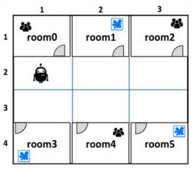
Our robot MARRtino works in a small hotel, whose map is represented as a grid world (see figure). MARRtino has the task of cleaning the rooms that are marked dirty in the map. MARRtino starts in the cell in front of room0 and can navigate the environment by moving inside the map in any of the 6 adjacent cells, that are traversable; it can also enter the hotel rooms by activating a specialized behavior, when is in the cell in front of the door. Once in a room MARRtino can clean it and then exit. MARRtino should not enter the rooms where it knows there are guests.

(a) Describe the domain in PDDL;

(b) Describe the problem in PDDL;

(c) Discuss the forward planning process to reach the goal, using a *perfect* heuristic that gives for each state the number of steps to reach the goal; for each step, show the current state, the applicable actions and the state resulting from the application of the chosen action.



Domain File

```
(:require :adl)

(:types room

(:predicates (at ?R - room)
             (guest ?R - room)
             (dirt ?R - room)
             (trav ?R - room)
             (front ?R1 ?R - room)
)

(:action clean
    (:parameters ?R1 ?R2 - room)
    (:precondition (and (front ?R1 ?R2) (at ?R1) (not guest ?R2) (dirty ?R2)))
    (:effects (not dirty ?R2))
)

(:action move
    (:parameters ?R1 ?R2 - room)
    (:preconditions (and (at ?R1) (trav ?R2)))
    (:effects (and (not at ?R1) (at ?R2))
)
```

