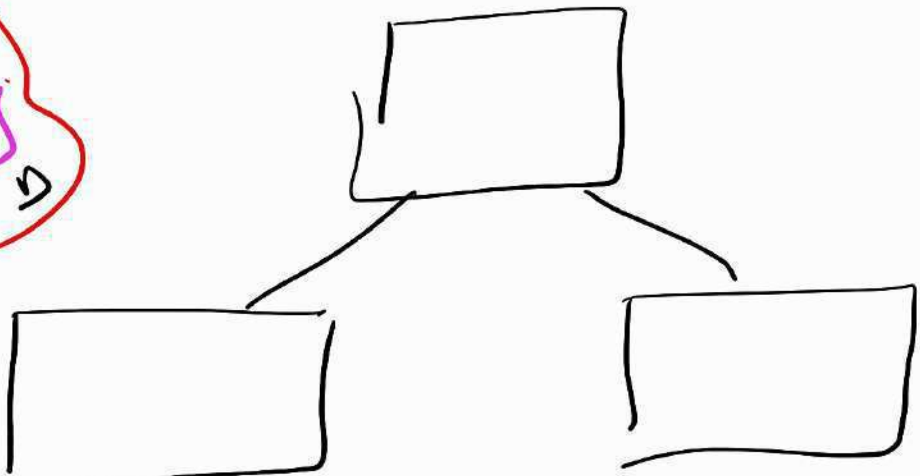
Informed Search:-

→ Utilizes additional knowledge to guide search process. while the other explores without specific information (or) guidance.

State-Space Search:-

Given a State Space.



Search Algorithm:-

How to intelligently explore a state space by following state transformation rules.

→ Choose an suitable search algorithm from the class of search problems.

→ Size of State Space.

Quick Recap

(1) States :- Perfect Info / Partial Info.

(a) Set of variables that define a state

(b) Domains for every variable.

(2) State Transformation Rules

(a) Set of valid rules — Deterministic outcome / Non-deterministic outcome

(3) Implicit Graph / State space.

(a) Single player : OR Graph

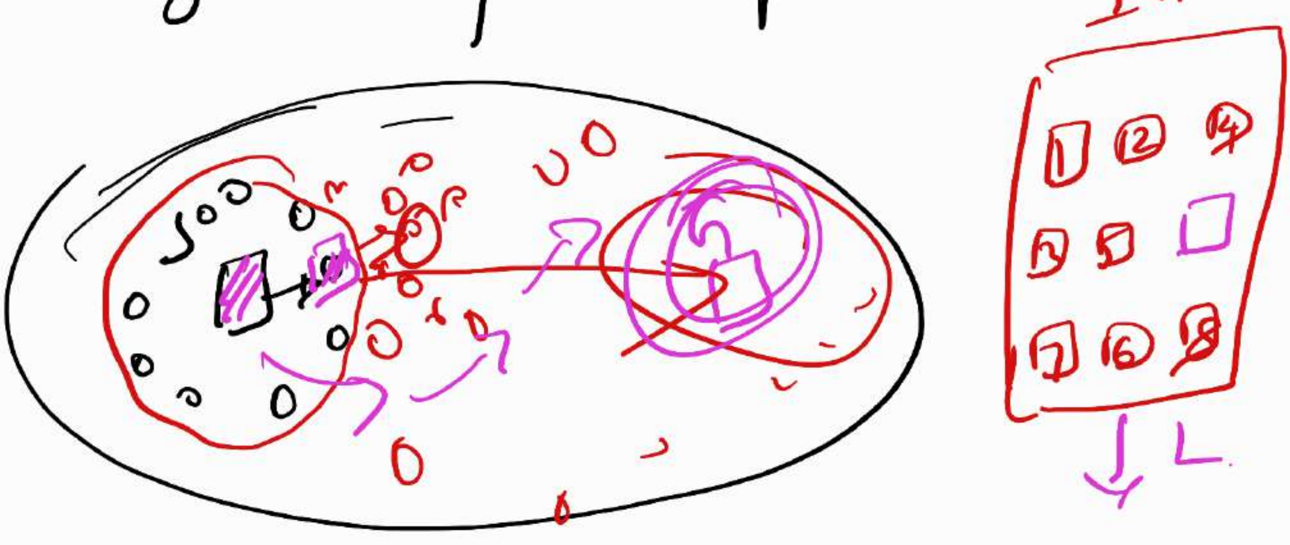
(b) Multiple player :- And/or graph, probabilistic graph

(4) Path Cost

Ex :- Move the peg from one valid state to another valid state.

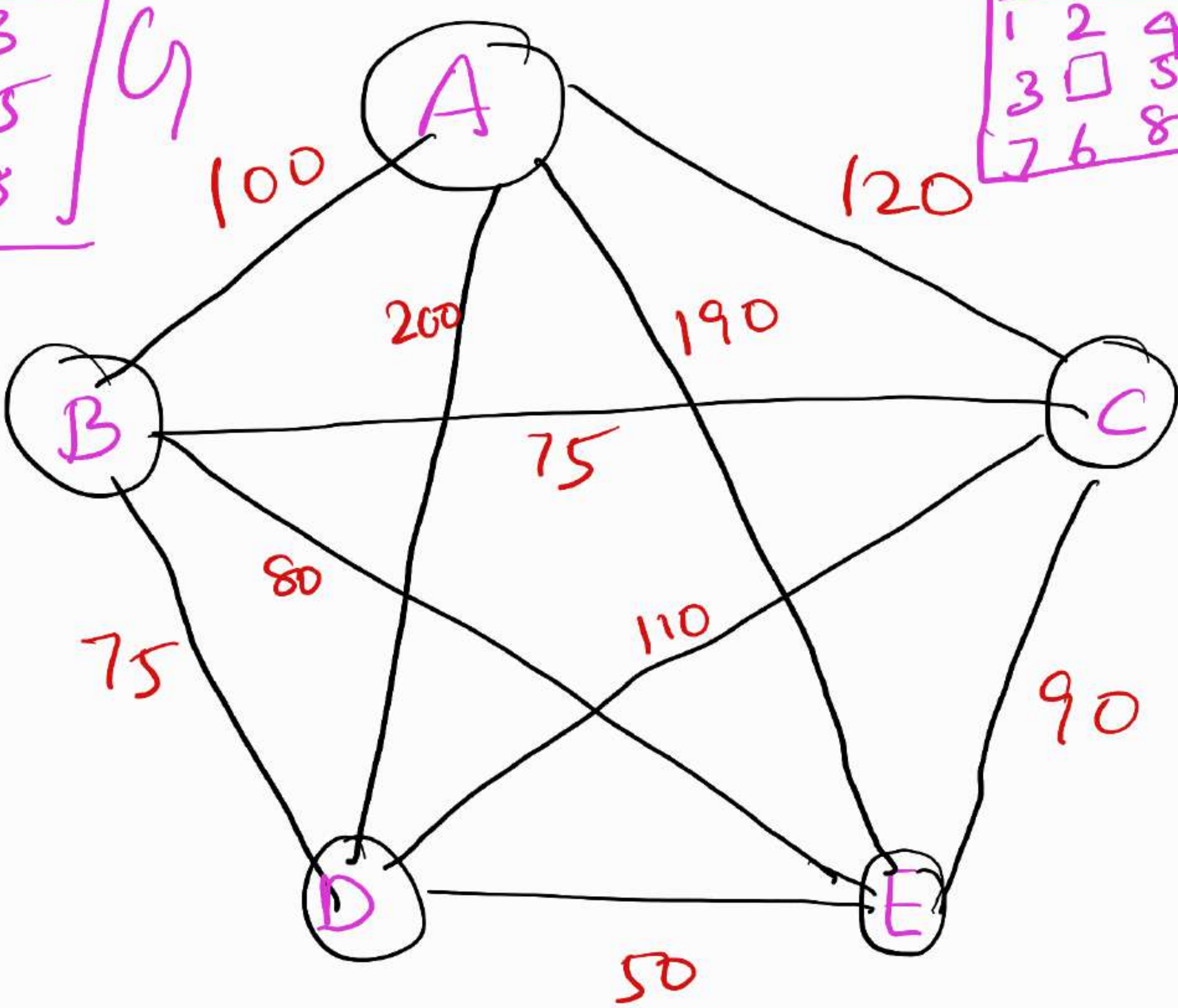
Traveling Sales person problem.

