Week 3: Heuristic Search

COMP90054 – AI Planning for Autonomy

Key concepts

- Heuristic Functions and their properties and relations
- Heuristic search algorithms
- State-space model and size of the problem

Problem 1: Task 1

Heuristic function

h(s) estimates the distance from the current state **s** to the *closest* goal state

$\boldsymbol{h^*(s)}$ is a **perfect heuristic,** the optimal cost from the current state to the goal state

Problem 1: Task 1

- *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 $s \xrightarrow{a} s'$.

Heuristic function's properties

4 properties:

- **Safe**: if a solution exists from state s, then $h(s) < \infty$
- **Goal-aware**: All goal states have a heuristic h = 0
- **Admissible**: never over-estimate the cost $h(s) \leq h^*(s)$
- **Consistent**: the cost diff between the parent and the child heuristics is never larger than the actual cost

$$\underbrace{h(s)}_{\text{parent}} - \underbrace{h(s')}_{\text{child}} \leq c(s, s')$$

Problem 1: Task 1

$h_1(s) \leq h_1^*(s)$ for all states

$4 < 6$

Admissible: for all $s \in S$, $h(s) \leq h^*(s)$

$h^*(s1) = \min(7, 12, 6) = 6$

Which Heuristics are admissible?

$h^*(s2) = 2+3 = 5$

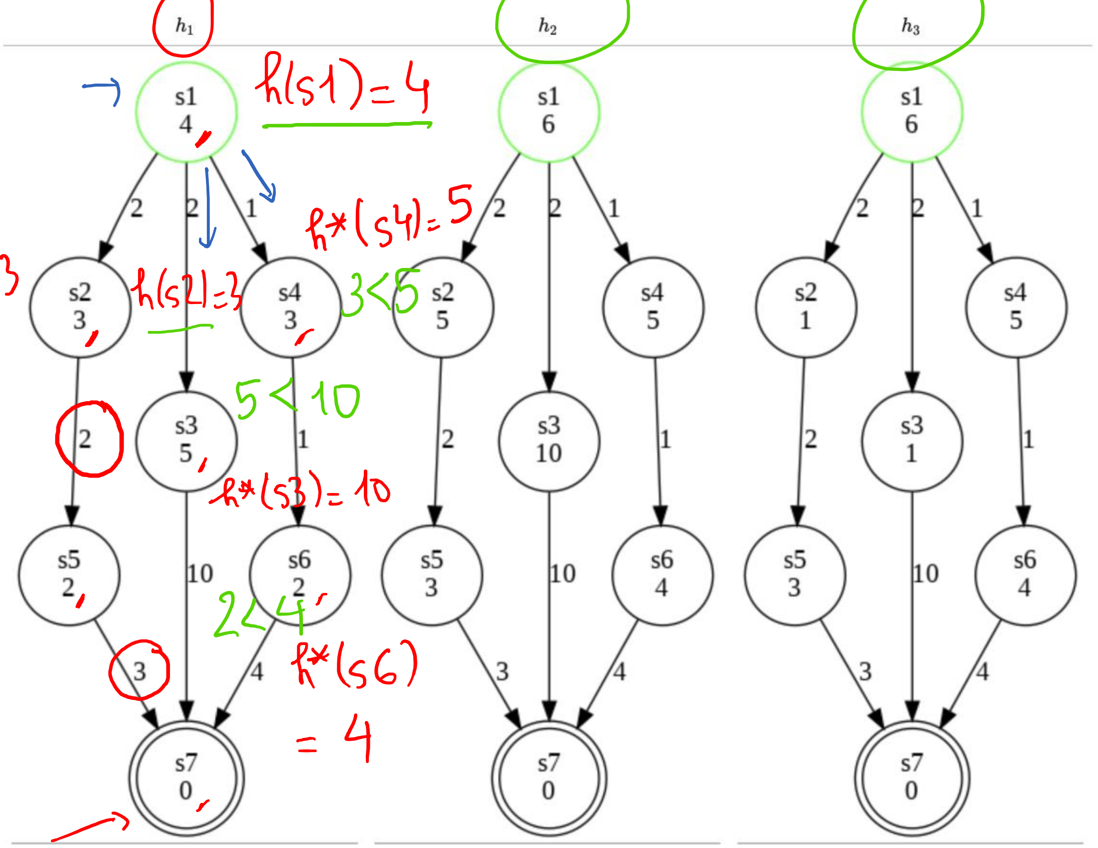
$3 < 5$

$h^*(s5) = 3$

$2 < 3$

$h^*(s7) = 0$

h1, h2, h3



$0 = 0$

Problem 1: Task 1

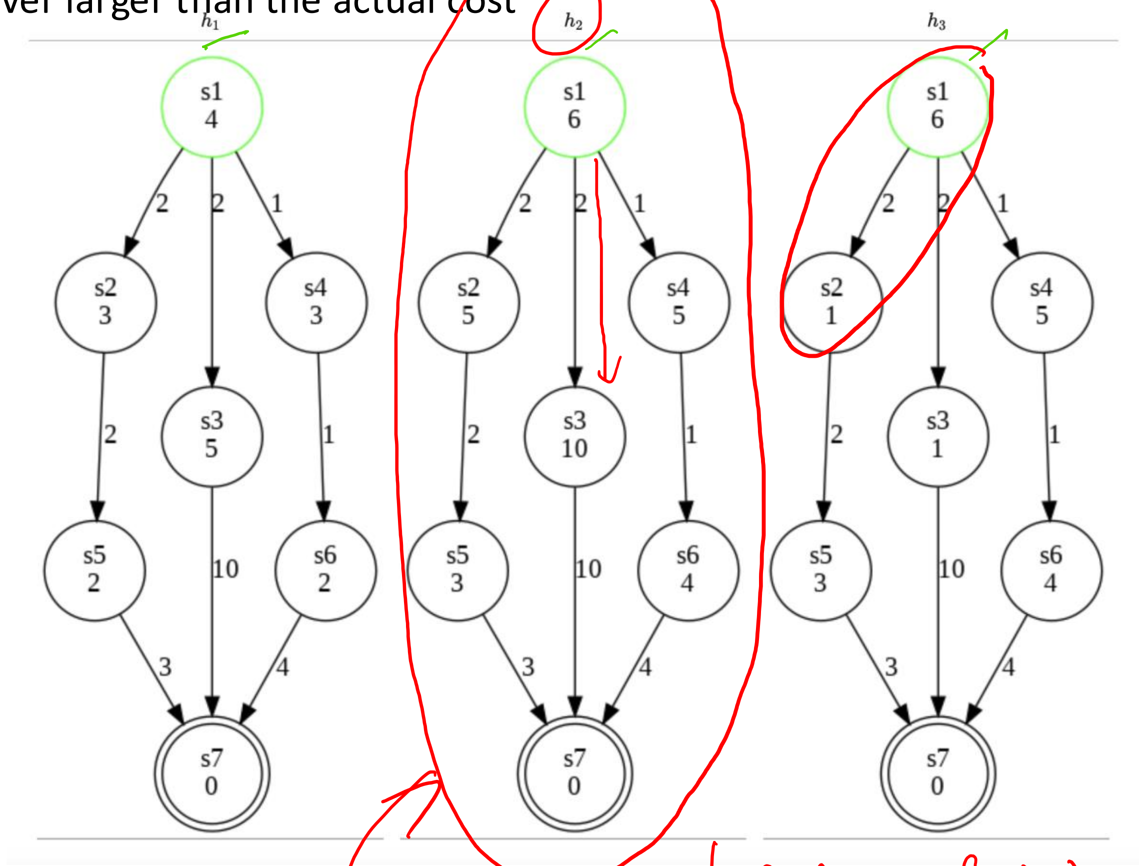
$h(s1) - h(s_3) = 6 - 10 = -4 < 2$

Consistent: the cost diff between the parent and the child heuristics is never larger than the actual cost

$$\underbrace{h(s)}_{\text{parent}} - \underbrace{h(s')}_{\text{child}} \leq \underbrace{c(s,s')}_{\text{actual cost}}$$

Which heuristics are consistent?

