Assignment13 – Planning

Problem 13.1 (Admissible Heuristics in Gripper)
Consider a problem where we have two rooms, A and B, one robot initially located in room A, and n balls that are also initially located in room A. The *goal* demands that all balls be located in room B. The robot can move between the rooms, it can pick up balls provided its gripper hand is free (see below), and it can drop a ball it is currently holding.

Answer the following questions with yes/no. Justify your answer.

1. Say that the robot has only one gripper, so that it can only hold one ball at a time. Is the number of balls not yet in room B an *admissible heuristic*?

   Solution: Yes: The solution must contain at least one separate drop action for each ball that is not yet currently in room B.

2. Say that the robot has only one gripper, so that it can only hold one ball at a time. Is the number of balls still in room A, multiplied by 4, an *admissible heuristic*?

   Solution: No: For example, in the initial state for $n = 1$, the length of an optimal solution is 3 (pick, move A B, drop), whereas the *value* of this *heuristic* is 4.

3. Say now the robot has two grippers, and it takes only one action to pick up two balls, and only one action to drop two balls. Is the number of balls not yet in room B an *admissible heuristic*?

   Solution: No: For example, if all but 2 balls are already in room B, and the robot is in room B and holds the 2 remaining balls, then the length of an optimal solution is 1, whereas the value of this *heuristic* is 2.

4. Say now the robot has two grippers, but picks up/drops each ball individually, so that it needs two actions to take two balls, and two actions to drop two balls. Is the number of balls not yet in room B an *admissible heuristic*?

   Solution: Yes, for the same reason as in (a).

Problem 13.2 (Partial Order Planning)

Consider the planning task (P, A, I, G) where

- facts $P = \{p, q, r, s\}$
- actions $A = \{X, Y, Z\}$ where the preconditions (above the box) and effects (below the box) of the actions are given by

$$\begin{array}{ccc} p & q & p \\ \boxed{\text{X}} & \boxed{\text{Y}} & \boxed{\text{Z}} \\ q & \neg p, r & \neg p, s \end{array}$$

- initial state $I = \{p\}$
- goal $G = \{r, s\}$

Our goal is to build a partially ordered plan. Recall that the steps consist of the actions plus the start and finish step; and that the effect of an action consists of the added facts and the negations of the deleted facts.

1. Give the start step and finish step.

Solution: $\begin{array}{c} \\ \boxed{\text{Start}} \\ p \end{array} \quad \begin{array}{c} r, s \\ \boxed{\text{Finish}} \end{array}$

2. Give all causal links between the steps.

Solution: $Start \xrightarrow{p} X, Start \xrightarrow{p} Z, X \xrightarrow{q} Y, Y \xrightarrow{r} Finish, Z \xrightarrow{s} Finish$

3. Give an example where a step clobbers a link.

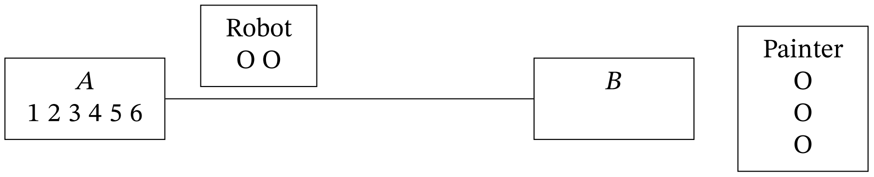
Solution: Both steps Y and Z clobber both of the links $\xrightarrow{p}$.

4. Give the temporal ordering that yields a partially ordered plan that solves the task.

Solution: $X \prec Z$ and $Z \prec Y$

Problem 13.3 (STRIPS)

Consider a set of objects $Obj = \{1,2,3,4,5,6\}$ that can be at location A or B. Currently all objects are at location A and **unpainted**. Eventually all objects are needed in location A and **painted**. At location B, a painting station is available that can paint up to 3 objects at a time. A robot is available (currently at location A) that can move up to 2 objects at a time from one location to another.



