COMP90054 Workshop 2

Geye Guo

Recap

- Blind search: only use basic search algorithm (BFS, DFS, ID)

	Complete	Optimal	Time Complexity	Space Complexity
BFS	T	T*	O(b^d)	O(b^d)
DFS	F	F	O(b^D)	O(b*d)
ID	T	T*	O(b^d)	O(b*d)

b = branching factor, d = depth of the optimal path
D = maximum depth of the problem

- Heuristic Search: additionally use the heuristic function to estimate the remaining cost (distance) to the goal state

A few notations for heuristic search

- $s, s', a, c(a)$
- $n = \langle s, f(n), g(n), n_{parent} \rangle$
- $h \leftrightarrow h(s)$, $h^* \leftrightarrow h^*(s)$

- Uniform cost search: f(n) = g(n)
- Greedy: f(n) = h(s)
- A*: f(n) = h(s) + g(n)
- WA*: f(n) = W * h(s) + g(n)

Weighted A*

- **$f(n) = g(n) + w * h(s)$**

- If $w == 0$: $f(n) = g(n)$ => uniform cost
- If $w == 1$: $f(n) = g(n) + h(s)$ => A*
- If $w ==$ infinite: $f(n) = h(s)$ => Greedy

Properties of Heuristic functions

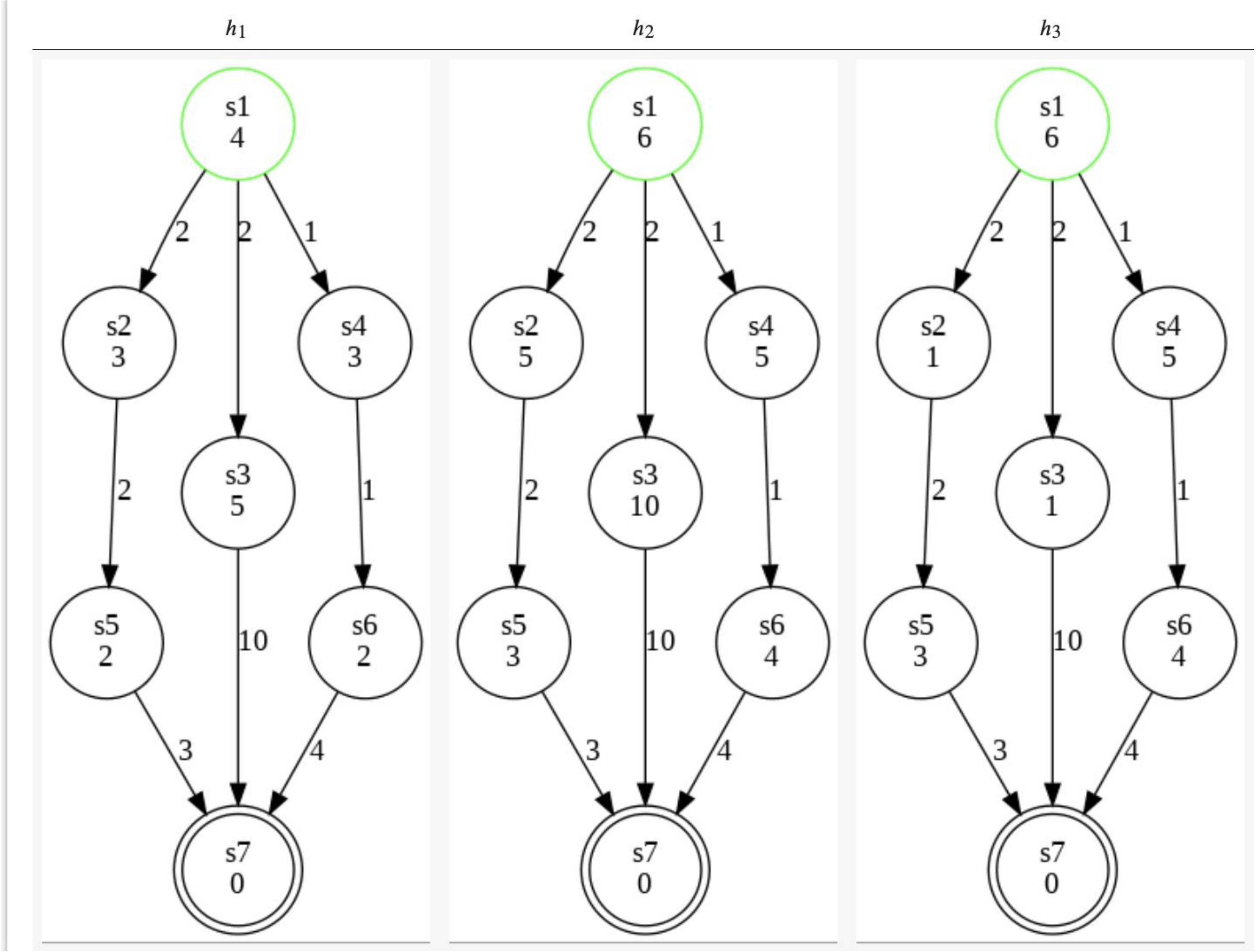
Definition (Safe/Goal-Aware/Admissible/Consistent). *Let Π be a planning task with state space $\Theta_\Pi = (S, L, c, T, I, S^G)$, and let h be a heuristic for Π. The heuristic is called:*

- *safe if $h^*(s) = \infty$ for all $s \in S$ with $h(s) = \infty$;*
- *goal-aware if $h(s) = 0$ for all goal states $s \in S^G$;*
- *admissible if $h(s) \leq h^*(s)$ for all $s \in s$;*
- *consistent if $h(s) \leq h(s') + c(a)$ for all transitions $s \xrightarrow{a} s'$.*

Dominant Relation

- If heuristic h1 dominates heuristic h2:
- Then we will have h1(s) >= h2(s), for all s belongs to state space S
- And both h1 and h2 need to be admissible

