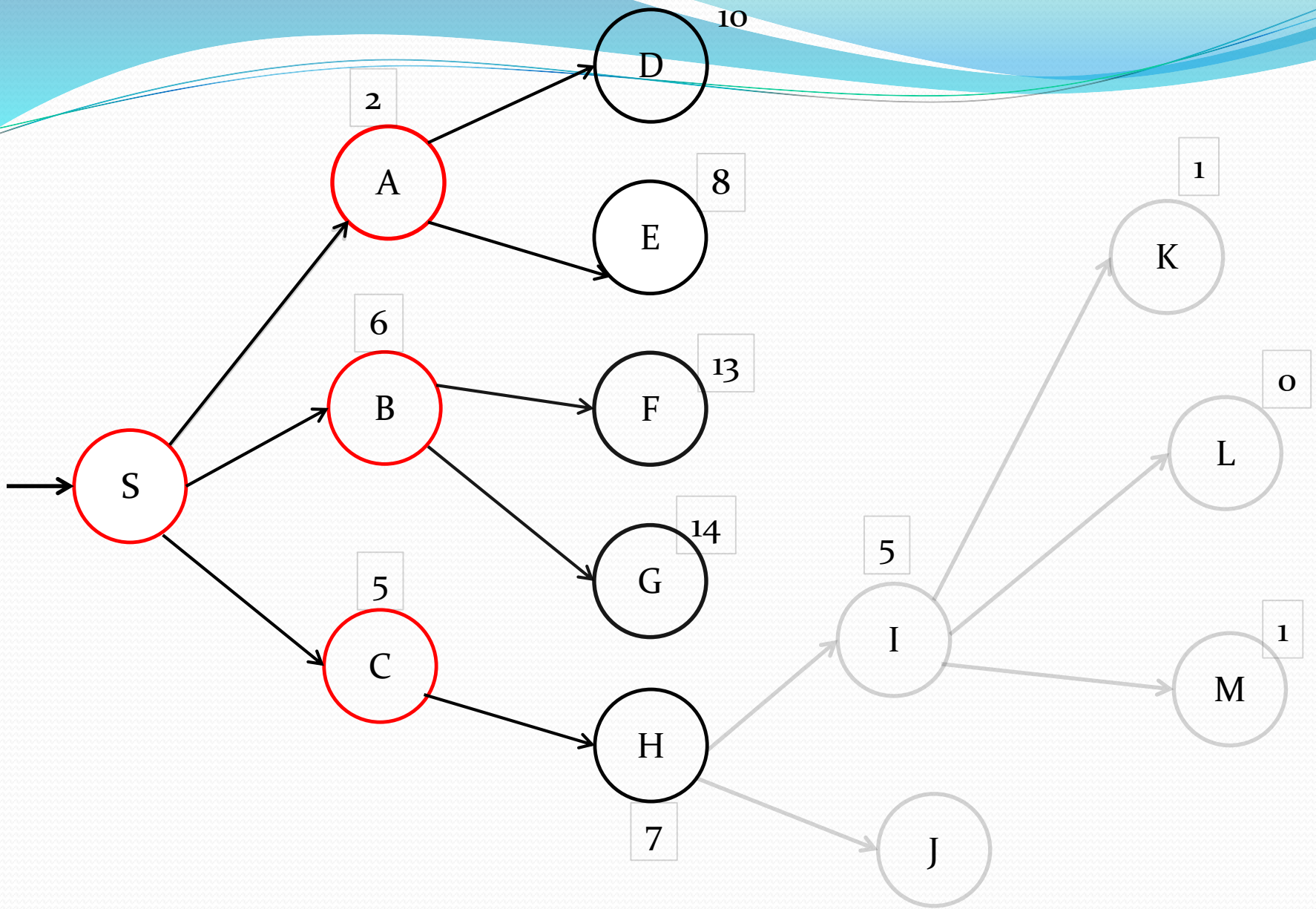
OPEN={ **A(2),C(5),B(6)**}
CLOSED={**S**}



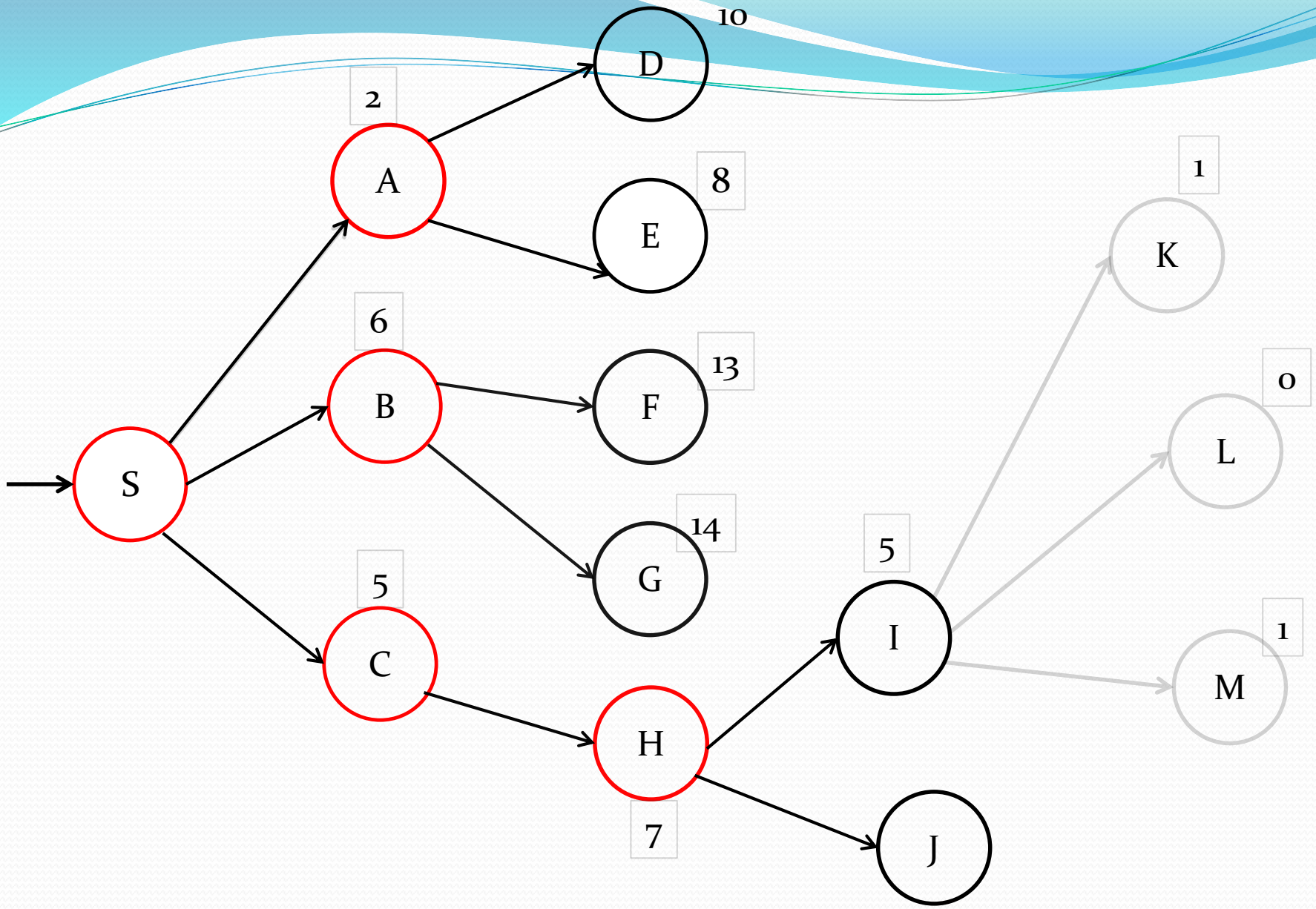
OPEN={ C(5), B(6), **E(8), D(10)**}
CLOSED={S,**A**}