We formalize this problem as a STRIPS task (P, A, I, G) where the set P of facts contains

- at(l, o) for $l \in \{A, B\}$ and $o \in (Obj \cup \{Robot\})$
- painted(o) for $o \in Obj$

and the set A of actions contains

- move(l, m, O) for $l, m \in \{A, B\}$, $O \subseteq Obj$, $\#(O) \leq 2$ given by
  - precondition: at(l, o) for all $o \in (O \cup \{Robot\})$
  - add list: at(m, o) for all $o \in (O \cup \{Robot\})$
  - delete list: same as precondition
- paint(O) for $O \subseteq Obj$, $\#(O) \leq 3$ given by
  - precondition: at(B, o) for all $o \in O$
  - add list: painted(o) for all $o \in O$
  - delete list: nothing

1. Give the initial state I and the goal G.

   Solution: at(A, o) for all $o \in (Obj \cup \{Robot\})$, goal: at$(A, o)$, painted$(o)$ for all $o \in Obj$

2. After applying move$(A, B, \{1,2\})$ in I, multiple actions are applicable. Give two of them.

   Solution: The applicable actions are move(B, A, O) and paint(O) for any $O \subseteq \{1,2\}$. Note: Among those, move$(B, A, \emptyset)$ and paint$(\{1,2\})$ are the not-obviously-suboptimal ones and pondering those helps with the next subquestion.

3. Give the value $h^*(I)$.

Solution: 9. An optimal plan is move(A, B, O), paint(O), move(B, A, O), repeated 3 times for *disjoint* sets O. move$(A, B, \{1, 2\})$, move$(B, A, \emptyset)$, move$(A, B, \{3, 4\})$, paint$(\{1, 2, 3\})$, move$(B, A, \{1, 2\})$, move$(A, B, \{5, 6\})$, paint$(\{4, 5, 6\})$, move$(B, A, \{3, 4\})$, move$(A, B, \emptyset)$, move$(B, A, \{5, 6\})$ takes 10 steps and is not optimal, but induces an optimal relaxed plan, in which some moves actions can be dropped.

---

4. Give the value $h^+(I)$.

Solution: 5. An optimal relaxed plan moves 2 objects 3 times, paints twice.

---

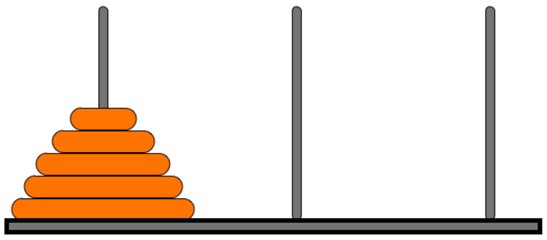
5. Let $U_s(l)$ and $P_s(l)$ be the numbers of unpainted and painted objects at location l in state s. For each of the following *heuristic* $h(s)$, say if it is *admissible*.
   1. $2 \cdot U_s(A) + U_s(B)$
   2. 0
   3. $U_s(A)$+roundDown$((U_s(A)+U_s(B))/3)$+roundDown$((P_s(B)+U_s(B))/2)$

Solution: 2 and 3.

1. too pessimistic, e.g., two unpainted objects in location A need 3 steps, *heuristic* yields 4
2. trivially *admissible* but useless
3. a good *heuristic*: unpainted objects in location A (resp. any objects in location B) need to be moved at least twice (resp. once) in groups of at most 2; unpainted objects must be painted in groups of at most 3.

Problem 13.4

Encode the Tower of Hanoi task below into PDDL. There are five discs and the goal is to move them from left to right according to the rules of Tower of Hanoi. Solve the PDDL encoding using FF. As your solution, submit a print-out of the following 3 files: The PDDL domain file "towersofhanoi-domain.pddl"; the PDDL problem file "towersofhanoi-problem.pddl", as well as a file "towersofhanoi-output.txt" containing FF's output.



Solution: PDDL Domain file:

```
(define (domain hanoi)
  (:requirements :strips)
  (:predicates (clear ?x)
               (on ?x ?y)
               (smaller ?x ?y)
  )

  (:action move
    :parameters (?disc ?from ?to)
    :precondition (and (smaller ?disc ?to)
                  (on ?disc ?from)
                  (clear ?disc)
                  (clear ?to)
    )
    :effect (and (clear ?from)
             (on ?disc ?to)
             (not (on ?disc ?from))
             (not (clear ?to))
    )
  )
)
```

PDDL Problem file:

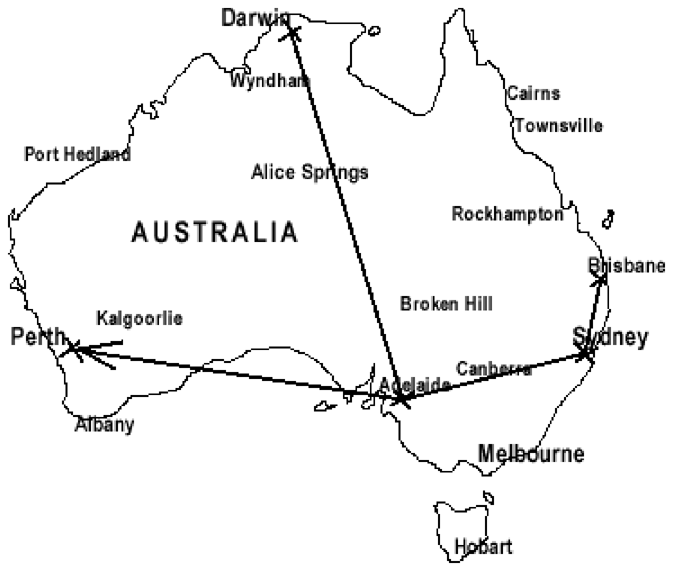
```
(define (problem hanoi)
  (:domain hanoi)
  (:objects p1 p2 p3 d1 d2 d3 d4 d5)
  (:init
    (smaller d1 p1)(smaller d1 p2)(smaller d1 p3)
    (smaller d2 p1)(smaller d2 p2)(smaller d2 p3)
    (smaller d3 p1)(smaller d3 p2)(smaller d3 p3)
    (smaller d4 p1)(smaller d4 p2)(smaller d4 p3)
    (smaller d5 p1)(smaller d5 p2)(smaller d5 p3)

    (smaller d1 d2)(smaller d1 d3)(smaller d1 d4)(smaller d1 d5)
    (smaller d2 d3)(smaller d2 d4)(smaller d2 d5)
    (smaller d3 d4)(smaller d3 d5)
    (smaller d4 d5)
    (clear p2)(clear p3)(clear d1)
    (on d1 d2)(on d2 d3)(on d3 d4)
    (on d4 d5)(on d5 p1)
  )
  (:goal
    (and (on d1 d2)(on d2 d3)(on d3 d4)(on d4 d5)(on d5 p3) )
  )
)
```

The output of FF: (some white spaces removed)

```
parsing domain file
domain 'HANOI' defined
 ... done.
ff: parsing problem file
problem 'HANOI' defined
 ... done.



Cueing down from goal distance: 5 into depth [1]
                               4 [1][2][3][4]
                               3 [1]

Enforced Hill-climbing failed !
switching to Best-first Search now.

advancing to distance : 5
                        4
                        3
                        2
                        1
                        0

ff: found legal plan as follows

step 0: MOVE D1 D2 P3
       1: MOVE D2 D3 P2
       2: MOVE D1 P3 D2
       3: MOVE D3 D4 P3
       4: MOVE D1 D2 D4
       5: MOVE D2 P2 D3
       6: MOVE D1 D4 D2
       7: MOVE D4 D5 P2
       8: MOVE D1 D2 D4
       9: MOVE D2 D3 D5
      10: MOVE D1 D4 D2
      11: MOVE D3 P3 D4
      12: MOVE D1 D2 P3
      13: MOVE D2 D5 D3
      14: MOVE D1 P3 D2
      15: MOVE D5 P1 P3
      16: MOVE D1 D2 P1
      17: MOVE D2 D3 D5
      18: MOVE D1 P1 D2
      19: MOVE D3 D4 P1
      20: MOVE D1 D2 D4
      21: MOVE D2 D5 D3
      22: MOVE D1 D4 D2
      23: MOVE D4 P2 D5
      24: MOVE D1 D2 D4
      25: MOVE D2 D3 P2
      26: MOVE D1 D4 D2
      27: MOVE D3 P1 D4
      28: MOVE D1 D2 P1
      29: MOVE D2 P2 D3
      30: MOVE D1 P1 D2


time spent: 0.00 seconds instantiating 200 easy, 0 hard action templates
             0.00 seconds reachability analysis, yielding 33 facts and 135 actions
             0.00 seconds creating final representation with 32 relevant facts
             0.00 seconds building connectivity graph
             0.00 seconds searching, evaluating 101 states, to a max depth of 4
             0.00 seconds total time
```

Problem 13.5 (STRIPS Planning)

Consider the road map of Australia given below. The task here is to visit Darwin, Brisbane and Perth, starting from Sydney.