Initial

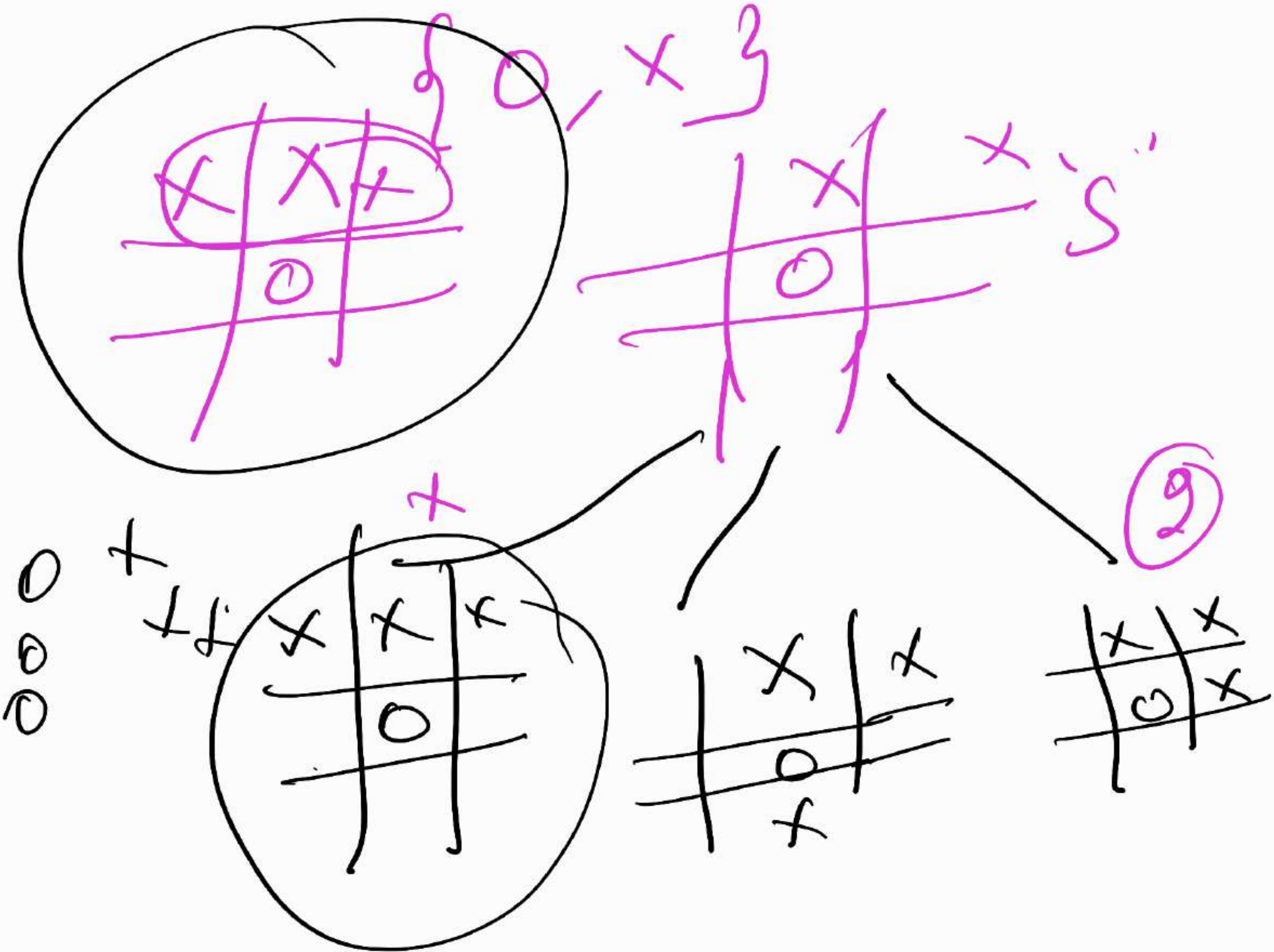


1	2	3
4	$\square$	5
6	7	8

1	2	4
3	$\square$	5
7	6	8



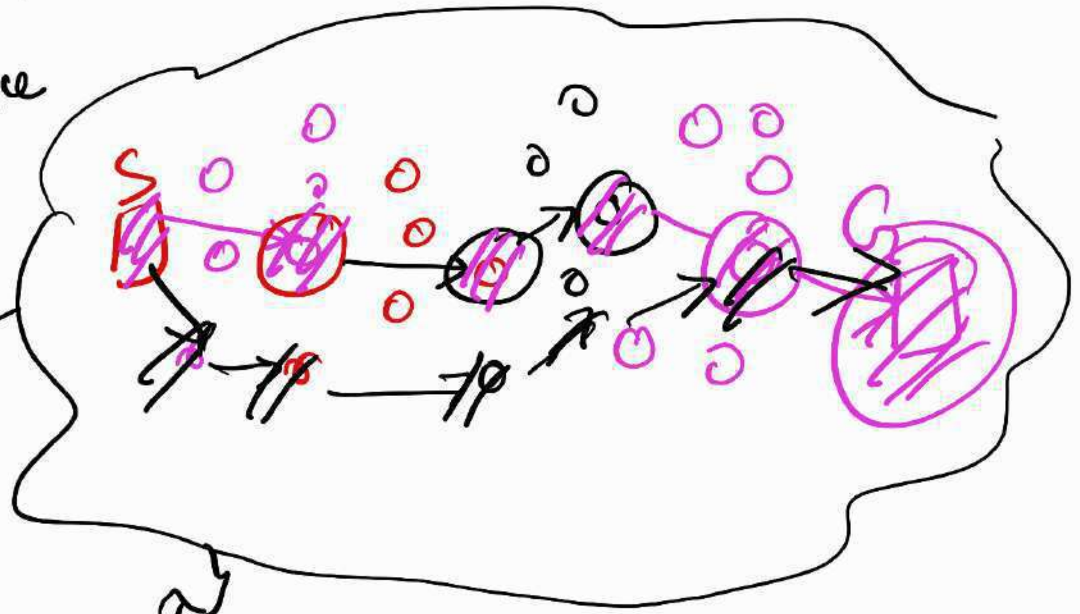
{O, X}



Domain -Independent Algorithms

User needs to understand the domain description and adopt to suitable problem solving.

State-Space Search



MoveGen(S) - A domain function

Goaltest(N) - A domain function

MoveGen(N) $\rightarrow$ {-, -O, -}

Define a State:-

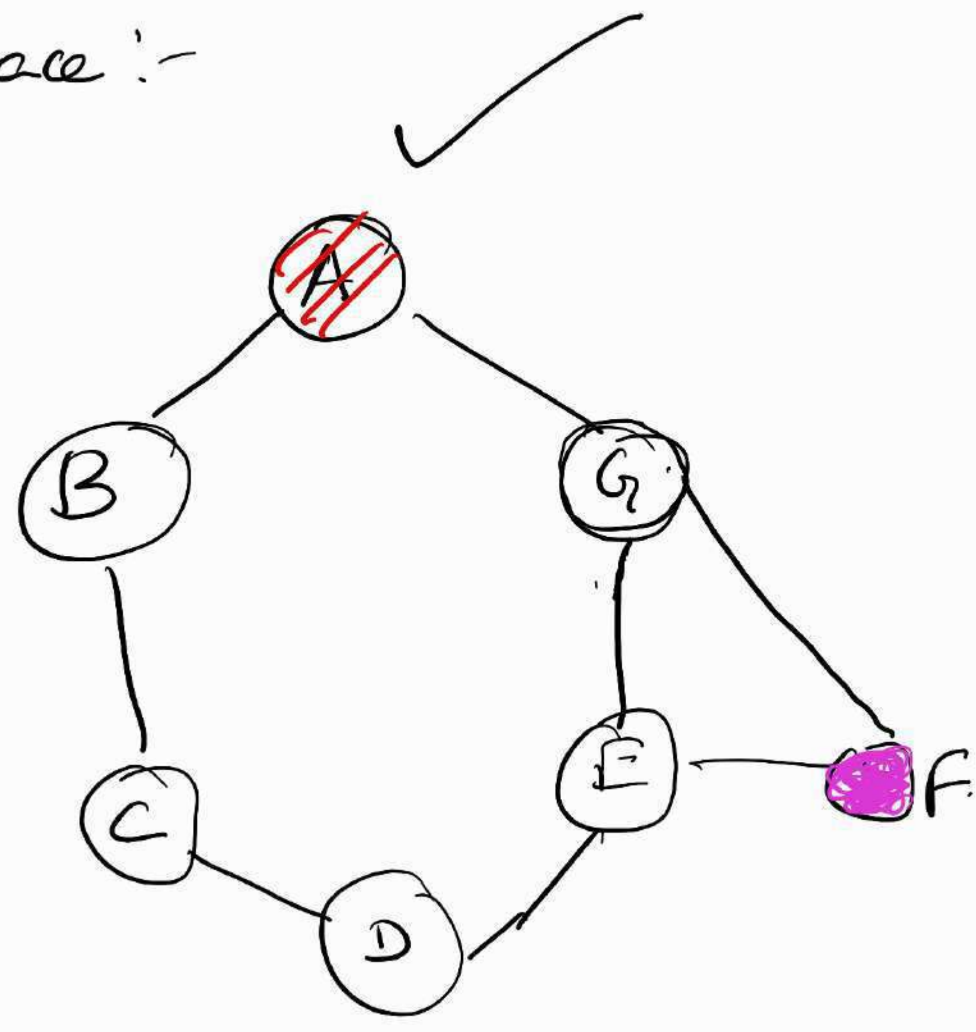
A Search algorithm must Choose a from set of Candidates.

OPEN:-

Search Algorithm:-

1. Generate & Test
2. Traverse.
3. Check.

State Space:-



Search Space:-

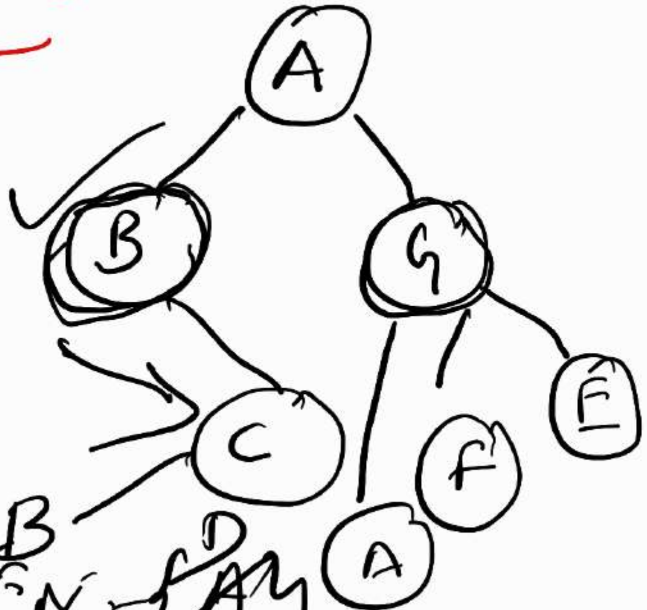
S :- {A}

G :- {F}

MoveGen( )