h1, h2



$h_2(s)$ vs $h_2^*(s)$

perfect heuristic

$h_3(s1) - h_3(s2)$
$= 6 - 1 = 5 > 2$

Problem 1: Task 1

$h_2(s1) \geq h_1(s1)$ $h_2(s1) \geq h_3(s1)$

Dominate relation

h1 dominates h2 if

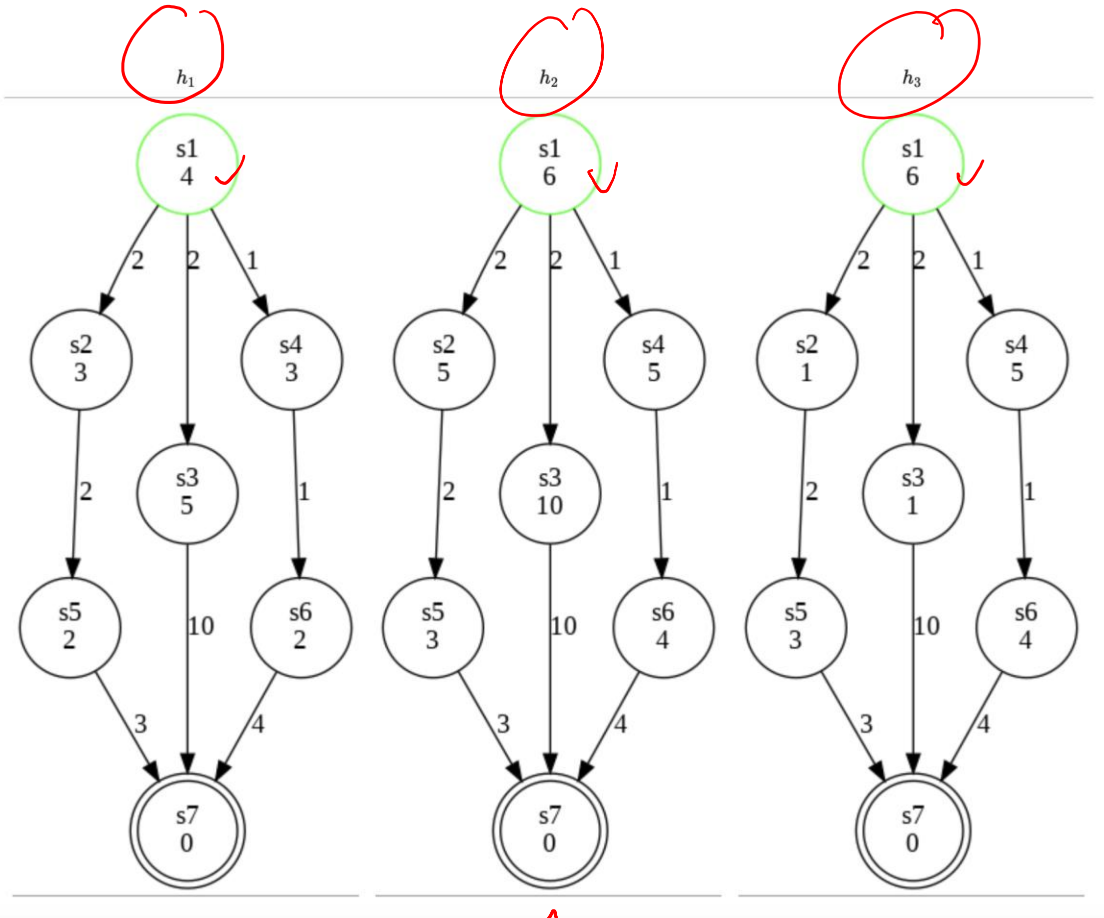
- both heuristics are admissible (1)
- h2 <= h1 <= h* for all s in S (2)

Does any of the heuristic dominate any other?

h2 dominates h1

h2 dominates h3

Note: If h1 >= h2, A* with h1 will generally expand fewer nodes than A* with h2



perfect heuristic.

Problem 1: Task 2

Heuristic search algorithms

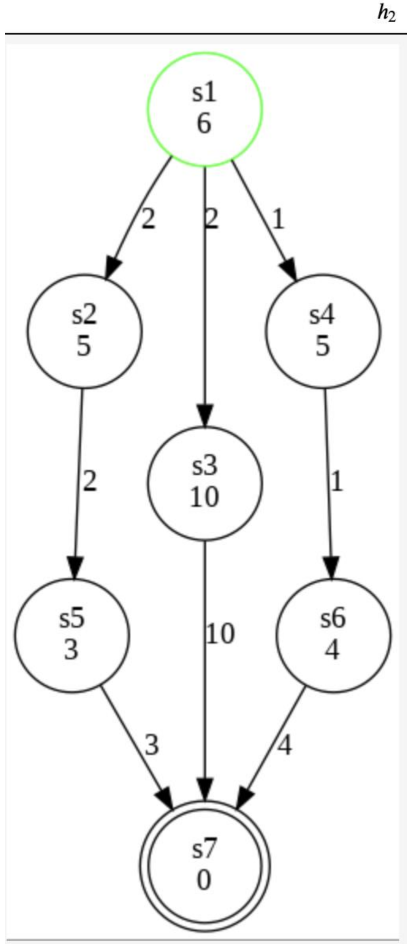
Search node: n = <s, f(n), g(n), parent n>
f(n) is a priority value for node in the priority queue

DS: priority queue

- Uniform-cost search (Dijkstra): f(n) = g(n) ← blind search
- Greedy best-first search: f(n) = h(s)
- A*: f(n) = h(s) + g(n)
- WA*: f(n) = W * h(s) + g(n)

heuristic search.

blind search: do not consider heuristic function.