The task is formalized in STRIPS as follows. Facts are $at(x)$ and $visited(x)$ where $x \in \{Adelaide, Brisbane, Darwin, Perth, Sydney\}$. The initial state is $\{at(Sydney), visited(Sydney)\}$, the goal is $\{visited(Brisbane), visited(Darwin), visited(Perth)\}$. The actions move along the roads, i.e., they are of the form

$$drive(x,y) : (\{at(x)\}, \{at(y), visited(y)\}, \{at(x)\})$$

where x and y have a direct connection according to the road map. **Each road can be driven in both directions, except for the road between Adelaide and**

Perth, which can only be driven from Adelaide to Perth, not in the opposite direction. In your answers to the following questions, use the abbreviations "v" for "visited", and "Ad", "Br", "Da", "Pe", "Sy" for the cities.

1. Give an optimal (shortest) plan for the initial state, if one exists; if no plan exists, argue why that is the case. Give an optimal (shortest) relaxed plan for the initial state, if one exists; if no relaxed plan exists, argue why that is the case. What is the h^* value and the h^+ value of the initial state? (When writing up a plan or relaxed plan, it suffices to give the sequence of action names.)

   Solution: Optimal plan: $drive(Sy, Br)$, $drive(Br, Sy)$, $drive(Sy, Ad)$, $drive(Ad, Da)$, $drive(Da, Ad)$, $drive(Ad, Pe)$. Optimal relaxed plan: $drive(Sy, Br)$, $drive(Sy, Ad)$, $drive(Ad, Da)$, $drive(Ad, Pe)$. $h^* = 6, h^+ = 4$.

   ---

2. Do the same in the modified task where the road between Sydney and Brisbane is also one-way, i.e., it can only be driven from Sydney to Brisbane, not in the opposite direction.

   Solution: Optimal plan: Does not exist because we must visit both Brisbane and Perth, but once we moved to either of these two, we cannot get back out again. Optimal relaxed plan: $drive(Sy, Br)$, $drive(Sy, Ad)$, $drive(Ad, Da)$, $drive(Ad, Pe)$. $h^* = \infty, h^+ = 4$.

   ---

3. Write up, in STRIPS notation, all states reachable from the initial state in at most *two* steps. Start at the initial state, and insert successors. Indicate successor states by edges. Annotate the states with their h^* values as well as their h^+ values.
4. Do the same in the modified task where the road between Sydney and Brisbane is one-way, i.e., it can only be driven from Sydney to Brisbane, not in the opposite direction.

Problem 13.6 (Relaxation)

We want to solve a *STRIPS planning task*.

1. Explain (in about 2 sentences) the purpose of *relaxed planning*.

   Solution: A *relaxed problem* is easier to solve than the original problem. Then the length of its *optimal plan* of the *relaxed problem* can be used as a *heuristic* for the original *problem*.

   ---

2. Explain (in about 2 sentences) why it is bad to *relax* too much or too little.

Solution: If we *relax* too little, the *relaxed problem* is still too difficult to solve. If we relax too much, the *relaxed problem* is so easy that it does not induce a useful *heuristic*.

Now consider a concrete task given by a *finite set B* of *size n* and

- *facts*: $inA(b)$, $inB(b)$, $held(b)$ for $b \in B$, $Rfree$, $RinA$, $RinB$
- *actionss*

action	*precondition*	*add list*	*delete list*
$move_A$	$RinB$	$RinA$	$RinB$
$move_B$	$RinA$	$RinB$	$RinA$
$pickup(b)$	$RinA, Rfree, inA(b)$	$held(b)$	$Rfree, inA(b)$
$release(b)$	$RinB, held(b)$	$inB(b), Rfree$	$held(b)$

- *initial state I*: $inA(b)$ for $b \in B$, $Rfree$, $RinA$
- *goal state*: $inB(b)$ for $b \in B$

3. Let h^+ be the *heuristic* obtained from the *delete relaxation*. Give the *value* of $h^+(I)$.

Solution: $2n + 1$ ($move_B, pickup(1), release(1), \dots, pickup(n), release(n)$)

---

4. Let h^+ be the *heuristic* obtained from the *only-adds relaxation*. Give the *value* of $h^+(I)$.

Solution: n ($release(1), \dots, release(n)$)