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

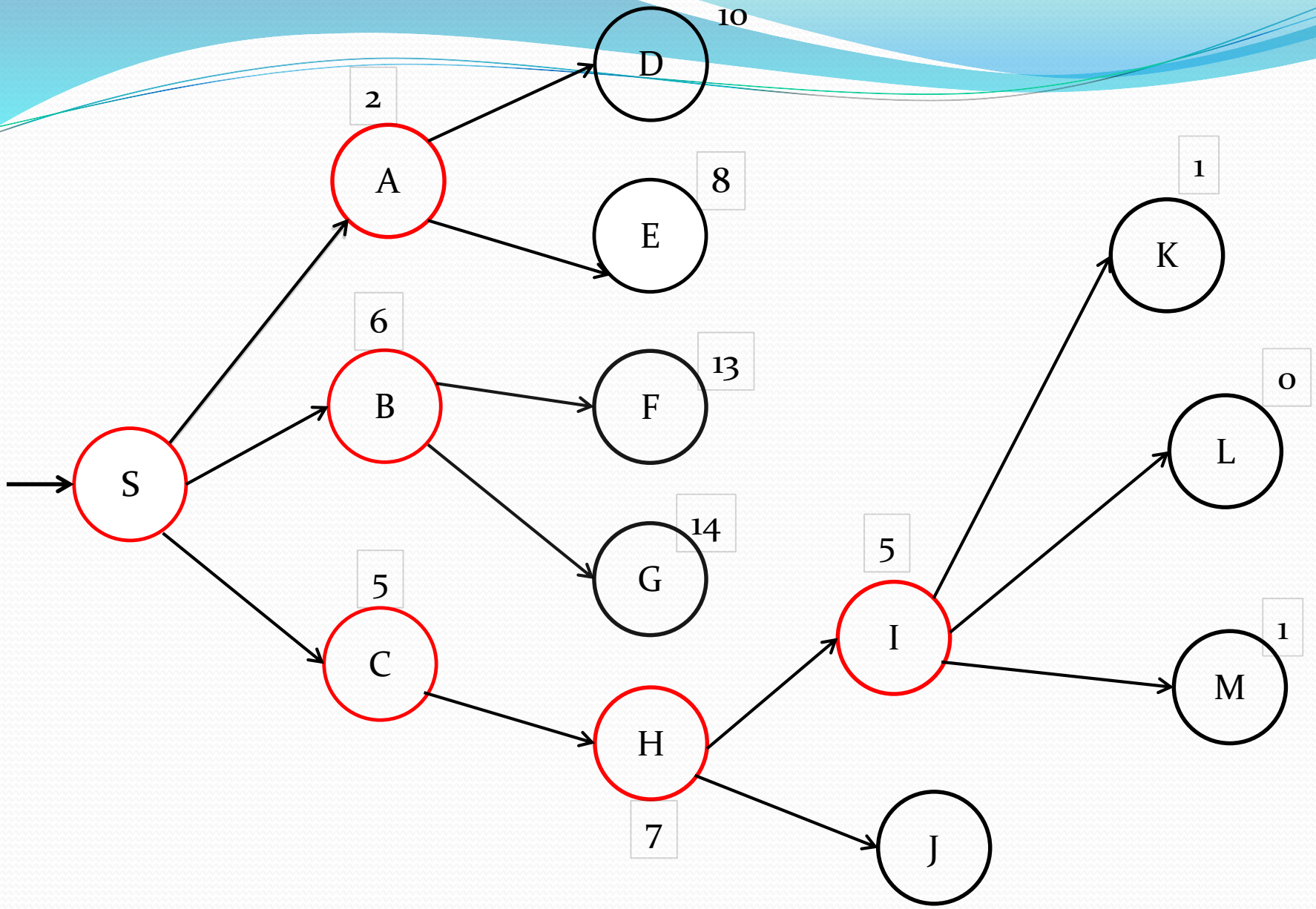


OPEN={H(7),E(8), D(10), **F(13),G(14)**}
CLOSED={S,A, C, **B**}

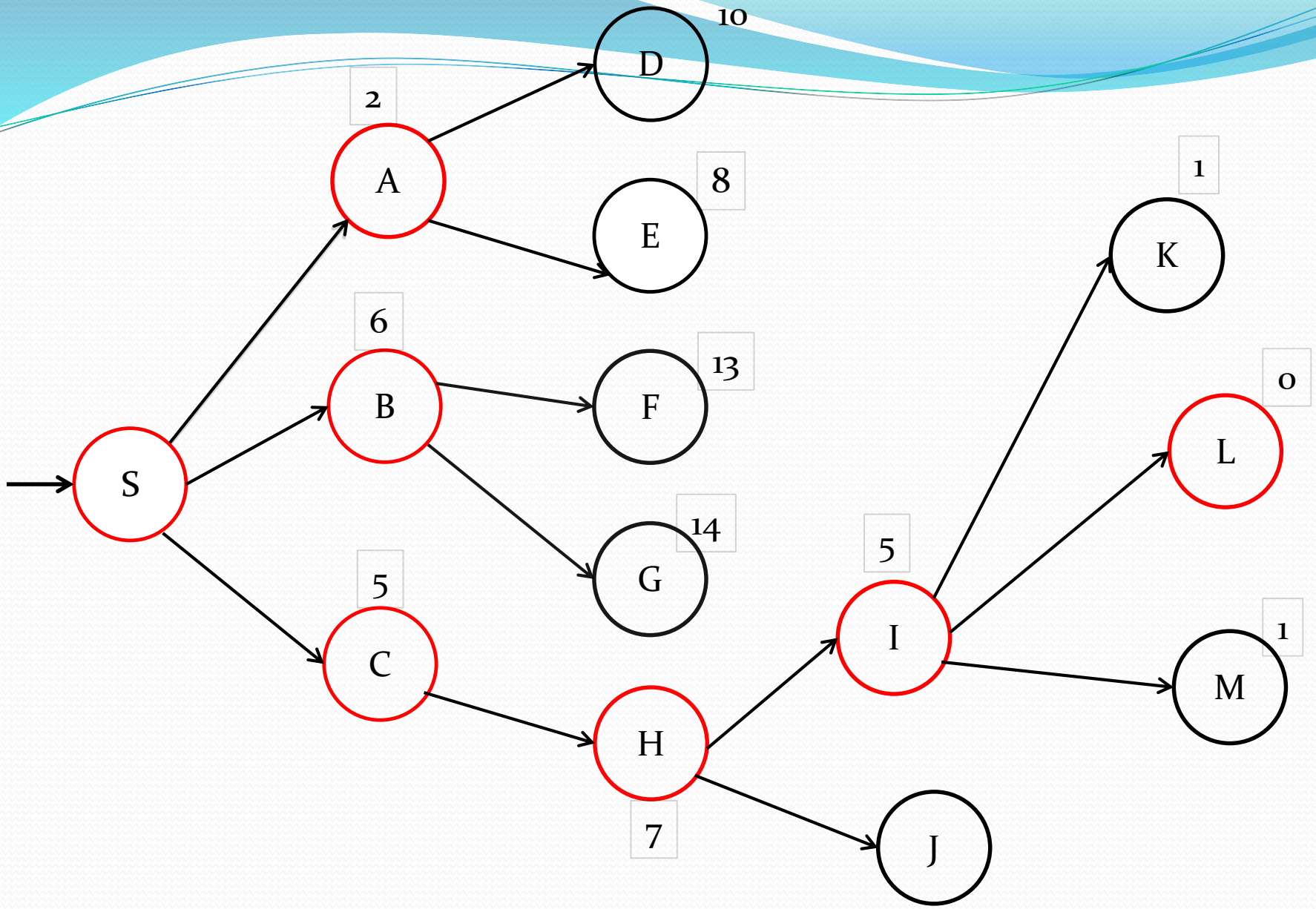


OPEN={**I(5), J(6),**H(7),E(8), D(10), F(13),G(14)}

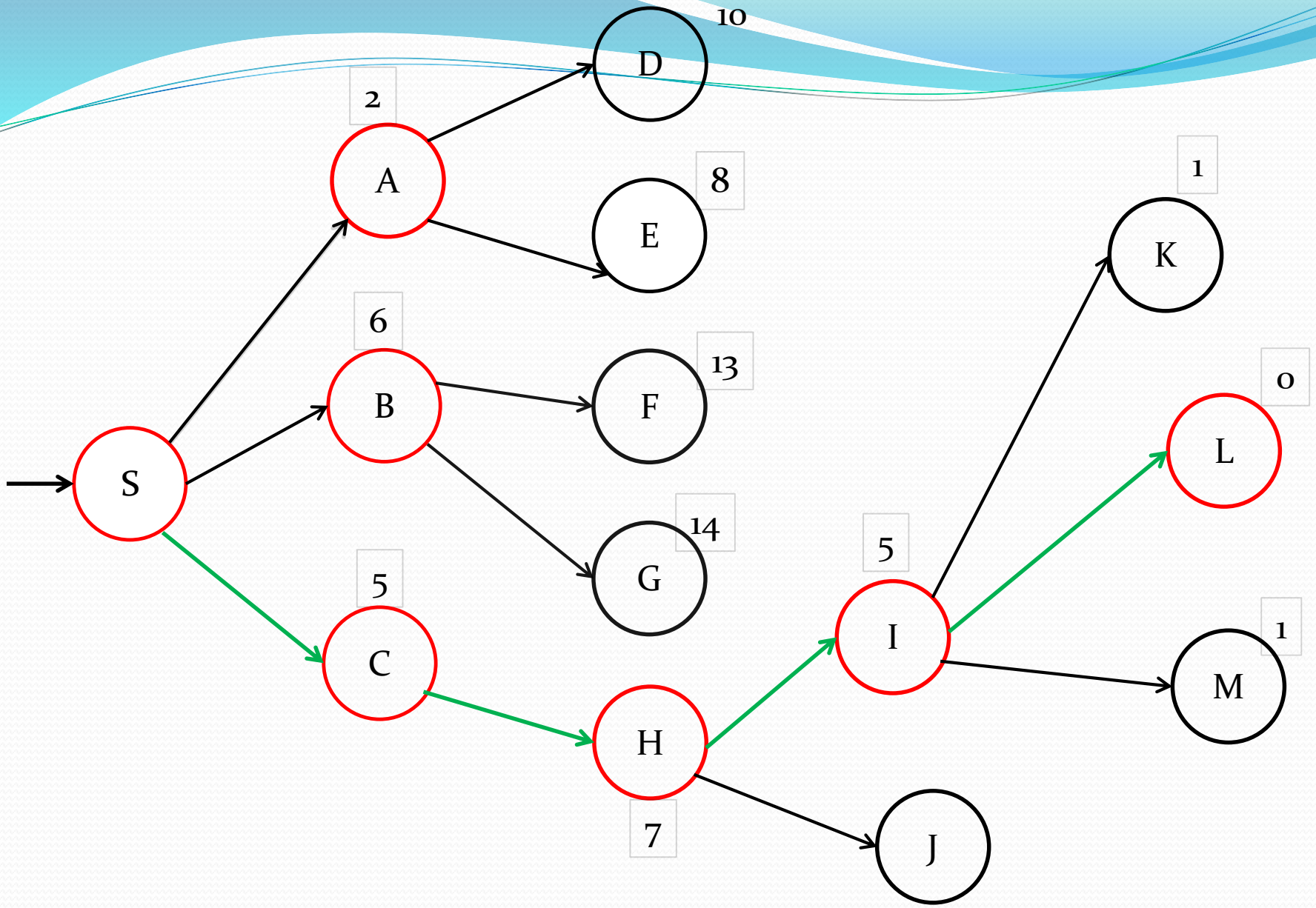