Problem 1: Task 2

Search node: n = <s, f(n), g(n), parent n>

f(n) = h(s)

Greedy best-first search

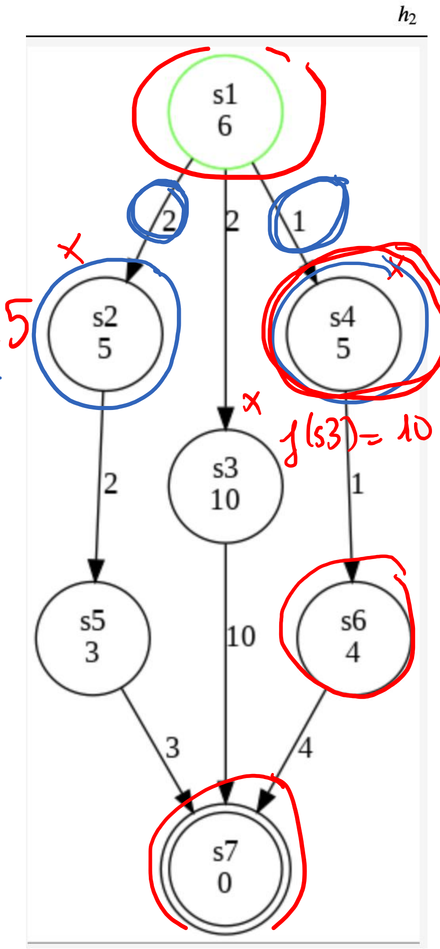
Step	Open (Priority Queue)	Close (Visited)
1	n0 = <s1, 6, 0, None>	
2	n1 = <s2, 5, 2, 0> n2 = <s3, 10, 2, 0> n3 = <s4, 5, 1, 0>	n0
3	n1 = <s2, 5, 2, 0> n2 = <s3, 10, 2, 0> n4 = <s6, 4, 2, 3>	n0, n3
4	n1 = <s2, 5, 2, 0> n2 = <s3, 10, 2, 0> n5 = <s7, 0, 6, 4>	n0, n3, n4
5	n1 = <s2, 5, 2, 0> n2 = <s3, 10, 2, 0>	n0, n3, n4, n5

Solution: s1 -> s4 -> s6 -> s7

f(n) = h(s)

f(s2) = 5

f(s4) = 5



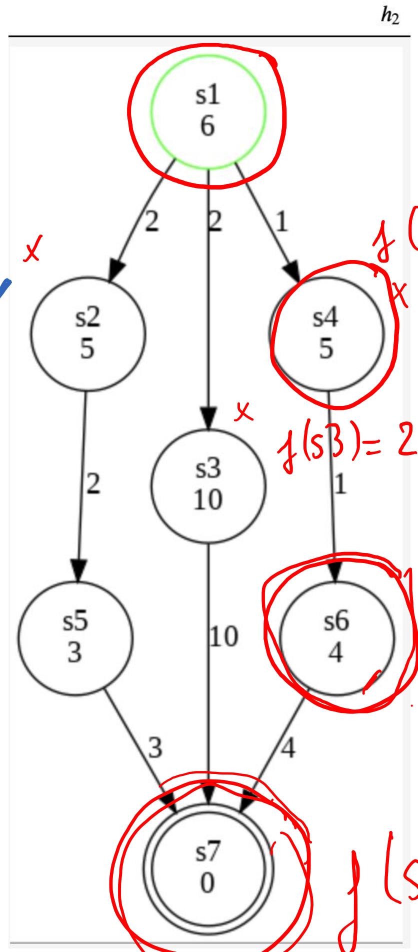
Problem 1: Task 2

Search node: n = <s, f(n), g(n), parent n>

f(n) = h(s) + g(n)

