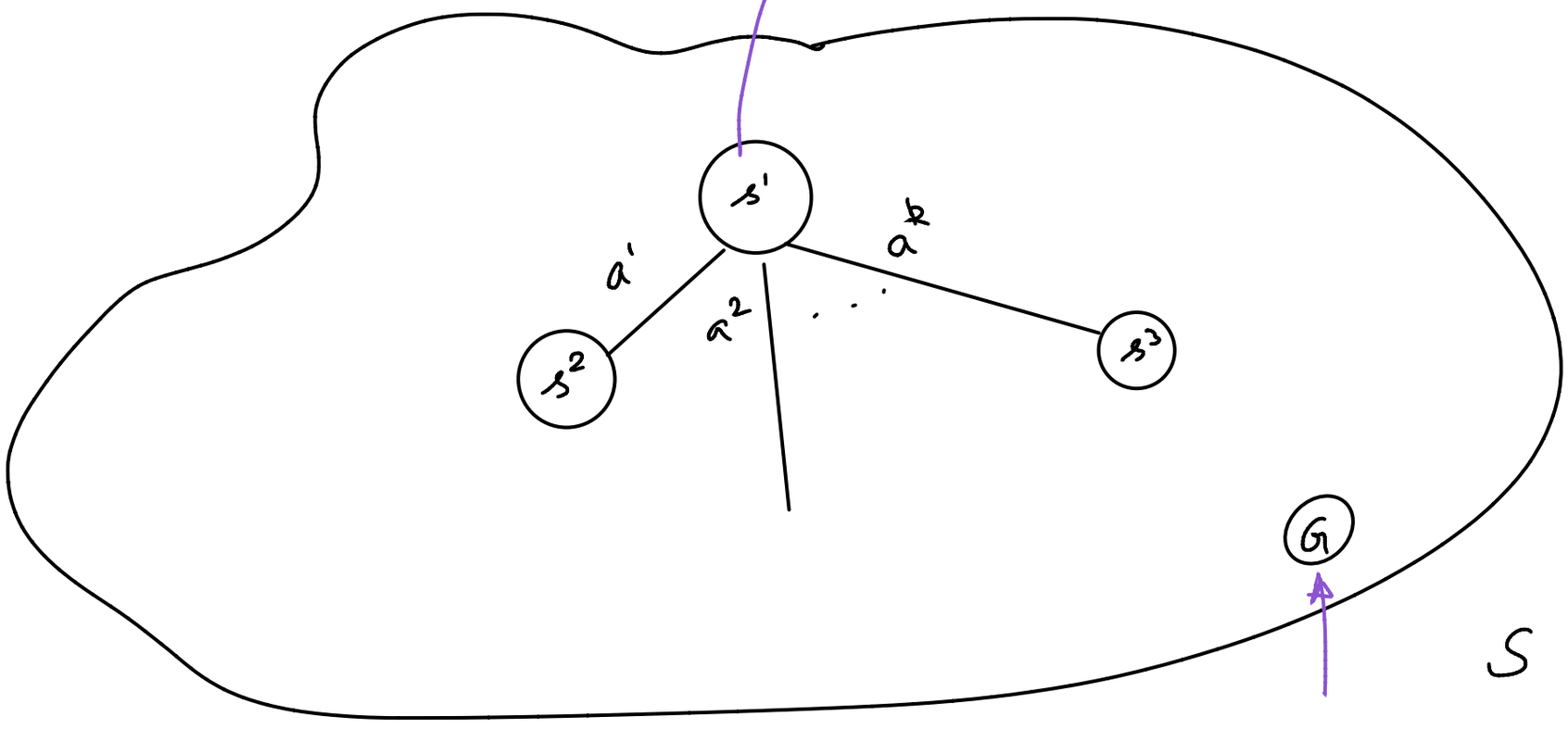
Graph Search

Breadth First , Depth First , Least Cost First , Heuristic

Applies to

- Graph Search
- Deterministic State Space Search

Rubik's Cube



16 - Puzzle

Goal

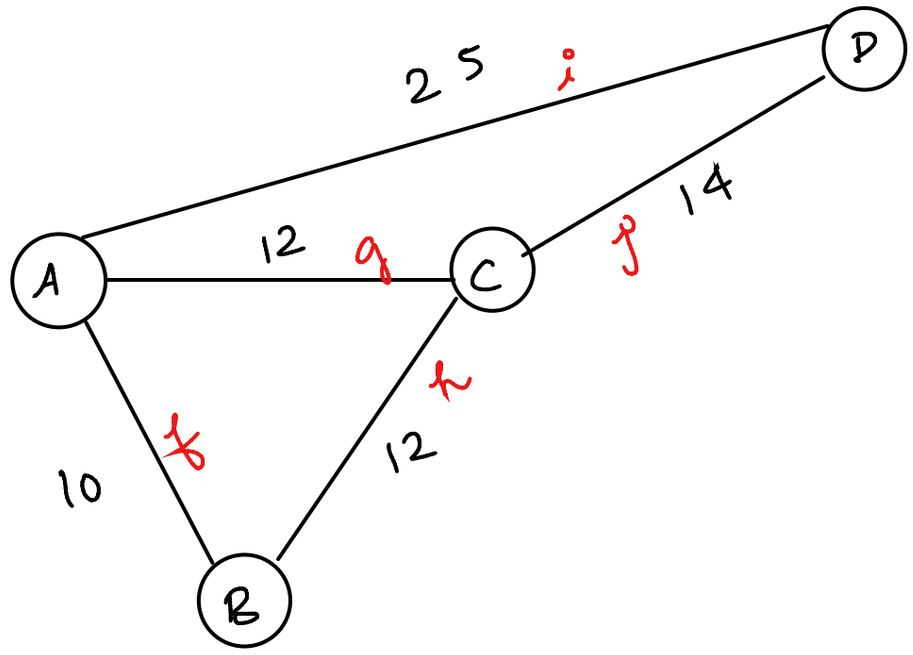
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	////

empty square

Start State

1	3	6	4
5	9	2	8
9	4	7	12
13	14	15	////

Road Network:



Costs are not same across edges.

First consider cost of edge to be 1.

Problem: Starting from start state (or start vertex) S we want to reach the Goal State G (or goal vertex)

Assume: unit step cost or edge cost = 1.

Data Structure :

- Search Tree (Separate from the graph data structure)
- Frontier (Implemented as a Priority Queue)
- Explored List (optional)

Node in Search Tree

* vertex / State : n. STATE / n. vertex
* parent : n. Parent
* action / edge : n. Action / n. Edge
* Path cost : n. COST

Tree Search : No explored List (potentially revisit nodes)

