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

$\boldsymbol{h}^*(\boldsymbol{s})$ 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
- **Consistent**: the cost diff between the parent and the child heuristics is never larger than the actual cost

$$\underbrace{h(s)}_{\text{heuristic of the parent node}} - \underbrace{h(s')}_{\text{child}} \leq \underbrace{c(a)}_{\text{actual cost}}$$

Problem 1: Task 1

$h^*(s)$?

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

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

Which Heuristics are admissible?

h1, h2, h3

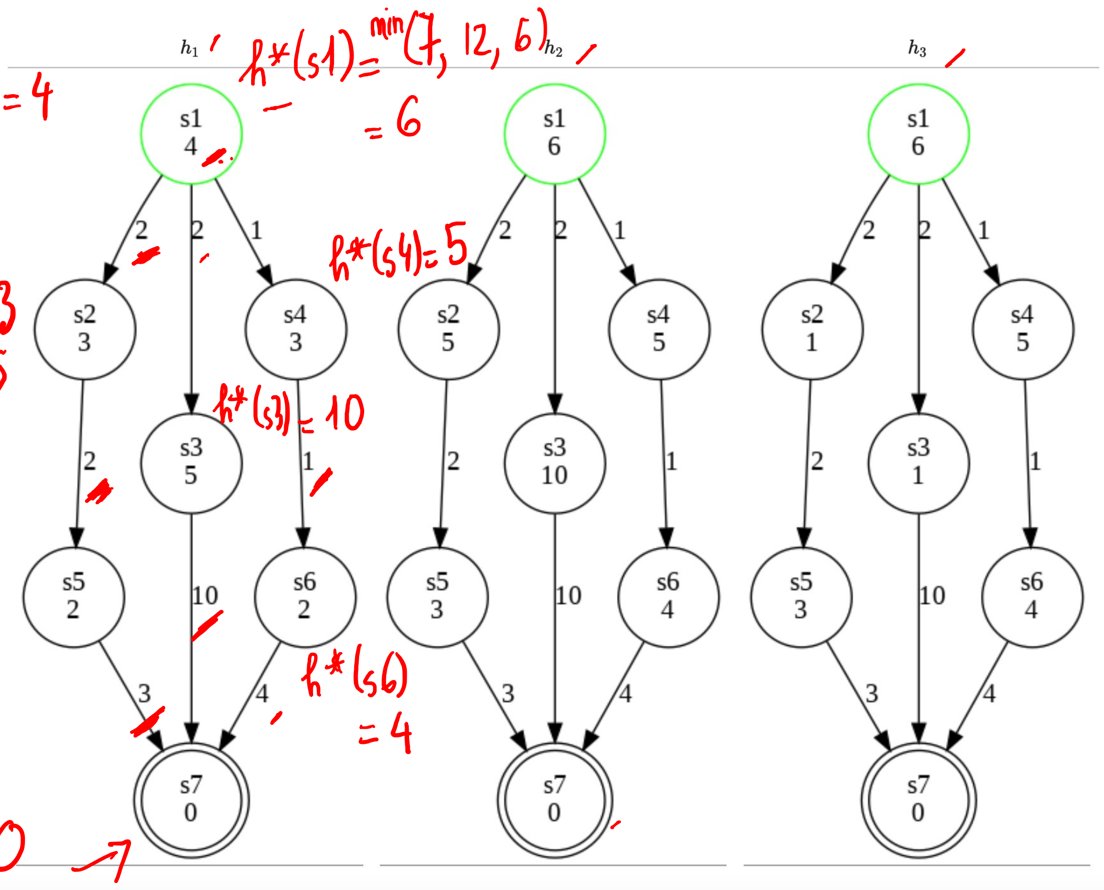
$h(s1) = 4$

$h(s2) = 3$

$h^*(s2) = 5$

$h^*(s5) = 3$

$h^*(s7) = 0$



Problem 1: Task 1

$h(s) - h(s') \leq c(a)$

parent child

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

$h(s1) - h(s2) = 6 - 1 = 5$

$> c(s_1, s_2) = 2$

$h(s1) = 4$

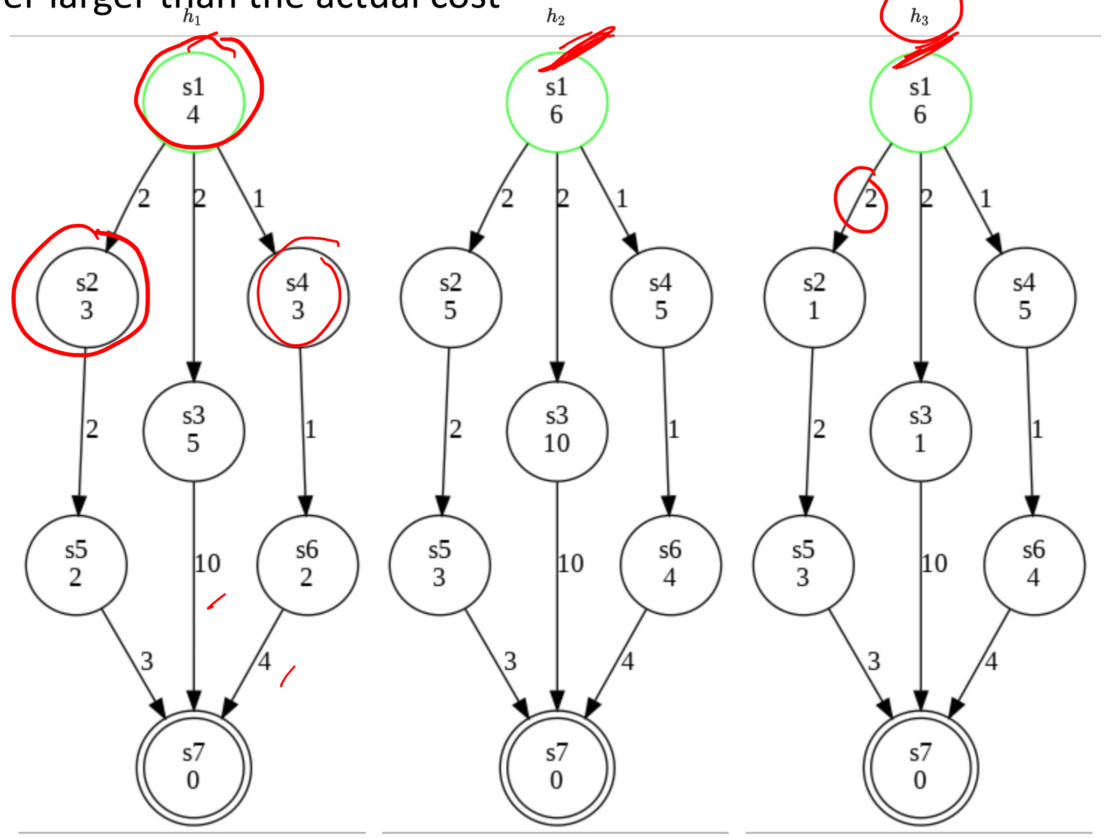
$h(s2) = 3$

$\rightarrow h(s1) - h(s2) = 1$

$< c(a) = 2$

Which heuristics are consistent?

