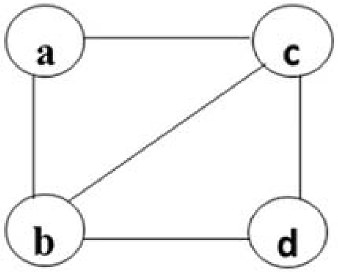
# Simple Search Algorithm-1 (1)

The first version of Simple Search algorithm.

```
Def SimpleSearch1()
Open=Start // {a}
While Open != Empty() then
    Pick some node 'N' from Open
    Open = Open – {N}
    If GOALTEST(N) = True then
      Return N
    Else
      Open = Open U {MOVEGEN(N)}
Return FAILURE
```

```
Open = {a}
While {a} != {}
  Pick 'a' from Open
   Open = {a} – {a} = {}
   if GOALTEST(a) = TRUE then
     return 'a'
    else
     Open = {} U { MOVEGEN(a) }
Return FAILURE
```



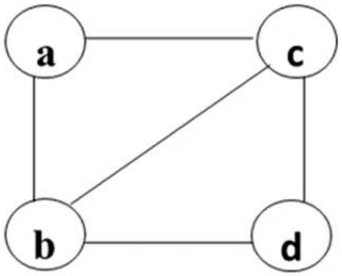
Example: Graph  
V= {a,b,c,d}  
E= {a-b,a-c,b-c,b-d,c-d}  
S ={a:[b,c],  
    b:[a,c,d],  
    c:[a,b,d]  
    d:[b,c] }  
Start = {a}  
Goal = {d}

# Simple Search Algorithm-1 (2)

The **GOALTEST(N)** and **MOVEGEN(N)**

```
Def GOALTEST(N)
  If N = Goal then
   Return TRUE
 Else
   Return FALSE

Def MOVEGEN(N)
Succ ={}
For N in S do
   Succ= Succ U {Children of N}
```

```
Open = {a}
While {a} != {}
   Pick 'a' from Open
    Open = {a} – {a} = {}
    if GOALTEST(a) = TRUE then
       return 'a'
     else
       Open = {} U { MOVEGEN(a) }
   //Open = {b,c}
Return FAILURE
```



Example: Graph
V= {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d, }
S ={a:[b,c],
b:[a,c,d],
c:[a,b,d]
d:[b,c] }
Start = {a}
Goal = {d}

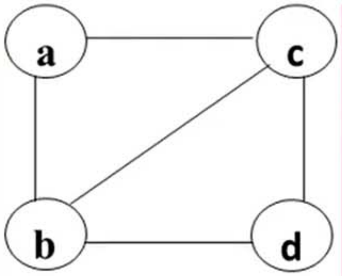
```
Def GOALTEST(a)
  If 'a' = 'd' then
   Return TRUE
 Else
   Return FALSE

Def MOVEGEN(a)
Succ ={}
For 'a' in S do
   Succ= { } U {b,c}
```

# Simple Search Algorithm-1 (3)

The **CLOSED** list

```
Open = {a}
While {a} != {}
  Pick 'a' from Open
   Open = {a} – {a} = {}
   if GOALTEST(a) = TRUE then
     return 'a'
    else
     Open = {} U { MOVEGEN(a) }
  //Open = {b,c}
Return FAILURE
```

```
Open = {b,c}
While {b,c} != {}
  Pick 'b' from Open
  Open = {b,c} –{b} = {c}
if GOALTEST(b) = TRUE then
     return 'c'
    else
     Open = {c} U { MOVEGEN(b) }
   //Open = {c} U {a,c,d}
  //Open = {a,c,d,c}
Return FAILURE
```



```
Example: Graph
V=  {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d, }
S ={a:[b,c],
     b:[a,c,d],
     c:[a,b,d]
     d:[b,c] }
Start = {a}
Goal = {d}
```

```
Def GOALTEST(b)
  If 'b' = 'd' then
   Return TRUE
 Else
   Return FALSE

Def MOVEGEN(b)
Succ ={}
For 'b' in S do
  Succ= { } U {a,c,d}
```

Open = {a,c,d,c} has 'a' at the beginning of the list so it will again visit the node 'a' and will be in never ending loop.