Graph Search : use explored List

Tree_Search

* Frontier = { start state /vertex }

* do

  * If Frontier = Empty ⇒ Search failed

  * Pick a node ∈ Frontier , Frontier = Frontier - Node — Selection

  * If node = Goal the stop (trace route back to start) — Goal Test

  * Frontier = Frontier + children(node) — Expansion

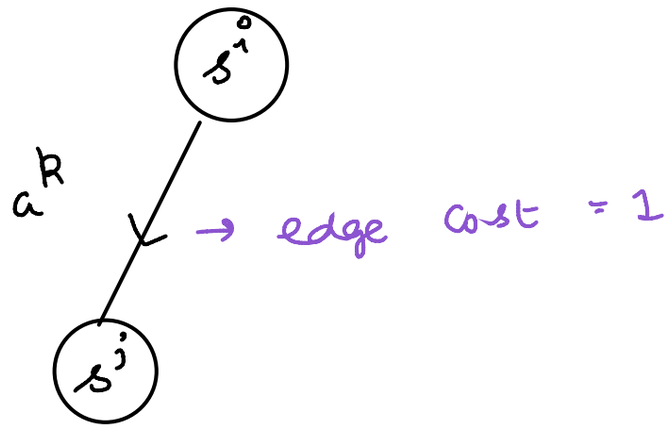
$S = \{ s^1, \ldots, s^n \}$

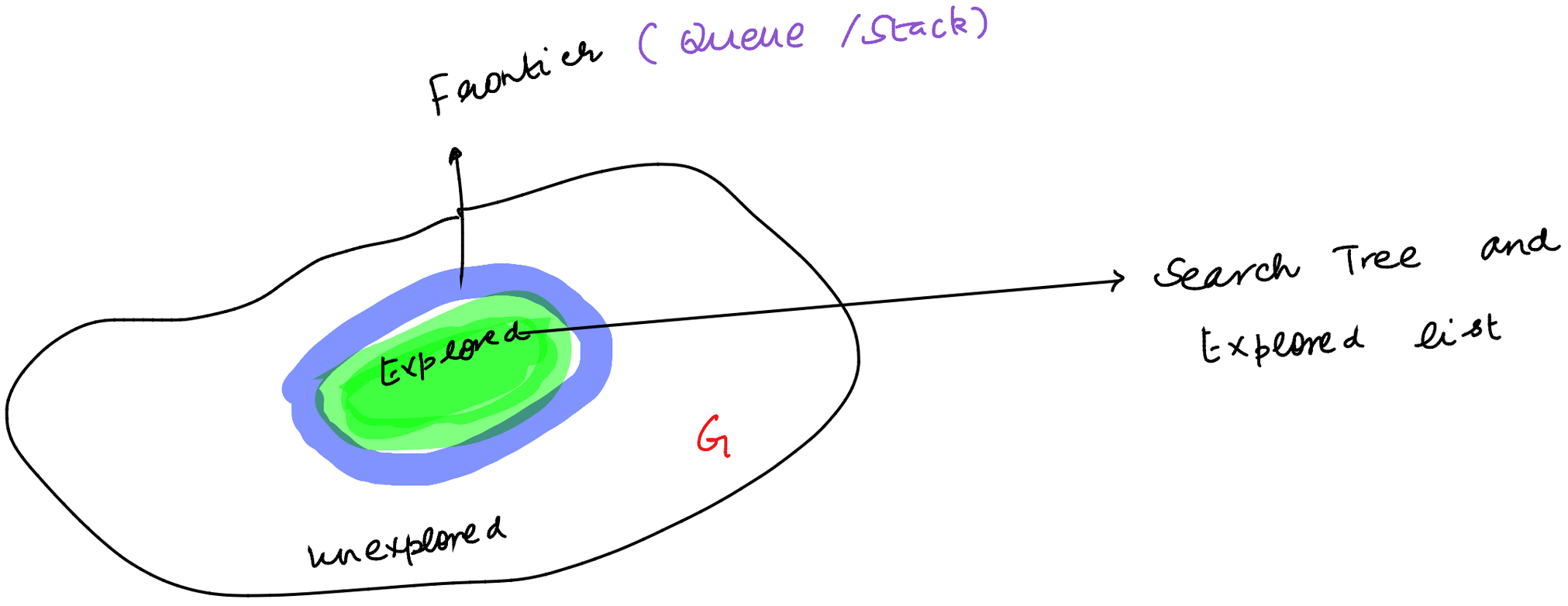
$A = \{ a^1, \ldots, a^k \}$

edges: actions

vertices: states

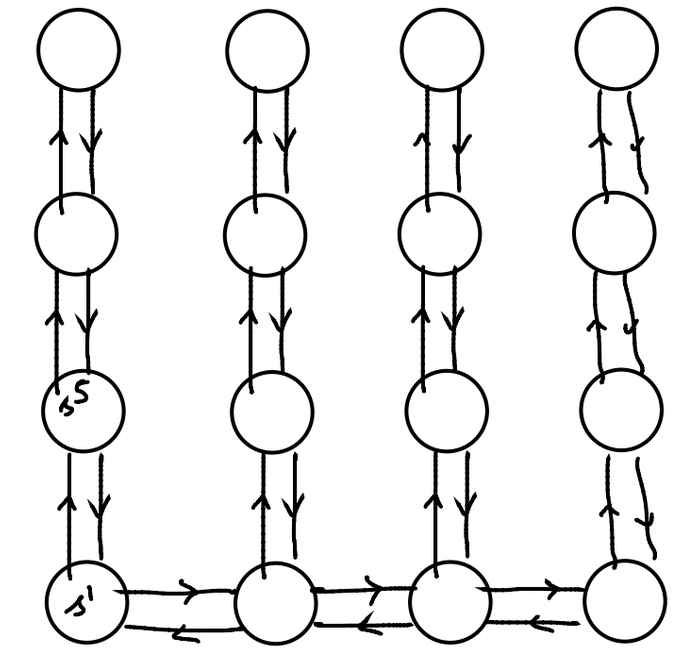
cost : 1




Frontier (Queue / Stack)
Explored
G
Unexplored
Search Tree and
Explored list

s^{13}	s^{14}	s^{15}	s^{16}
s^{9}	s^{10}	s^{11}	s^{12}
s^{5}	s^{6}	s^{7}	s^{8}
s^{1}	s^{2}	s^{3}	s^{4}