h1, h2



Problem 1: Task 1

Dominate relation

h1 dominates h2 if

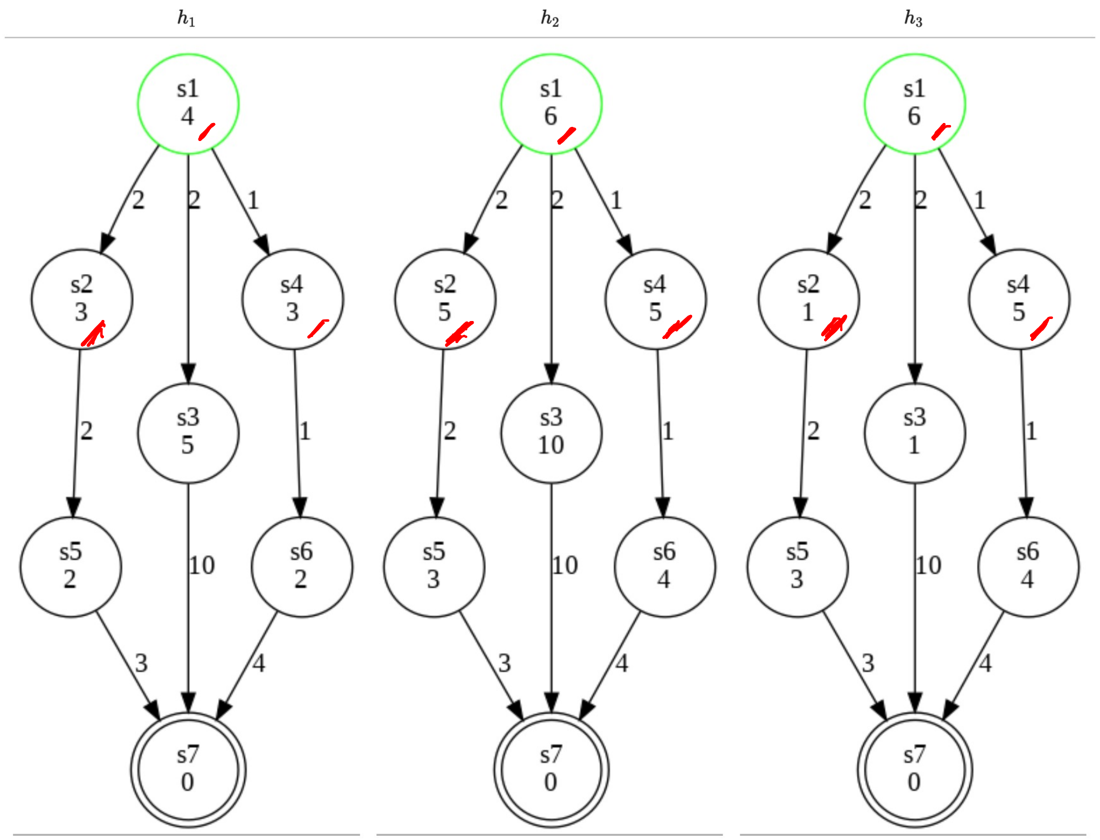
- both heuristics are admissible √
- h2 <= h1 <= h* for all s in S √

Does any of the heuristic dominate any other?

h2 dominates h1

h2 dominates h3

perfect heuristic



Problem 1: Task 2

priority value

Heuristic search algorithms

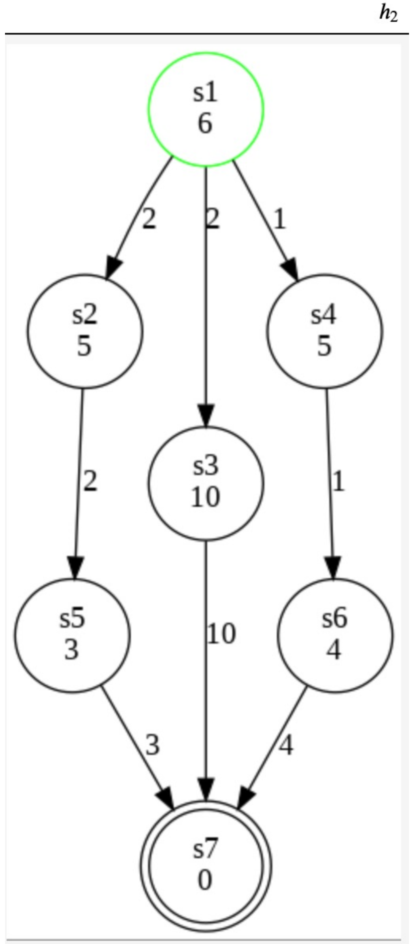
Search node: n = <s, f(n), g(n), parent n>

f(n) is a priority value for node in the priority queue

DS: priority queue

- Uniform-cost search (Dijkstra): f(n) = g(n) (h(n)=0) → 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)

