SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Planning III

PROF LIM KWAN HUI

50.021 Artificial Intelligence

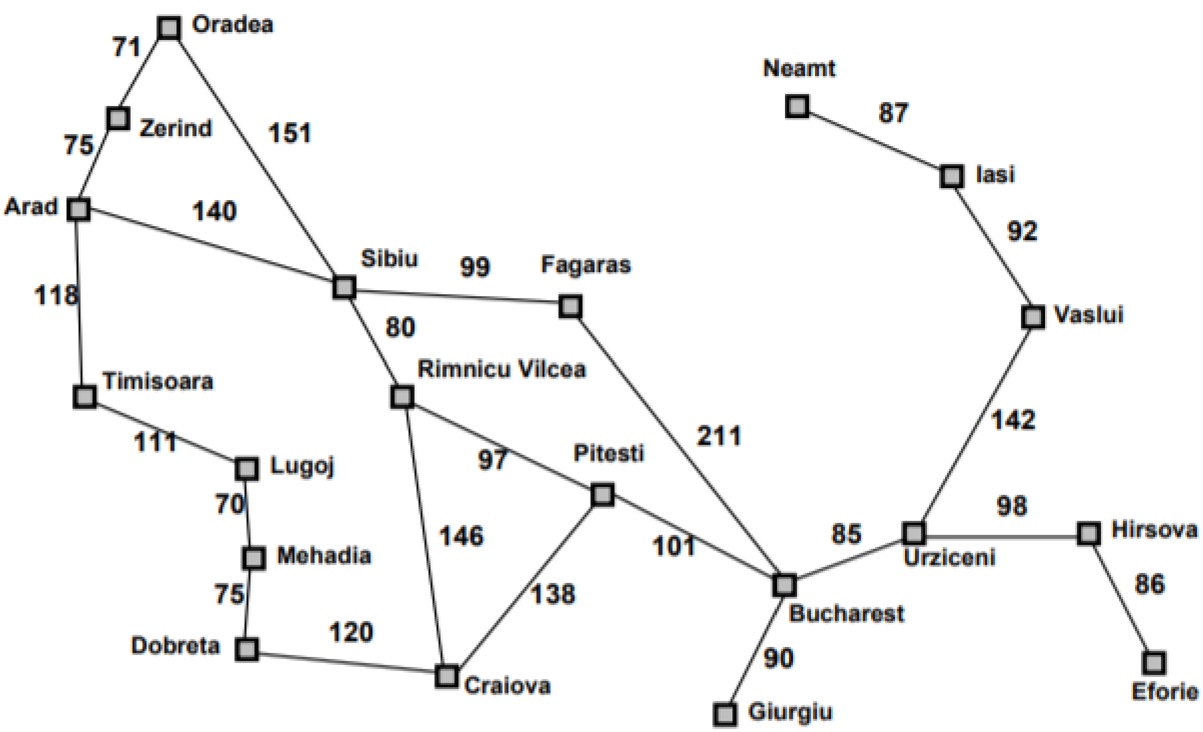
The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.

Recap: Heuristics

- Heuristics
  - $h(n)$ = estimated cost from n to goal
- Admissible Heuristic
  - $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost from n
- Application in informed search
  - E.g., greedy search, A* search

Recap: Heuristics

- E.g., straight line distance from current location to goal destination

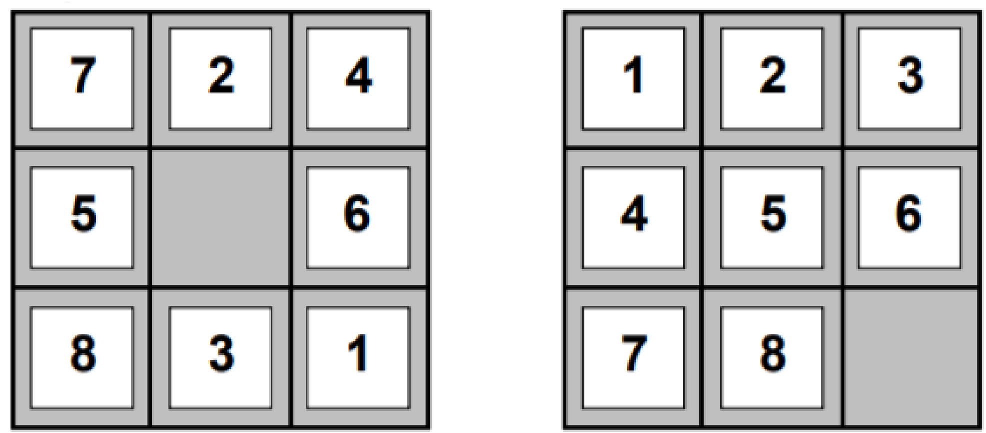


Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Recap: Admissible Heuristics