A	{B, G}
B	{A, C}
C	{B, D}
D	{C, E}
E	{F, G, D}
G	{A, F, E}
F	{E, G}

Search Tree:-



OPEN: ~~{A}~~

OPEN: {B, G

OPEN = {G, C, D}

Closed: {A, B}

basic search ( ) // with Closed set

OPEN ← {S}
Closed ← { }
While OPEN is not empty

(Head, tail) do pick some 'N' from OPEN

OPEN ← ~~OPEN ← OPEN - {N}~~

Closed ← Closed ∪ {N}

Closed ← → $\boxed{\text{IF GoalTest(N) = TRUE}}$

then return N

else

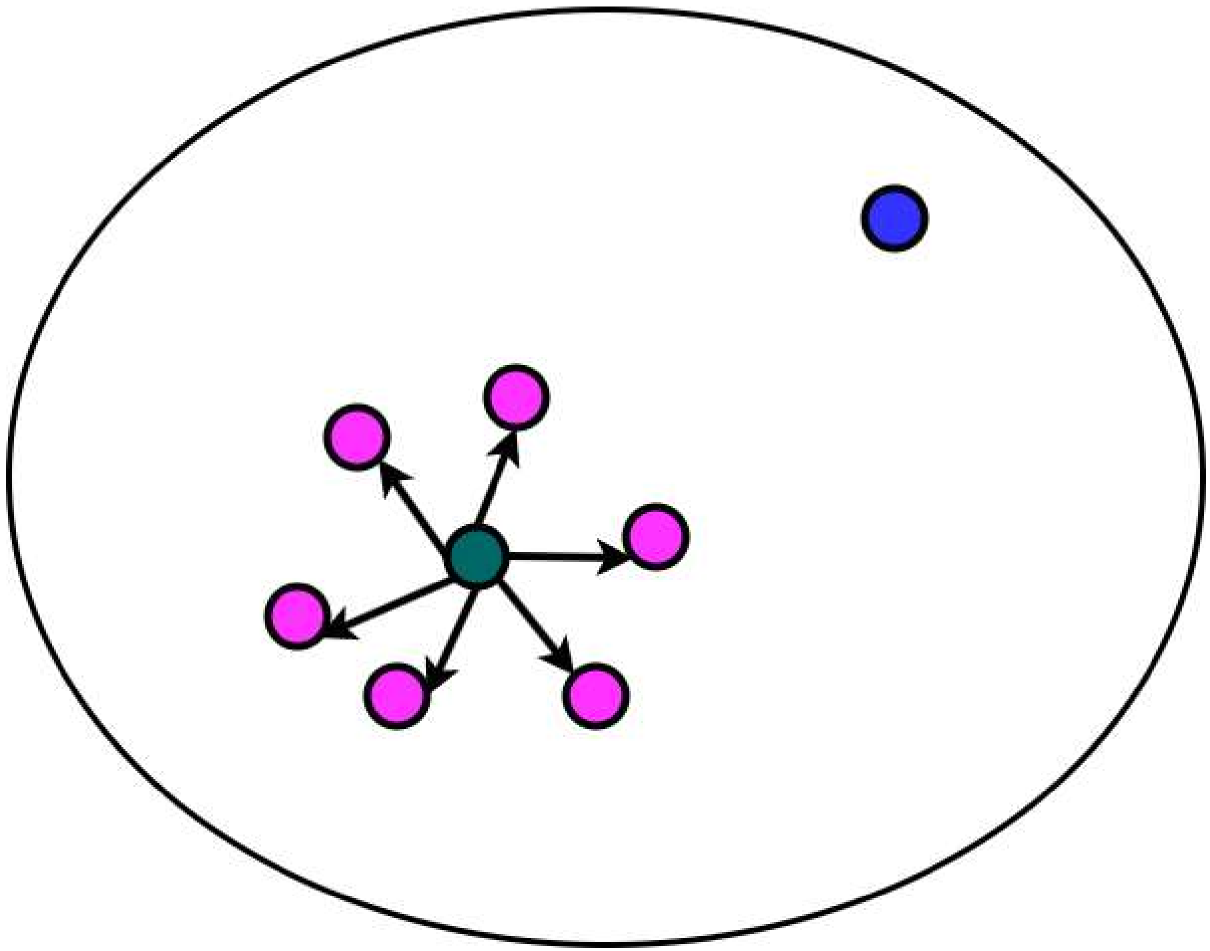
SET DIFFERENCE

OPEN ← OPEN ∪ {MOVEGEN(N) - Closed}

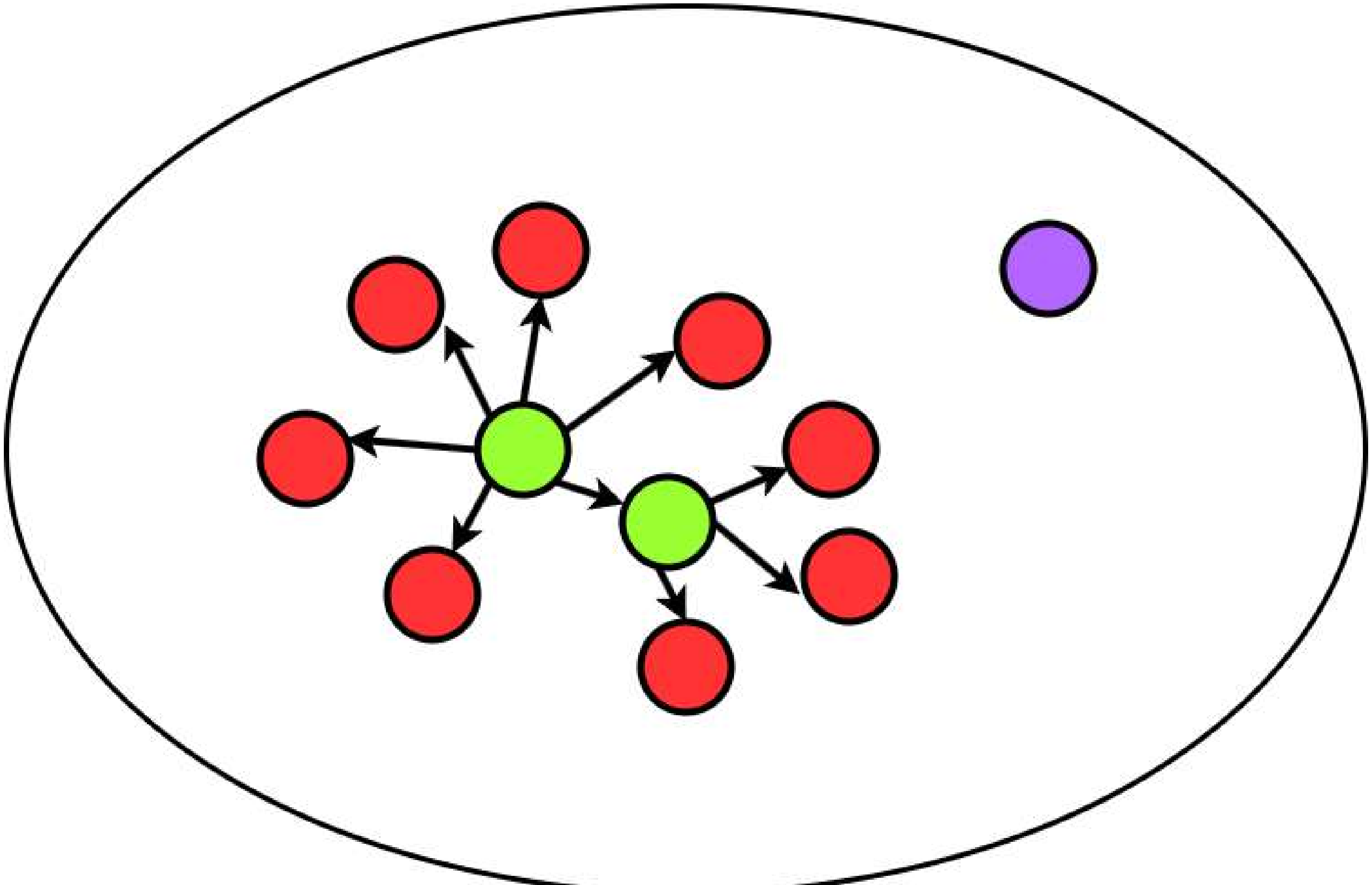
return failure.

- Already on OPEN)

