# 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** $h(s)$

**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

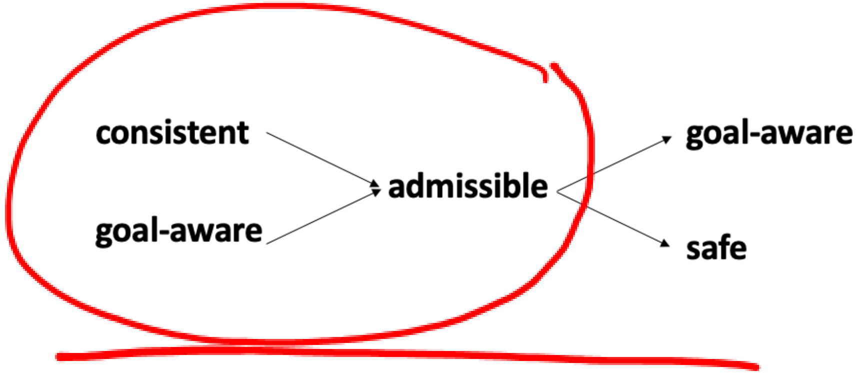
$h^*(s)$

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

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

# Problem 1: Task 1

$h(s) \le h^*(s)$

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

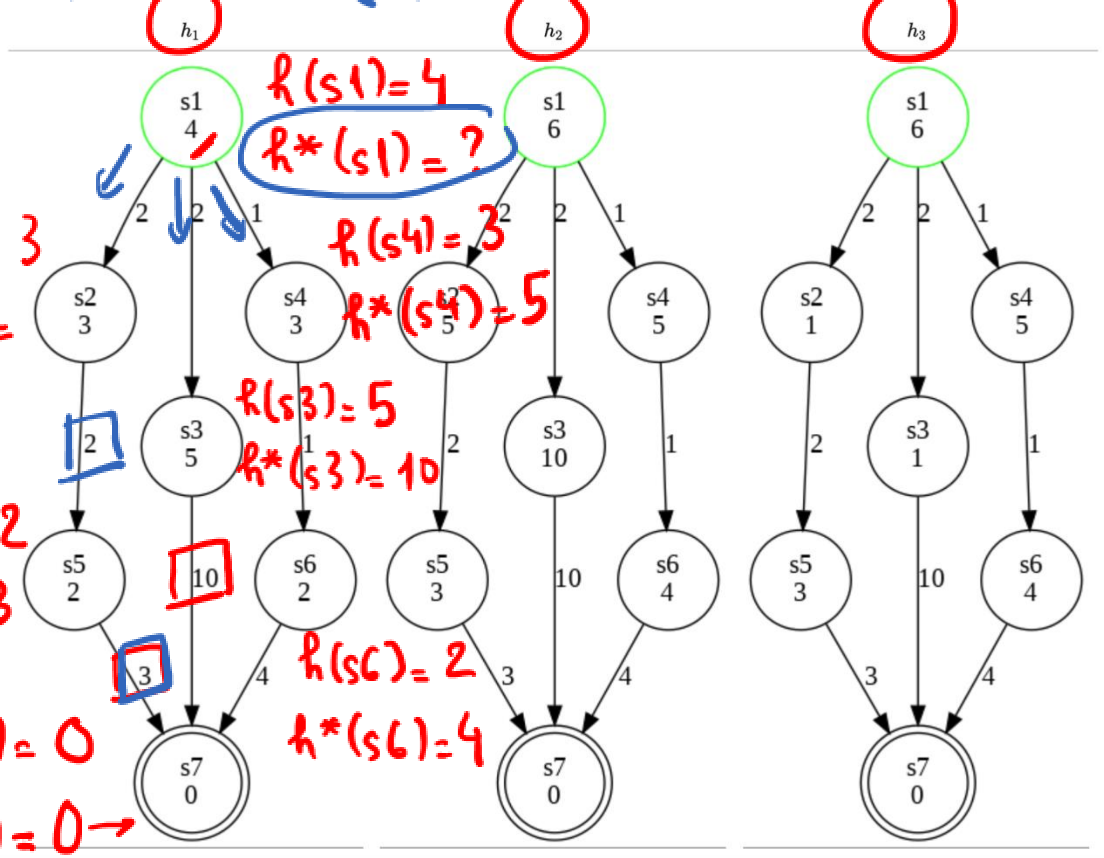
**Which Heuristics are admissible?**

h1, h2, h3

$h(s) \le h^*(s)$

$\forall s \in S$

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