- E.g., for the 8-puzzle:
- h1(n) = number of misplaced tiles
- h2(n) = total Manhattan distance (i.e., no. of squares from desired location of each tile)



Start State　　Goal State

- h1(S) = 6
- h2(S) = 4+0+3+3+1+0+2+1 = 14

Recap: Problem Relaxation

- **Admissible heuristics** can be derived from the **exact solution cost** of a **relaxed version** of the problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then h1(n) gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then h2(n) gives the shortest solution
- Key point
  - The optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

Planning Problem Relaxation

- How do we relax a planning problem based on STRIPS?

- Recall that STRIPS is formally defined as a 4-tuple ***(P,O, I, G)***
  - ***P*** - a set of propositional variables
  - ***O*** - a set of operators (i.e., actions). Each operator comprises
    - pre_a - facts that must be true before the action can be performed
    - add_a - facts that will change to true when/after the action can be performed
    - del_a - facts that will change to false when/after the action can be performed
  - ***I*** - the initial state of the world, true/false assignments to ***P***
  - ***G*** - the goal state of the world

Delete-relaxed Problem

- How do we relax a planning problem based on STRIPS?
  - ***Delete-relaxed: remove the negation of facts in all operators***

- Recall that STRIPS is formally defined as a 4-tuple ***(P,O, I, G)***
  - ***P*** - a set of propositional variables
  - ***O*** - a set of operators (i.e., actions). Each operator comprises
    - pre_a - facts that must be true before the action can be performed
    - add_a - facts that will change to true when/after the action can be performed
    - ~~***del_a* - facts that will change to false when/after the action can be performed**~~
  - ***I*** - the initial state of the world, true/false assignments to ***P***
  - ***G*** - the goal state of the world

Example: Robot World

- Original problem
- ***P*** - a set of propositional variables. E.g., inA(x), inB(x), dooropen(x,y)
- ***O*** - a set of operators (i.e., actions). E.g., kickball(a,b)
  - pre_a - inA(robot), inA(ball), dooropen(A,B)
  - add_a - inB(ball)
  - del_a - inA(ball)
- ***I*** - the initial state of the world. E.g., inA(robot), inA(ball), dooropen(A,B)
- ***G*** - the goal state of the world. E.g., inB(ball)

Example: Relaxed Robot World

- Delete-relaxed problem
- ***P*** - a set of propositional variables. E.g., inA(x), inB(x), dooropen(x,y)
- ***O*** - a set of operators (i.e., actions). E.g., kickball(a,b)
  - *pre*$_a$ - inA(robot), inA(ball), dooropen(A,B)
  - *add*$_a$ - inB(ball)
  - ~~**del**$_a$ **- inA(ball)**~~
- ***I*** - the initial state of the world. E.g., inA(robot), inA(ball), dooropen(A,B)
- ***G*** - the goal state of the world. E.g., inB(ball)

Delete-relaxed Problem

- Why is this an easier problem?
  - Recall that a solution/plan comprises a sequence of actions

- Every plan that solves the original problem (with deletes), also solves the delete-relaxed problem
  - Since the goal is for certain facts to be set to true, it does not matter if other additional facts are true.

Exercise: Is it admissible?

- Given this problem definition:
  - Initial State: ¬x ¬y
  - Goal: x y
  - Actions: a1: precondition: nil, postcondition : x
    a2: precondition: x, postcondition: ¬x y
- What are the actions needed to reach the goal?
- How can you relax this problem?

Exercise: Is it admissible?

- Relaxed problem definition:
  - Initial State: ¬x ¬y
  - Goal: x y
  - Actions: a1: precondition: nil, postcondition : x
    a2: precondition: x, postcondition: ~~¬x~~ y

- What are the actions needed to reach the goal? **a1, a2, a1 (optimal)**
- How can you relax this problem? **Remove the ¬x from a2**

Exercise: Is it admissible?