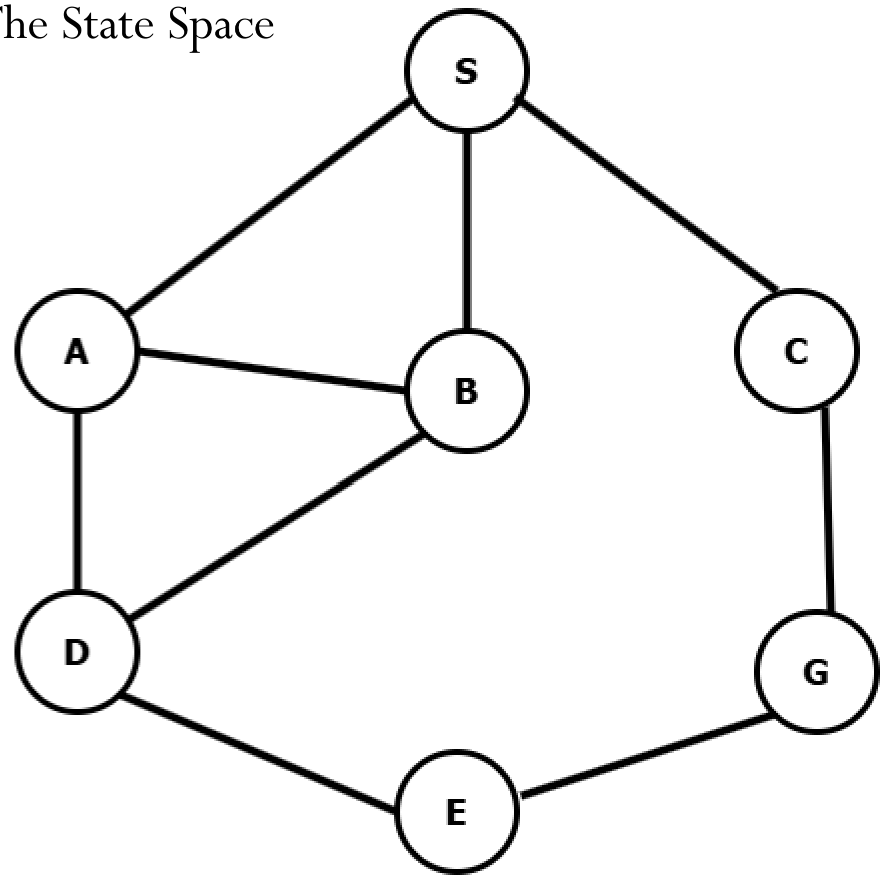
Moves:State transformation



The set OPEN of candidates



The State Space



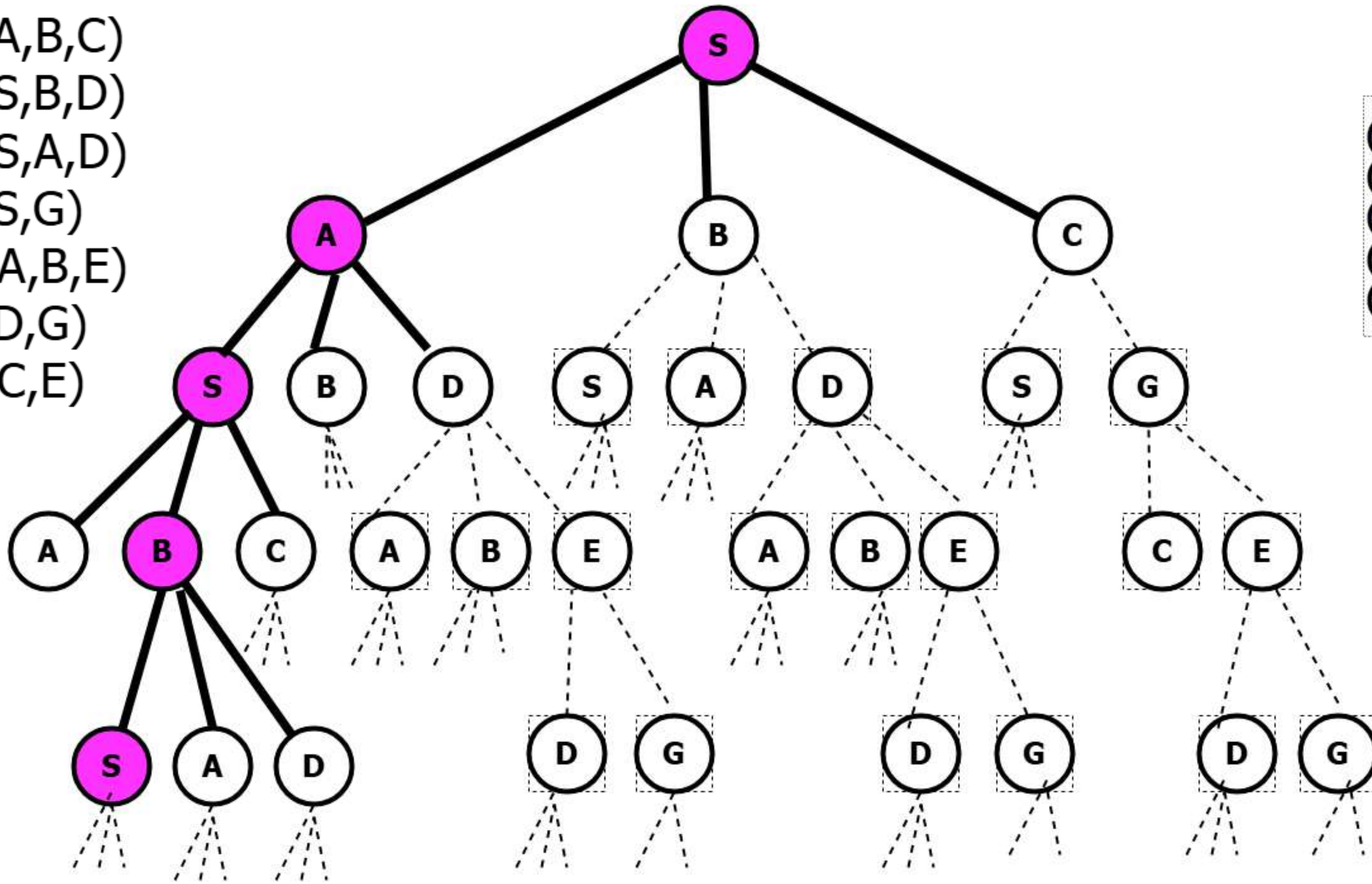
The MoveGen Function

S->(A,B,C)
A->(S,B,D)
B->(S,A,D)
C->(S,G)
D->(A,B,E)
E->(D,G)
G->(C,E)

Basicsearch

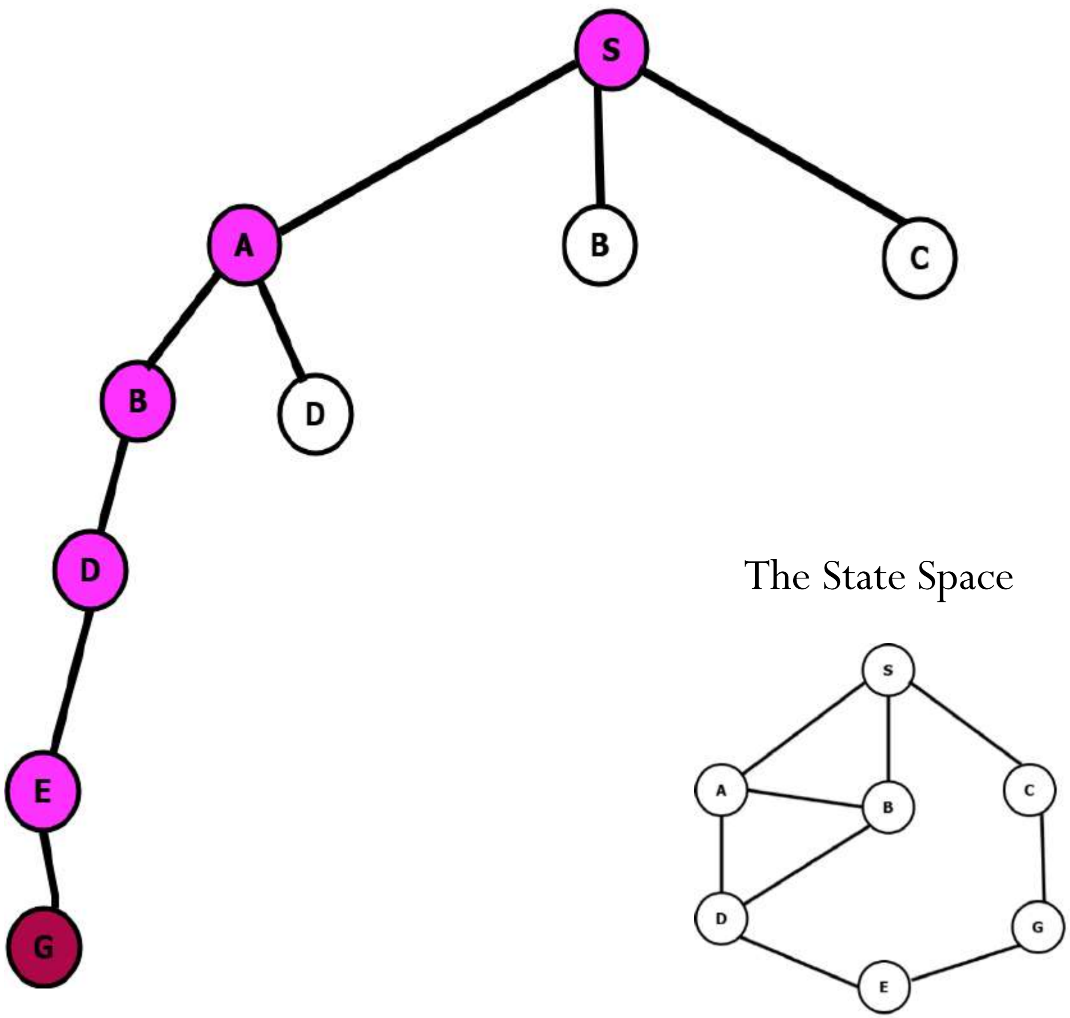
The MoveGen Function

S->(A,B,C)
A->(S,B,D)
B->(S,A,D)
C->(S,G)
D->(A,B,E)
E->(D,G)
G->(C,E)



(S)
(ABC)
(SBDBC)
(ABCBDBC)
(ASADCBDBC)