$A = \{ \uparrow, \downarrow, \leftarrow, \rightarrow \}$

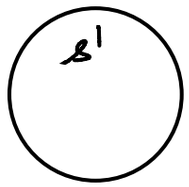


START STATE = s^{1}

GOAL STATE = s^{16}

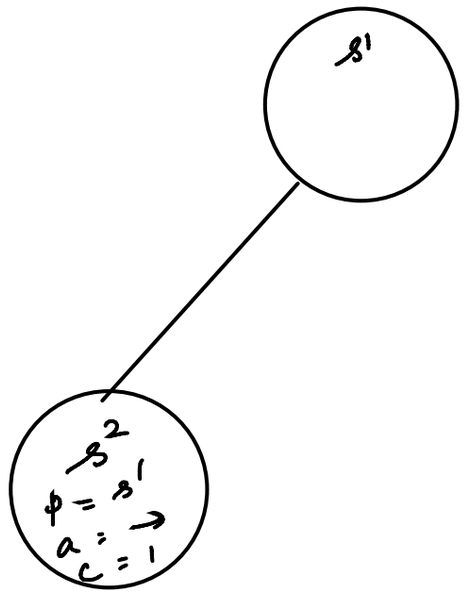
Frontier =

s' leaves



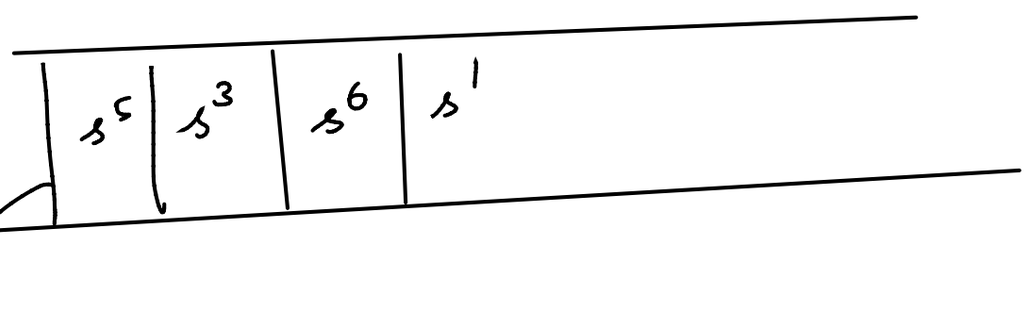
Frontier =

s^2 leaves.



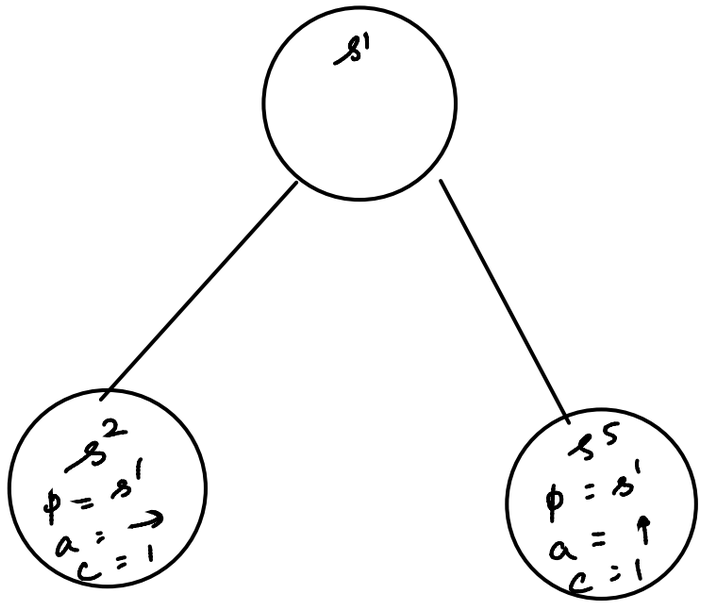
s^2 is not Goal, so expand s^2

Frontier =

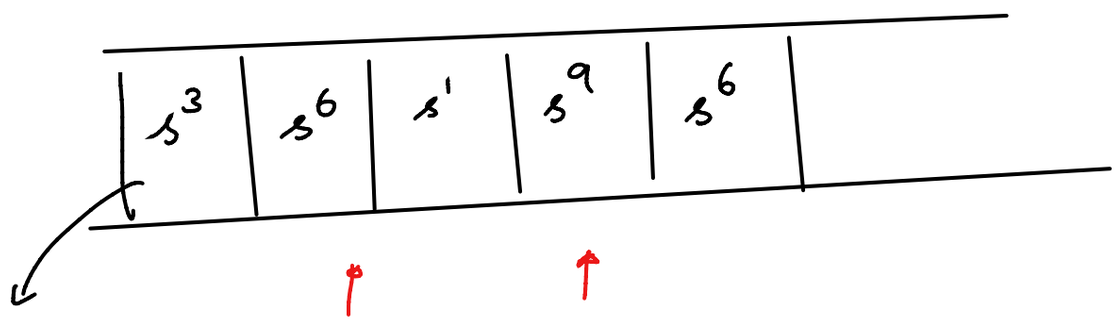