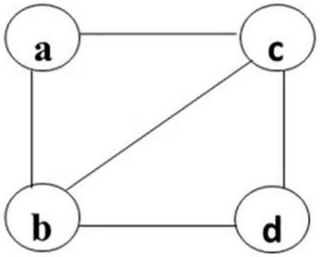
# Simple Search Algorithm-2

The second version of Simple Search algorithm.

```
Def SimpleSearch2()
Open=Start // {a}
Closed={}
While Open != Empty() then
      Pick some node 'N' from Open
      Open = Open – {N}
      Closed = Closed U {N}
      If GOALTEST(N) = True then
        Return N
      Else
        Open = Open U {MOVEGEN(N) – Closed}
Return FAILURE
```

```
Open = {a}
Closed = {}
While {a} != {}
  Pick 'a' from Open
   Open = {a} – {a} = {}
   Closed = {} U {a} = {a}
   if GOALTEST(a) = TRUE then
     return 'a'
    else
     Open = {} U {MOVEGEN(a) – Closed}
   //Open = {} U { {b,c} – {a} } = {b,c}
Return FAILURE
```

```
Open = {b,c}
Closed = {a,b}
Open = {c} U { {a,c,d} – {a,b} } = {c} U {c,d}  = {c,d}
```



Example: Graph<br>
V=  {a,b,c,d}<br>
E= {a-b,a-c,b-c,b-d,c-d}<br>
S ={a:[b,c],<br>
   b:[a,c,d],<br>
   c:[a,b,d]<br>
   d:[b,c] }<br>
Start = {a}<br>
Goal = {d}

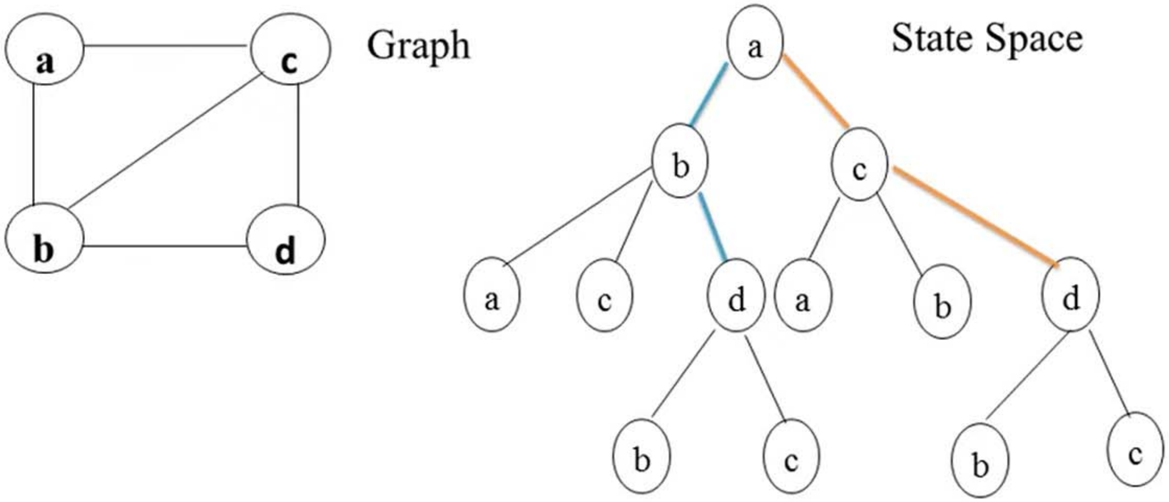
# Depth First Search (1)

The **Depth First Search** (**DFS**) is a search technique which searches to the ***lowest depth*** or ***ply*** of the tree.

The **DFS** uses ***Stack*** as a data structure (**OPEN** list) to process the given state space.