Search Tree for a basic search2



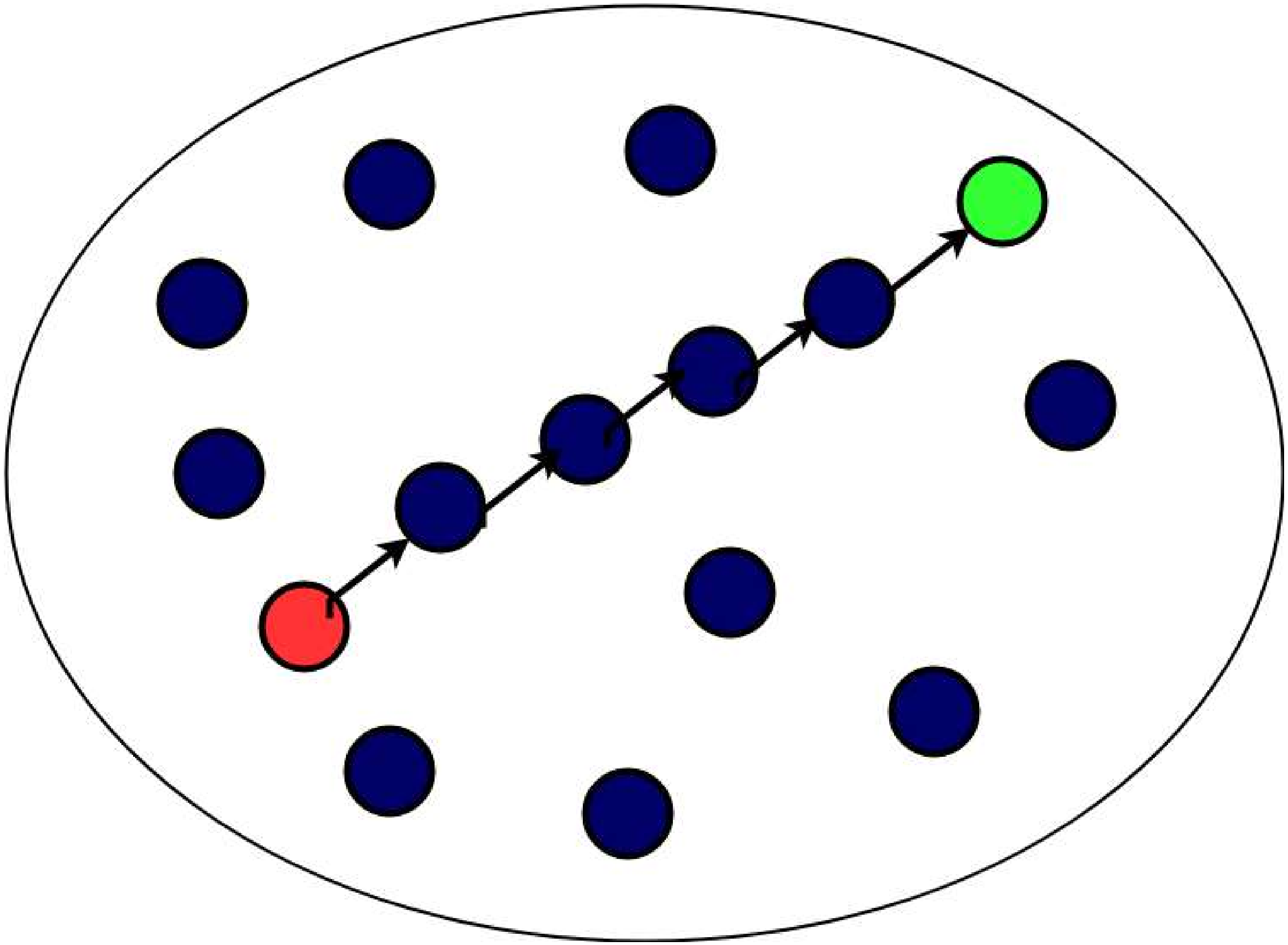
OPEN	CLOSED
(S)	()
(ABC)	(S)
(BDBC)	(AS)
(DDBC)	(BAS)
(EDBC)	(DBAS)
(GDBC)	(EDBAS)

The MoveGen Function

S->(A,B,C)
A->(S,B,D)
B->(S,A,D)
C->(S,G)
D->(A,B,E)
E->(D,G)
G->(C,E)

The State Space

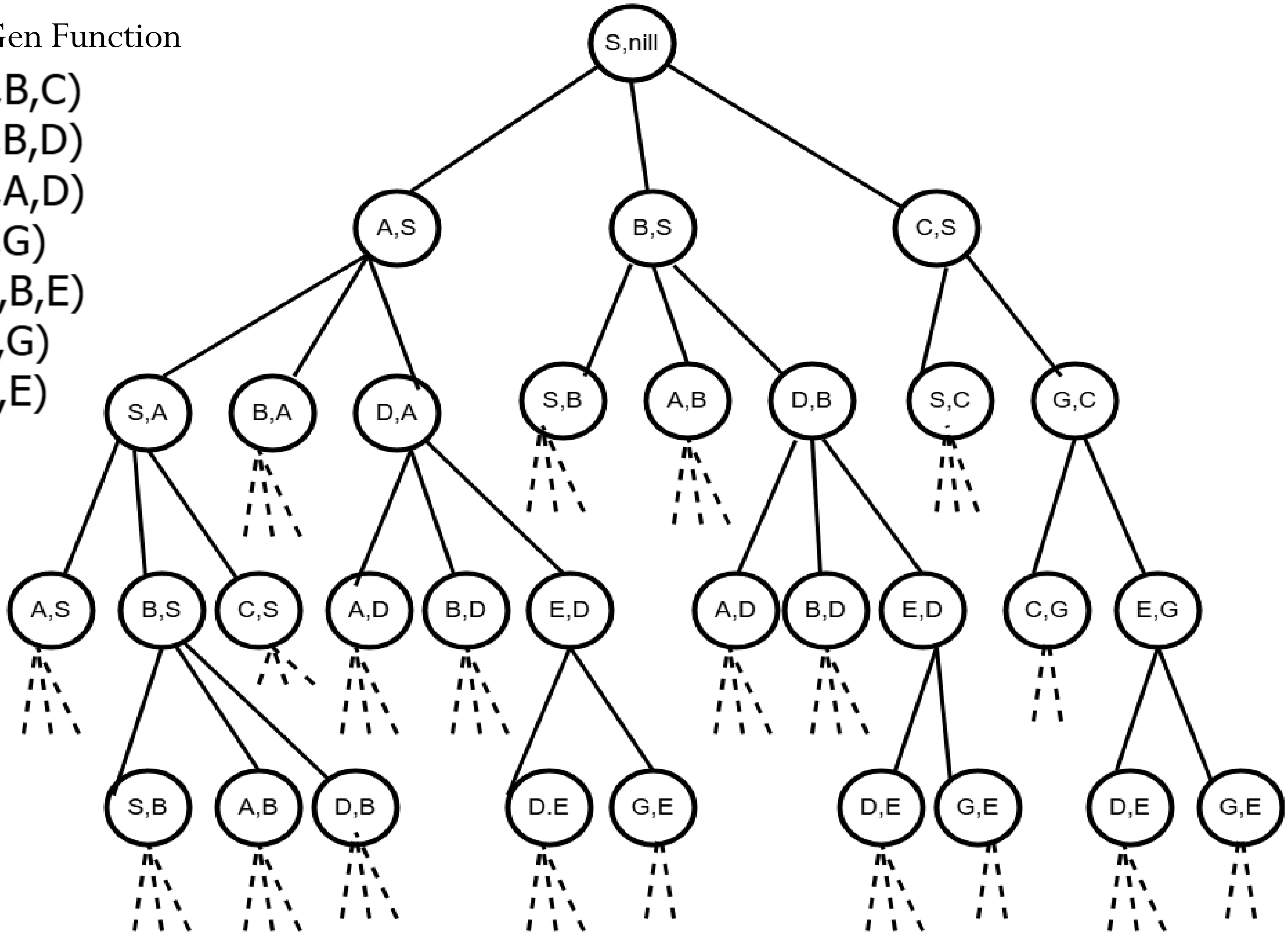
A Solution



Node Pairs in the Search Tree

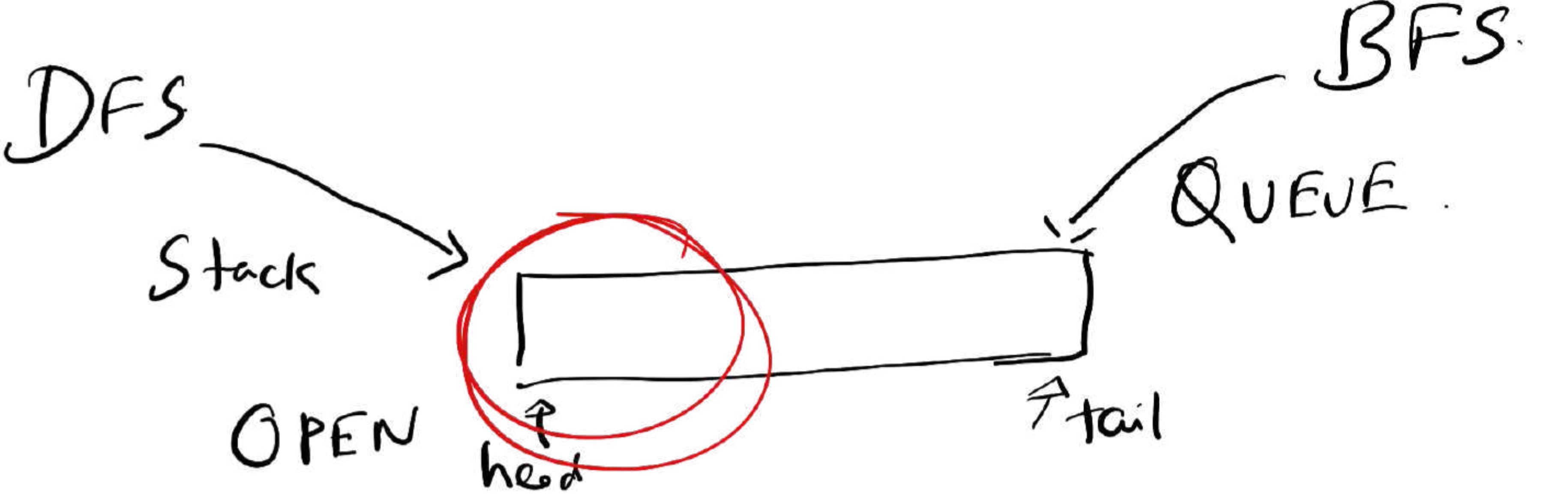
The MoveGen Function

S->(A,B,C)
A->(S,B,D)
B->(S,A,D)
C->(S,G)
D->(A,B,E)
E->(D,G)
G->(C,E)