Problem 1



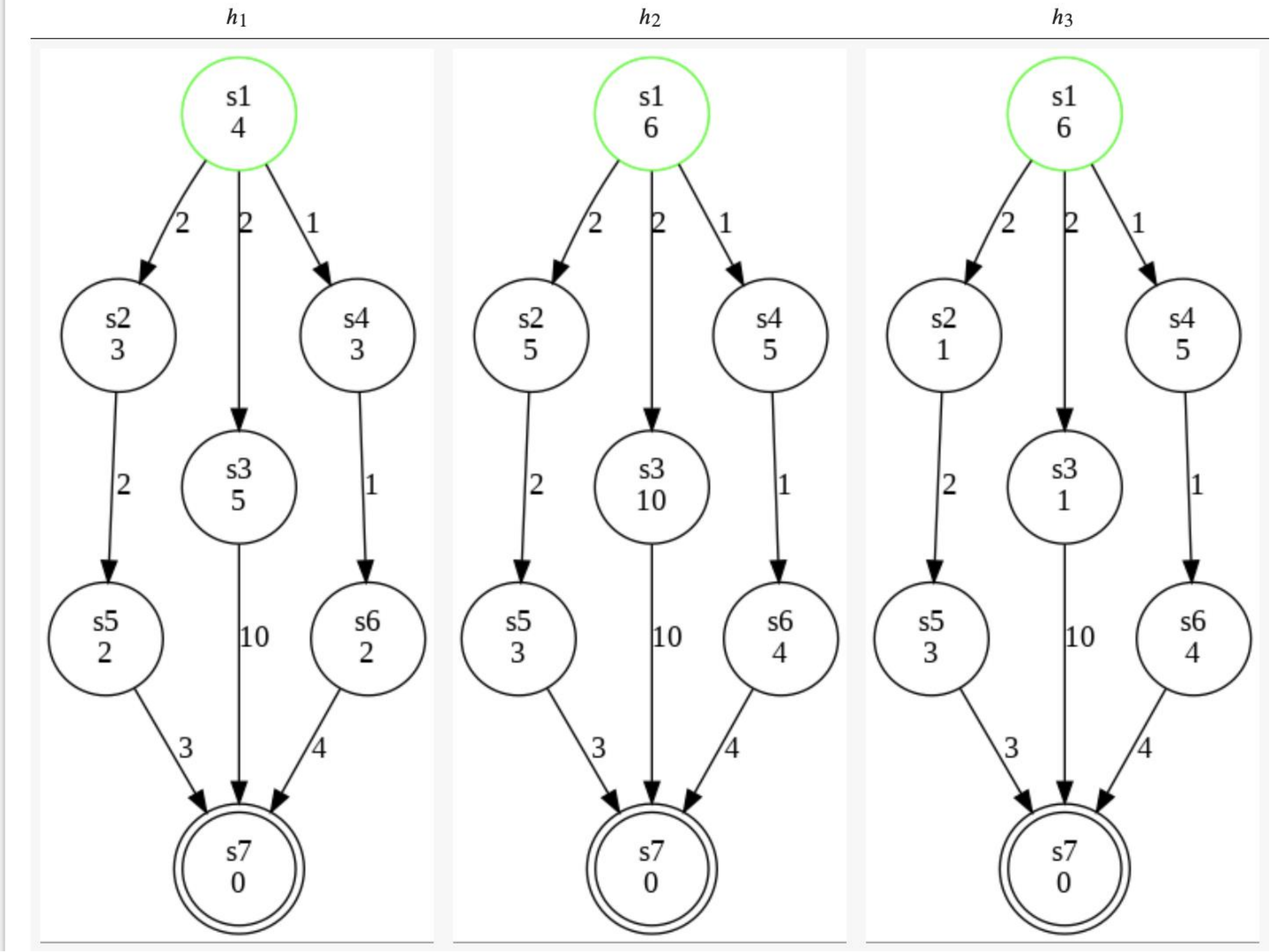
h_1
h_2
h_3
s1 4
s2 3
s3 5
s4 3
s5 2
s6 2
s7 0
s1 6
s2 5
s3 10
s4 5
s5 3
s6 4
s7 0
s1 6
s2 1
s3 1
s4 5
s5 3
s6 4
s7 0
2
2
1
2
1
10
3
4

Task 1

- **Which heuristics are admissible?**
- **Which are consistent?**
- **Does any of the heuristics dominate any other?**

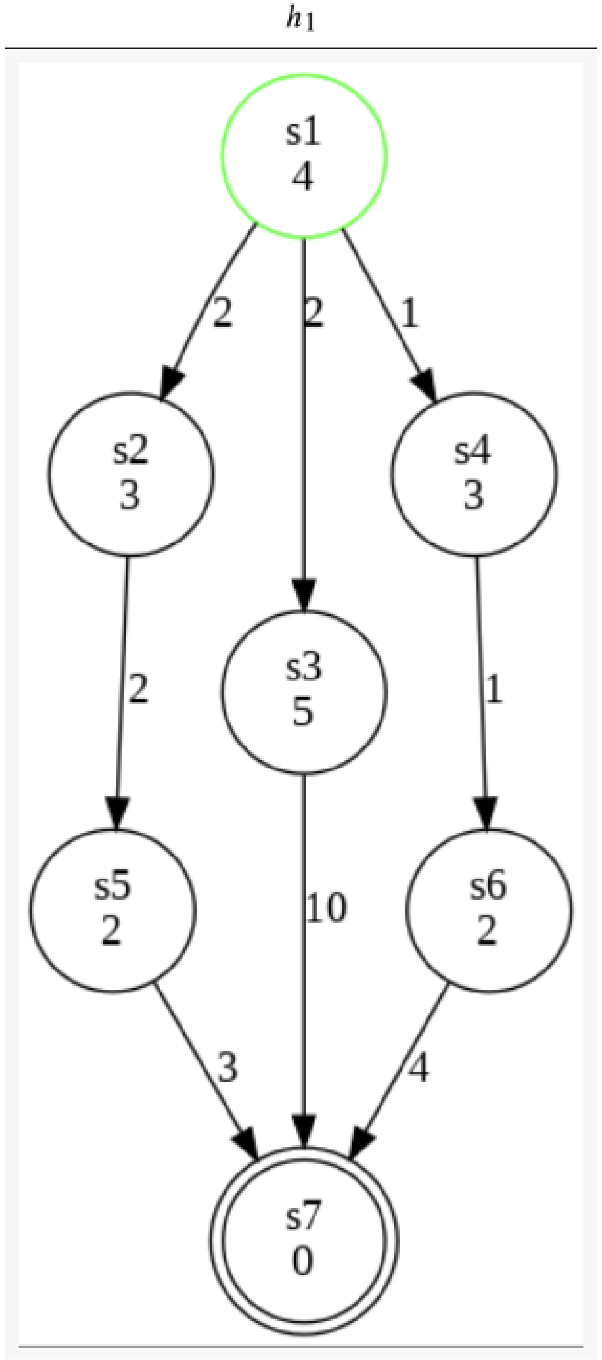
- *admissible* if $h(s) \leq h^*(s)$ for all $s \in s$;
- *consistent* if $h(s) \leq h(s') + c(a)$ for all transitions $s \xrightarrow{a} s'$.

