

COMP90054 Workshop 1

Geye Guo

Icebreaker Activity

High Level Picture

Different AI Research directions

- Only focus on AI especially AI planning -> acting rationally
- ML/NLP/CV -> acting like humans

3 approaches:

- Programming-based
- Learning-based: RL/TD (2nd half of this semester)
- Models-based: AI planning

What is AI planning?

Classical Planning Problem

Not every problem belongs to classical planning problem!!

Requirements:

- Single-agent
- Static environment
- **Deterministic action:**
 - Every action only has a certain outcome, and you know what that outcome will be
 - Counterexample: coin toss -> probabilistic actions
-

How to solve a Classical Planning Problem

- The idea of general AI solving problem: Problem => solver => solution
- Comes to AI planning: 1. Problem (Model) => 2. Planner => 3. Plan

1. Model

- **State-space model:** any Classical Planning Problem can be represented by a state-space model
- **STRIPS:** PDDL

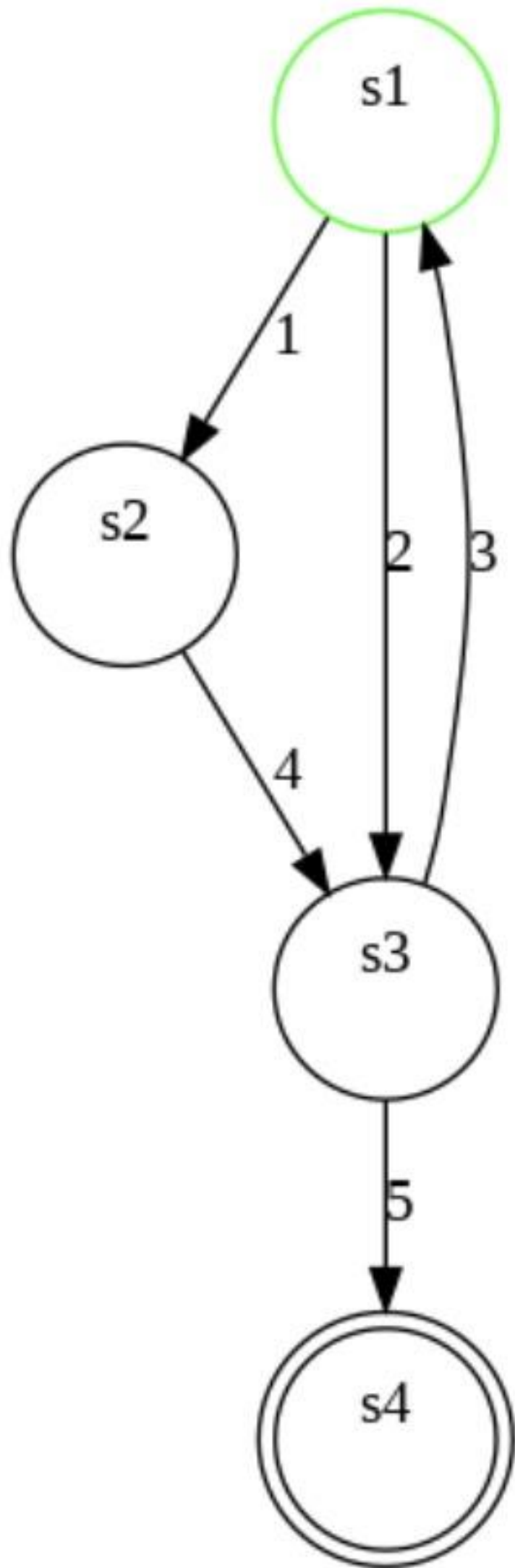
2. Planner => powered by the search algorithms

- **Blind search:** BFS, DFS, ID, Uniform-Cost, IW....
- **Heuristic Search** (Informed search)