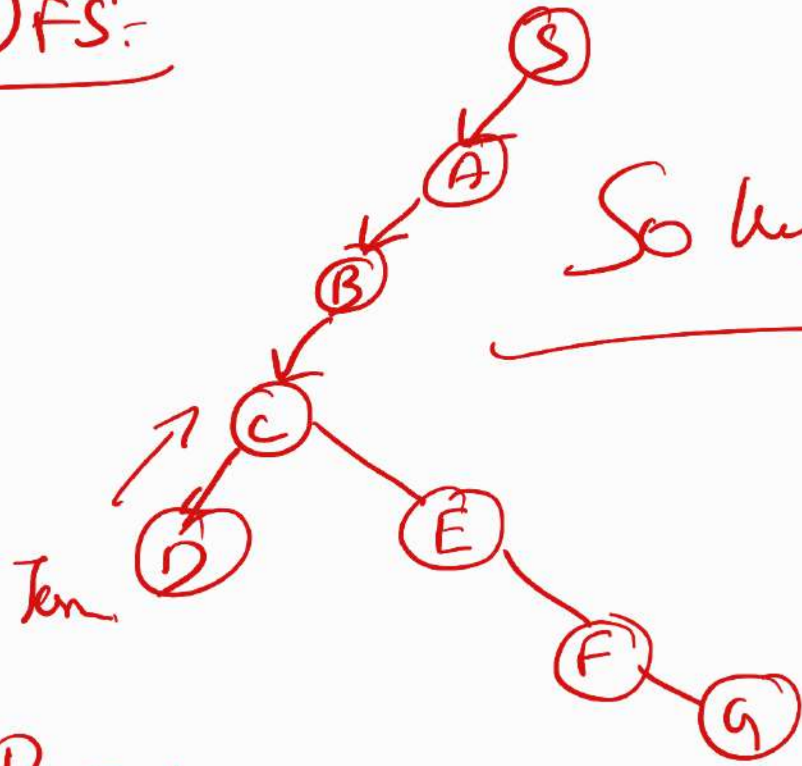


In BFS & DFS:-

* Instead of set, linked list datastructe is used
* So, the exploration of state starts from head node.
* Both BFS & DFS explore the same search tree. But, order is diffarent.



DFS:-

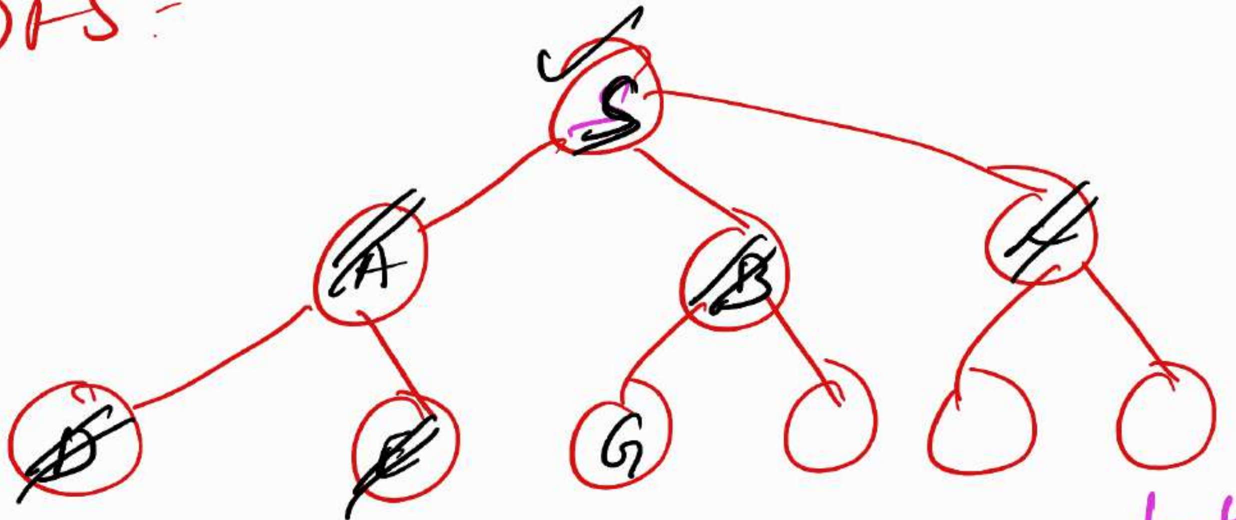


G.