$h(s1) = 4$

$h^*(s1) = ?$

$h(s2) = 3$

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

$h(s4) = 3$

$h^*(s4) = 5$

$h(s3) = 5$

$h^*(s3) = 10$

$h(s5) = 2$

$h^*(s5) = 3$

$h(s6) = 2$

$h^*(s6) = 4$

$h(s7) = 0$

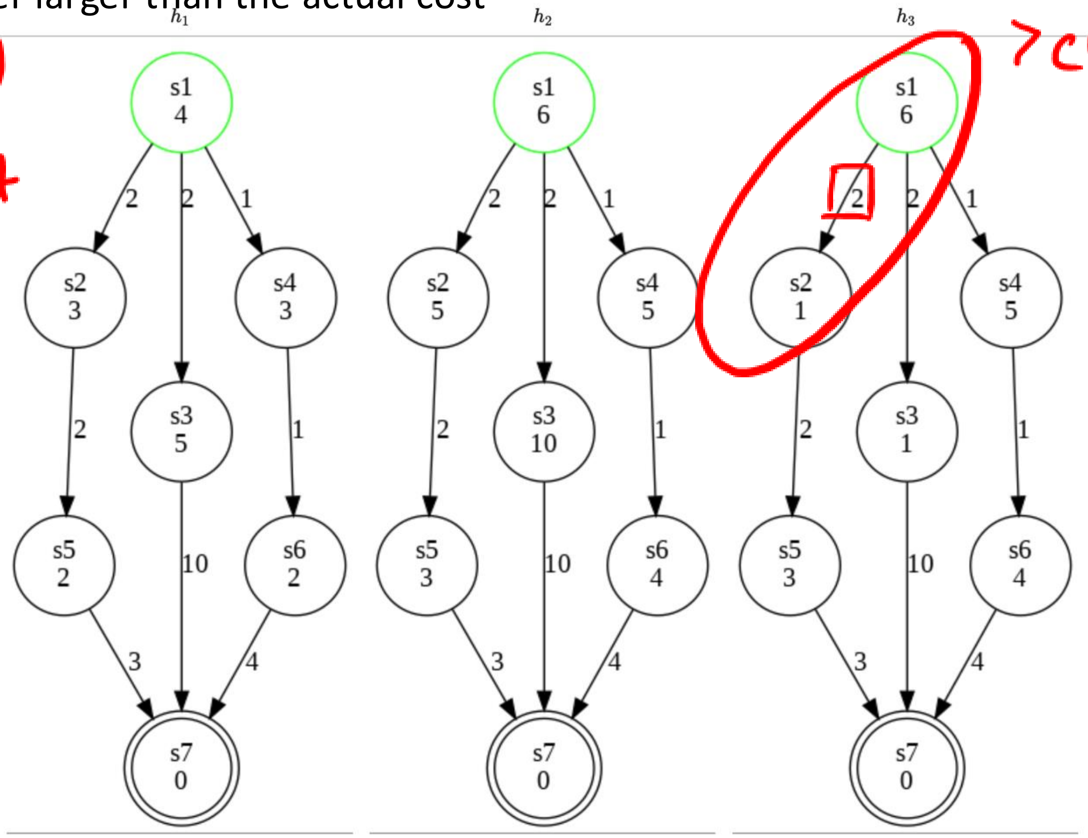
$h^*(s7) = 0$

# Problem 1: Task 1

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

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

$h(s1) - h(s2) = 5 > c(s1, s2)$




→ h3 X

**Which heuristics are consistent?**

h1, h2

# Problem 1: Task 1

~ how many nodes that you can expand.