CLOSED={S,A, C, B,**H**}



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

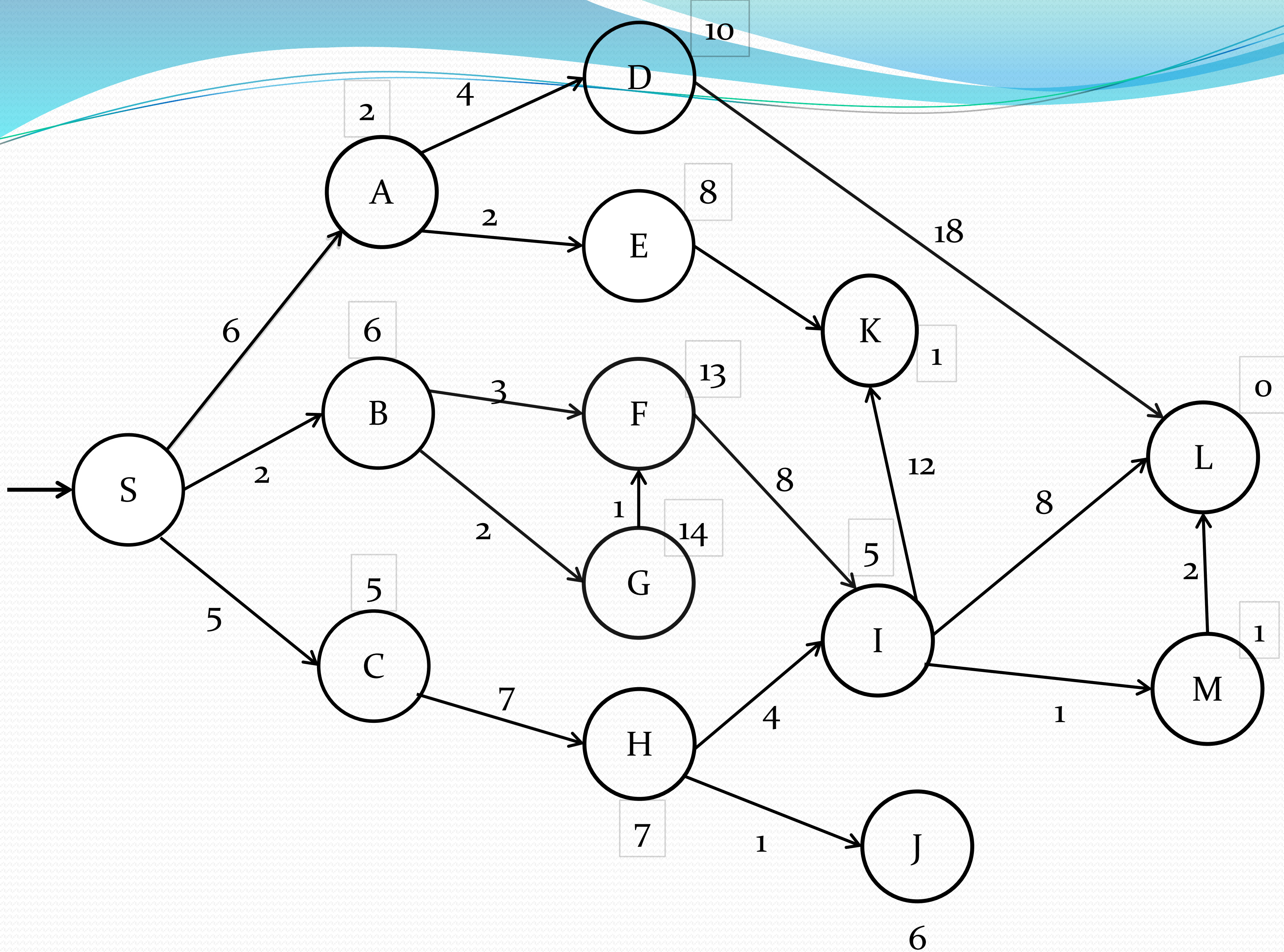
A* Algorithm

- 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.

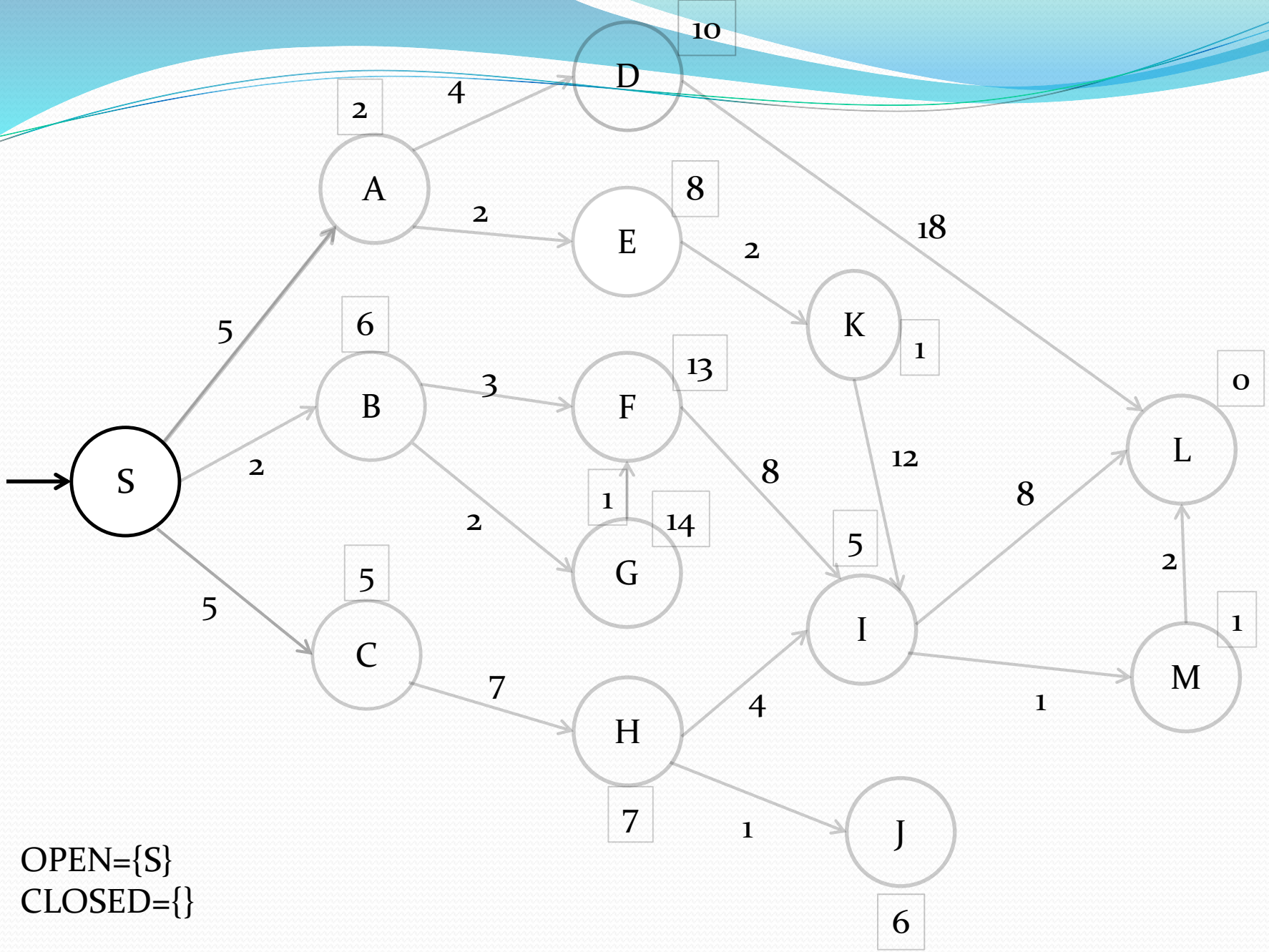
- A* takes the evaluation function as:
- F(n)=g(n)+h(n)
- where,