Solution path

OPEN:-

BFS:-

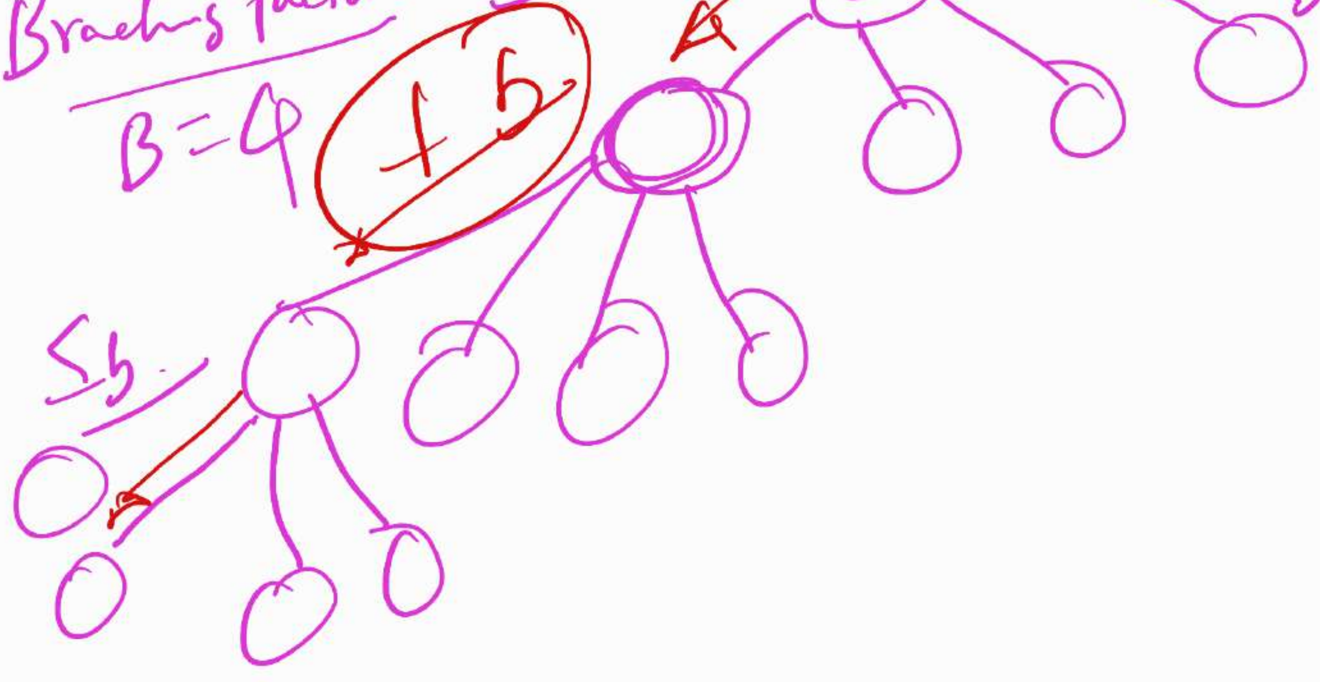


b + b + b

Branching factor:- 5

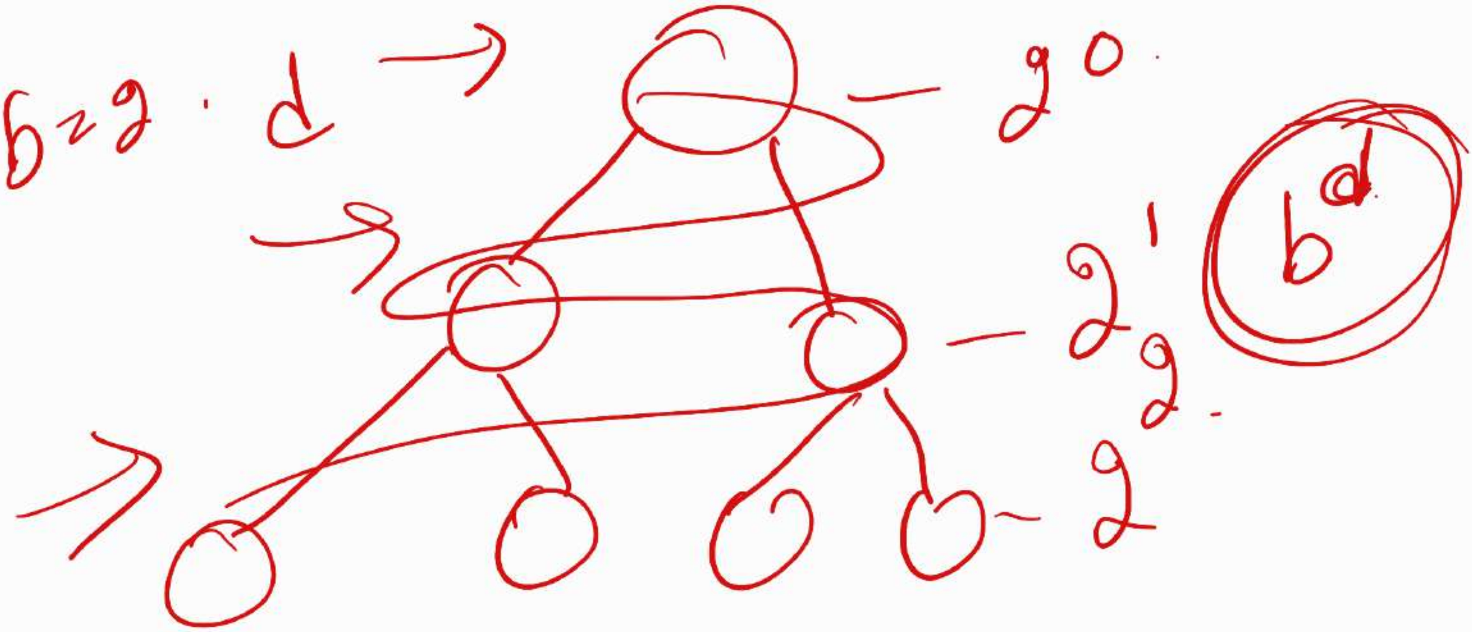
$B = 4$

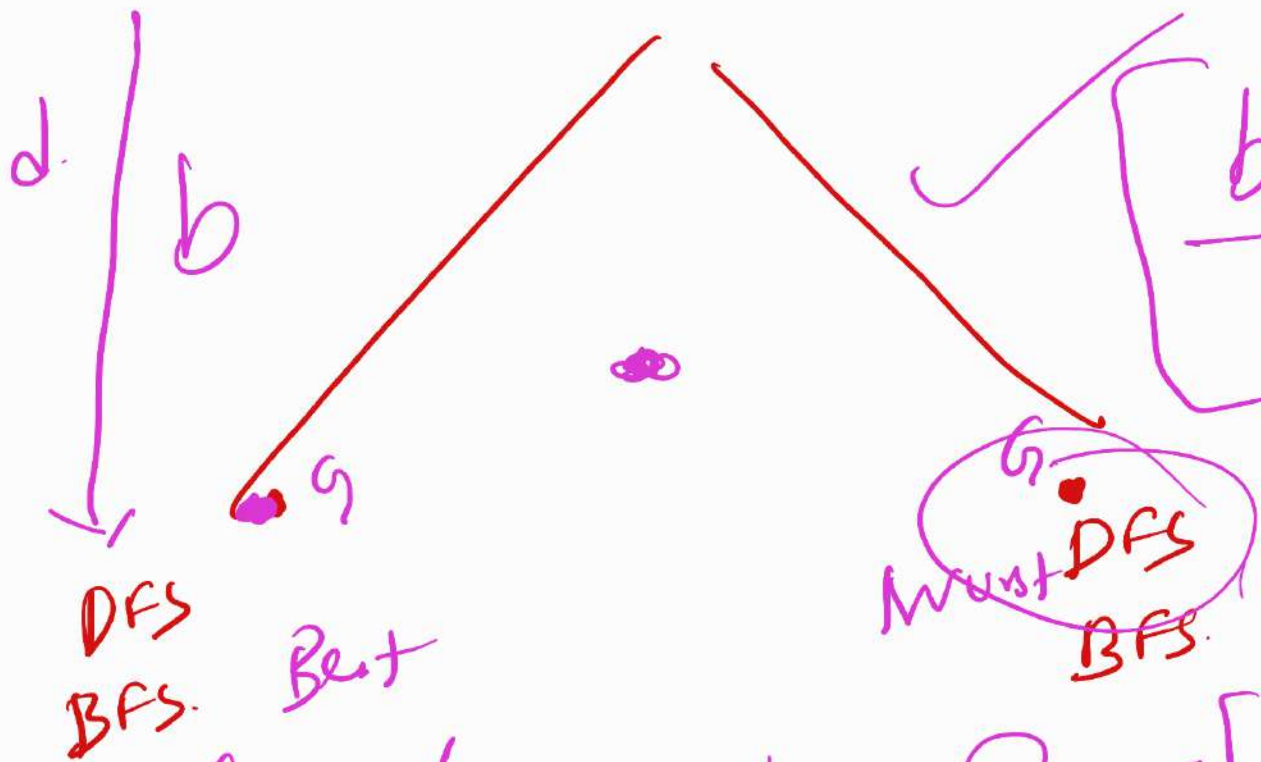
+b



b =

5b.





$$\left[\frac{b^d-1}{b-1}\right]$$

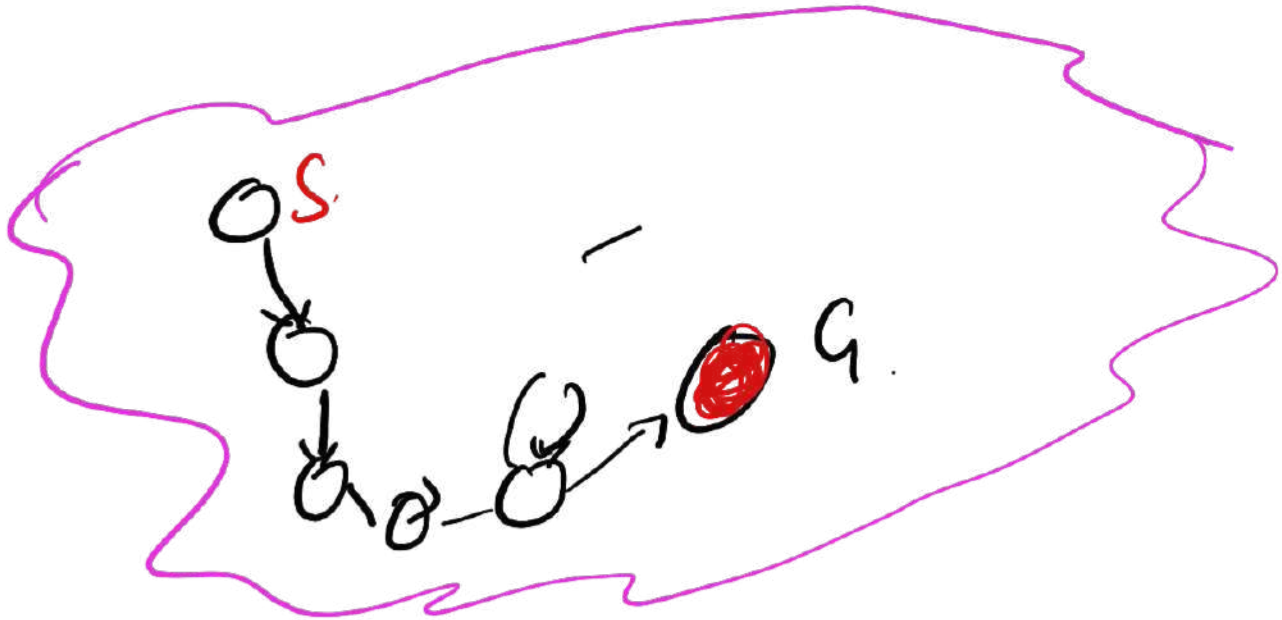
$$\delta_{DFS} = d+1 \qquad \Omega_{DFS} = \left[\frac{b^{d+1}-1}{b-1}\right]$$

$$\delta_{BFS} = \left[\frac{b^d-1}{b-1}\right]+1 \qquad \Omega_{BFS} = \left[\frac{b^{d+1}-1}{b-1}\right]$$

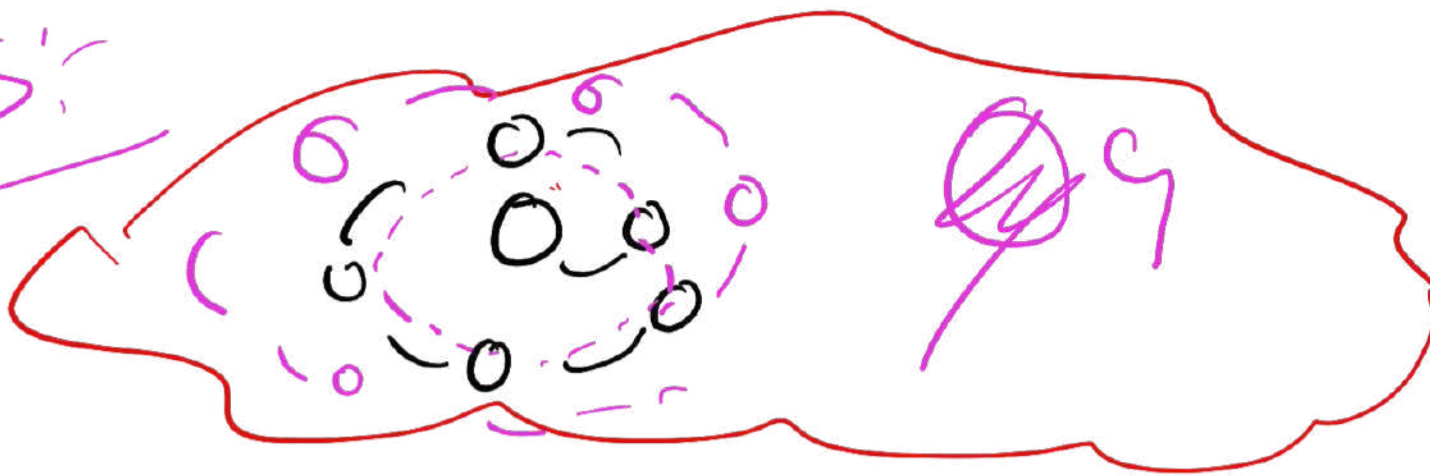
Exponentially $\rightarrow$ $O(b^d) \qquad O(b^d)$

Convergence

DFS:-



BFS:-



OPEN=STACK

```
DFS(S)
OPEN ← (S, null) : [ ]
CLOSED ← empty list
while OPEN is not empty
    nodePair ← head OPEN
    (N, _ ) ← nodePair
    if GoalTest(N) = TRUE
        return ReconstructPath(nodePair, CLOSED)
    else CLOSED ← nodePair : CLOSED
        children ← MoveGen(N)
        newNodes ← RemoveSeen(children, OPEN, CLOSED)
        newPairs ← MakePairs(newNodes, N)
        OPEN ← newPairs ++ (tail OPEN)
return empty list
```