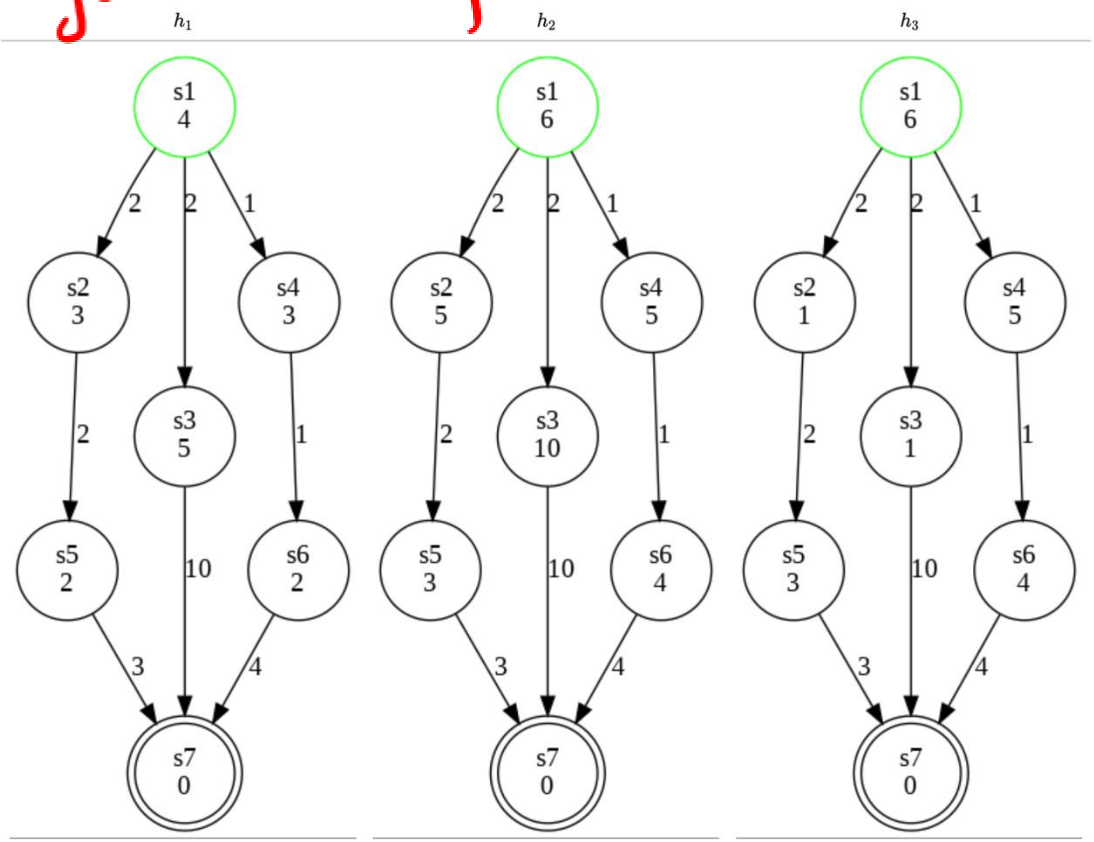
**Dominate relation**

h1 dominates h2 if

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

$h_1(s) \geq h_2(s) \ \forall s \in S$

**Does any of the heuristic dominate any other?**




h2 dominates h1

h2 dominates h3

(1) All admissible

(2) $h_2$ is the perfect heuristic

→ $h_2 \geq h_1$

$h_2 \geq h_3$

# Problem 1: Task 2

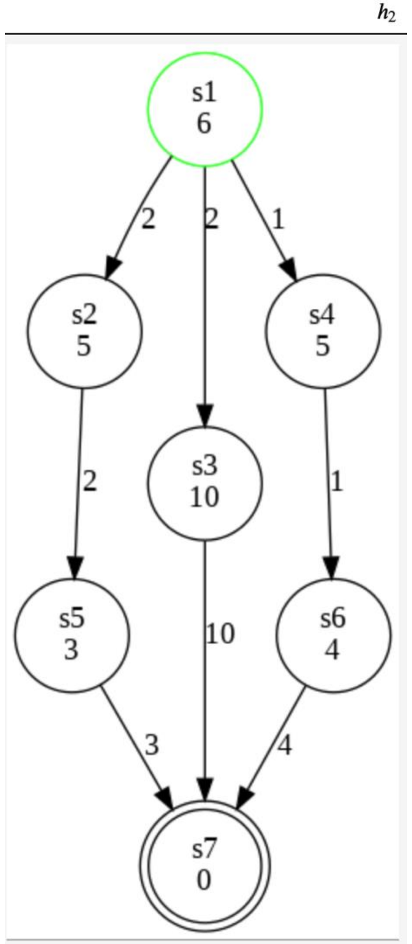
**Heuristic search algorithms**

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)
- Greedy best-first search: f(n) = h(s)
- A*: f(n) = $\boxed{\text{h(s) + g(n)}}$
- WA*: f(n) = W * h(s) + g(n)