3. Plan

- A sequence of actions: a1-> a2 ->....
- **Not a set of actions**

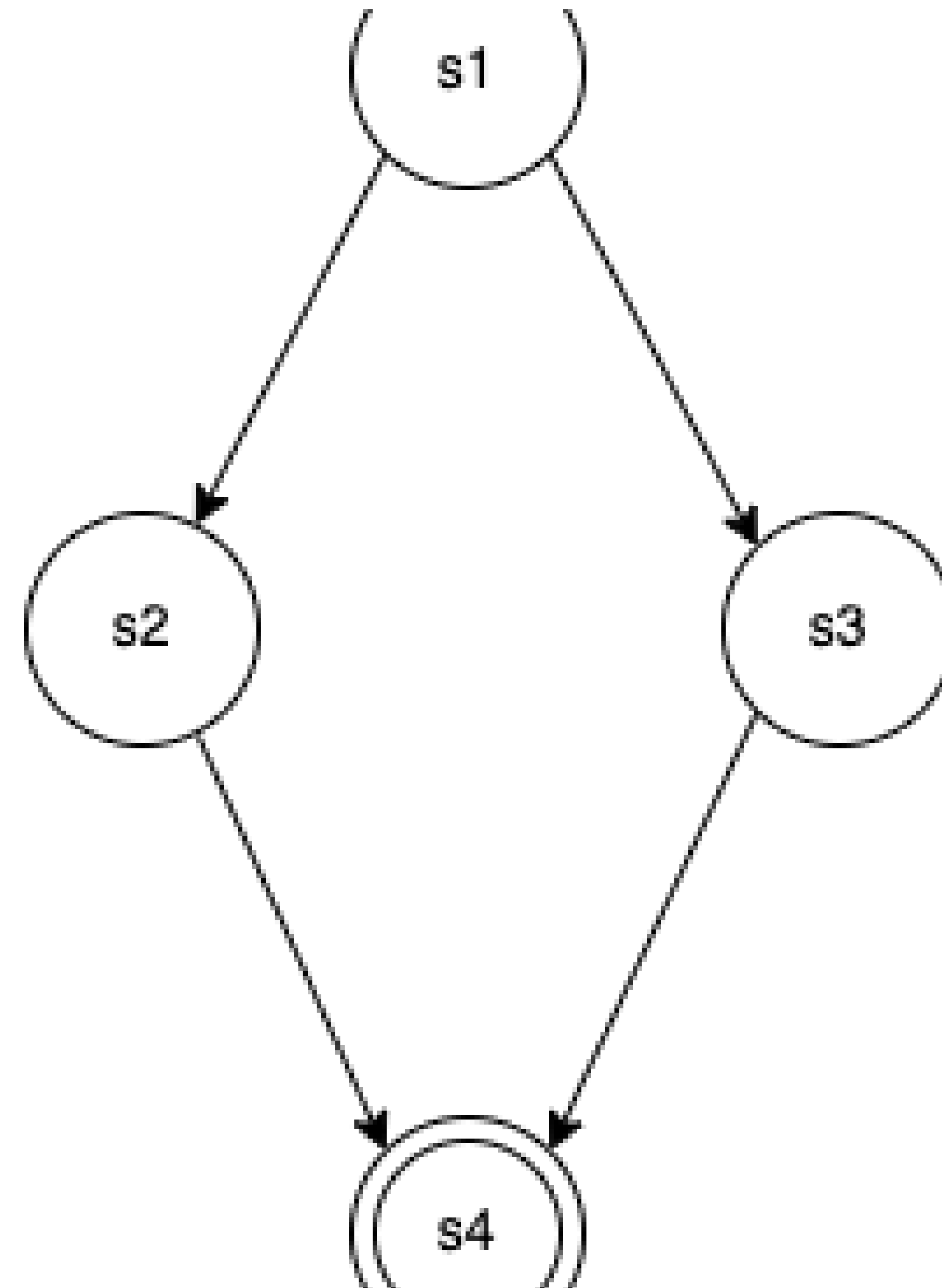
Problem 1: State-transition graph



- State space $S = \{ ? \}$
- Initial State
- Goal State
- Action
- Transition Function
- Cost Function

Search Node vs State

What is the difference between a node and a state?