s^5 leaves.

s^5 is not Goal state



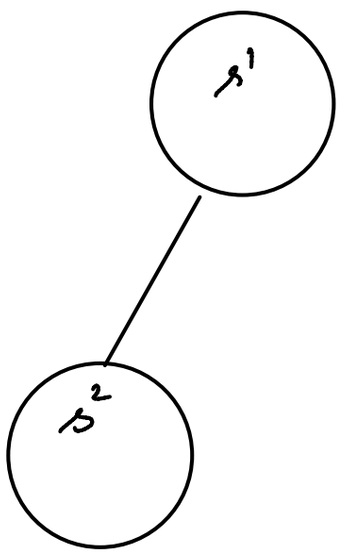
Frontier =

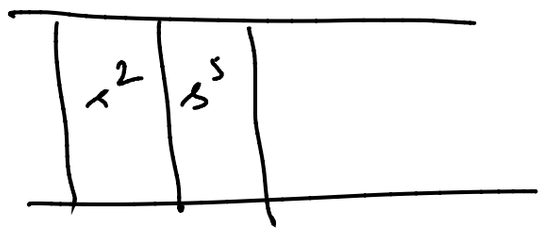


- s^6 gets repeated
- In Tree Search we don't stop

s^{13}	s^{14}	s^{15}	s^{16}
s^{9}	s^{10}	s^{11}	s^{12}
s^{5}	s^{6}	s^{7}	s^{8}
s^{1}	s^{2}	s^{3}	s^{4}

s^{13}	s^{14}	s^{15}	s^{16}
s^{9}	s^{10}	s^{11}	s^{12}
3	s^{6}	s^{7}	s^{8}
1	2	4	s^{4}