A*

Step	Open (Priority Queue)	Close (Visited)
1	n0 = <s1, 6, 0, None>	
2	n1 = <s2, 7, 2, 0> n2 = <s3, 12, 2, 0> n3 = <s4, 6, 1, 0>	n0
3	n1 = <s2, 7, 2, 0> n2 = <s3, 12, 2, 0> n4 = <s6, 6, 2, 3>	n0, n3
4	n1 = <s2, 7, 2, 0> n2 = <s3, 12, 2, 0> n5 = <s7, 6, 6, 5>	n0, n3, n4
5	n1 = <s2, 7, 2, 0> n2 = <s3, 12, 2, 0>	n0, n3, n4, n5



f(s2)= 2+5=7

f(s4)=1+5 =6

f(s3)= 2+10=12

f(s6)= 2+4 =6

f(s7)= 6

Solution: s1 -> s4 -> s6 -> s7

Problem 1: Task 2

Search node: n = <s, f(n), g(n), parent n>

f(n) = W * h(s) + g(n)   W = 2

WA* (W = 2)

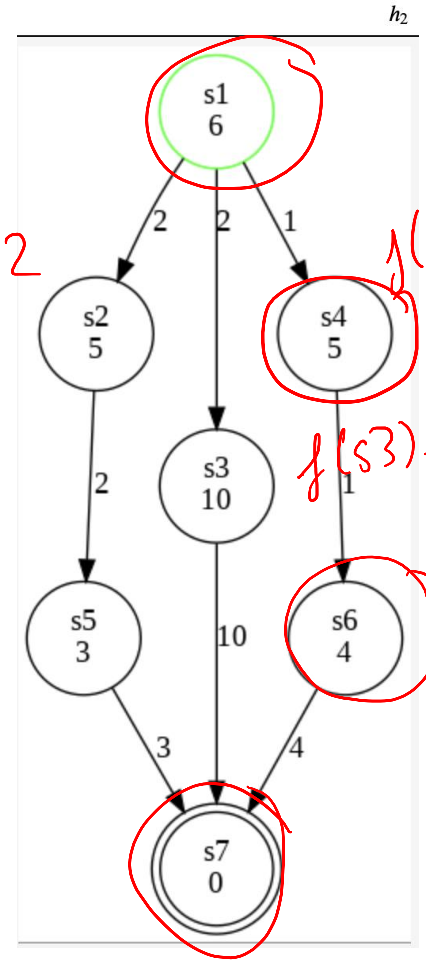
Step	Open (Priority Queue)	Close (Visited)
1	n0 = <s1, 12, 0, None>	
2	n1 = <s2, 12, 2, 0> n2 = <s3, 22, 2, 0> n3 = <s4, 11, 1, 0>	n0
3	n1 = <s2, 12, 2, 0> n2 = <s3, 22, 2, 0> n4 = <s6, 10, 2, 3>	n0, n3
4	n1 = <s2, 12, 2, 0> n2 = <s3, 22, 2, 0> n5 = <s7, 6, 6, 4>	n0, n3, n4
5	n1 = <s2, 12, 2, 0> n2 = <s3, 22, 2, 0>	n0, n3, n4, n5

Solution: s1 -> s4 -> s6 -> s7

f(s2) = 2x5+2 = 12

f(s4) = 2x5 +1 = 11

f(s3) = 2x10+2 = 22



Problem 1: Task 2

Heuristic algorithms

Which is the path returned as solution? (using $h2$ and A* as example)

s1 -> s4 -> s6 -> s7

Is this the optimal plan? Has the algorithm proved this? (using $h2$ and A* as example)

Yes. h2 is both admissible and consistent

Note about A* optimality

A* will return an optimal solution:

- If using A* with re-opening (lecture slides) and heuristic function is admissible
- If using A* without re-opening (original algo) and heuristic function is both admissible and consistent

A^* (with duplicate detection and re-opening)

open := **new** priority queue ordered by ascending $g(state(\sigma)) + h(state(\sigma))$
open.insert(make-root-node(init()))
closed := $\emptyset$
best-g := $\emptyset$ /* maps states to numbers */
while not *open*.empty():
    σ := *open*.pop-min()
    if $state(\sigma) \notin closed$ **or** $g(\sigma) < best\text{-}g(state(\sigma))$:
        /* re-open if better g; note that all σ' with same state but worse g are *behind* σ in *open*, and will be skipped when their turn comes */
        closed := *closed* $\cup \{state(\sigma)\}$
        best-g$(state(\sigma)) := g(\sigma)$
        if is-goal(state(σ)): **return** extract-solution(σ)
        for each $(a, s') \in$ succ$(state(\sigma))$:
            σ' := make-node(σ, a, s')
            if $h(state(\sigma')) < \infty$: *open*.insert($\sigma'$)
return unsolvable

re visit a closed state

State space model

Problem 2

(1) (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 $\boxed{V' \text{ remaining to be visited,}}$ or a $\boxed{\text{set of coordinates } V \text{ already visited}}$. What's the difference between them