Problem 1


h_1
h_2
h_3
s1 4
s2 3
s3 5
s4 3
s5 2
s6 2
s7 0
2
2
1
2
1
3
10
4
s1 6
s2 5
s3 10
s4 5
s5 3
s6 4
s7 0
2
2
1
2
1
3
10
4
s1 6
s2 1
s3 1
s4 5
s5 3
s6 4
s7 0
2
2
1
2
1
3
10
4

Task 2

- Choose one Heuristic and perform A*
- Choose one Heuristic and perform Greedy
- Choose one Heuristic and perform WA*



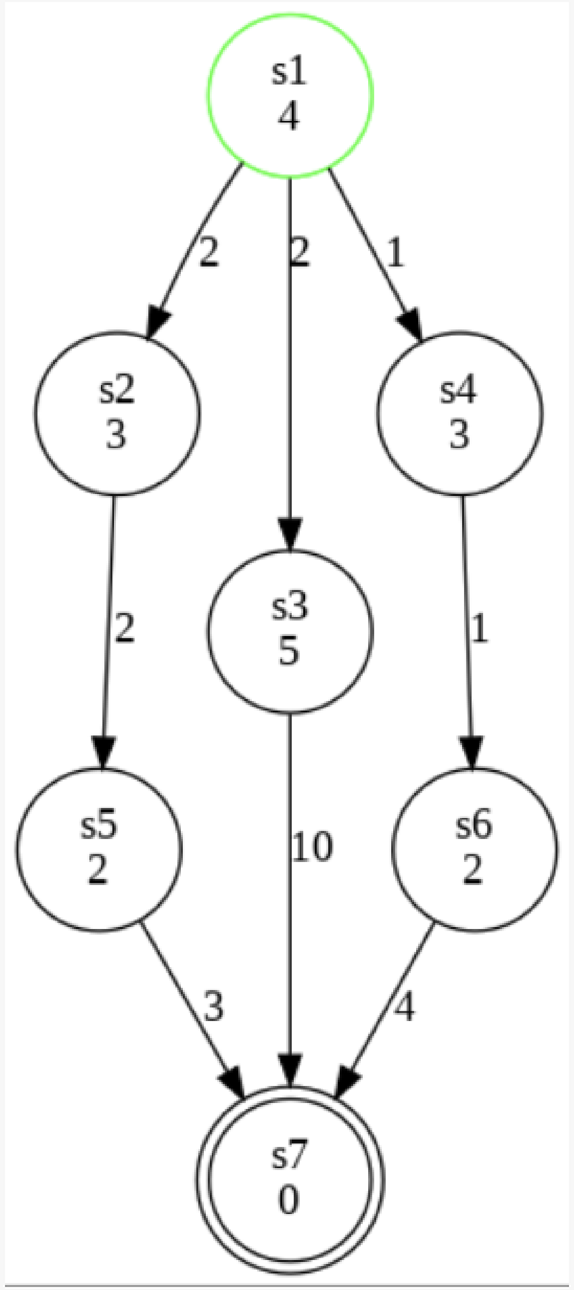
Node expansion order of A*, h1

When pop up a node from the data structure:

1. Check if current node n contains the goal state

2. Generate children nodes, and put into data structure

h_1

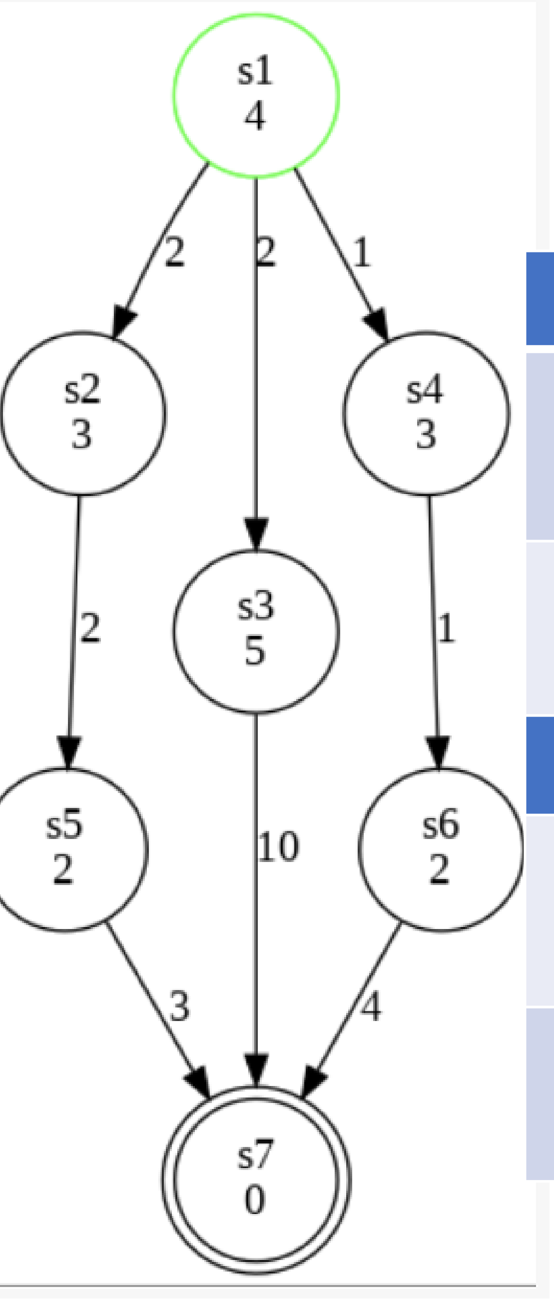


Node expansion order of A*, h1

	I0	I1	I2
Open	n0 = <s1, 4, 0, null>	n1 = <s2, ?, ?, n0> n2 = <s3, ?, ?, n0> n3 = <s4, ?, ?, n0>	
Closed		n0	