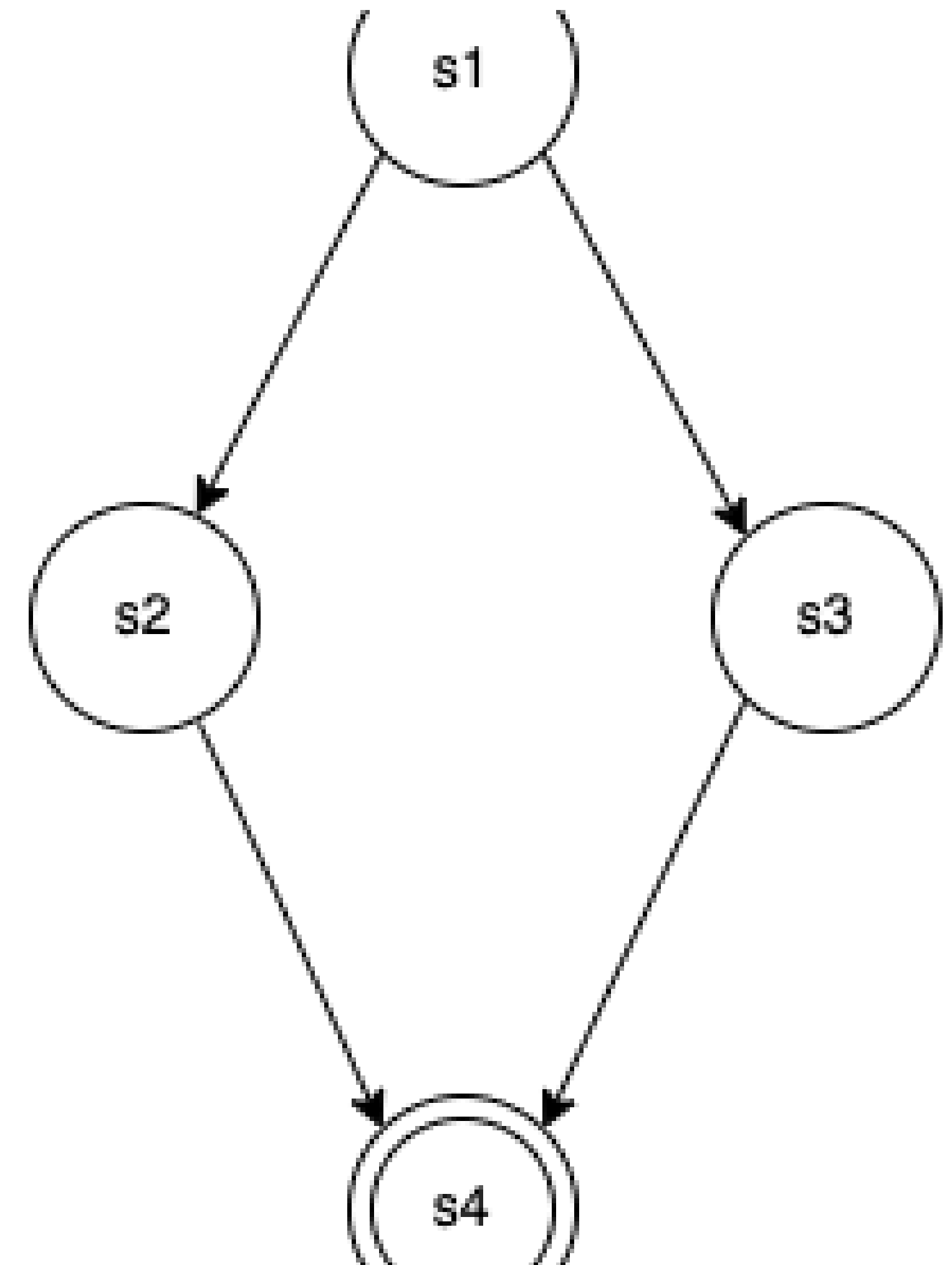
Search Node vs State

What is the difference between a node and a state?