- Relaxed problem definition:
  - Initial State: ¬x ¬y
  - Goal: x y
  - Actions: a1: precondition: nil, postcondition : x

    a2: precondition: x, postcondition: ~~¬x~~ y

- What are the actions needed to reach the goal? **a1, a2, a1 (optimal)**
- How can you relax this problem? **Remove the ¬x from a2**
- What are the actions now for the relaxed problem?
- Is the new set of actions admissible?

Exercise: Is it admissible?

- Relaxed problem definition:
  - Initial State: ¬x ¬y
  - Goal: x y
  - Actions: a1: precondition: nil, postcondition : x

    a2: precondition: x, postcondition: ~~¬x~~ y

- What are the actions needed to reach the goal? **a1, a2, a1 (optimal)**
- How can you relax this problem? **Remove the ¬x from a2**
- What are the actions now for the relaxed problem? **a1, a2**
- Is the new set of actions admissible? **Yes. Less steps than the optimal**

h_+ Heuristic

- **Definition:** The optimal plan (or minimal number of actions) for a delete-relaxed problem (no deletes) is called a h+ heuristic.

- We can use the optimal plan for a delete-relaxed problem as heuristic for a state in the original problem
  - The minimal number of steps to solve a delete-relaxed problem can not be larger than the minimal number of steps to solve the original problem
  - So it never overestimates the cost in the original problem, h(n) <= h*(n)

- The h+ heuristic is admissible by design for every state of the original problem.
  - E.g., based on the exercise earlier:
    - Original problem plan: 3 steps
    - Delete-relaxed plan: 2 steps (2 <= 3, so admissible)

h_+ Heuristic: Issues

- Recall the use of a heuristic
  - You calculate the heuristic each time a new state is generated/searched
  - Easy for path planning and 8-puzzle (use direct and Manhattan distances)

- For planning problems, need to compute h+ heuristic at each newly generated state
  - h+ heuristic means computing the optimal plan for a delete-relaxed problem
  - So this means we are solving for multiple delete-relaxed problems, each time based on a generated state as start state (**expensive!**)

- Although delete-relaxed is an easier problem, expensive to compute h+ multiple times. So we **<u>need to approximate h+</u>**.

Faster Planner Search Heuristics (h_{add},h_{max}, h_{FF})

- Assumption: Delete-relaxed problem (no delete actions)

The general idea is as follows:

- Every action can be applied only if all facts in its precondition are true
- Given a set of true facts, we can check what action are applicable.
- If we perform these valid actions, we create a new set of facts (due to postconditions)
  - Since there are no deletes in this delete-relaxed problem, the number of true facts will only accumulate

Faster Heuristics: Overall Idea

1. F_0 is the initial set of true facts
2. A_0 is the set of actions that can be applied on F_0 (recall that there are no deletes). This will result in F_0 plus some added facts. Defined as $F_1 = F_0$+ plus the added facts that come from applying every action a A_0.

f1 bigger

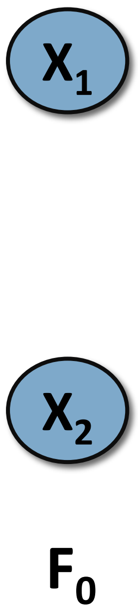
3. Now F_1 is again a set of facts. Apply all applicable actions that could not be applied before. Let's call this set of newly applicable actions A_1.
4. Iterate this: one has a set F_i, apply all applicable actions A_i to yield F_{i+1}.
5. Terminate at iteration number M when the set F_M of facts contains all goal facts.

Faster Heuristics: Overall Idea

- Given this (delete-relaxed) problem definition:
  - Variables: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$
  - Initial State: x_1, x_2
  - Goal: x_3, x_4, x_5, x_8
  - Actions:
    - o_1 : precond: x_1, postcond: x_3, x_4
    - o_2 : precond: x_2, postcond: x_5
    - o_3 : precond: x_3, postcond: x_6
    - o_4 : precond: x_5, postcond: x_7
    - o_5 : precond: x_6, postcond: x_8