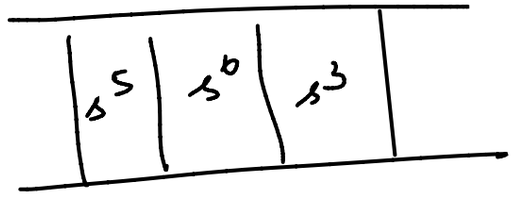




Frontier

s^1

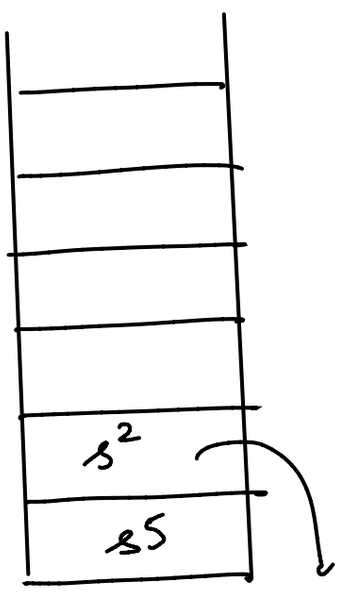
Explored list



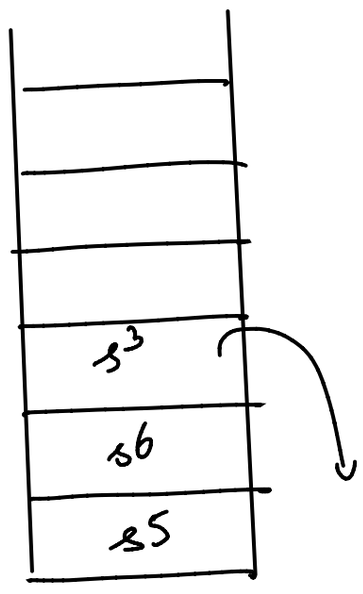
s^2
s^1

Explored list

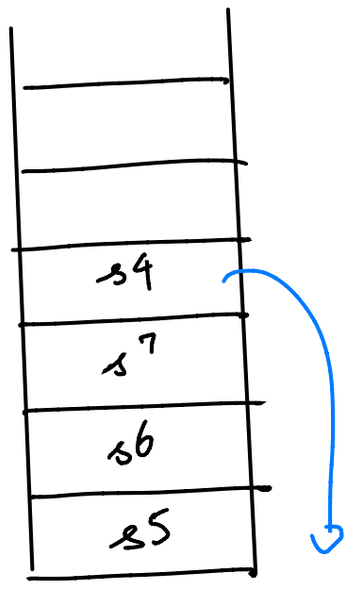
Frontier Implemented as a stack in Graph Search



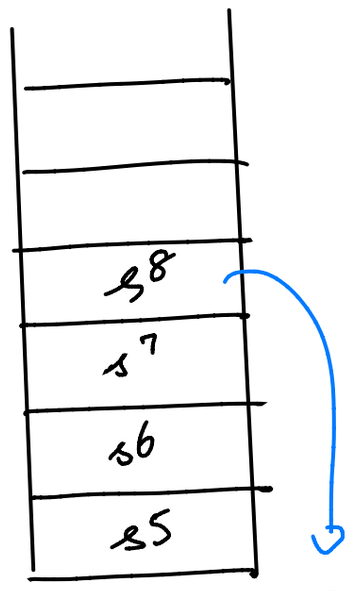
s^2 leaves.



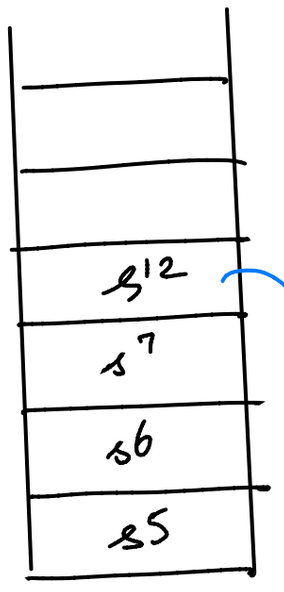
s^3 leaves.