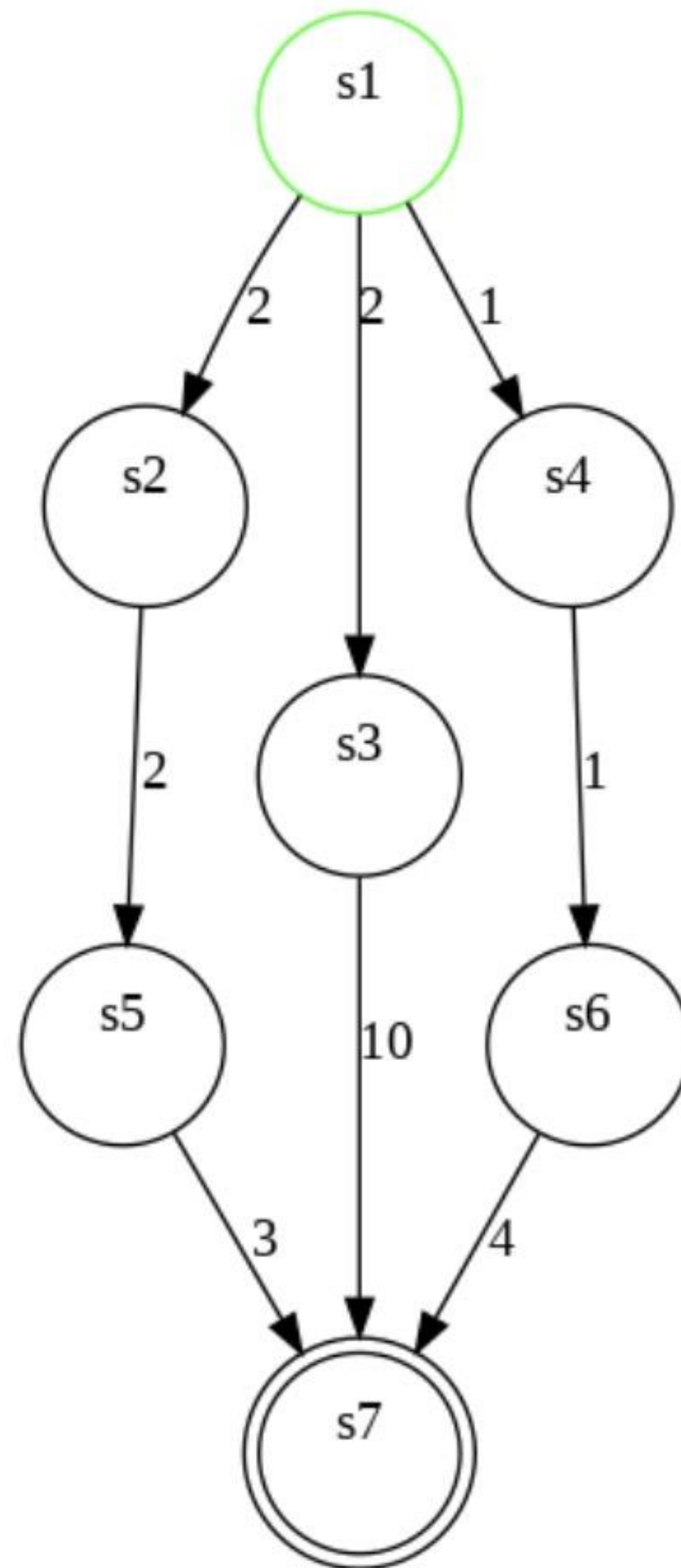
Node contains more information than a state, it may contain:

- State
- Accumulated cost
- Parent information

Why we need nodes?



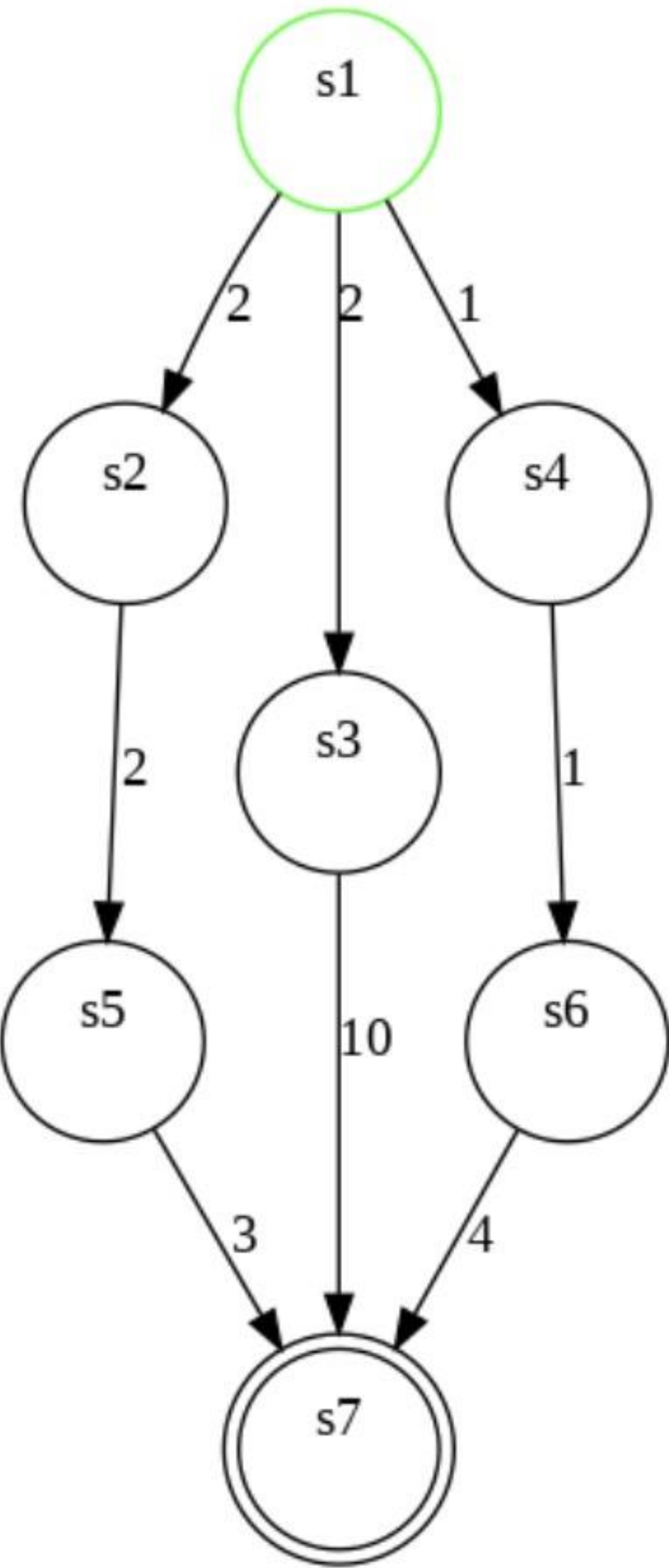
Problem 2



Task 1

Discuss with others, and finish the node expansion order for each algorithm

BFS Expansion

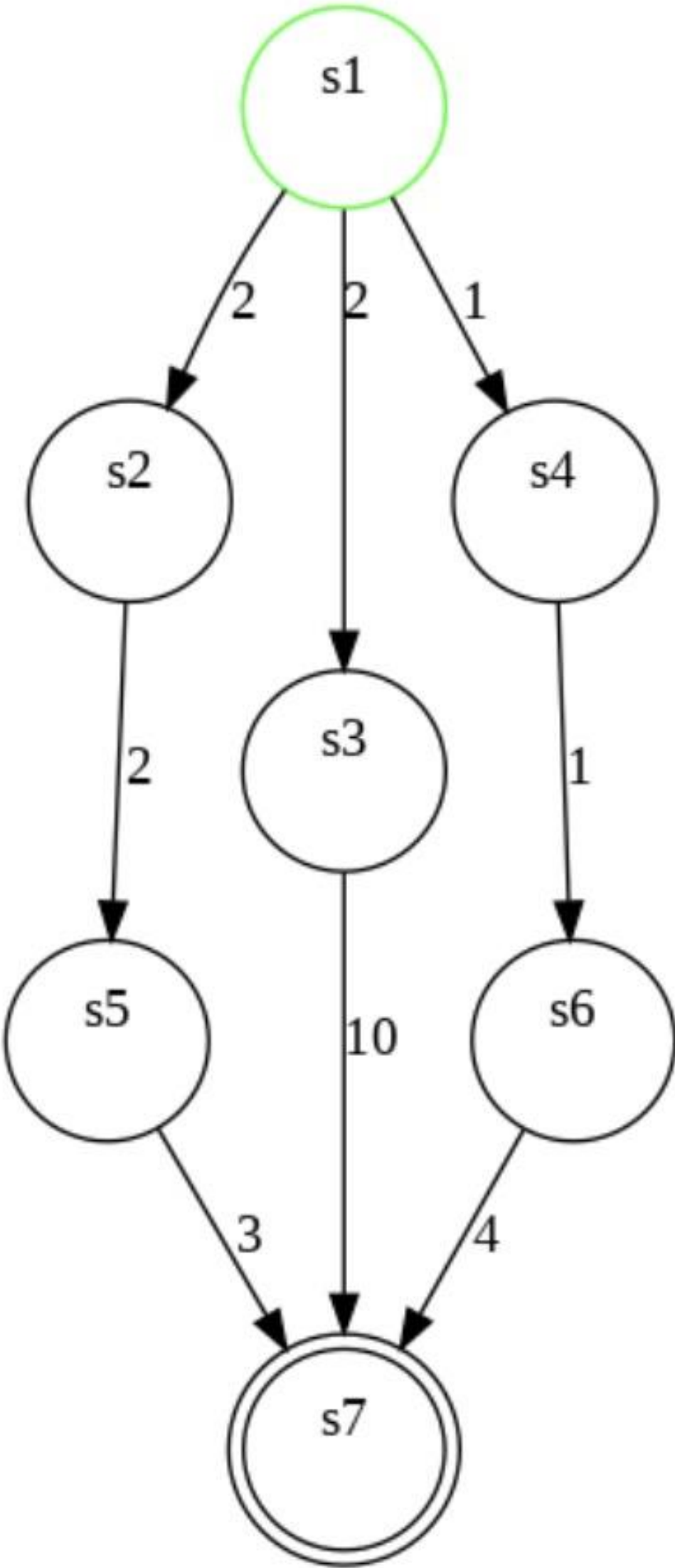


	Iteration 0	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Open							
Closed							

When pop up a node from the queue:

- 1. Check if current node n contains the goal state
- 2. Generate children nodes, and put into data structure

