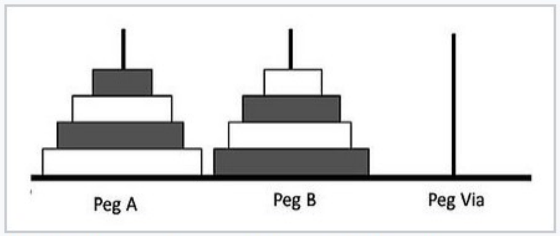
Dhaka University of Engineering & Technology, Gazipur

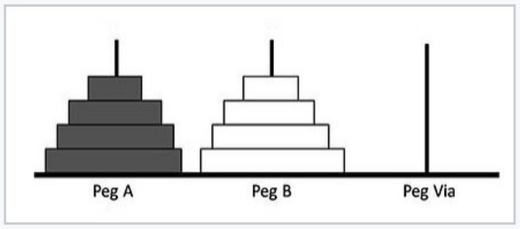
Department of Computer Science and Engineering

Course Code: CSE-4623 Course Title: Artificial Intelligence

Task-2 Documentation



Initial configuration of bicolor Towers of Hanoi (n=4)



Final configuration of bicolor Towers of Hanoi (n=4)

Submitted To

Dr. Fazlul Hasan Siddiqui
Professor, Department of CSE
Dhaka University of Engineering & Technology, Gazipur

Submitted By

Md. Masud Rana
ID: 154103
4th Year 2nd semester
CSE,DUET

Bi-color Towers of Hanoi (N=4)

The rules of the puzzle are essentially the same of monochrome Towers of Hanoi:
1) disks are transferred between pegs one at a time.
2) At no time may a bigger disk be placed on top of a smaller one.
3) The difference is that now for every size there are two disks: one black and one white.
4) same size different color disks placed on each other.
5) Also, there are now two towers of disks of alternating colors.

The goal of the puzzle is to make the towers monochrome. The biggest disks at the bottom of the towers are assumed to swap positions.

Here I use N=4 that mean 4 white disk and 4 black disk.
Used atom F
(on X Y) that describe X is placed top of Y
(clear X) that describe no disk on the X
(smaller X Y) that describe Y is smaller then X

Used action O
(MOVE ?disk ?from ?to)
this action moves disk from location to to location. Before move it check parameter and prediction

```
 :precondition (and (smaller ?to ?disc) (on ?disc ?from)
                    (clear ?disc) (clear ?to))
```

this action check disk is clear or not, to location is clear or not and disk is smaller or not to ‘to’ disk and finally make effect

```
:effect  (and (clear ?from)
              (on ?disc ?to)
              (not (on ?disc ?from))
              (not (clear ?to)))
```

Initial state:
Describe the initial picture of Problem. I named 3 base as peg1 peg2 peg3, and back disk as b and white disk as w. Smallest disk is W1 and b1, and largest Disk W4 and B4

```
      ----      b1        ----      w1              |
    -------     w2      -------     b2              |
  ----------    b3    ----------    w3              |
 ------------- w4   ------------- b4                |
----------------------------------------------------------
```

Goal State:

```
      ----      b1        ----      w1              |
    -------     b2      -------     w2              |
  ----------    b3    ----------    w3              |
 ------------- b4   ------------- w4                |
----------------------------------------------------------
```

problem.pddl, domain.pddl and plan.pddl is attested with zip folder

Planner Use: Fast downward

Here I use fast downward offline planner (http://www.fast-downward.org)
use Fast downward planner A* search with landmark-cut heuristic. I found optimal plan with cast 67.
A* search give optimal solution.

Planner commend

```
./fast-downward.py domain.pddl task.pddl --search "astar(lmcut())"
```

output:

```
[t=1.88461s, 10948 KB] Plan length: 67 step(s).
[t=1.88461s, 10948 KB] Plan cost: 67
[t=1.88461s, 10948 KB] Expanded 18624 state(s).
[t=1.88461s, 10948 KB] Reopened 365 state(s).
[t=1.88461s, 10948 KB] Evaluated 18657 state(s).
[t=1.88461s, 10948 KB] Evaluations: 18657
[t=1.88461s, 10948 KB] Generated 66938 state(s).
[t=1.88461s, 10948 KB] Dead ends: 0 state(s).
[t=1.88461s, 10948 KB] Expanded until last jump: 18621 state(s).
[t=1.88461s, 10948 KB] Reopened until last jump: 365 state(s).
[t=1.88461s, 10948 KB] Evaluated until last jump: 18654 state(s).
[t=1.88461s, 10948 KB] Generated until last jump: 66932 state(s).
[t=1.88461s, 10948 KB] Number of registered states: 18657
[t=1.88461s, 10948 KB] Int hash set load factor: 18657/32768 =
0.569366
[t=1.88461s, 10948 KB] Int hash set resizes: 15
[t=1.88461s, 10948 KB] Search time: 1.88039s
[t=1.88461s, 10948 KB] Total time: 1.88461s
Solution found.
Peak memory: 10948 KB
```