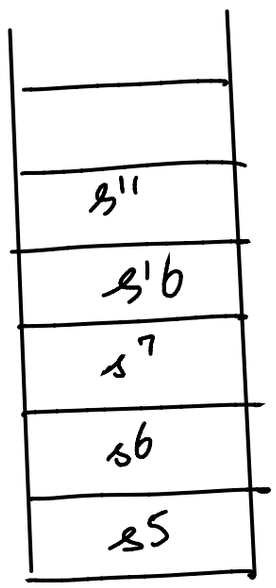
s^4 leaves

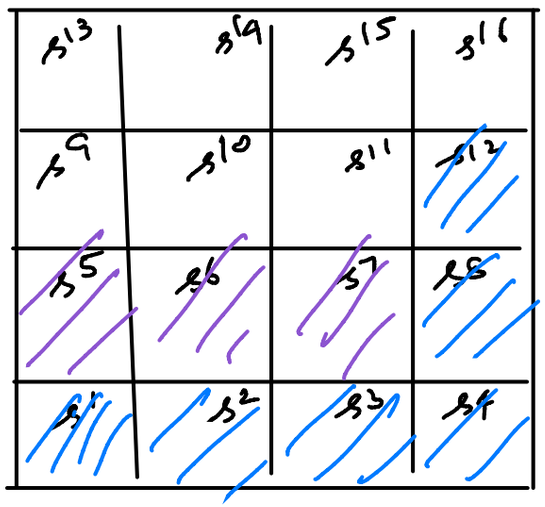


s^8 leaves

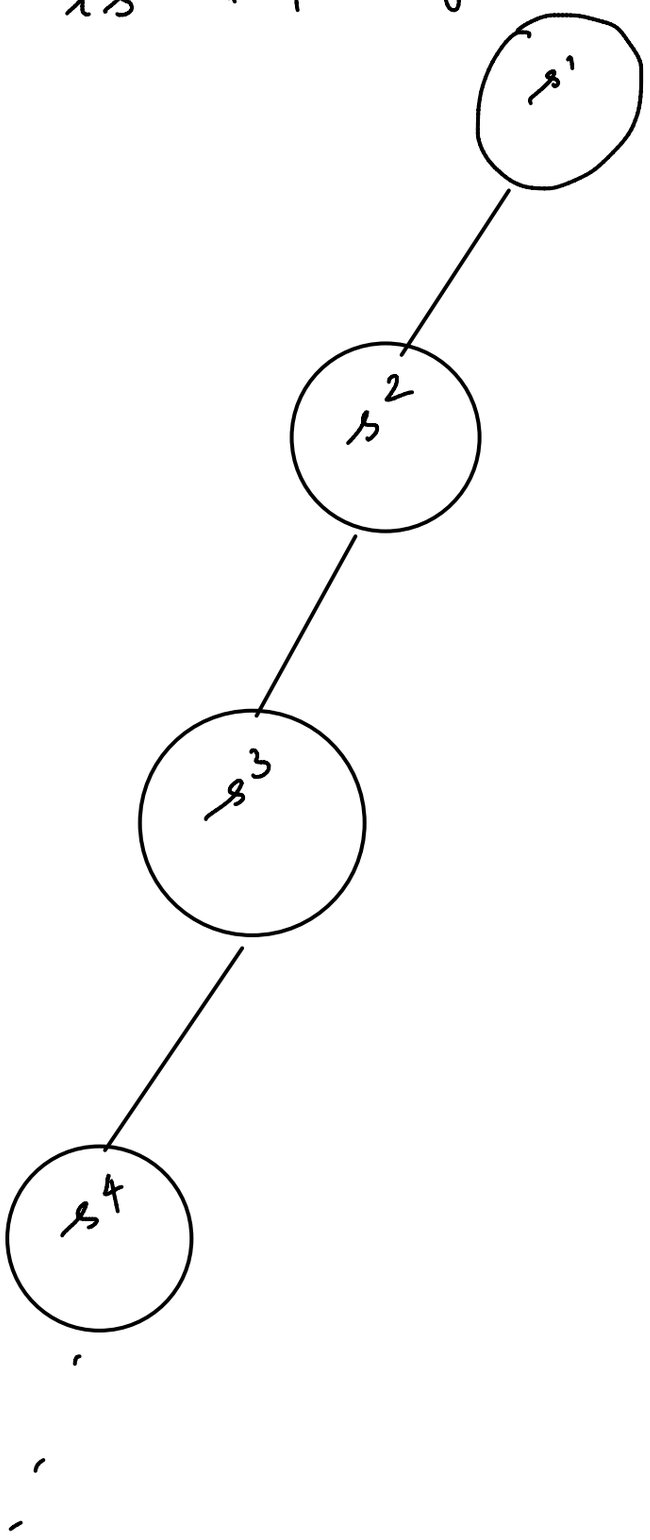


s^{12} leaves





Expansion is Depth first



Frontier : Queue : Breadth First Search

" : Stack : Depth "