	I3	I4	I5
Open			
Closed			

h_1



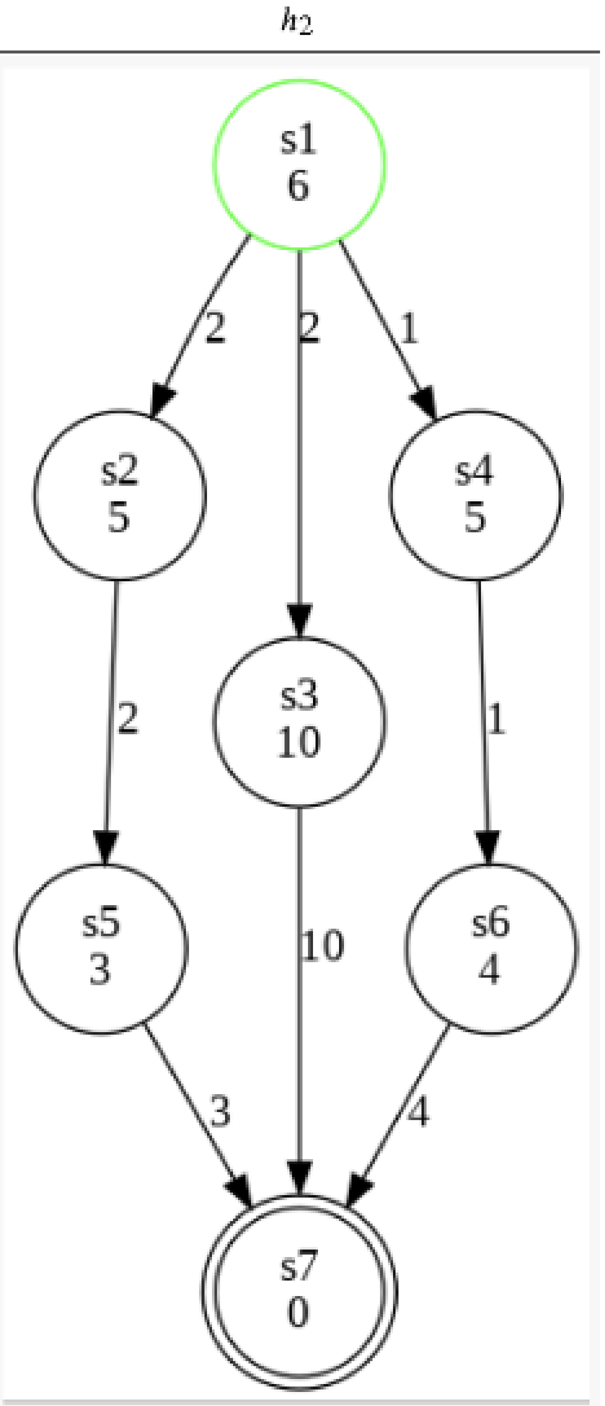
Node expansion order of A*, h1

	I0	I1	I2
Open	n0 = <s1, 4, 0, null>	n1 = <s2, 5, 2, n0> n2 = <s3, 7, 2, n0> n3 = <s4, 4, 1, n0>	n1 = <s2, 5, 2, n0> n2 = <s3, 7, 2, n0> n4 = <s6, 4, 2, n3>
Closed		n0	n0, n3

	I3	I4	I5
Open	n1 = <s2, 5, 2, n0> n2 = <s3, 7, 2, n0> n5 = <s7, 6, 6, n4>	n6 = <s5, 6, 4, n1> n2 = <s3, 7, 2, n0> n5 = <s7, 6, 6, n4>	
Closed	n0, n3, n4	n0, n1,n3, n4	n0, n1, n3, n4, n5

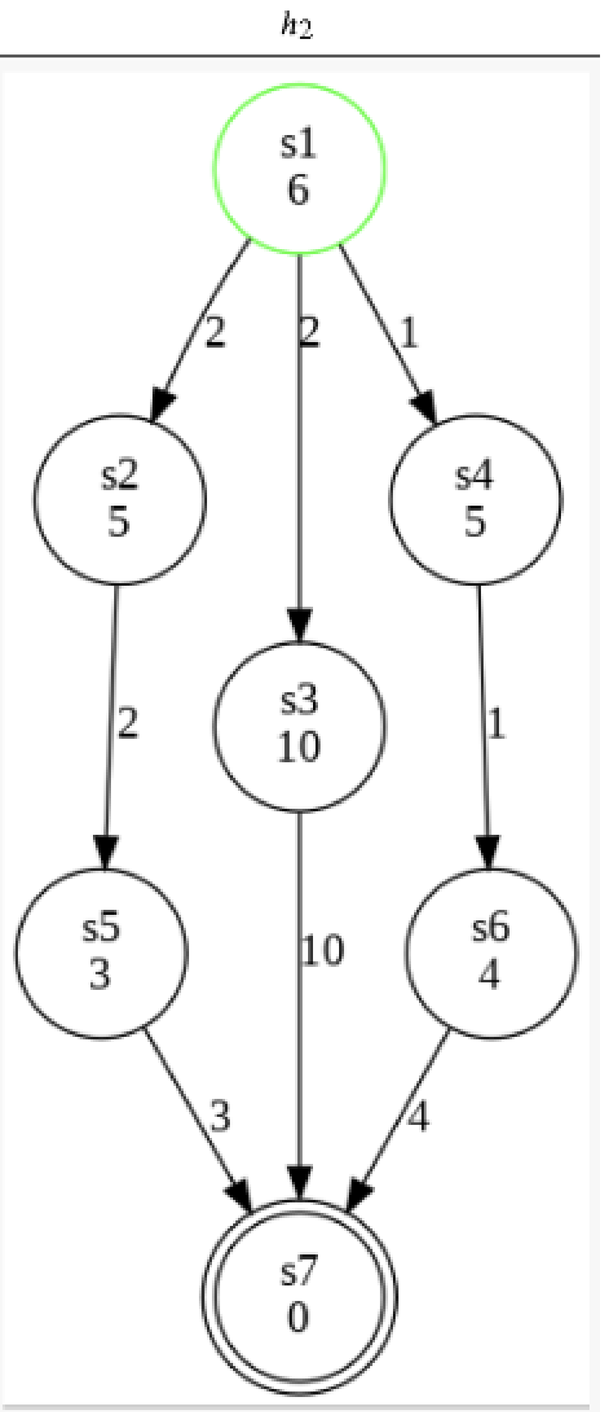
Task 2

- Choose h2 and perform A*
- Choose h2 and perform Greedy
- Choose h2 and perform WA*, weight =2