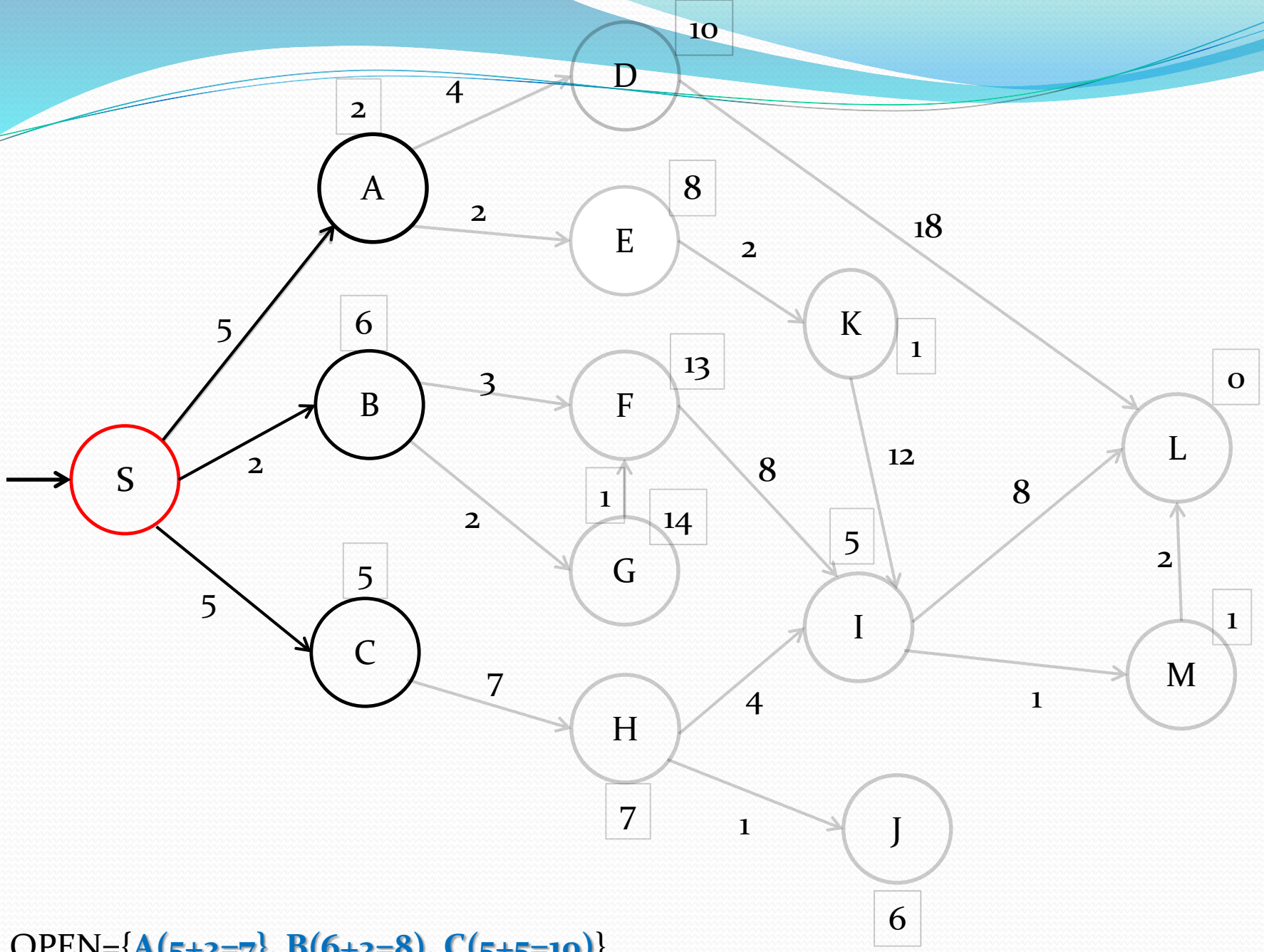
g(n)= cost(or distance) of the current node from **start state.**