Priority Queue

+ count : BFS

- count : DFS

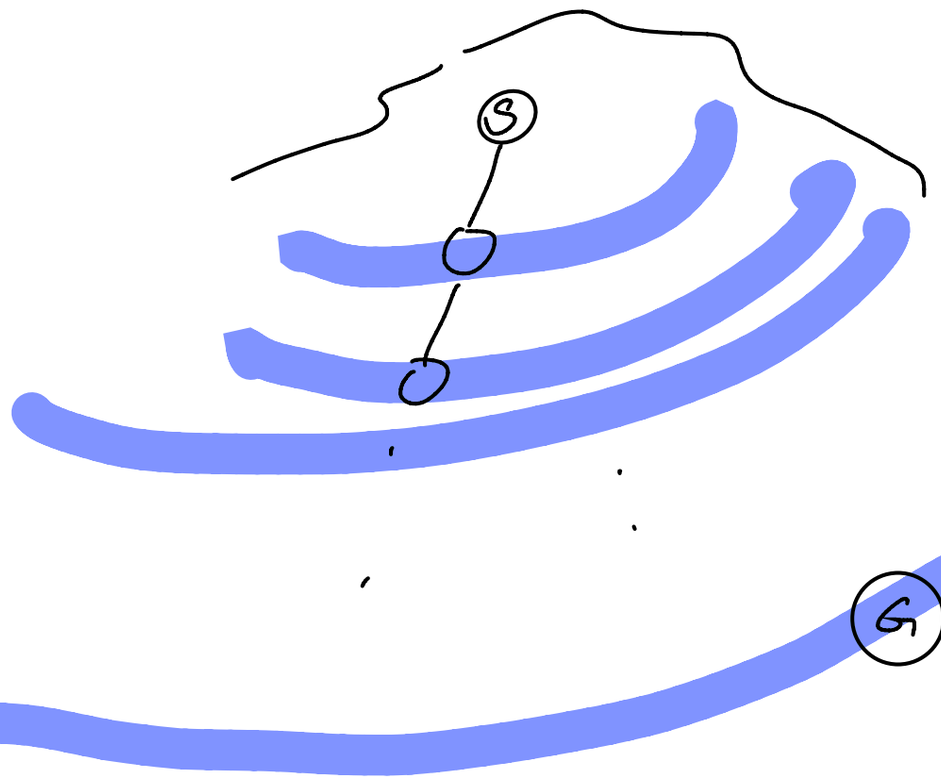
Performance Criteria:

- Completeness: Does it actually find the solution (not necessarily the best)
- Optimality: Does it find the optimal " solution?
  - Time
  - Space

Assume : unit cost

BFS

* Completeness ~ (graph & Tree)

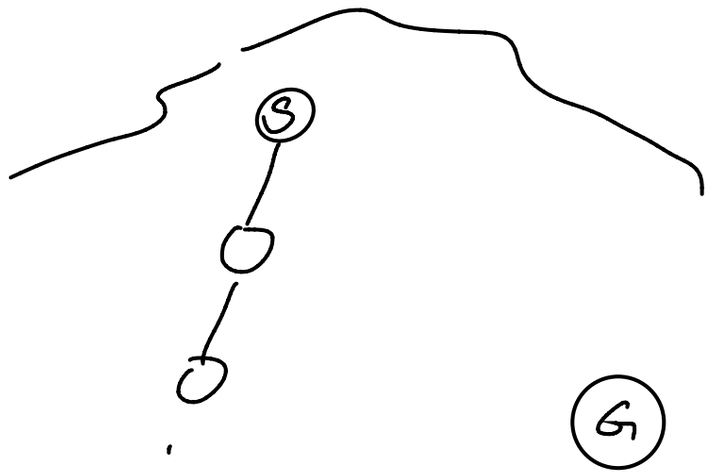


* Optimality ~

DFS

X ~ (Search space is finite)