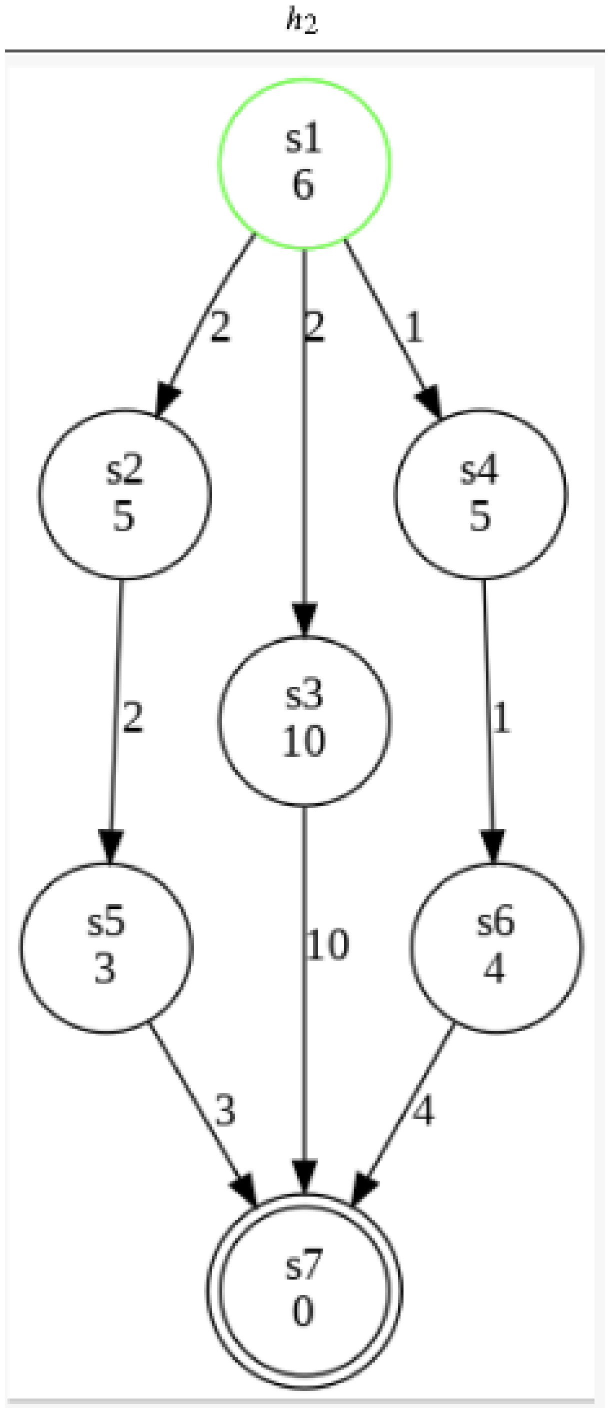
Node expansion order of A*, h2

```
nodes = [
# (state, fn, accumulated cost, id of parent node)
('s1', 6, 0, None),
('s4', 6, 1, 0),
('s6', 6, 2, 1),
('s7', 6, 6, 2)
]
```



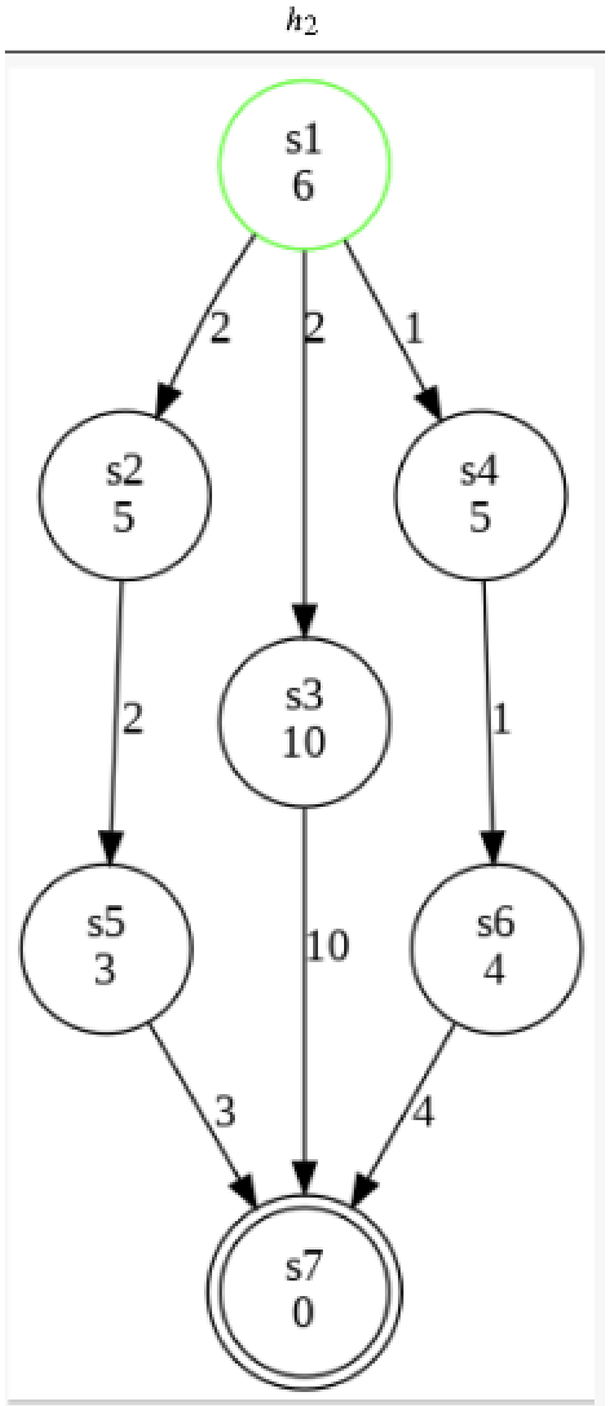
Node expansion order of Greedy, h2

```
nodes = [
# (state, fn, accumulated cost, id of parent node)
('s1', 6, 0, None),
('s4', 5, 1, 0),
('s6', 4, 2, 1),
('s7', 0, 6, 2)
]
```



Node expansion order of WA*, h2

```
nodes = [
# (state, fn, accumulated cost,
id of parent node)
('s1', 12, 0, None),
('s4', 11, 1, 0),
('s6', 10, 2, 1),
('s7', 6, 6, 2)
]
```



Task 2

- Which is the path returned as the solution?