BFS Expansion



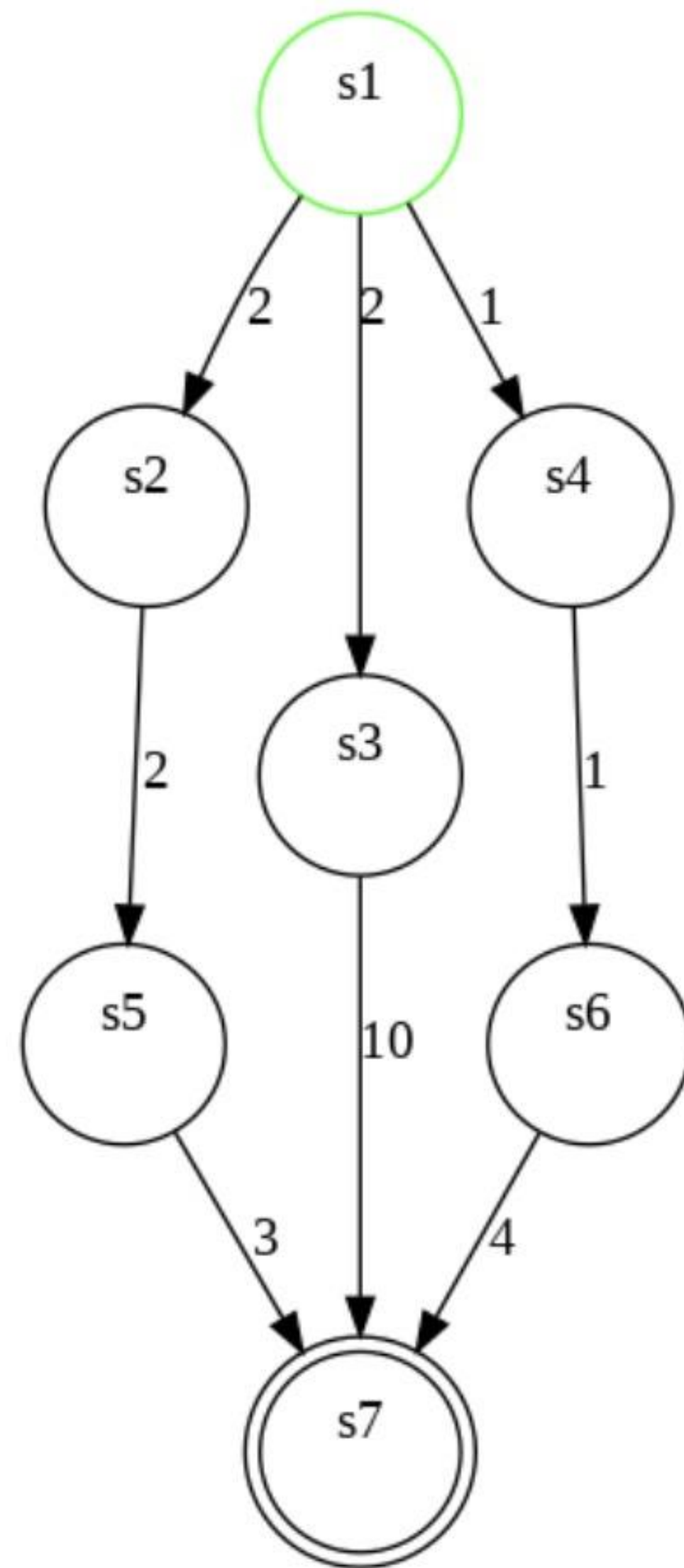
	Iteration 0	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Open	$n0 = \langle s1, 0, \text{null} \rangle$	$n1 = \langle s2, 2, n0 \rangle$ $n2 = \langle s3, 2, n0 \rangle$ $n3 = \langle s4, 1, n0 \rangle$	$n2$ $n3$ $n4 = \langle s5, ?, n1 \rangle$	$n3$ $n4$ $n5 = \langle s7, 12, n2 \rangle$	$n4$ $n5$ $n6 = \langle s6, 2, n3 \rangle$	$n5$ $n6$ $n7 = \langle s7, 7, n4 \rangle$	$n6$ $n7$
Closed		$n0$	$n0, n1$	$n0, n1, n2$	$n0, n1, n2, n3$	$n0, n1, n2, n3, n4$	$n0, n1, n2, n3, n4, n5$

Queue: $n0, n1, n2, n3, n4, n5, n6, n7$

When pop up a node from the queue:

1. Check if current node n contains the goal state
2. Generate children nodes, and put into data structure

Problem 2 Task 1

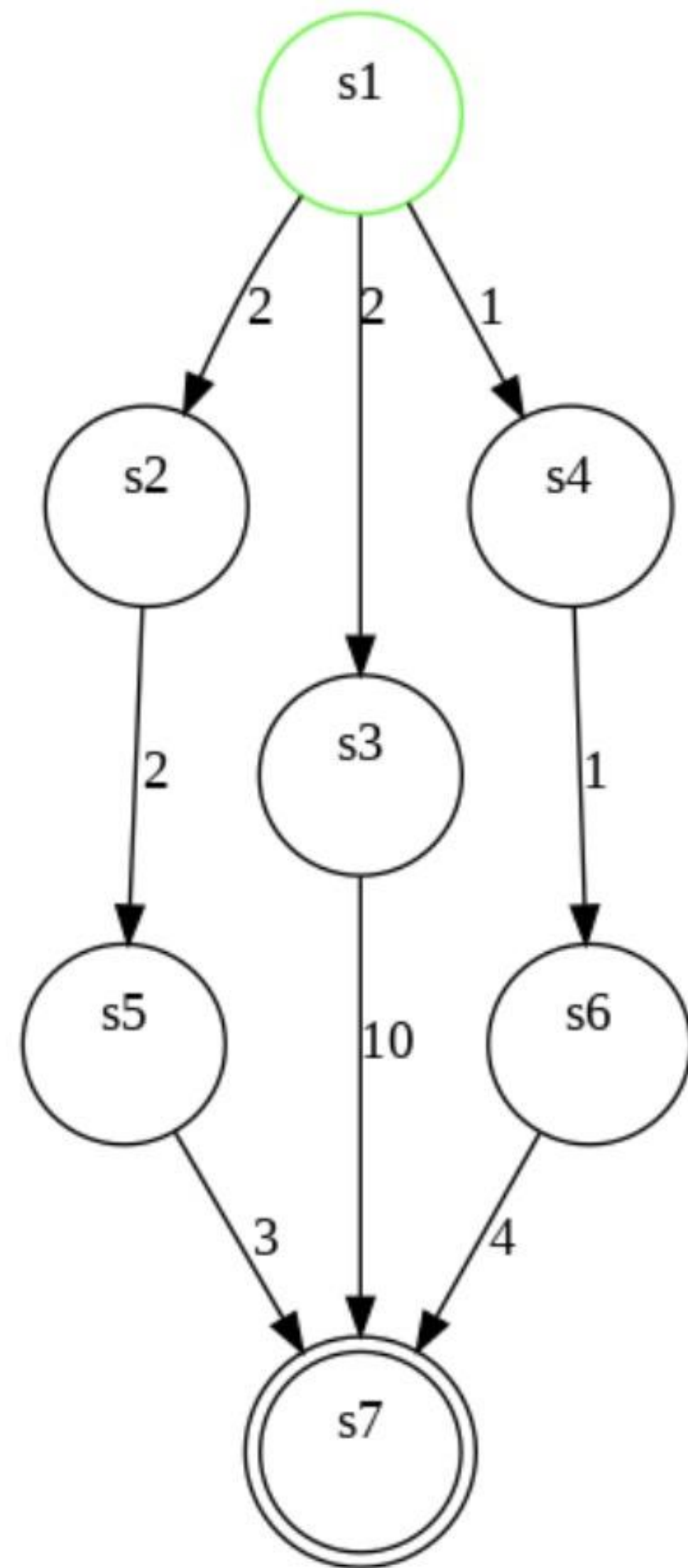


- **Depth-First Search**

```
Nodes = [  
    ('s1', 0, None) ,  
    ('s2', 2, 0) ,  
    ('s5', 4, 1) ,  
    ('s7', 7, 2) ]
```