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

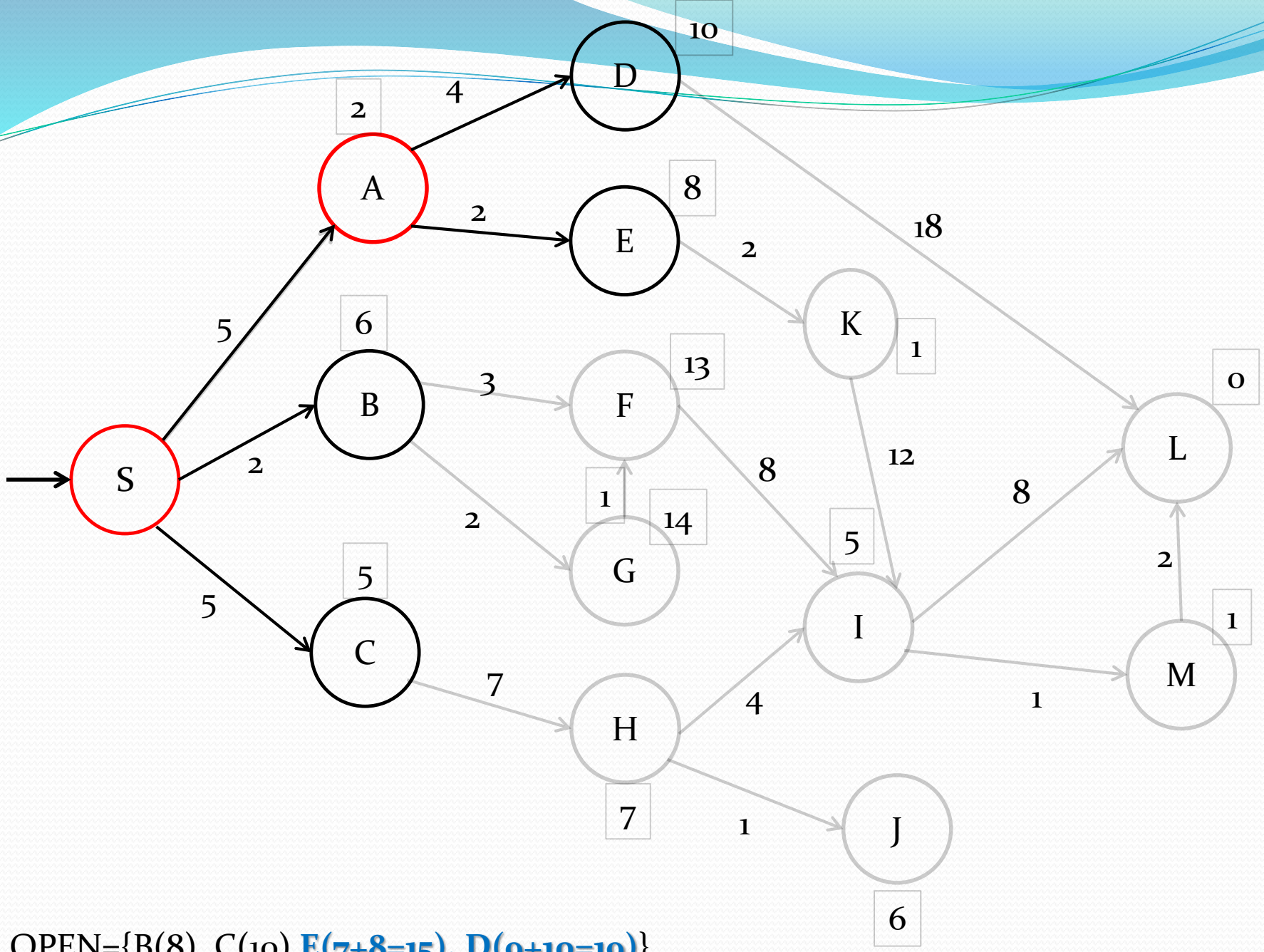


S
A
B
C
D
E
F
G
H
I
J
K
L
M
10
2
4
8
2
6
6
13
3
18
1
12
8
0
2
1
14
8
5
5
2
5
1
7
4
1
7
1
6

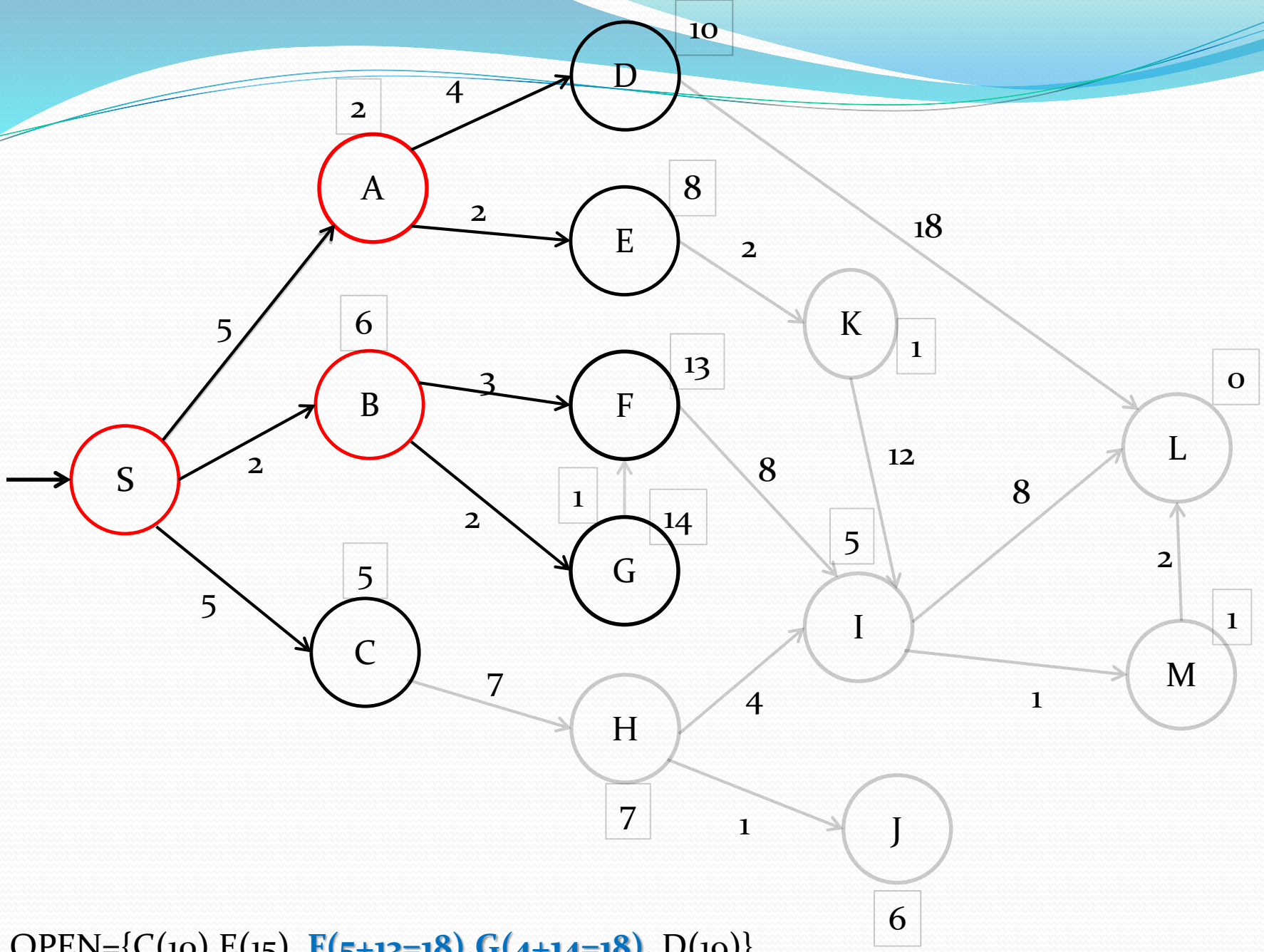

S
A
B
C
D
E
F
G
H
I
J
K
L
M
5
2
5
4
2
3
2
1
7
4
1
2
8
12
8
18
1
2
1
10
8
13
14
5
7
6
0
1
OPEN={S}
CLOSED={}



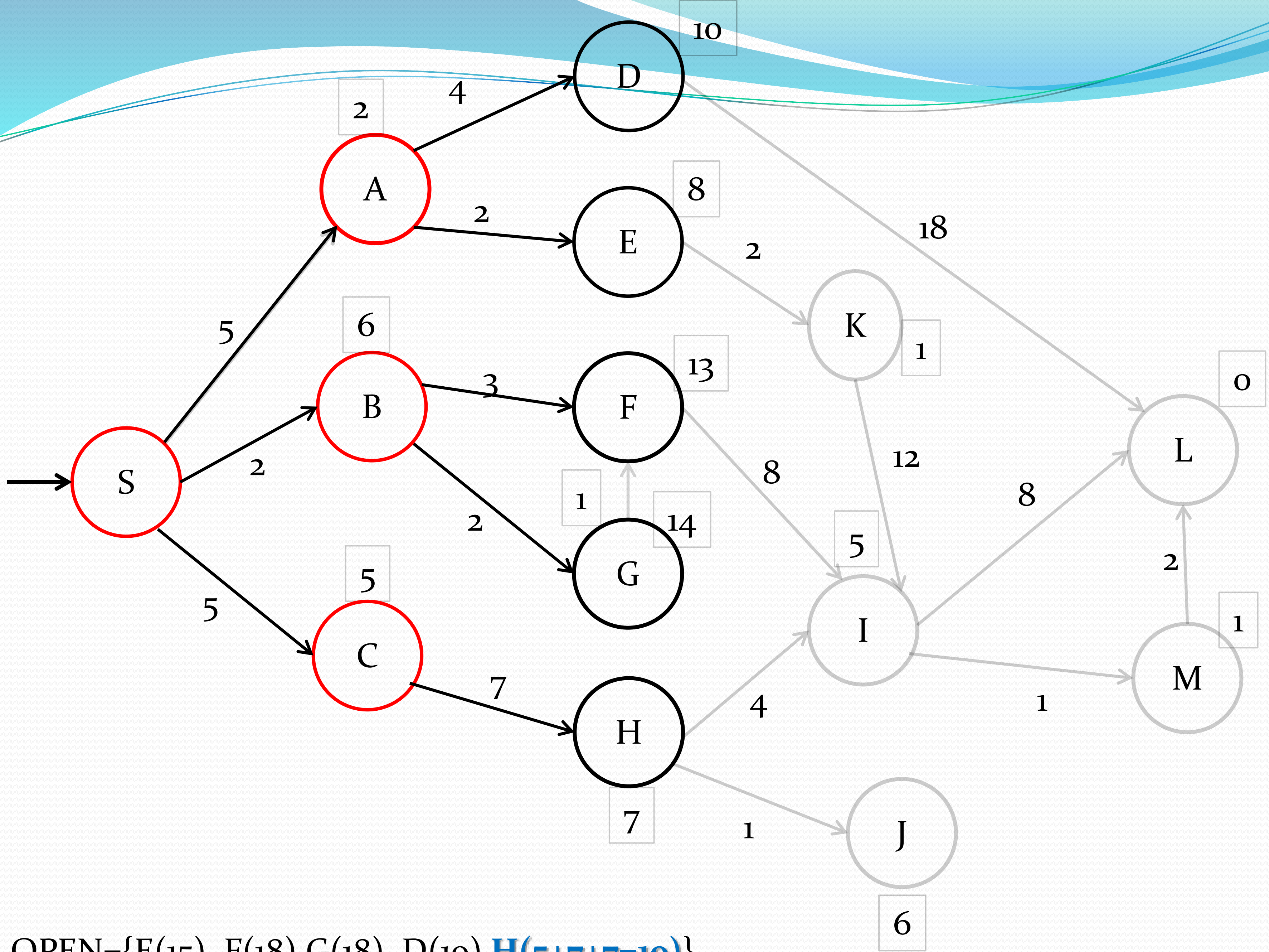
OPEN={A(5+2=7}, B(6+2=8), C(5+5=10)}
CLOSED={S}



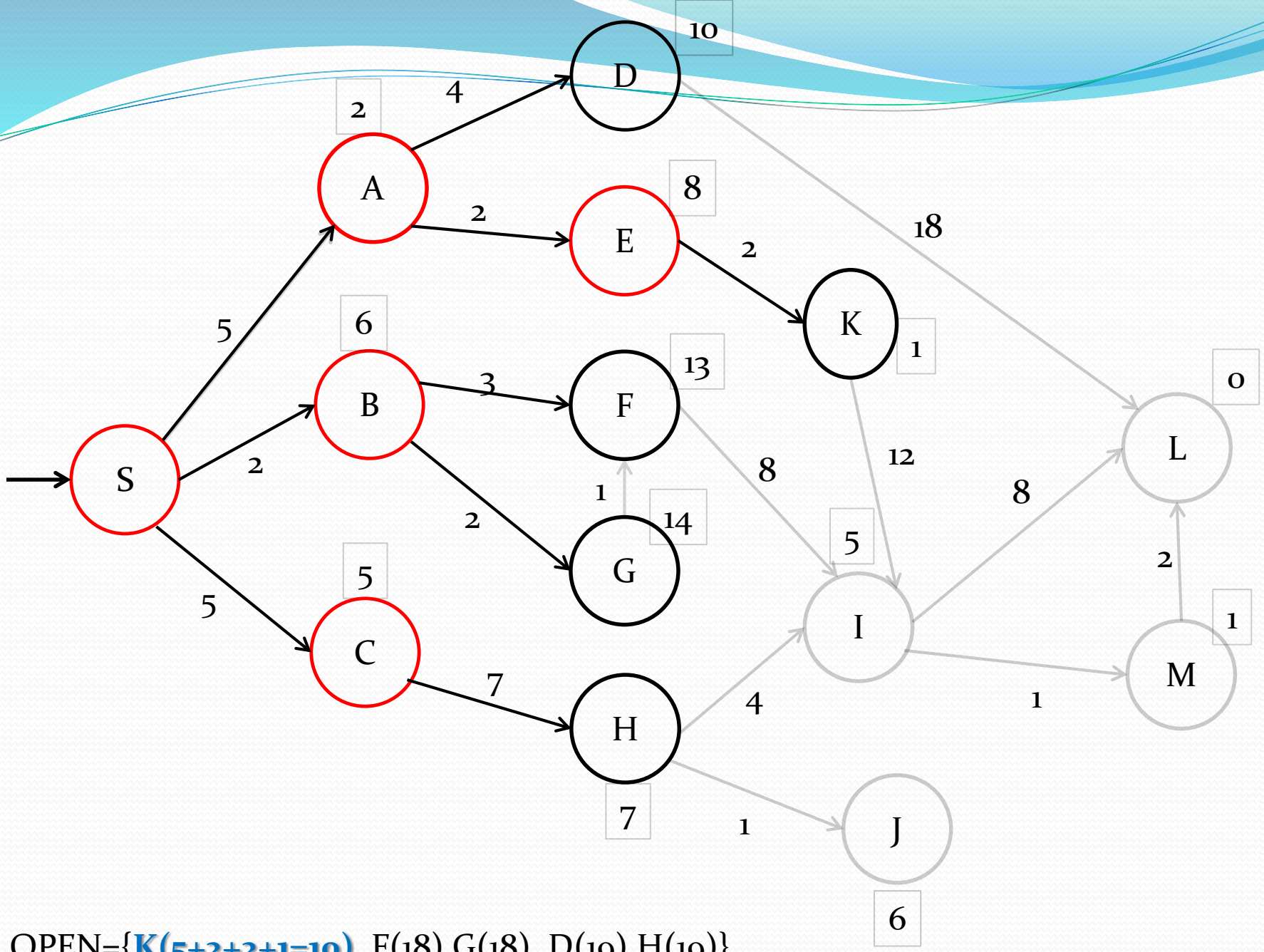
OPEN={B(8), C(10),**E(7+8=15), D(9+10=19)**}
CLOSED={S,**A**}