- Is this the optimal plan?

(using h2 and A* as example)

Recap

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.

Recap

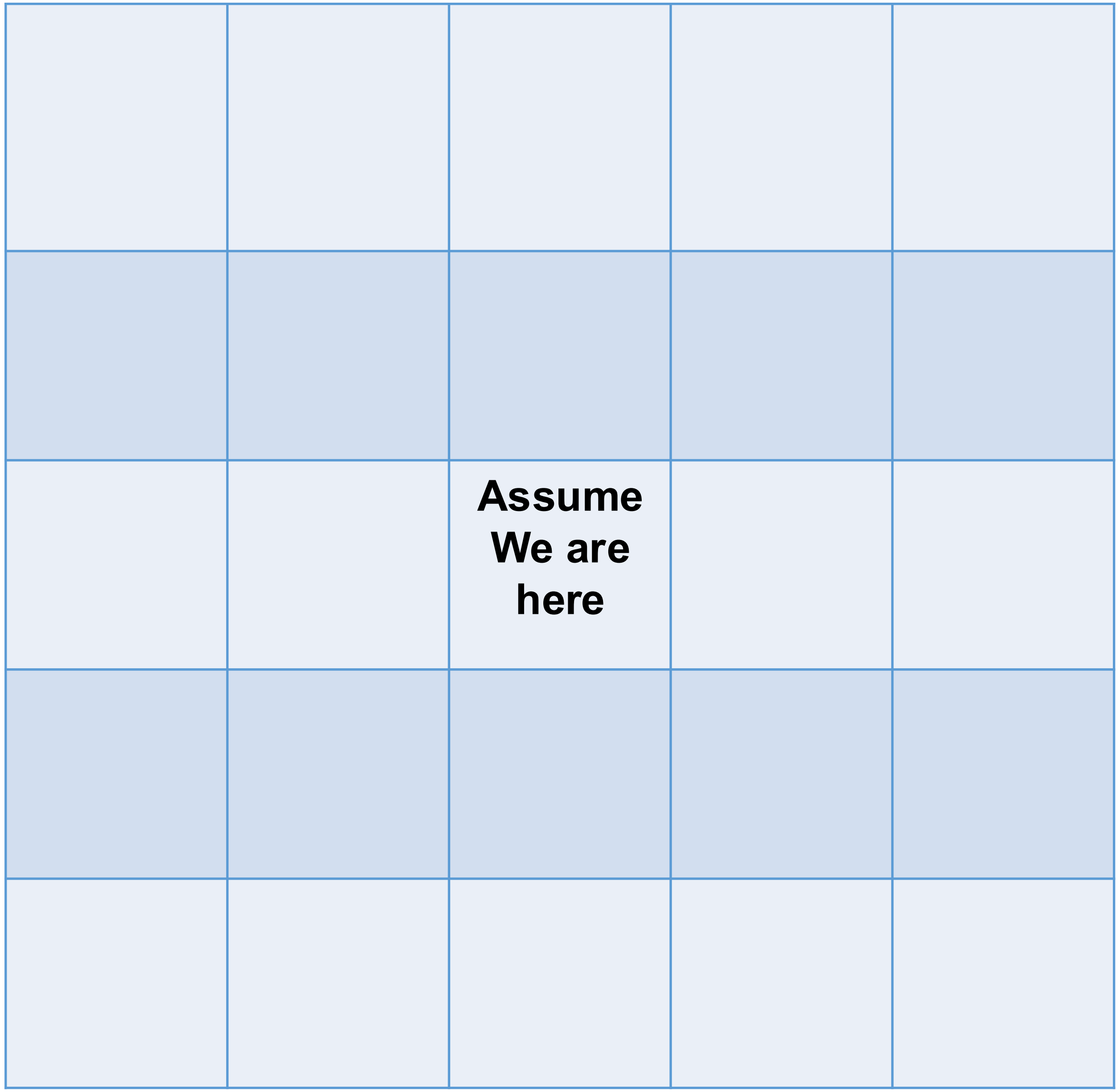
Let V' be the set contain visited cities:

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

Problem 2

- Consider an $m \times m$ **Manhattan Grid**, and a set of coordinates G to visit in any order.
- **Hint**: Consider a set of coordinates V' remaining to be visited, or a set of coordinates V already visited. What's the difference between them
- Formulate a state-based search problem to find a tour of all the desired points (i.e. define a state space, applicable actions, transition and cost functions).
- What is the branching factor of this search?
- What is the size of the state space in terms of m and G?

m x m Grid



P = {S, s0, SG, A, T, C}

Consider 2 ways :

- Using a set of coordinates V' remaining to be visited,
- Or a set of coordinates V already visited.
- What's the difference between them

Problem 2

- $S = \{\langle x, y, V' \rangle \mid x, y \in \{0, \ldots, m-1\} \wedge V' \subseteq G\}$