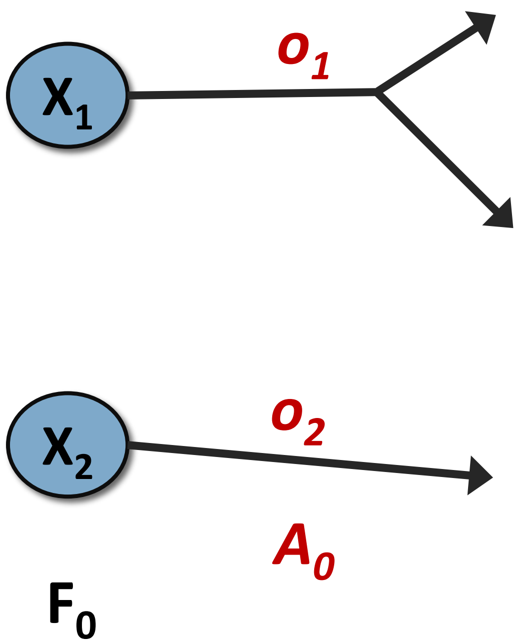
Faster Heuristics: Overall Idea

1. F_0 is the initial set of true facts



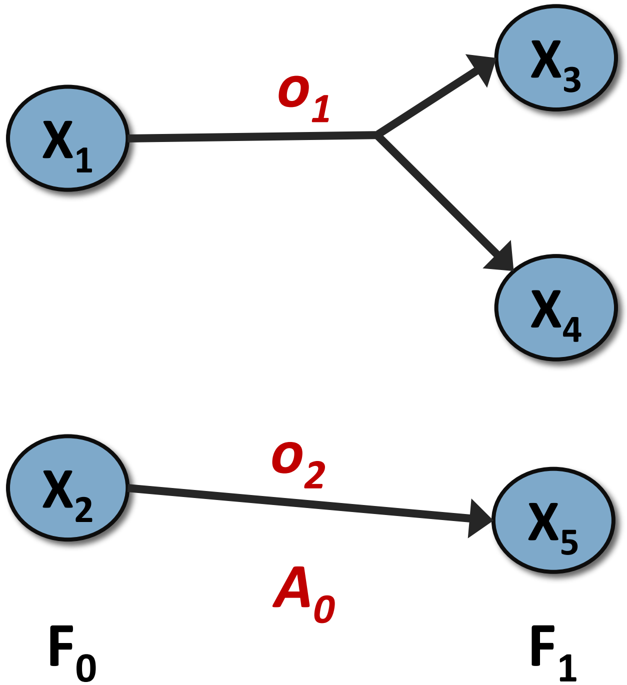
Faster Heuristics: Overall Idea

2. A_0 is the set of actions that can be applied on F_0 (recall that there are no deletes).



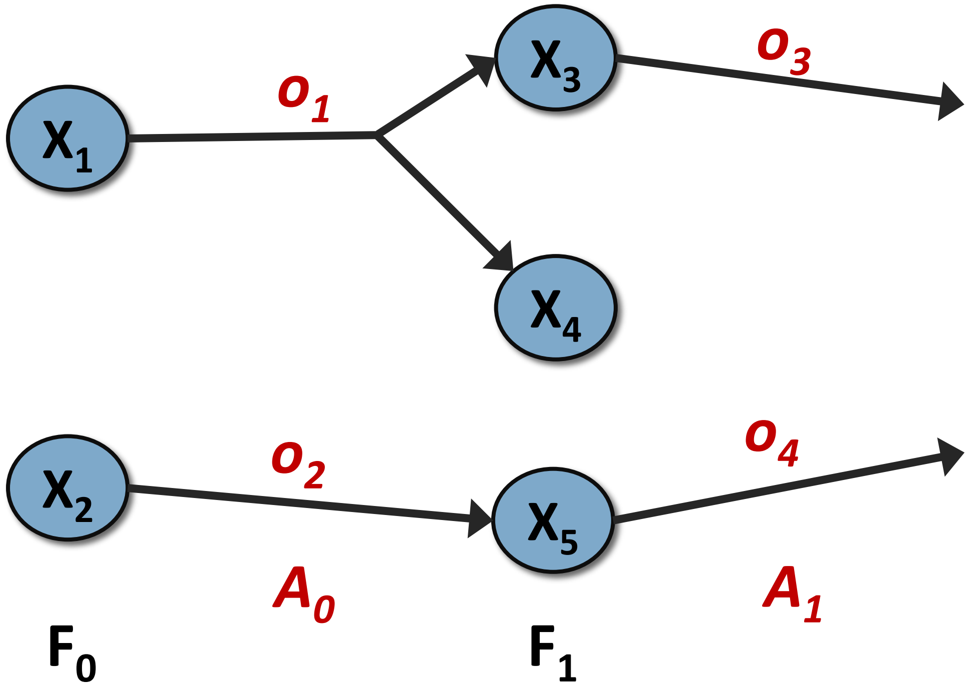
Faster Heuristics: Overall Idea

2. A_0 is the set of actions that can be applied on F_0 (recall that there are no deletes). This will result in F_0 plus some added facts. Defined as $F_1 = F_0$+ plus the added facts that come from applying every action a A_0.