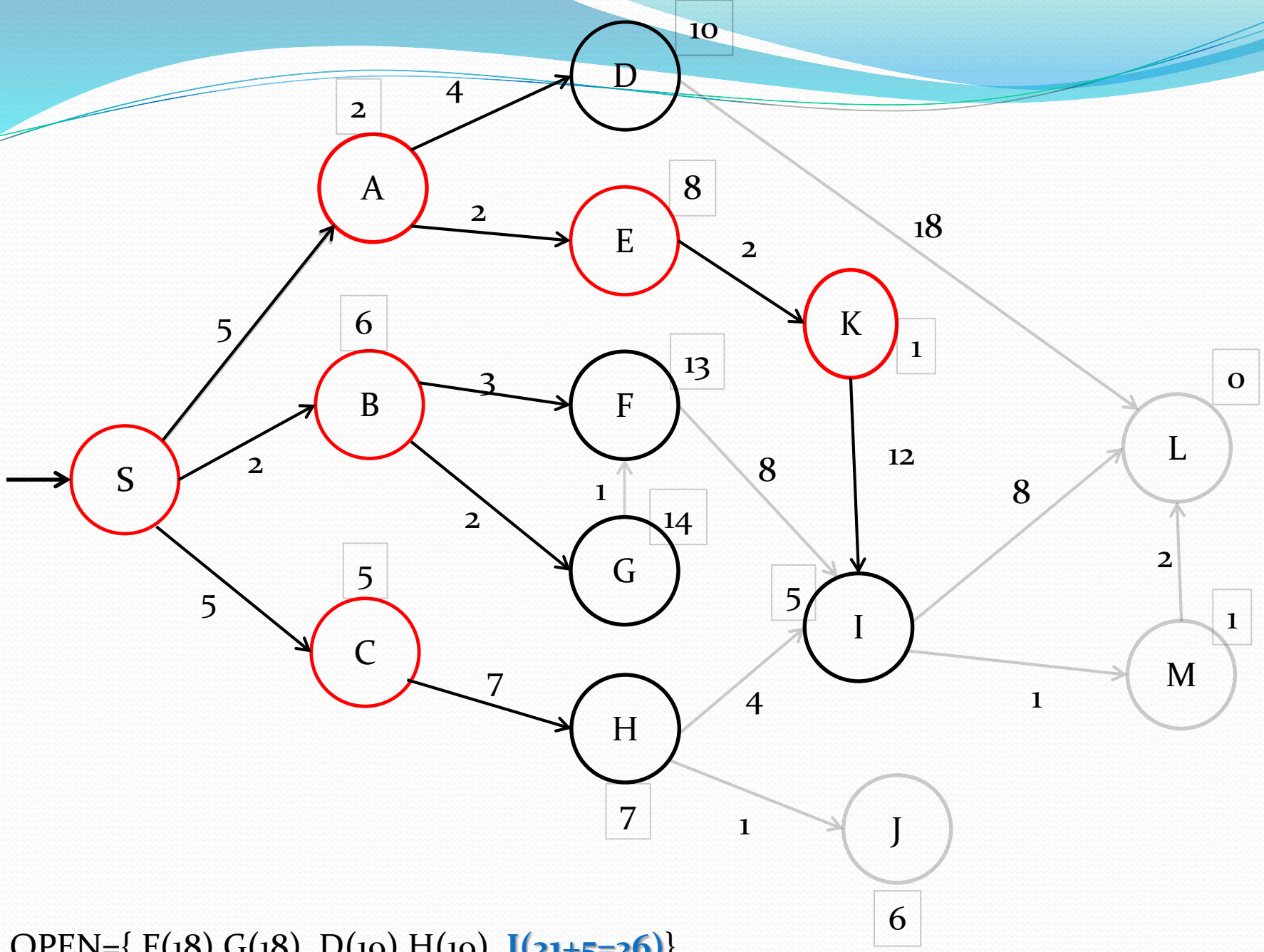
OPEN={C(10),E(15), **F(5+13=18),G(4+14=18),** D(19)}
CLOSED={S,A,**B**}



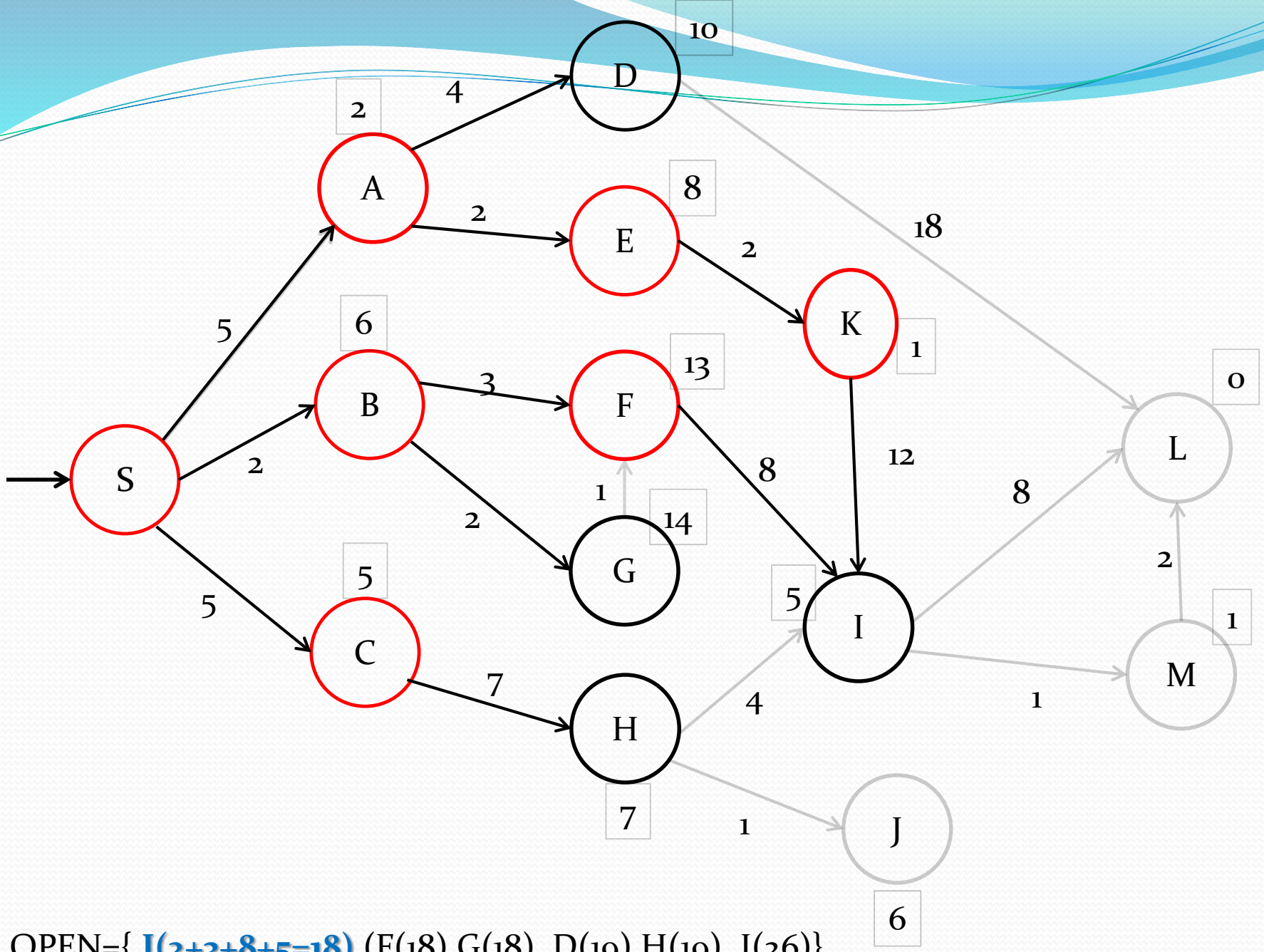
OPEN={E(15), F(18),G(18), D(19),**H(5+7+7=19)**}
CLOSED={S,A,B,**C**}