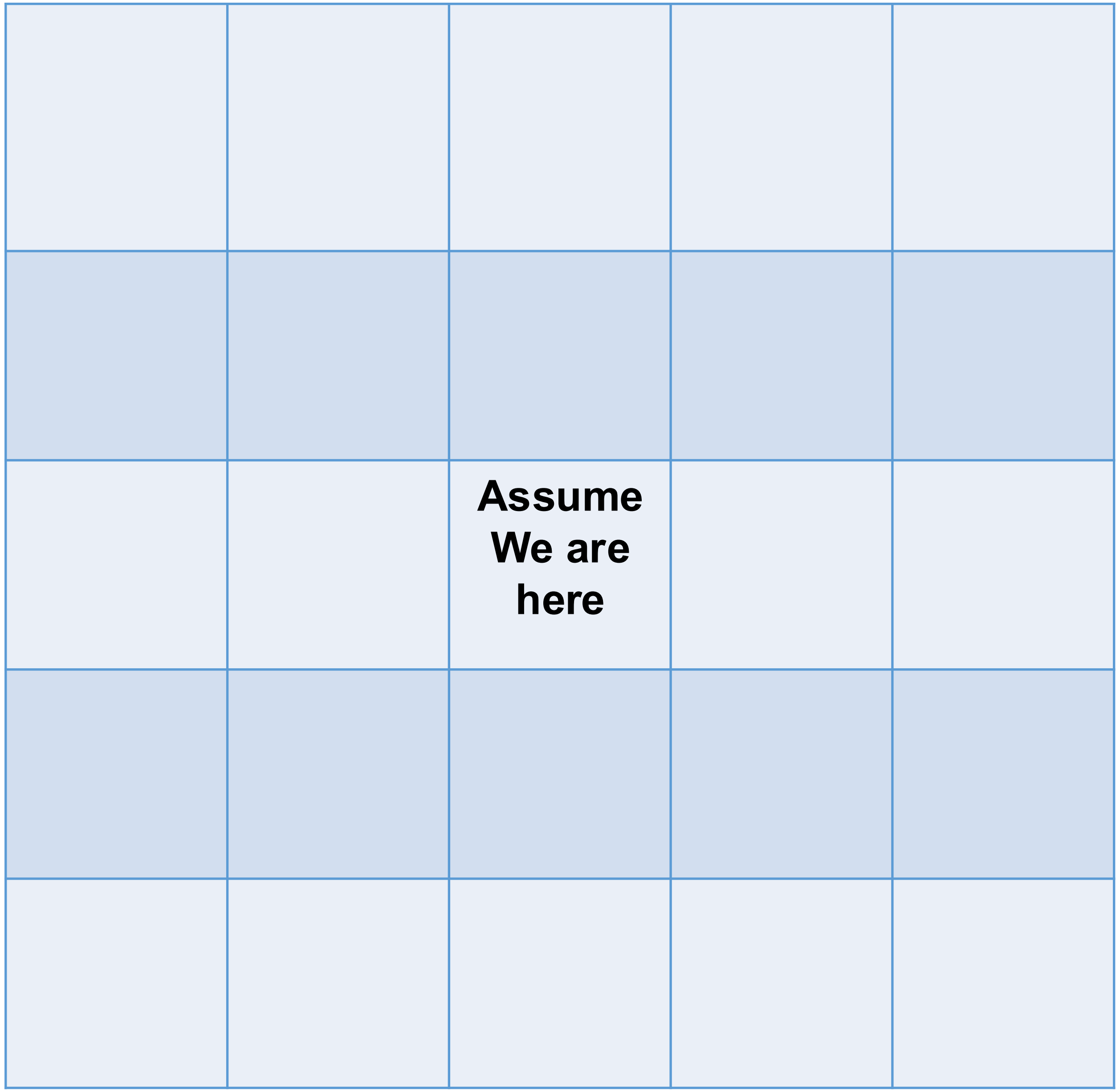
- $s_0 = \langle (0,0), G \setminus \{(0,0)\} \rangle$

- $S_G = \{\langle (x,y), \{\} \rangle \mid x, y \in \{0, \ldots, m-1\}\}$

- $A(\langle x, y, V' \rangle) = \{(dx, dy) \mid$
  - $dx, dy \in \{-1, 0, 1\}$
  - $\wedge\ |dx| + |dy| = 1$
  - $\wedge\ x + dx, y + dy \in \{0, \ldots, m-1\}$
  - $(x + dx, y + dy) \notin W\ \}$
- $T(\langle x, y, V' \rangle, (dx, dy)) = \langle x + dx, y + dy, V' \setminus \{(x + dx, y + dy)\} \rangle$