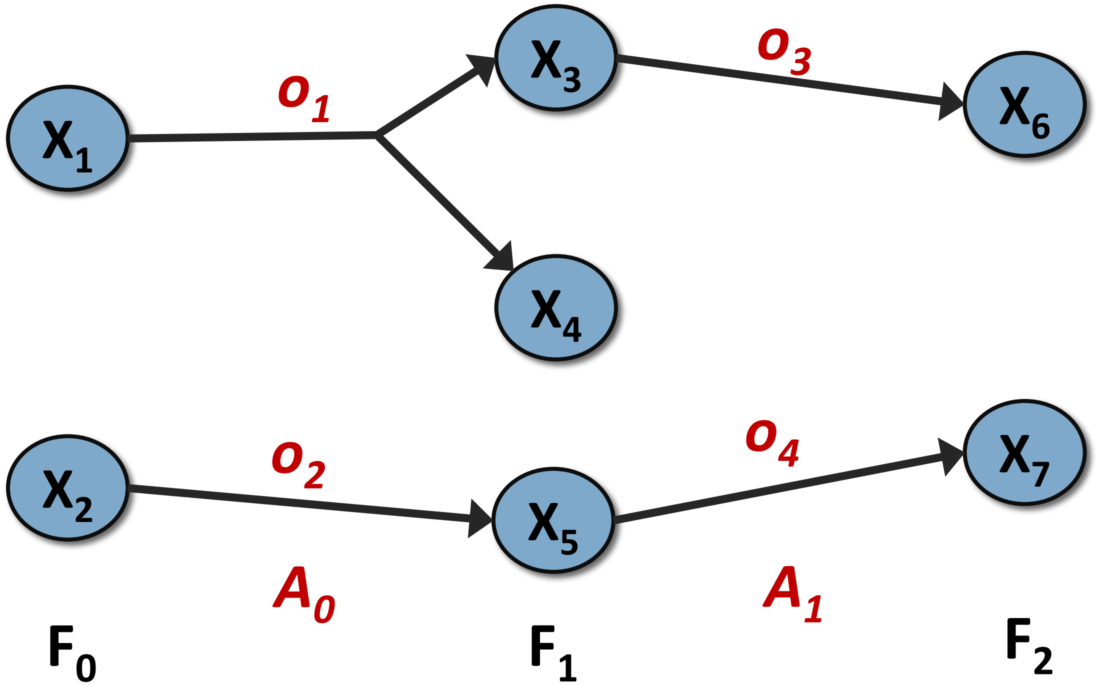
Faster Heuristics: Overall Idea

3. Now F_1 is again a set of facts. Apply all applicable actions that could not be applied before. Let's call this set of newly applicable actions A_1.