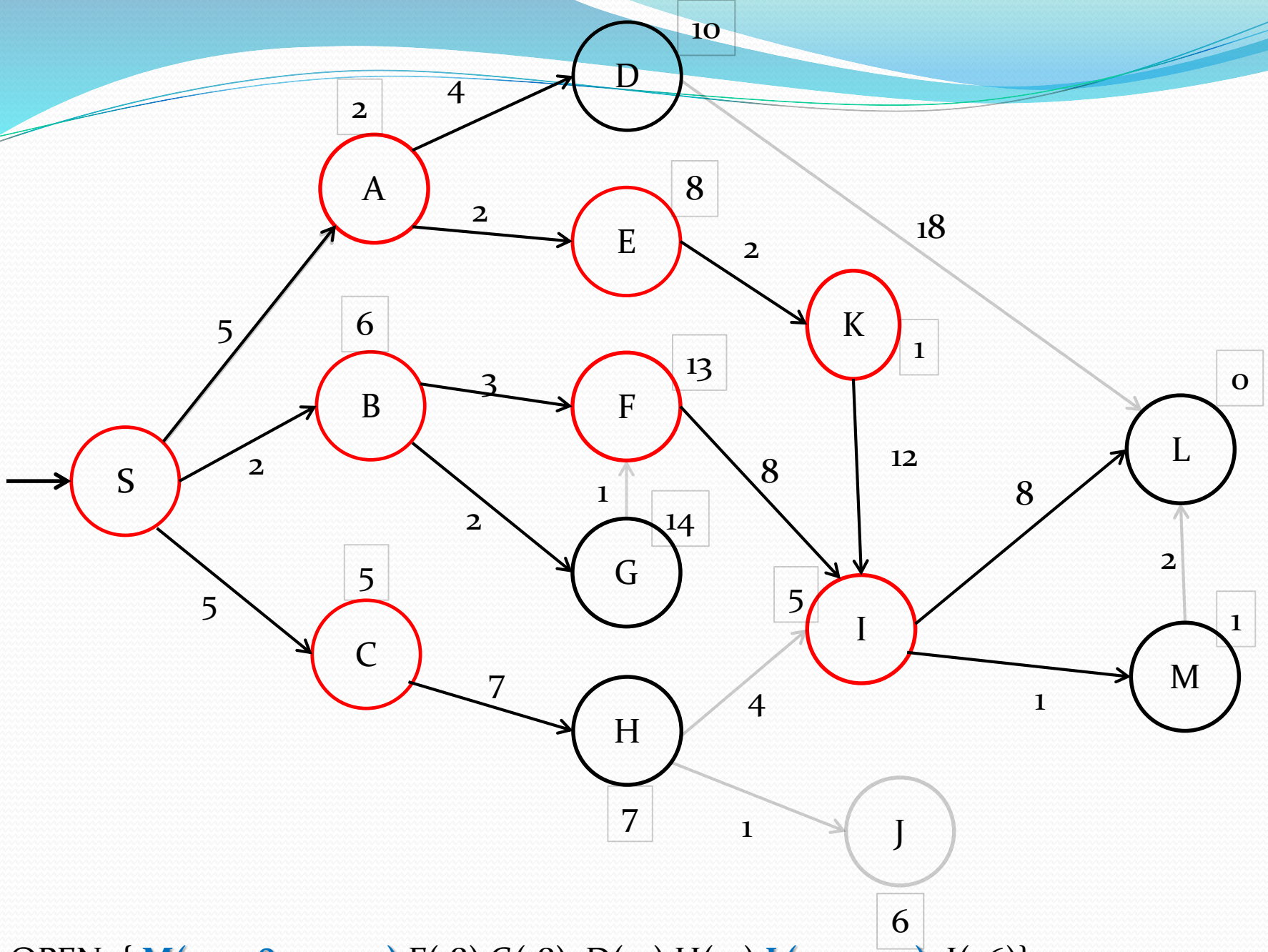
OPEN={**K(5+2+2+1=10)**, F(18),G(18), D(19),H(19)}
CLOSED={S,A,B,C,**E**}



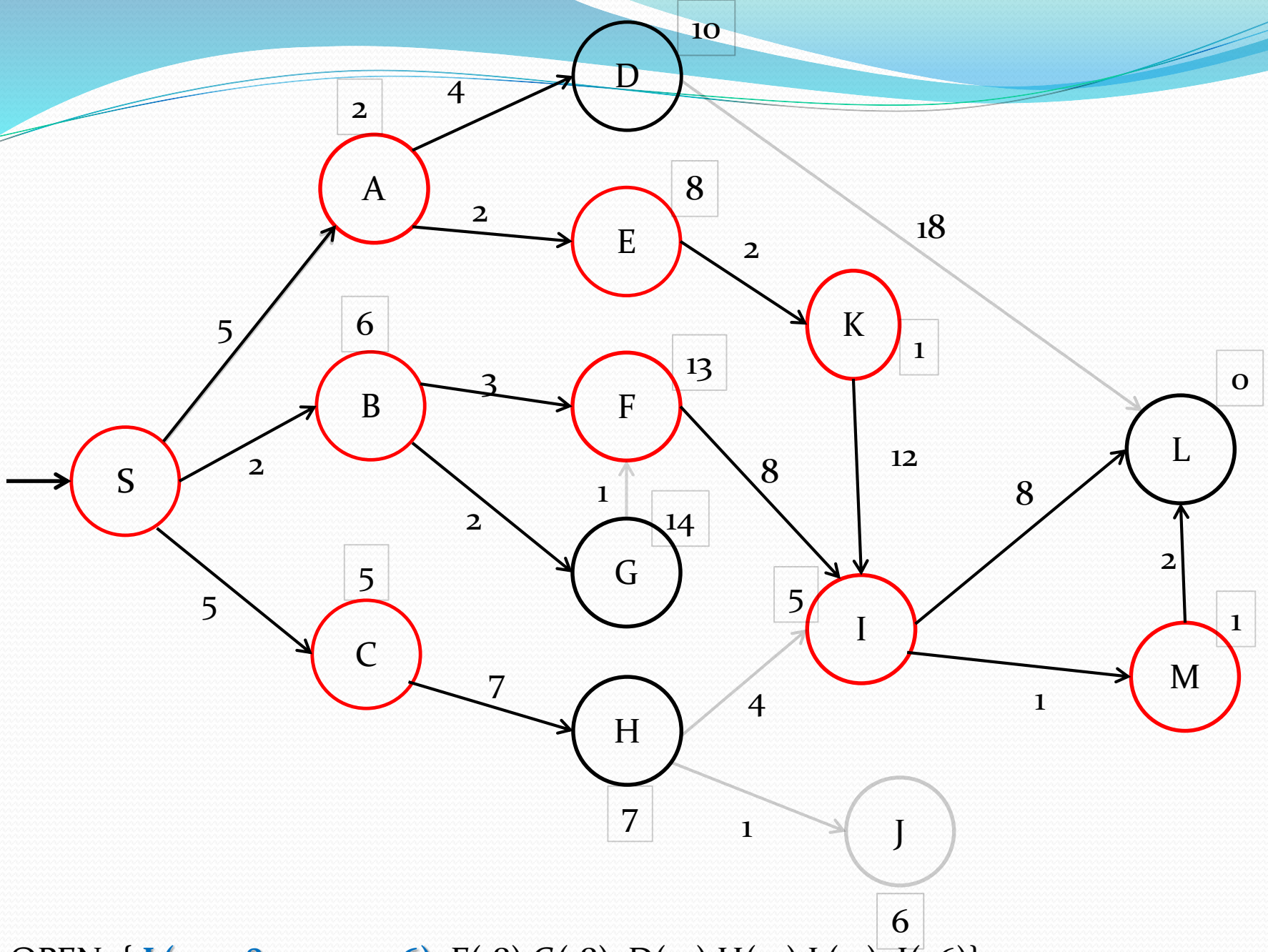
OPEN={ F(18),G(18), D(19),H(19), **I(21+5=26)**}
CLOSED={S,A,B,C,E,**K**}