# Problem 1: Task 2

priority value = heuristic value

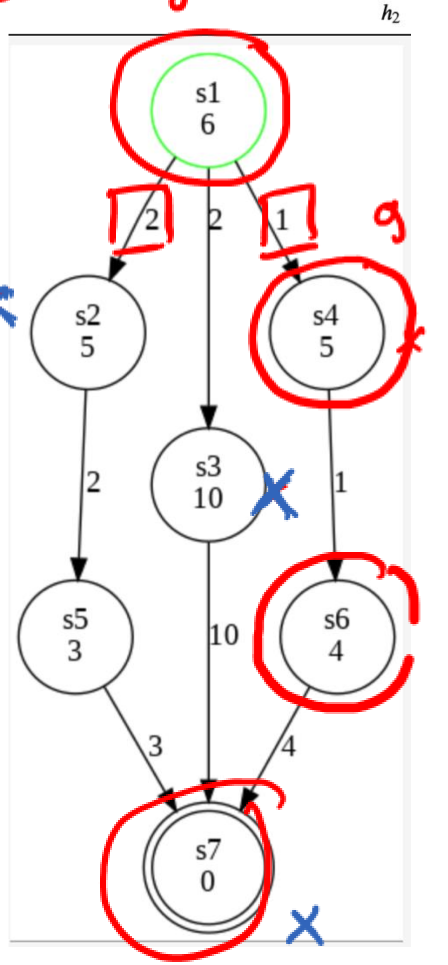
f(s1) = 6

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

**Greedy best-first search**

f(s2) = 5

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




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

# Problem 1: Task 2

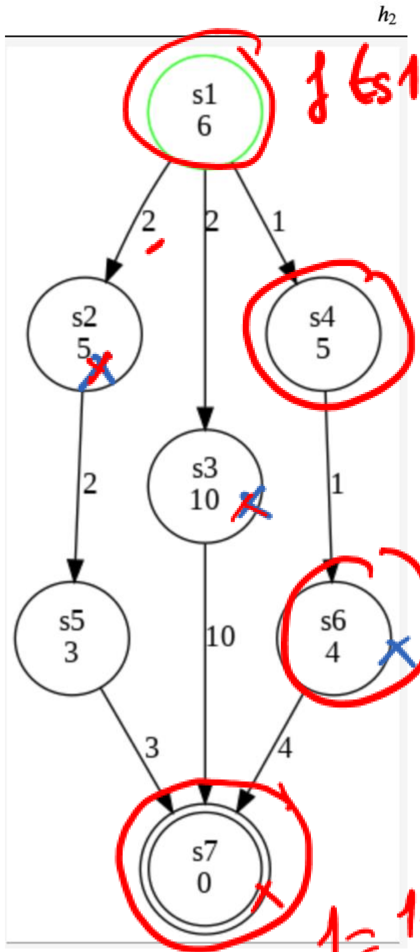
$f(n) = h + g$

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><br>n2 = <s3, 12, 2, 0><br>n3 = <s4, 6, 1, 0> | n0 |
| 3 | n1 = <s2, 7, 2, 0><br>n2 = <s3, 12, 2, 0><br>n4 = <s6, 6, 2, 3> | n0, n3 |
| 4 | n1 = <s2, 7, 2, 0><br>n2 = <s3, 12, 2, 0><br>n5 = <s7, 6, 6, 5> | n0, n3, n4 |
| 5 | n1 = <s2, 7, 2, 0><br>n2 = <s3, 12, 2, 0> | n0, n3, n4, n5 |

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




$f(s1) = 6$

$f = 5+2 = 7$

$f = 1+1+4+0 = 6$

# Problem 1: Task 2

W=2

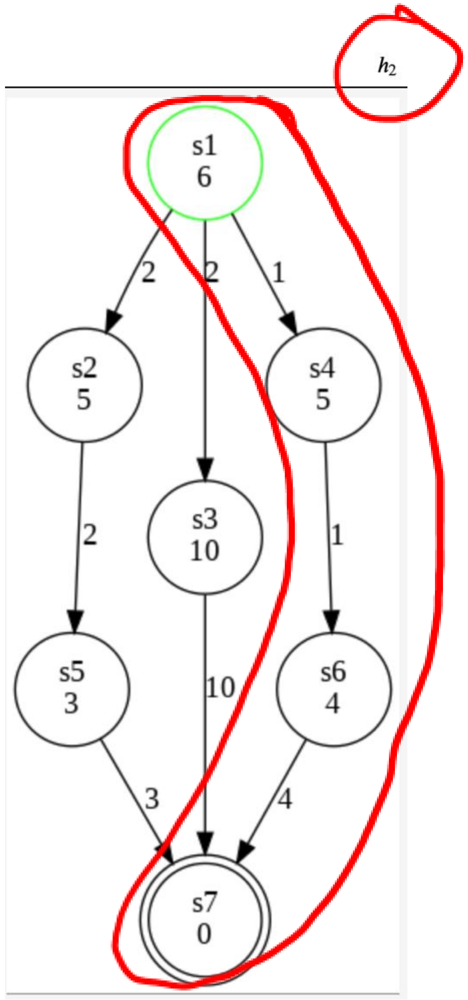
Search node: n = <s, f(n), g(n), parent n>

$\boxed{f(n) = W * h(s) + g(n)}$

**WA* (W = 2)**

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

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



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

re-visit the closed states

# 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

| (x,y) | | g1 |
|---|---|---|
| | | |
| | g2 | |

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

a state = <current coordinate, a set of remaining coordinates> (1)

a state = <current coordinate, a set of visited coordinates> (2)

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

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

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

- (1) : $|G|$
- (2) : $|m \times m|$

$|G| \leq |m \times m|$.

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

**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

**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>

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

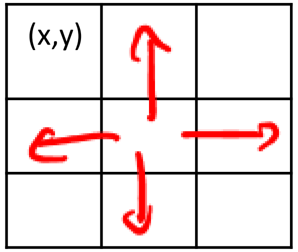
**Cost** $c(a, s) = 1$

# 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

| | | |
|---|---|---|
| (x,y) | | $g_1$ |
| | | |
| | $g_2$ | |

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 size of the state space in terms of m and G?**

$$\underbrace{m^2}_{\text{current coordinates}} \times \underbrace{2^{|G|}}_{\text{remaining coordinates.}}$$

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

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

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

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

$\rightarrow \{g_1, g_2\}$ $\rightarrow 2^{|G|}$

visited x visited x

visited

not