Example: Graph
V= {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d, }

Start = {a}
Goal = {d}



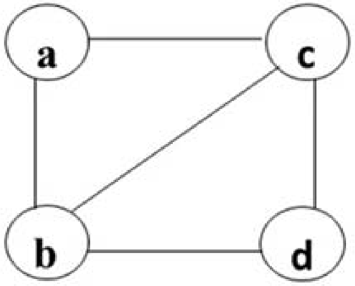
S ={ a:[b,c], b:[a,c,d], c:[a,b,d], d:[b,c] }

# Depth First Search (2)

## Algorithm DFS()

```
Def DFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS) then
    Pick front node 'N' from Open
    Open = Open – {N}
    If GOALTEST(N) = TRUE  then
        State = SUCCESS
        Closed = APPEND (Closed , {N})
    Else
        Closed = APPEND (Closed , {N})
        Child = { MOVEGEN(N) }
        Child = {Child - Open}
        Child= {Child  - Closed}
        Open = APPEND (Child,Open)
    End If
End While
Return State
```

```
Def GOALTEST(N)
  If N = Goal then
    Return TRUE
 Else
    Return FALSE

Def MOVEGEN(N)
Succ ={}
For N in S do
   Succ= Succ U  {Children of N}

Return Succ

Def APPEND(list1, list2)
New_list = list1  + list2
Return New_list
```



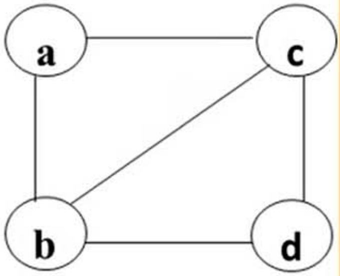
Example: Graph
V=  {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d}
S ={a:[b,c],
 b:[a,c,d],
 c:[a,b,d]
 d:[b,c] }
Start = {a}
Goal = {d}

# Depth First Search (3)

## Step-1: Iteration

```
Def DFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS)
    Pick front node 'N' from Open
    Open = Open – {N}
    If GOALTEST(N) = TRUE then
        State = SUCCESS
        Closed = APPEND (Closed , {N})
    Else
        Closed = APPEND (Closed , {N})
        Child = { MOVEGEN(N) }
        Child = {Child - Open}
        Child= {Child  - Closed}
        Open = APPEND (Child,Open)
    End If
End While
Return State
```

```
DFS({a})
Open  = {a}
Closed = {}
State = FAILURE
While ({a} <> Empty) AND (FAILURE <> SUCCESS)
    N='a'
    Open = {a} – {a} = {}
    If GOALTEST(a) = TRUE then
        State = SUCCESS
        Closed = APPEND (Closed , {N})
    Else
        Closed = APPEND ({}, {a}) = {a}
        Child = {b,c}
        Child = {b,c} – {} = {b,c}
        Child = {b,c} – {a} = {b,c}
        Open = APPEND ({b,c},{}) = {b,c}
    End If
End While
Return State
```



Example: Graph
V=  {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d}
S ={a:[b,c],
b:[a,c,d],
c:[a,b,d]
d:[b,c] }
Start = {a}
Goal = {d}

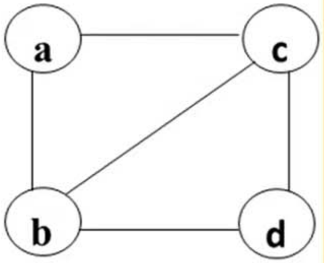
**Open= {b,c}**
**Close = {a}**

# Depth First Search (4)

## Step-2: Iteration

```
Def DFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS) then
      Pick front node 'N' from Open
      Open = Open – {N}
      If GOALTEST(N) = TRUE then
                State = SUCCESS
                Closed = APPEND (Closed , {N})
      Else
                Closed = APPEND (Closed , {N})
                Child = { MOVEGEN(N) }
                Child = {Child - Open}
                Child= {Child  - Closed}
                Open = APPEND (Child,Open)
     End If
End While
Return State
```

```
DFS({a})


State = FAILURE
While ({b,c} <> Empty) AND (FAILURE <> SUCCESS)
      N='b'
      Open = {b,c} – {b} = {c}
      If GOALTEST(b) = TRUE  then
                State = SUCCESS
                Closed = APPENDClosed , {N})
      Else
                Closed = APPEND ({a}, {b}) = {a,b}
                Child = {a,c,d}
                Child = {a,c,d} – {c} = {a,d}
                Child = {a,d} – {a} = {d}
                Open = APPEND ({d},{c}) = {d,c}
       End If
End While
Return State
```



Example: Graph<br>
V=  {a,b,c,d}<br>
E= {a-b,a-c,b-c,b-d,c-d}<br>
S ={a:[b,c],<br>
    b:[a,c,d],<br>
    c:[a,b,d]<br>
    d:[b,c] }<br>
Start = {a}<br>
Goal = {d}