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**}

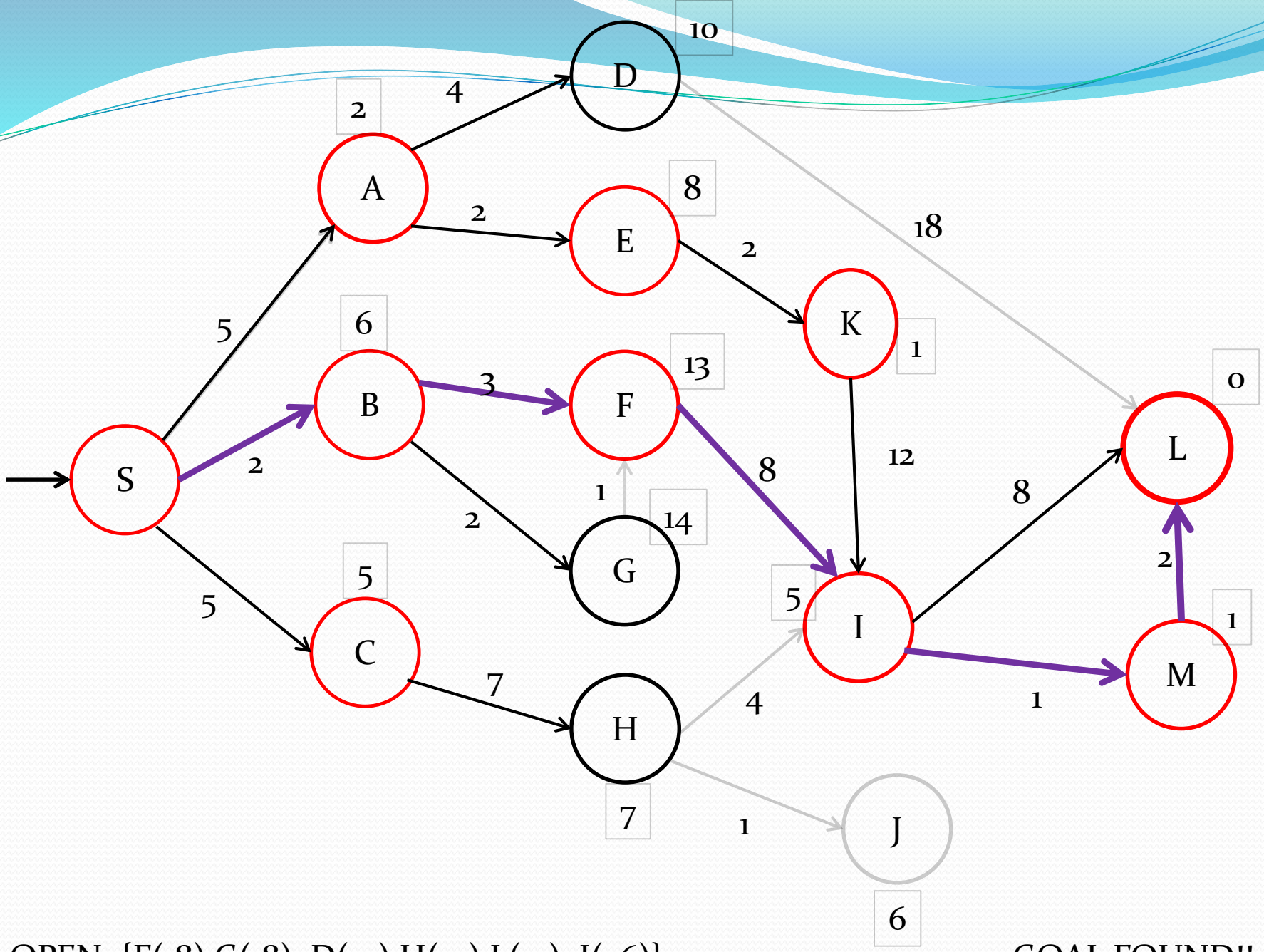


OPEN={ **M(2+3+8+1+1=15)**,F(18),G(18), D(19),H(19),**L(21+0=21)** ,I(26)}
CLOSED={S,A,B,C,E,K,**F,I**}



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**}

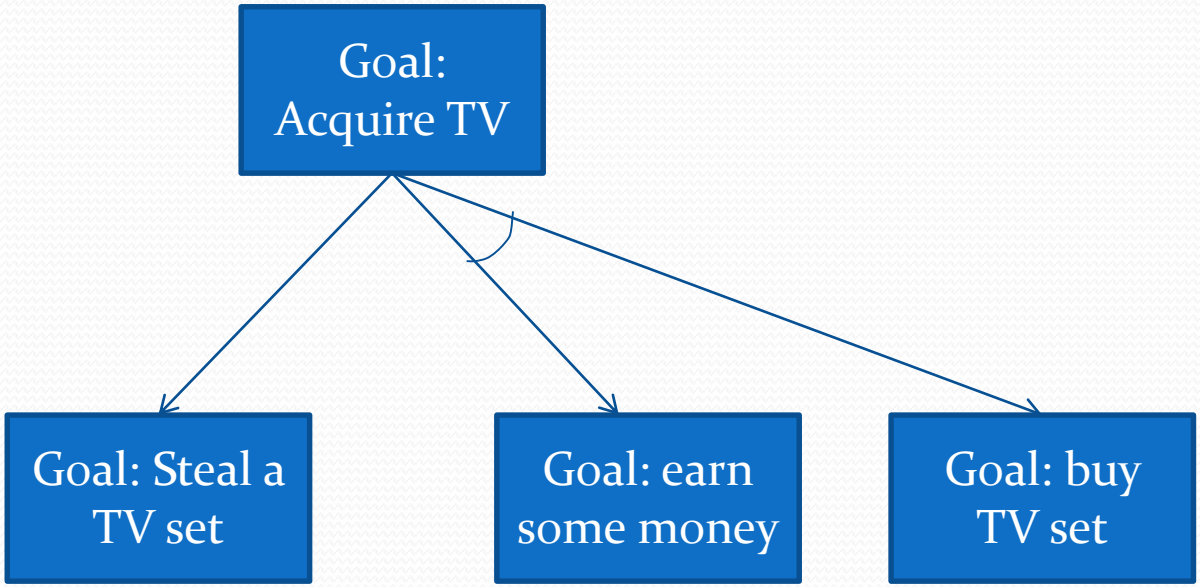


OPEN={F(18),G(18), D(19),H(19),L(21) ,I(26)}

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 $X_1, X_2, X_3, \ldots\ldots X_n$

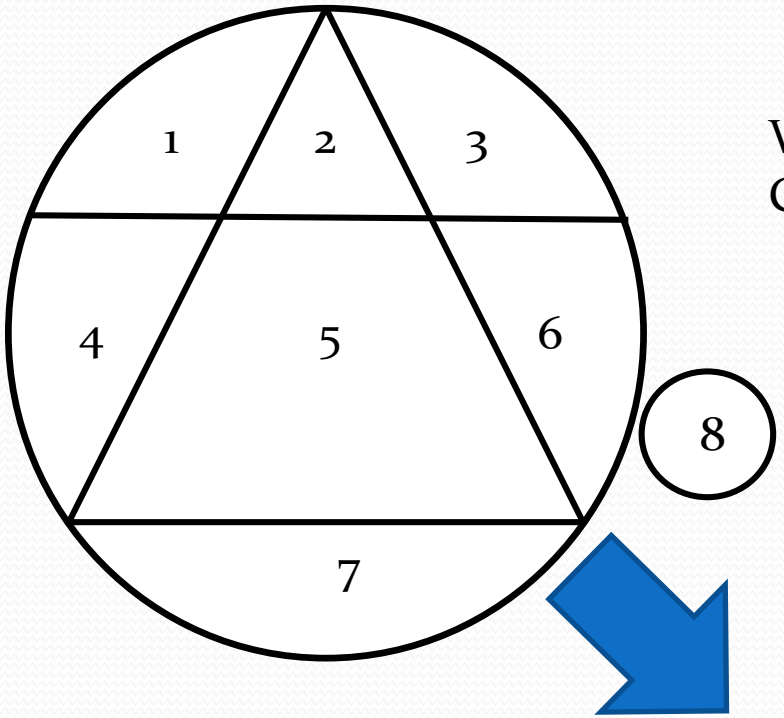
Constraints $C_1, C_2, \ldots\ldots C_n$

Variables have to be assigned with values $V_1, V_2, \ldots\ldots V_n$

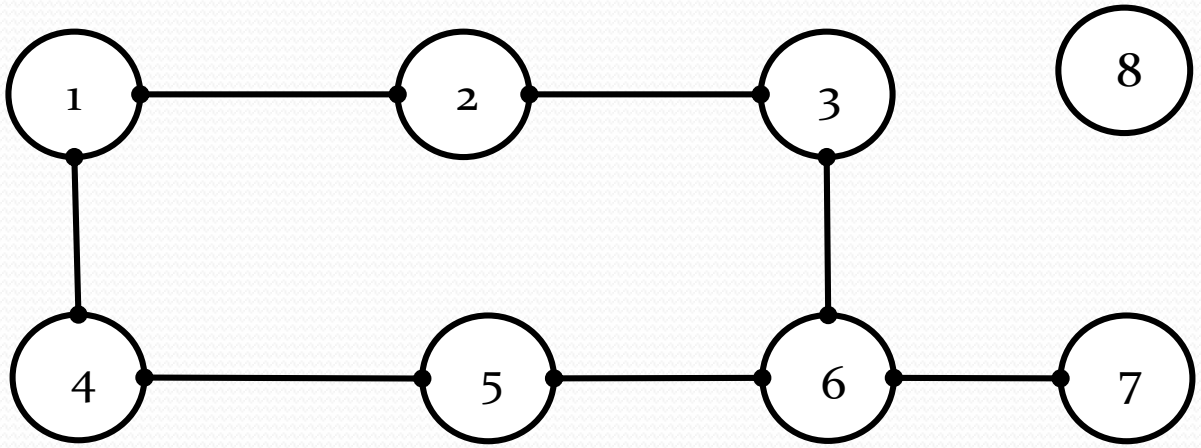
Such that none of the constraints are violated.

- 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
```

	C_3	C_2	C_1
	T	W	O
+	T	W	O
F	O	U	R

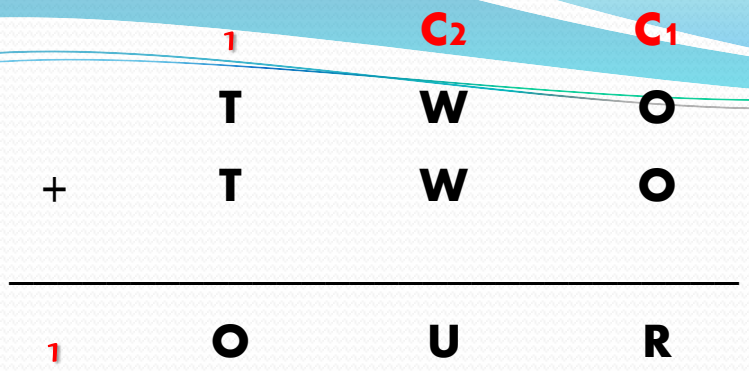
STEP 1:

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

$C_3=1$ $F=1$

Thus,

	1	C_2	C_1
	T	W	O
+	T	W	O
1	O	U	R



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	C_1
	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	C_1
	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	C_2	C_1
	6	W	O
+	6	W	O
1	O	U	R

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

$\boxed{O=3}$ Accepted till now

	1	C_2	C_1
	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	C_2	C_1
	7	W	O
+	7	W	O
1	O	U	R

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

	1	C_2	C_1
	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	C_2	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
	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