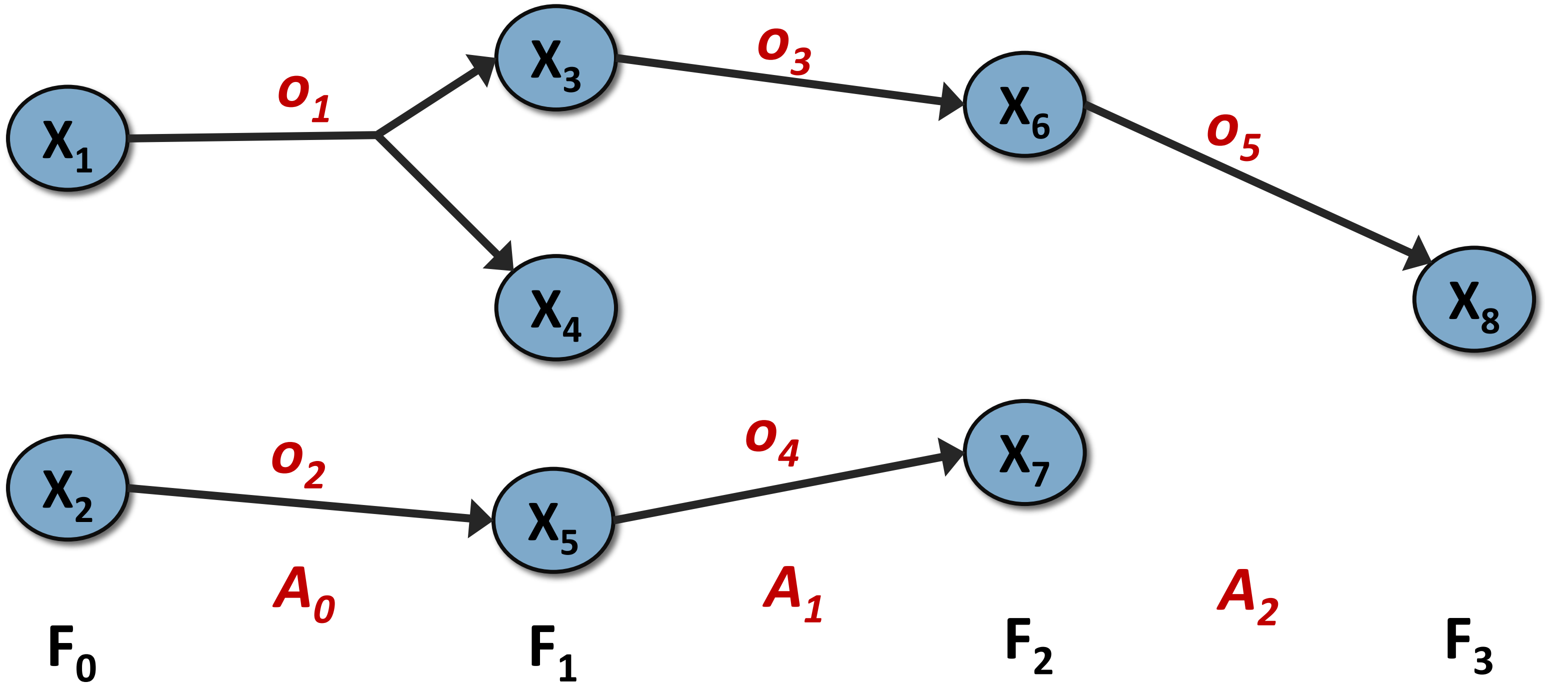
Faster Heuristics: Overall Idea

4. Iterate this: once we have a set F_i, apply all applicable actions A_i to yield F_{i+1}.



Faster Heuristics: Overall Idea

5. Terminate at iteration number M when the set F_M of facts contains all goal facts.



Faster Heuristics: Overall Idea

- Concise representation using facts (F) and actions (A)
  - F_0 $= x_1, x_2$
  - A_0 $= o_1, o_2$
  - F_1 $= x_1, x_2, x_3, x_4, x_5$
  - A_1 $= o_3, o_4$
  - F_2 $= x_1, x_2, x_3, x_4, x_5, x_6, x_7$
  - A_2 $= o_5$
  - F_3 $= x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

Faster Heuristics: Overall Idea

- Concise representation using facts (F) and actions (A)
  - $F_0 \quad = x_1, x_2$
  - $A_0 \quad = o_1, o_2$
  - $F_1 \quad = x_1, x_2, x_3, x_4, x_5$
  - $A_1 \quad = o_3, o_4$
  - $F_2 \quad = x_1, x_2, x_3, x_4, x_5, x_6, x_7$
  - $A_2 \quad = o_5$
  - $F_3 \quad = x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

- What does this tell us in terms of achieving goals (facts)?
  - It tells us the level/index where a fact was achieving (set to true) ***for the first time***. E.g., x_3 at level 1, x_7 at level 2, x_8 at level 3

Faster Heuristics: Overall Idea

- Concise representation using facts (F) and actions (A)
  - F_0 = x_1, x_2
  - A_0 = o_1, o_2
  - F_1 = x_1, x_2, x_3, x_4, x_5
  - A_1 = o_3, o_4
  - F_2 = $x_1, x_2, x_3, x_4, x_5, x_6, x_7$
  - A_2 = o_5
  - F_3 = $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

- Why do we care about the level/index of a fact?
  - It tells us the ***number of actions (i.e., cost) required*** to achieve that fact
  - This is also useful as we examine the h_{max}, h_{add} heuristics

Next

- Learn more about the h_{max} and h_{add} heuristics
- Understand how to compute h_{max} and h_{add} based on what we did