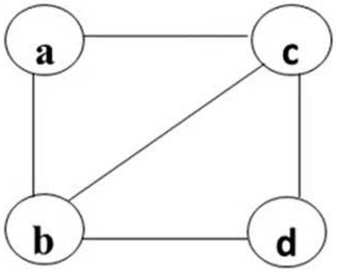
**Open= {d,c}**<br>
**Close = {a,b}**

# Depth First Search (5)

## Step-3: Iteration

```
Def DFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS) then
    Pick front node 'N' from Open
    Open = Open – {N}
    If GOALTEST(N) = TRUE then
            State = SUCCESS
            Closed = APPEND (Closed , {N})
    Else
            Closed = APPEND (Closed , {N})
            Child = { MOVEGEN(N) }
            Child = {Child - Open}
            Child= {Child  - Closed}
            Open = APPEND (Child,Open)
    End If
End While
Return State
```

```
DFS({a})


State = FAILURE
While ({d,c} <> Empty) AND (FAILURE <> SUCCESS)
    N='d'
    Open = {d,c} – {d} = {c}
    If GOALTEST(d) = TRUE then
            State = SUCCESS
            Closed = APPEND( {a,b} , {d}) = {a,b,d}
    Else
            Closed = APPEND (Closed , {N})
            Child = { MOVEGEN(N) }
            Child = {Child - Open}
            Child= {Child  - Closed}
            Open = APPEND (Child,Open)
    End If
End While
Return State
```



Example: Graph
V= {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d}
S ={a:[b,c],
b:[a,c,d],
c:[a,b,d]
d:[b,c] }
Start = {a}
Goal = {d}

**Open= {c}**
**Close = {a,b,d}**

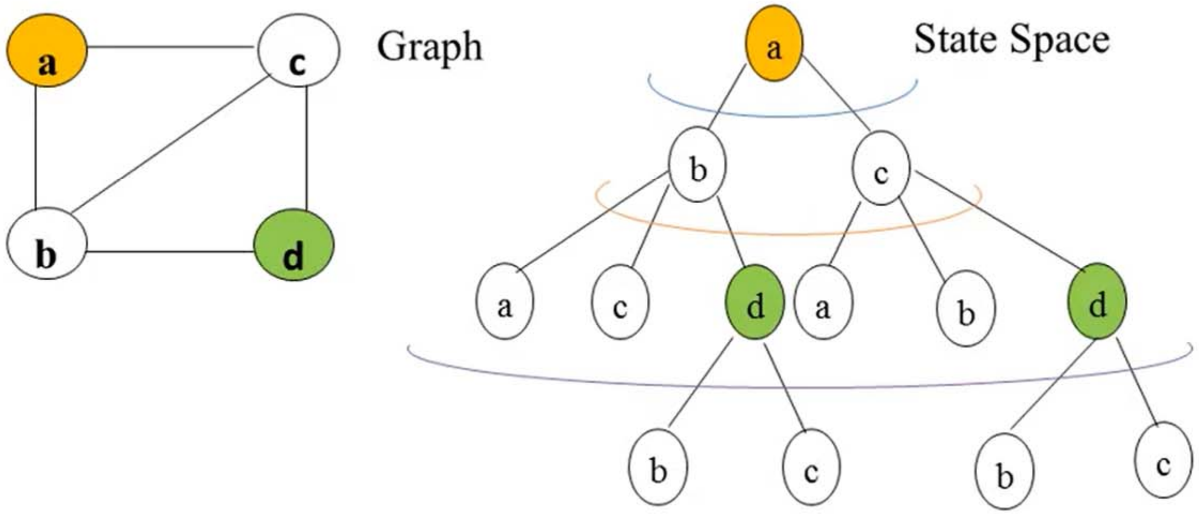
# Breadth First Search (1)

The **Breadth First Search** (**BFS**) is a search technique which searches the tree ***level wise*** and ***left to right*** at each level of the tree.

The **BFS** uses ***Queue*** as a data structure (**OPEN** list) to process the given state space.