(x,y)		

Formulate a state-based search problem to find a tour to all the desired points

State space model:
P = $<s_0, S, S_G, A, f, c>$

a state = <current coordinate, a set of remaining coordinates>

1st way

Initial state $s_0 = <(0,0), G \setminus \{(0,0)\}>$

Goal state $S_G = \{<(x,y), \{\}> \mid x, y \in \{0, \dots, m-1\}\}$

State $S = \{<(x,y), V'> \mid x, y \in \{0, \dots, m-1\} \wedge V' \subseteq G\}$

Action $A(<(x,y), V'>) = \{(dx, dy) \mid dx, dy \in \{-1, 0, 1\}$
$\wedge\ |dx| + |dy| = 1$
$\wedge\ x + dx, y + dy \in \{0, \dots, m-1\}\}$

Transition $f(<(x,y), V'>, (dx, dy)) = <(x+dx, y+dy),\ V' \setminus \{(x+dx, y+dy)\}>$

Cost $c(a, s) = 1$

Problem 2

size of the state space

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 to all the desired points

(x,y)		

State space model: a state = <current coordinate, a set of visited coordinates>

2nd way

P = <s_0, S, S_G, A, f, c>

Initial state $s_0 = <(0,0), \{(0,0)\}>$

Goal state $S_G = \{<(x,y), V> \mid x, y \in \{0, \dots, m-1\} \wedge G \subseteq V\}$

State $S = \{<(x,y), V> \mid x, y \in \{0, \dots, m-1\} \wedge V \subseteq \{(x', y') \mid x', y' \in \{0, \dots, m-1\}\}\}$

Action $A(<(x,y), V>) = \{(dx, dy) \mid dx, dy \in \{-1, 0, 1\}$
$\wedge\ |dx| + |dy| = 1$
$\wedge\ x + dx, y + dy \in \{0, \dots, m-1\}\}$

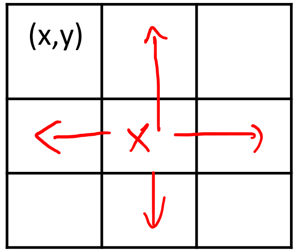
Transition $f(<(x,y), V>, (dx, dy)) = <(x+dx, y+dy),\ V \cup \{(x+dx, y+dy)\}>$

Cost $c(a, s) = 1$

Problem 2

Consider an $m \times m$ **Manhattan Grid**, and a set of coordinates G to visit in any order.

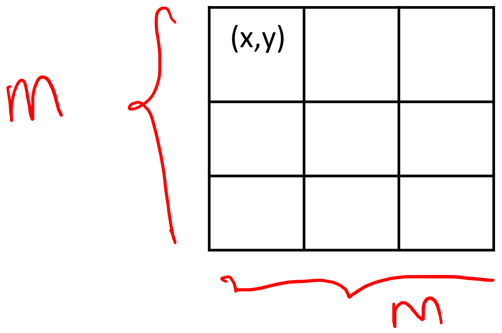
Hint: Consider a set of coordinates V' remaining to be visited, or a set of coordinates V already visited. What's the difference between them



What is the branching factor of the search?

4 (branching factor = max number of child nodes)

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

(1) (2)

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

2 < visit / not visited yet.

(1) If using V' (a set of remaining coordinates), then $m^2 \times 2^{|G|}$

(2) If using V (a set of visited coordinates), then $m^2 \times 2^{|m \times m|}$

If you have a small $|G|$.
$|G| < |m \times m|$
→ it's much better to do the 1st way

State $\boldsymbol{S} = \{< (x,y), V' > | x,y \in \{0,\ldots,m-1\} \wedge V' \subseteq G\}$

State $\boldsymbol{S} = \{< (x,y), V > | x,y \in \{0,\ldots,m-1\} \wedge V \subseteq \{(x',y') \mid x',y' \in \{0,\ldots,m-1\}\}\}$