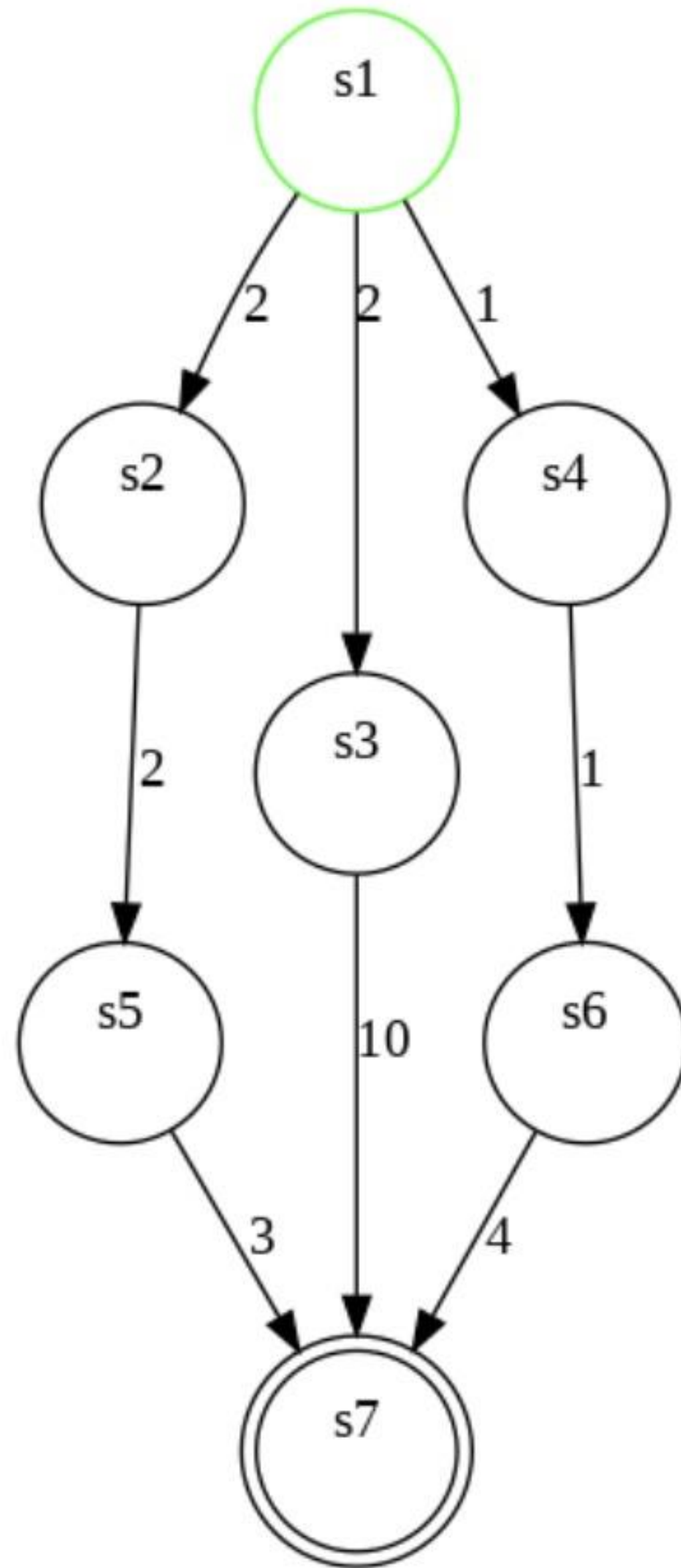
- # (state, accumulated cost, index of parent node)

Problem 2 Task 1



- **ID (iterative deepening)**

```
Nodes = [  
  ('s1', 0, None), #depth limit = 0  
  ('s2', 2, 1),  
  ('s3', 2, 1),  
  ('s4', 1, 1),  
  ('s1', 0, None), #depth limit = 1  
  ('s2', 2, 5),  
  ('s5', 4, 6),  
  ('s3', 2, 5),  
  ('s7', 12, 8)]
```

Problem 2

Task 2

Q1: What is the solution found by each algorithm?

Q2: What is the actual optimal solution?

Q3: Explain under which conditions the algorithms guarantee optimality.

Q4: Can any of the previous algorithms be adapted to account for $g(n)$ in order to make it optimal?

	Complete	Optimal	Time Complexity	Space Complexity
BFS	T	T*	$O(b^d)$	$O(b^d)$
DFS	F	F	infinity	$O(b \cdot d)$
ID	T	T*	$O(b^d)$	$O(b \cdot d)$

b = branching factor
d = depth of the optimal path

Problem 3

Describe a simple example of *Travelling Salesman Problem* along with its corresponding **State Space Model**.

Definition should be brief, clear, and *compact* (*compact* means using mathematical notation to define sets, i.e. $S = \{x | x \in V\}$ to define that there are as many states as elements in the set V , and pseudo-code, i.e. to define the transition function.)

1. State space S
2. Initial state $s_0 \in S$
3. Set of goal states $S_G \subseteq S$
4. Applicable actions function $A(s)$ for each state $s \in S$
5. Transition function $f(s, a)$ for $s \in S$ and $a \in A(s)$
6. Cost of each action $c(a)$ for $a \in A(s)$

Hint: Consider a set of cities V to visit in any order, a starting city location v_{start} , and a set of edges E specifying if there's an edge from two cities $\langle v_1, v_2 \rangle$. Let V' be the set of cities has been visited.

Problem 3

Let V' be the set contain visited cities:

- $S = \{\langle v_{current}, V' \rangle \mid v_{current} \in V \wedge V' \subseteq V\}$
- $s_0 = \langle v_{start}, \{v_{start}\} \rangle$
- $S_G = \{\langle v_{current}, V \rangle \mid v_{current} \in V\}$
- $A(\langle v_{current}, V' \rangle) = \{\langle v_{current}, v_{next} \rangle \mid \langle v_{current}, v_{next} \rangle \in E\}$
- $f(\langle v_{current}, V' \rangle, \langle v_{current}, v_{next} \rangle) = \langle v_{next}, V' \cup \{v_{next}\} \rangle$
- $c(\langle v_{current}, v_{next} \rangle) = cost(\langle v_{current}, v_{next} \rangle)$