weight

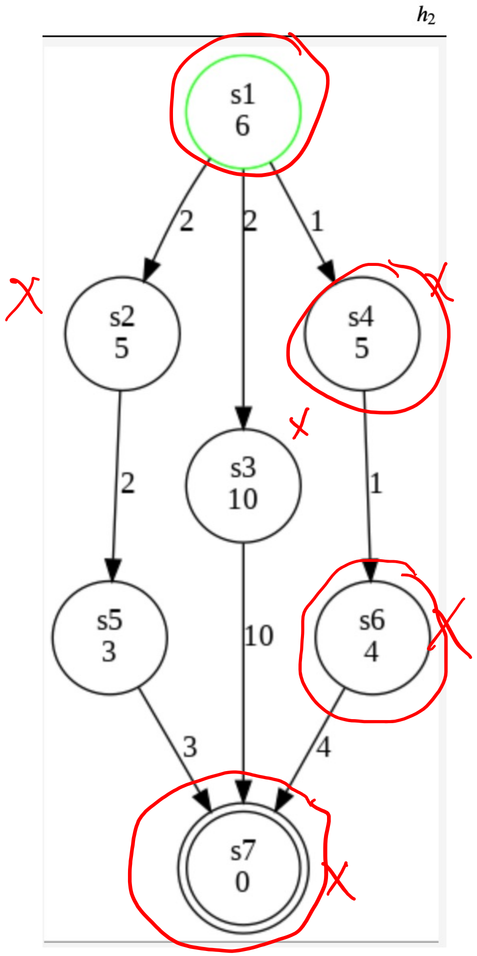


Problem 1: Task 2

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

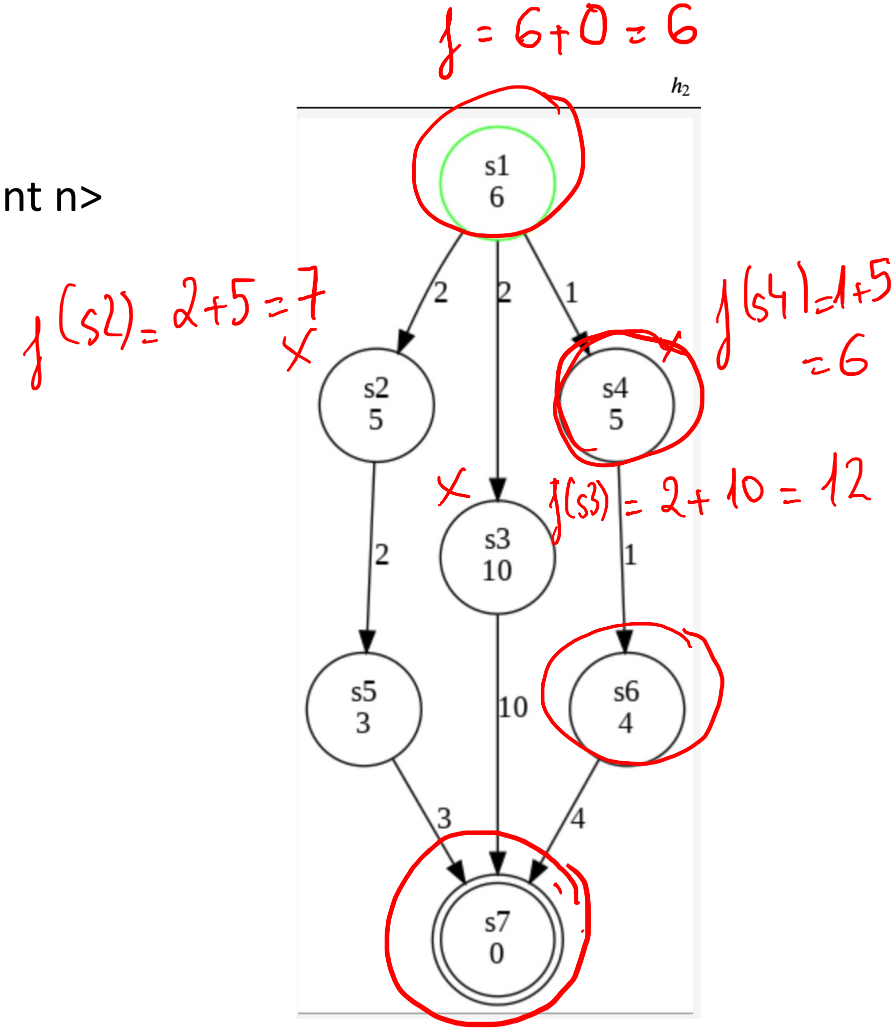
Problem 1: Task 2

A*

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

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

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



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

Problem 1: Task 2

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

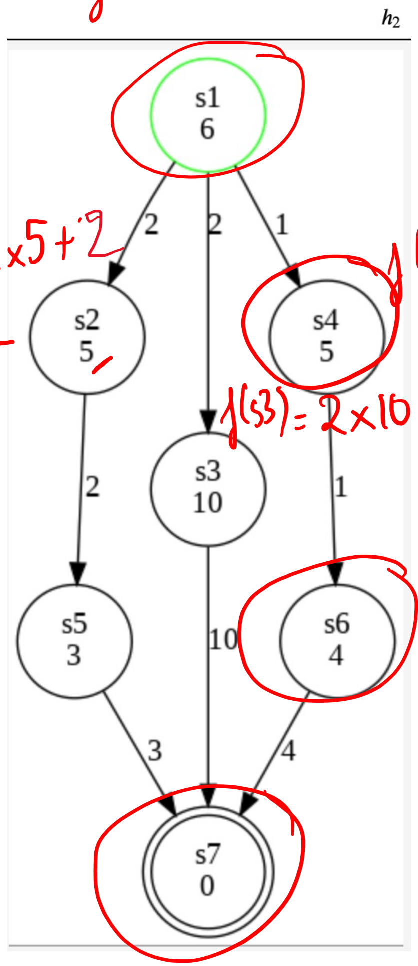
f(n) = W * h(s) + g(n)

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(s1) = 2 × 6 + 0 = 12



f(s2) = 2×5+2 = 12

f(s4) = 2×5+1 = 11

f(s3) = 2×10+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

X

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

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

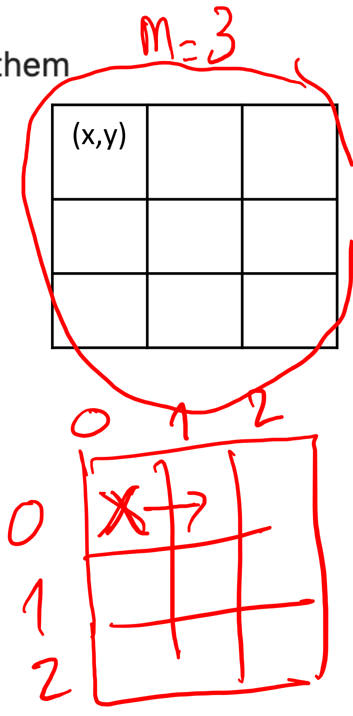
open := **new** priority queue ordered by ascending $g(\textit{state}(\sigma)) + h(\textit{state}(\sigma))$
open.insert(make-root-node(init()))
closed := $\emptyset$
best-g := $\emptyset$ /* maps states to numbers */
while not *open*.empty():
    σ := *open*.pop-min()
    if $\textit{state}(\sigma) \notin \textit{closed}$ **or** $g(\sigma) < \textit{best-g}(\textit{state}(\sigma))$:
        /* 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* $\cup$ $\{\textit{state}(\sigma)\}$
        $\textit{best-g}(\textit{state}(\sigma)) := g(\sigma)$
        if is-goal(state(σ)): **return** extract-solution(σ)
        for each $(a, s') \in$ succ(*state*(σ)):
            σ' := make-node(σ, a, s')
            if $h(\textit{state}(\sigma')) < \infty$: *open*.insert($\sigma'$)
return unsolvable

re-visit a closed state

Problem 2

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

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

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

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

a state = <current coord, a set of visited coordinates> (2nd way)

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

current remaining

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

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

remaining

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

current state

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

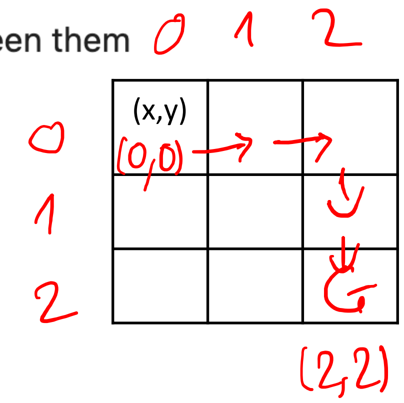
current state action next state

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

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

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

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

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

Current

Vall

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

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

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

next state

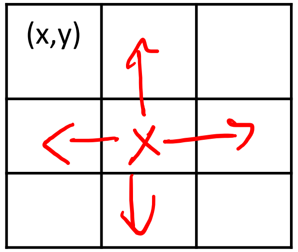
Cost c(a, s) = 1

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? = max no of child nodes



4 (branching factor = max number of child nodes)

Problem 2

(x,y)		
		X

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?

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

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

size of the grid.

If you have a small G, it is much better to apply the 1st way

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

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