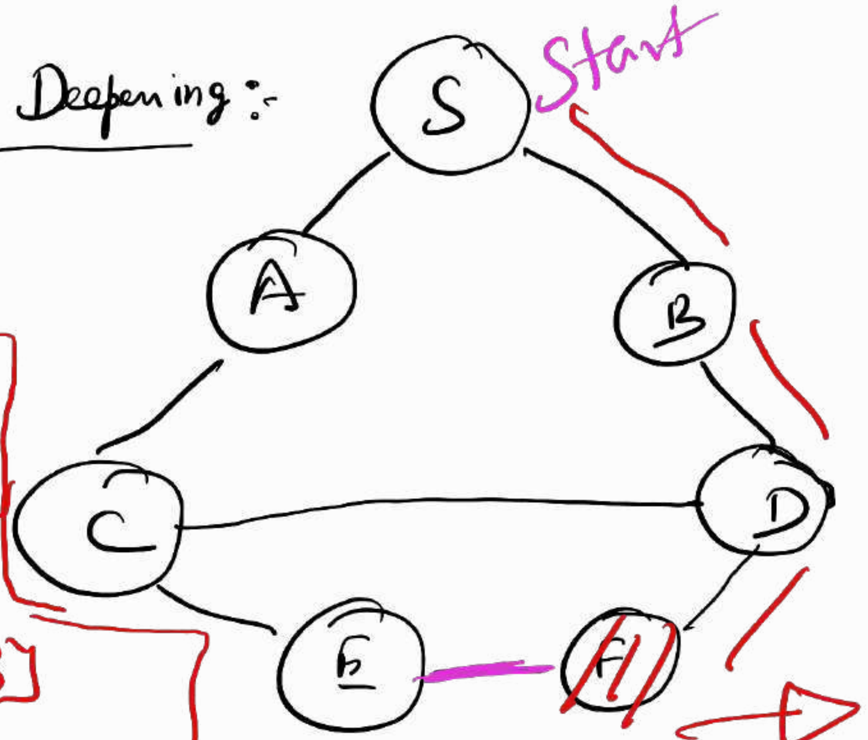
For BFS(S): //OPEN=QUEUE

Replace the line 13 in DFS (S) , with

OPEN<----- (tail OPEN) ++ new Pairs // So, QUEUE datastructure

(Linear/Exponential	DFS	BFS
TIME	Exponential	Exponential.
SPACE	Linear	Exponential.
Solution	No Guarantee (Infinite Space)	Shortest Path.
Completeness	Not for Infinite Search Space	Path exits

Iterative-Deepening :-



MONEGEN

$S \rightarrow \{A, B\}$

$A \rightarrow \{S, C\}$

$B - \{S, D\}$

$C \rightarrow \{A, D, E\}$

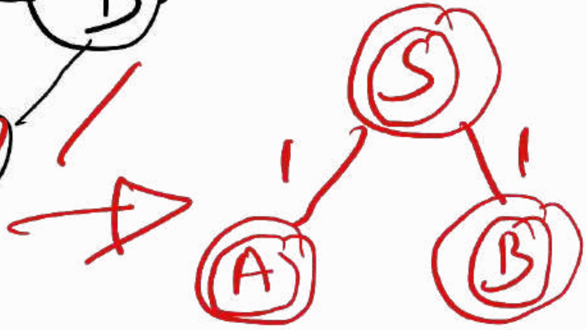
$D \rightarrow \{B, C, F\}$

$E \rightarrow \{F, C\}$

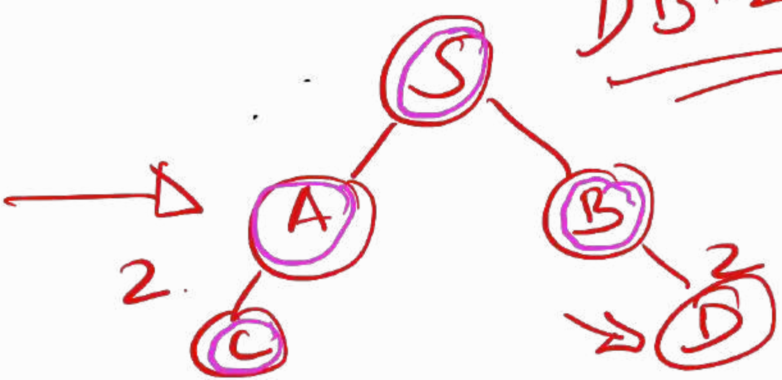
$F \rightarrow \{E, D\}$

DB-0

DB-1



DB-2

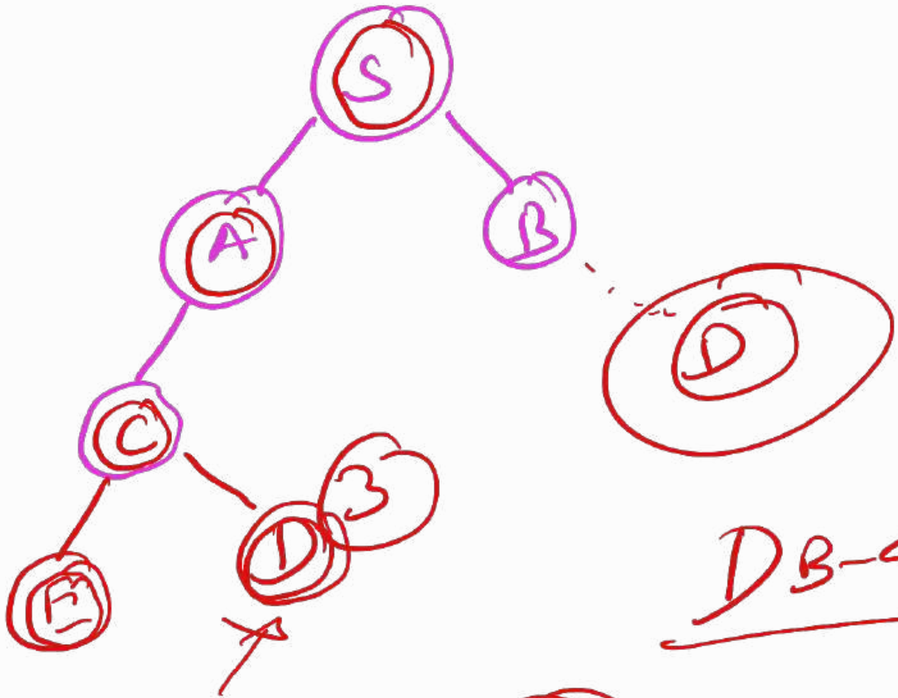


```
DB-DFS(S, depthBound)
    OPEN ← (S, null, 0) : [ ]
    CLOSED ← empty list
    while OPEN is not empty
        nodePair ← head OPEN
        (N, _, depth) ← nodePair
        if GOALTEST(N) = TRUE
            return RECONSTRUCTPATH(nodePair, CLOSED)
        else CLOSED ← nodePair : CLOSED
            if depth < depthBound
                children ← MOVEGEN(N)
                newNodes ← REMOVESEEN(children, OPEN, CLOSED)
                newPairs ← MAKEPAIRS(newNodes, N, depth + 1)
                OPEN ← newPairs ++ tail OPEN
            else OPEN ← tail OPEN
    return empty list
```

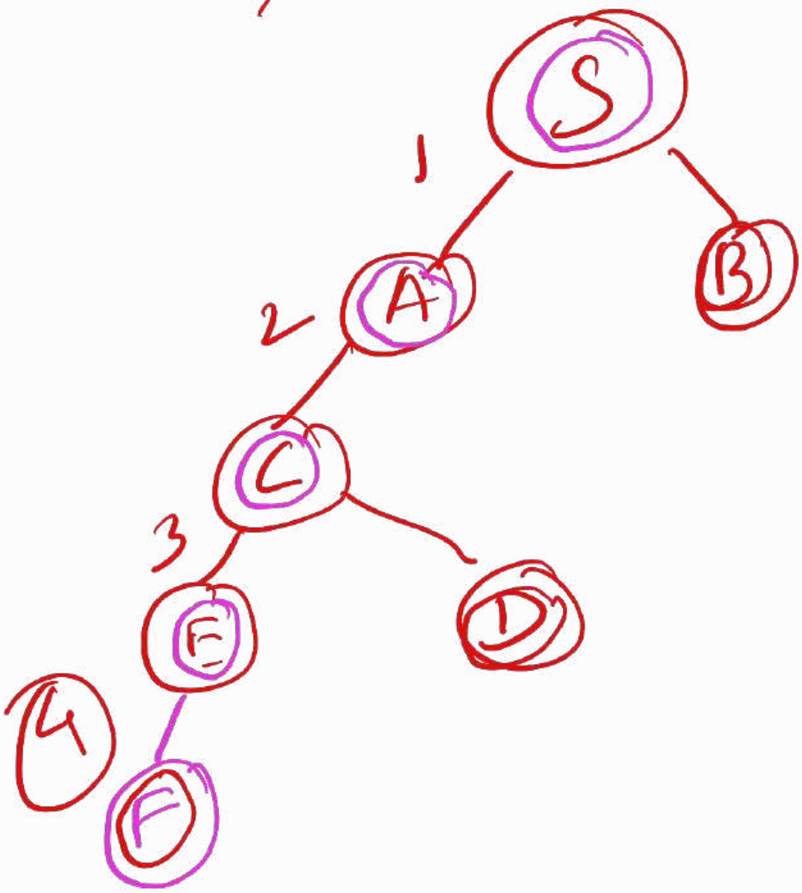
d

1<1

DB-3



DB-4



DFID (Start)

depthbound ← 1

While true

do DepthboundedDFS (Start, depthbound)

depthbound ← depthbound+1

* DFID does a ~~Series~~ repetetive call of DBDFS with ↑ depthbound

Both BFS & DFS are oblivious of the goal.

* Predetermined Trajectory.
* a metric about distance is required.

Informed Strategies