Example: Graph
V= {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d}

Start = {a}
Goal = {d}



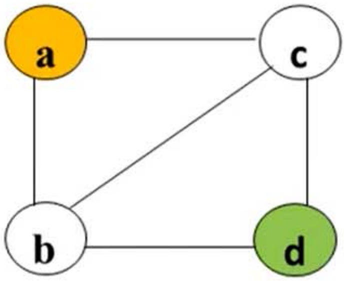
S ={ a:[b,c], b:[a,c,d], c:[a,b,d], d:[b,c] }

# Breadth First Search (2)

## Algorithm BFS()

```
Def BFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS) then
      Pick  front node 'N' from Open
      Open = Open – {N}
      If GOALTEST(N) = TRUE  then
            State = SUCCESS
            Closed = APPEND (Closed , {N})
      Else
            Closed = APPEND (Closed , {N})
            Child = { MOVEGEN(N) }
            Child = {Child - Open}
            Child= {Child  - Closed}
            Open = APPEND (Open, Child)
      End If
End While
Return State
```

```
Def GOALTEST(N)
 If N = Goal then
   Return TRUE
 Else
   Return FALSE

Def MOVEGEN(N)
Succ ={}
For N in S do
   Succ= Succ U  {Children of N}

Return Succ

Def APPEND(list1, list2)
New_list = list1 + list2
Return New_list
```



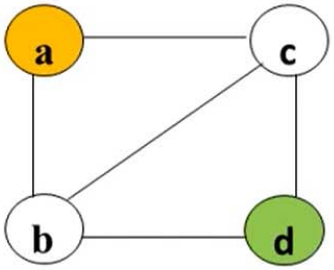
Example: Graph  
V= {a,b,c,d}  
E= {a-b,a-c,b-c,b-d,c-d}  
S ={a:[b,c],  
b:[a,c,d],  
c:[a,b,d]  
d:[b,c] }  
Start = {a}  
Goal = {d}

# Breadth First Search (3)

## Step-1: Iteration

```
Def BFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS)
    Pick front node 'N' from Open
    Open = Open – {N}
    If GOALTEST(N) = TRUE then
            State = SUCCESS
            Closed = APPEND (Closed , {N})
    Else
            Closed = APPEND (Closed , {N})
            Child = { MOVEGEN(N) }
            Child = {Child - Open}
            Child= {Child  - Closed}
            Open = APPEND (Open, Child)
    End If
End While
Return State
```

```
BFS({a})
Open  = {a}
Closed = {}
State = FAILURE
While ({a} <> Empty) AND (FAILURE <> SUCCESS)
    N='a'
    Open = {a} – {a} = {}
    If GOALTEST(a) = TRUE  then
            State = SUCCESS
            Closed = APPEND (Closed , {N})
    Else
            Closed = APPEND ({}, {a}) = {a}
            Child = {b,c}
            Child = {b,c} – {} = {b,c}
            Child = {b,c} – {a} = {b,c}
            Open = APPEND ({}, {b,c}) = {b,c}
    End If
End While
Return State
```



Example: Graph<br>
V=  {a,b,c,d}<br>
E= {a-b,a-c,b-c,b-d,c-d}<br>
S ={a:[b,c],<br>
    b:[a,c,d],<br>
    c:[a,b,d]<br>
    d:[b,c] }<br>
Start = {a}<br>
Goal = {d}

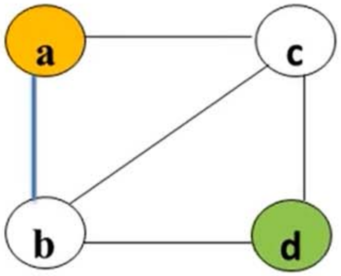
**Open= {b,c}**<br>
**Close = {a}**

# Breadth First Search (4)

## Step-2: Iteration

```
Def BFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS) then
    Pick front node 'N' from Open
    Open = Open – {N}
    If GOALTEST(N) = TRUE then
        State = SUCCESS
        Closed = APPEND (Closed , {N})
    Else
        Closed = APPEND (Closed , {N})
        Child = { MOVEGEN(N) }
        Child = {Child - Open}
        Child= {Child  - Closed}
        Open = APPEND (Open, Child)
    End If
End While
Return State
```

```
BFS({a})


State = FAILURE
While ({b,c} <> Empty) AND (FAILURE <> SUCCESS)
    N='b'
    Open = {b,c} – {b} = {c}
    If GOALTEST(b) = TRUE then
        State = SUCCESS
        Closed = APPEND(Closed , {N})
    Else
        Closed = APPEND ({a}, {b}) = {a,b}
        Child = {a,c,d}
        Child = {a,c,d} – {c} = {a,d}
        Child = {a,d} – {a,b} = {d}
        Open = APPEND ({c}, {d}) = {c,d}
    End If
End While
Return State
```