- $c(a, s) = 1$

m x m Grid

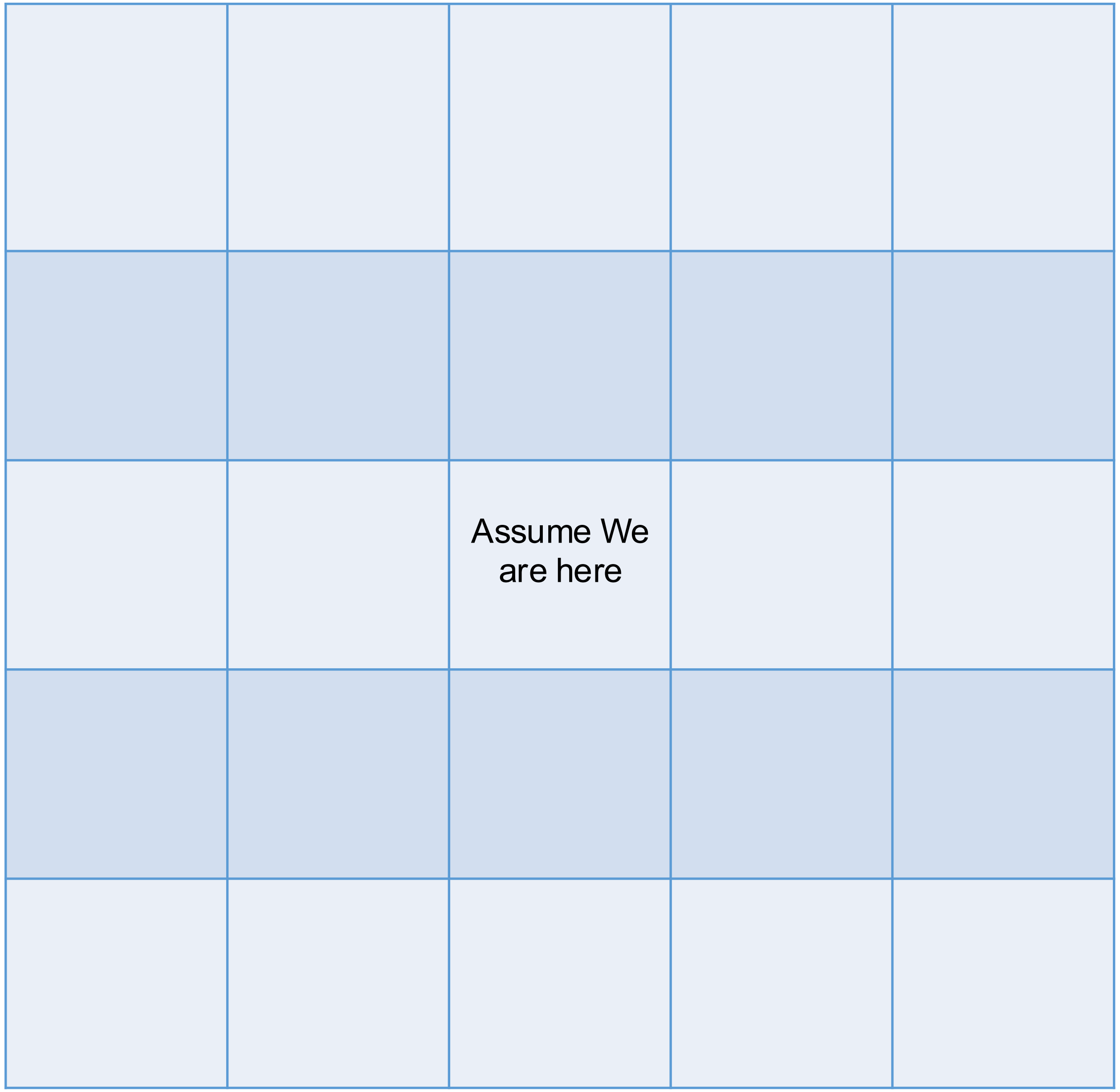


P = {S, s0, SG, A, T, C}

Consider 2 ways :

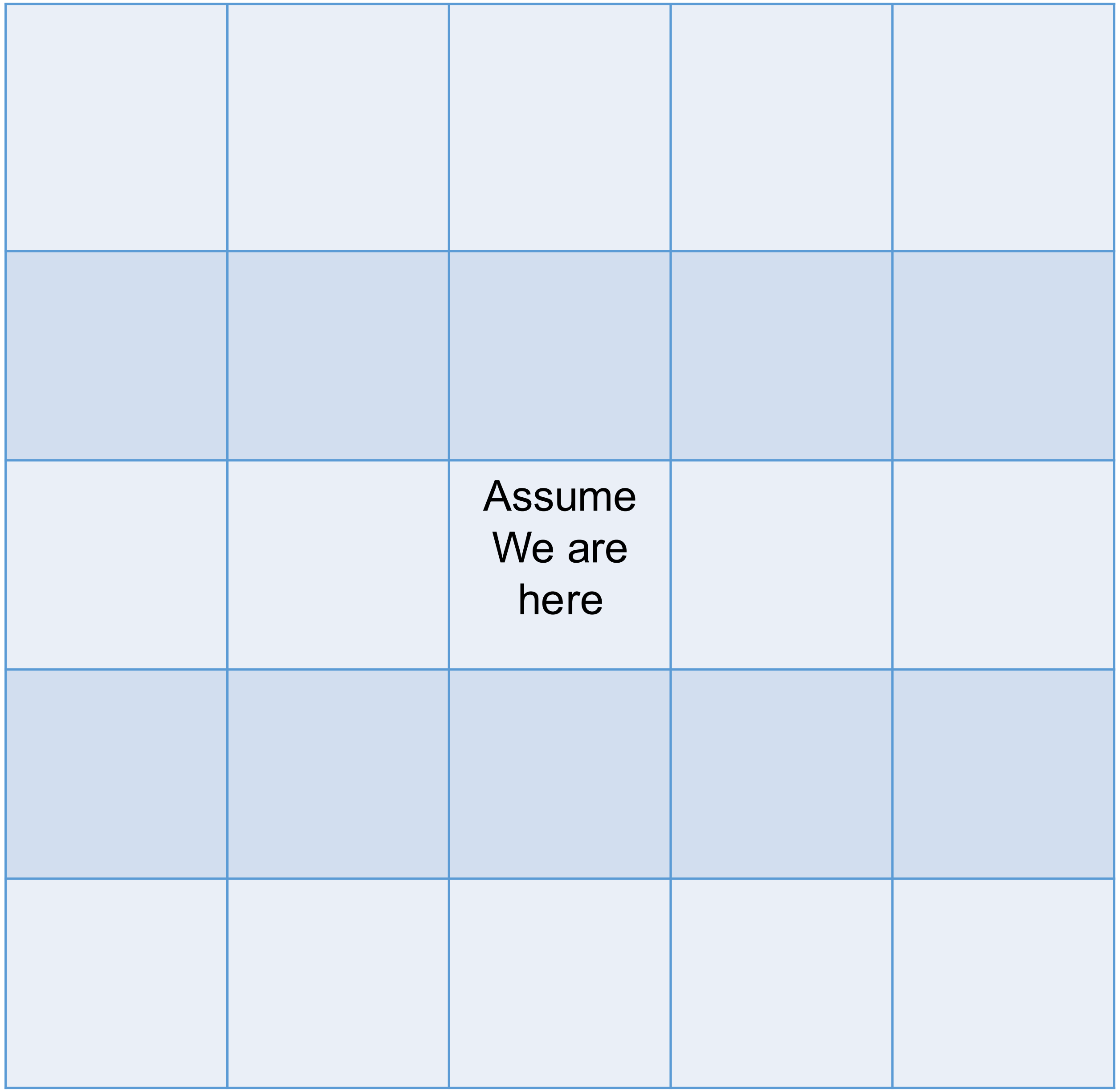
- Using a set of coordinates V' remaining to be visited,
- Or a set of coordinates V already visited.
- What's the difference between them

m x m Grid



- What is the branching factor of this search?
- What is the size of the state space in terms of m and G?

m x m Grid



- What is the branching factor of this search?

- What is the size of the state space in terms of m and G?

If using V', then $m^2 \times 2^{|G|}$

If using V, then $m^2 \times 2^{|m \times m|}$