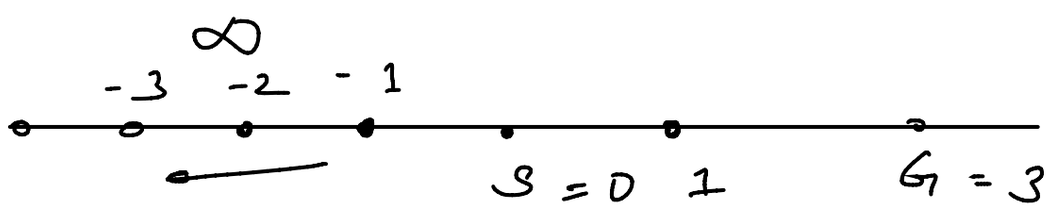
Tree Graph

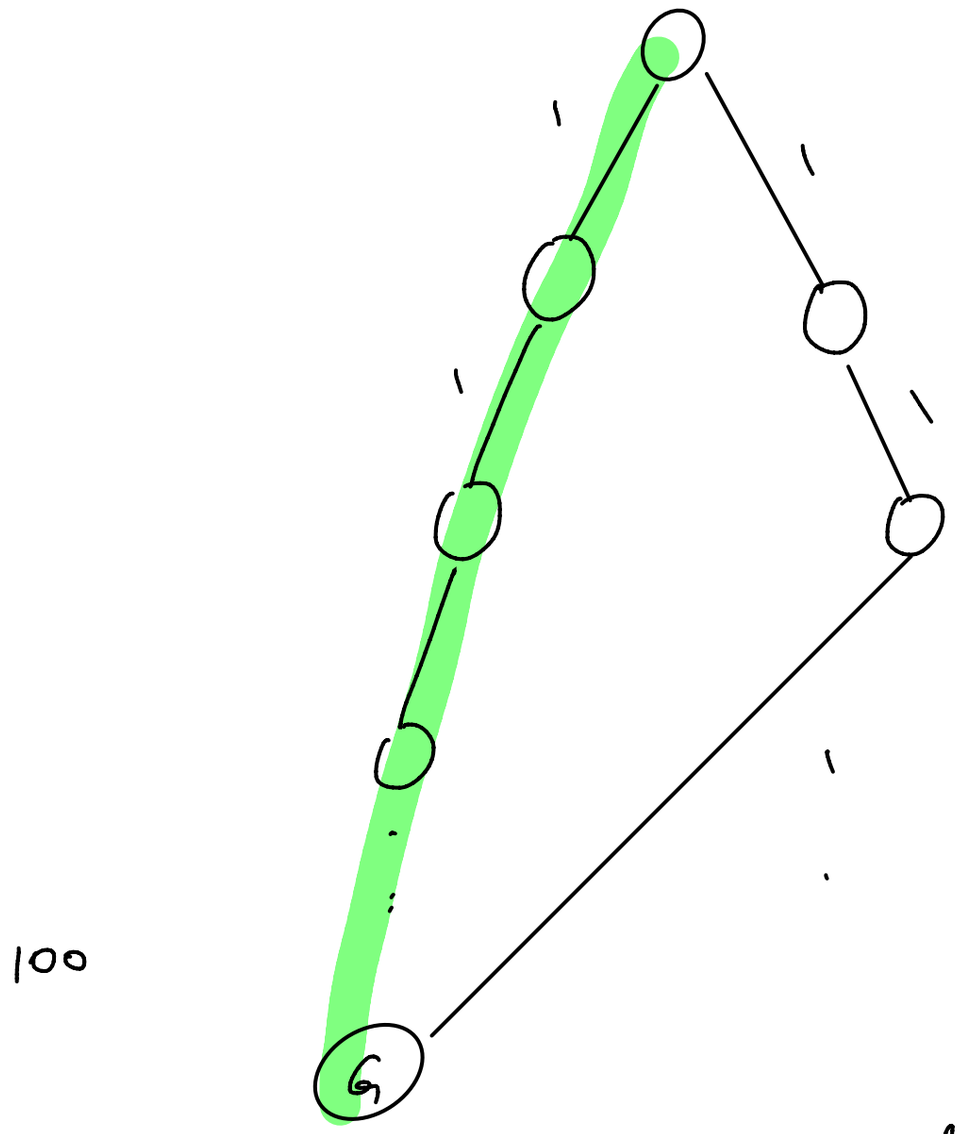


1

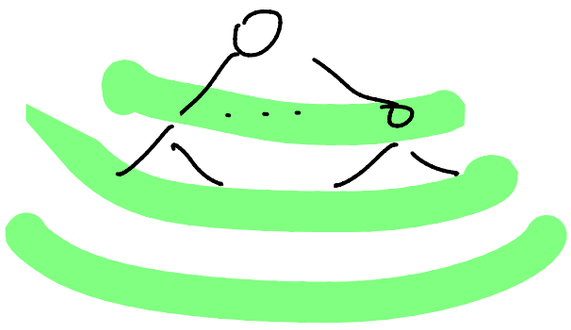
1

X





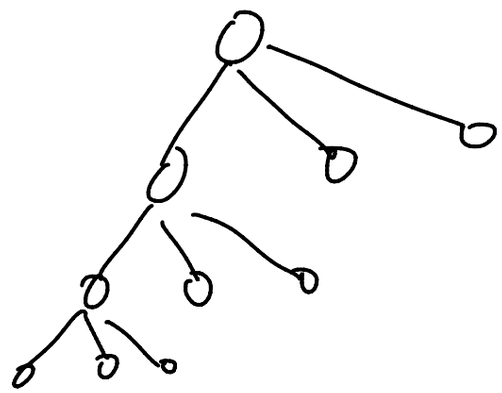
* Space $O(b^d)$



Assume: solution/Goal is at depth m

* Time $O(b^m)$

$O(bd)$



$O(b^m)$