Example: Graph
V=  {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d}
S ={a:[b,c],
    b:[a,c,d],
    c:[a,b,d]
    d:[b,c] }
Start = {a}
Goal = {d}

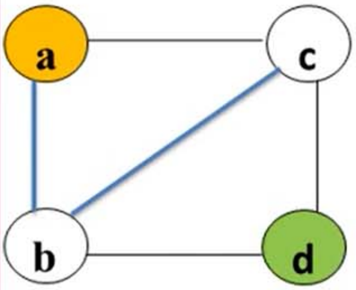
**Open= {c,d}**
**Close = {a,b}**

# Breadth First Search (5)

## Step-3: Iteration

```
Def BFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS) then
    Pick front node 'N' from Open
    Open = Open – {N}
    If GOALTEST(N) = TRUE then
        State = SUCCESS
        Closed = APPEND (Closed , {N})
    Else
        Closed = APPEND (Closed , {N})
        Child = { MOVEGEN(N) }
        Child = {Child - Open}
        Child= {Child  - Closed}
        Open = APPEND (Open, Child)
    End If
End While
Return State
```

```
BFS({a})


State = FAILURE
While ({c,d} <> Empty) AND (FAILURE <> SUCCESS)
    N='c'
    Open = {c,d} – {c} = {d}
    If GOALTEST(c) = TRUE then
        State = SUCCESS
        Closed = APPEND(Closed , {N})
    Else
        Closed = APPEND( {a,b} , {c}) = {a,b,c}
        Child = {a,b,d}
        Child = {a,b,d} – {d} = {a,b}
        Child = {a,b} – {a,b,c} = {}
        Open = APPEND ({d}, {}) = {d}
    End If
End While
Return State
```



Example: Graph  
V= {a,b,c,d}  
E= {a-b,a-c,b-c,b-d,c-d}  
S ={a:[b,c],  
b:[a,c,d],  
c:[a,b,d]  
d:[b,c] }  
Start = {a}  
Goal = {d}

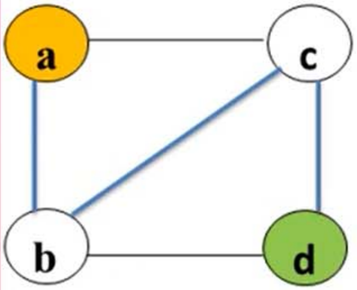
**Open= {}**  
**Close = {a,b,c}**

# Breadth First Search (6)

## Step-4: Iteration

```
Def BFS(Start)
Open=Start
Closed={}
State = FAILURE
While (Open <> Empty ) AND  (State <> SUCCESS) then
    Pick front node 'N' from Open
    Open = Open – {N}
    If GOALTEST(N) = TRUE then
        State = SUCCESS
        Closed = APPEND (Closed , {N})
    Else
        Closed = APPEND (Closed , {N})
        Child = { MOVEGEN(N) }
        Child = {Child - Open}
        Child= {Child  - Closed}
        Open = APPEND (Open, Child)
    End If
End While
Return State
```

```
BFS({a})


State = FAILURE
While ({d} <> Empty) AND (FAILURE <> SUCCESS)
    N='d'
    Open = {d} – {d} = {}
    If GOALTEST(d) = TRUE then
        State = SUCCESS
        Closed = APPEND({a,b,c} , {d}) = {a,b,c,d}
    Else
        Closed = APPEND (Closed , {N})
        Child = { MOVEGEN(N) }
        Child = {Child - Open}
        Child= {Child  - Closed}
        Open = APPEND (Open, Child)
    End If
End While
Return State
```



Example: Graph
V=  {a,b,c,d}
E= {a-b,a-c,b-c,b-d,c-d}
S ={a:[b,c],
    b:[a,c,d],
    c:[a,b,d]
    d:[b,c] }
Start = {a}
Goal = {d}

**Open= {d}**
**Close = {a,b,c,d}**