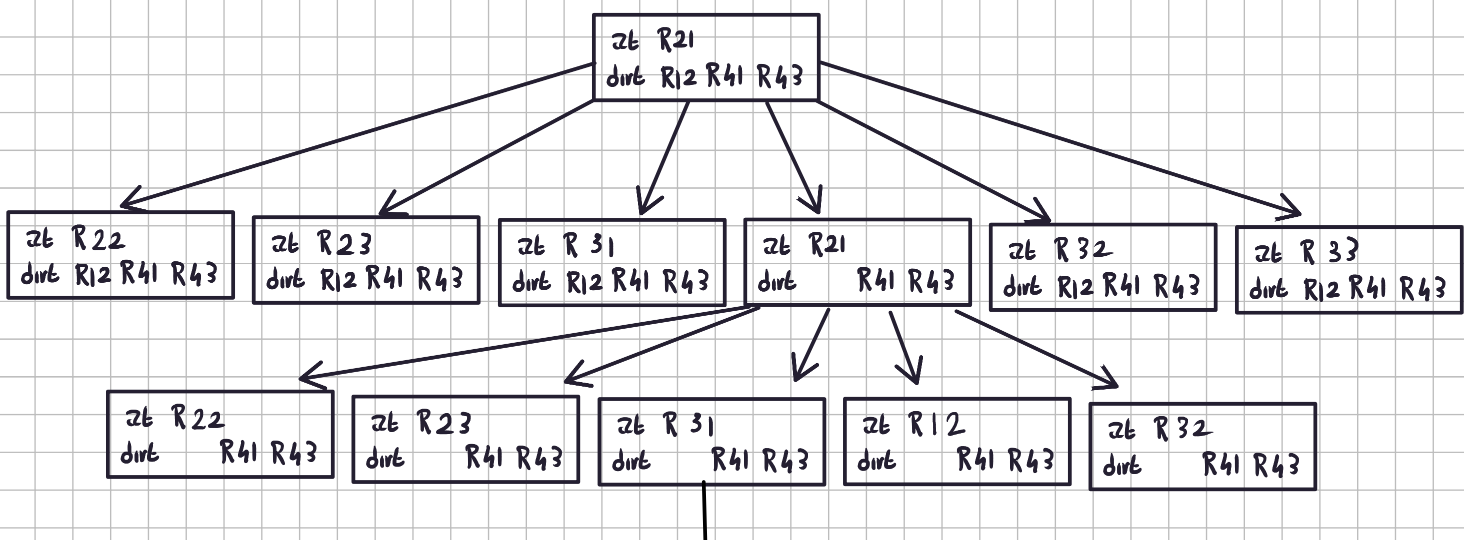
Problem File

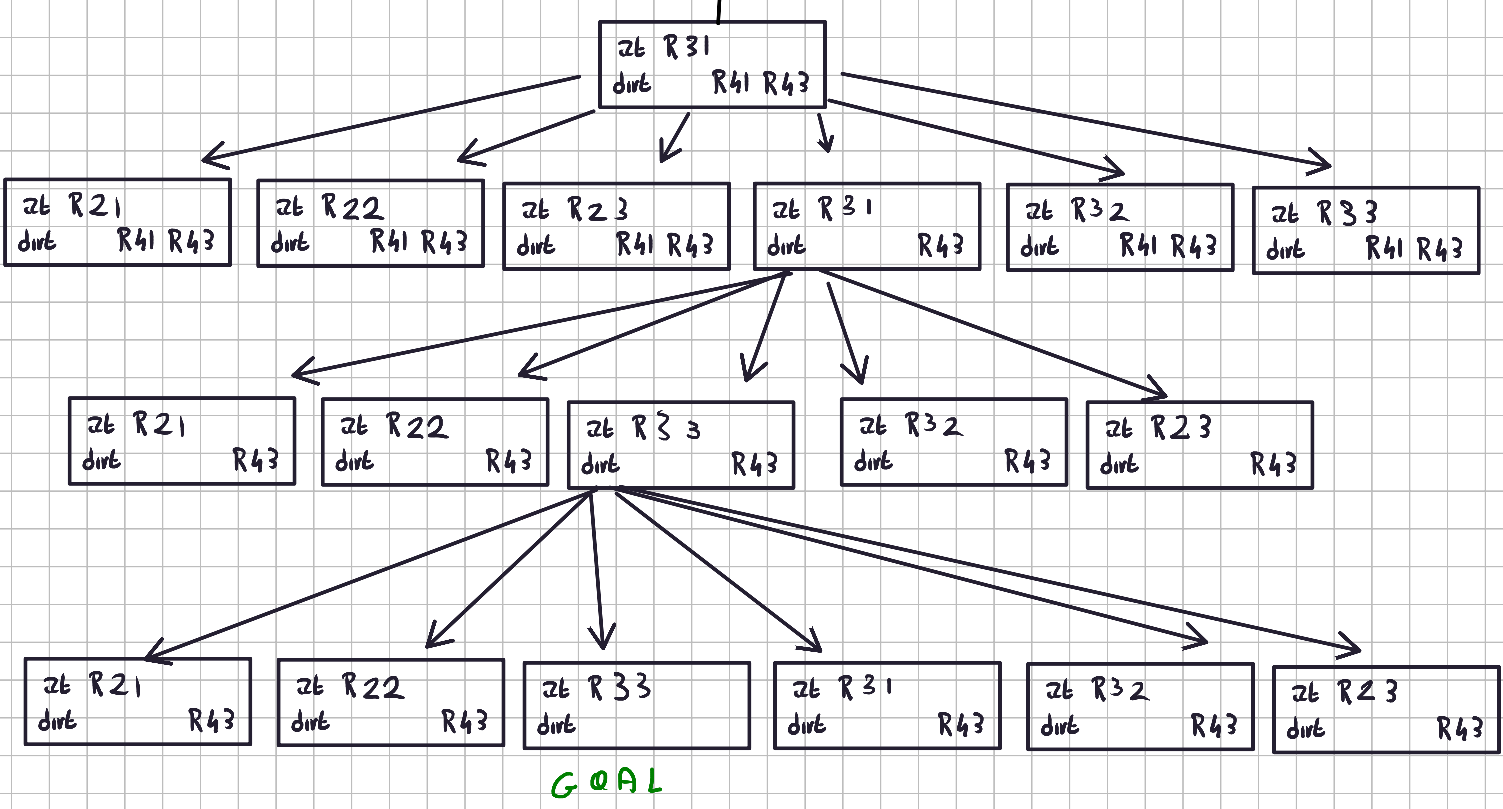
```
(:objects R11 R12 R13 R21 R22 R23 R31 R32 R33 R41 R42 R43 - room)

(:init (at R21) (guest R11) (guest R13) (guest R42) (dirty R12) (dirty R41) (dirty R43) (trav R21)
       (trav 22) (trav 23) (trav 31) (trav 32) (trav 33) (front R21 R11) (front R22 R12) (front R23 R13)
       (front R31 R41) (front R32 R42) (front R33 R43)
)

(:goal (forall (?R - room) (not (dirty ?R))))
```

We denote a state with the assignment of true predicates (omitting constant assignment)





Illustrate the procedure for generating a clausal (conjunctive) normal form of a generic formula in first-order logic.

(a) provide the CNF of the formula $A \wedge B \leftrightarrow C \vee D$
(b) provide the CNF of the formula $\forall x\, \exists y\, ((F(x) \wedge G(x, y)) \rightarrow \exists z\, H(x, z))$

Children are happy if (and only if) someone makes them happy. Parents make their children happy. Anna is the mother of Paola.

(a) Represent the above sentences in first-order logic.
(b) Transform them into CNF.
(c) Is Paola a happy child? If she is, prove it using resolution. Otherwise, consider adding some knowledge to prove it (except for the trivial addition of $Happy(Paola)$) and show the resolution proof.

For a generic formula we have to:

remove $\rightarrow$: $A \rightarrow B \equiv \neg A \vee B$
remove $\leftrightarrow$: $A \leftrightarrow B \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$
push $\neg$: $\neg(A \wedge B) \equiv \neg A \vee \neg B$
$\neg(A \vee B) \equiv \neg A \wedge \neg B$

distribute $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

(a) $A \wedge B \leftrightarrow C \vee D \equiv$

$A \wedge [(B \wedge C) \vee (\neg B \wedge \neg C)] \vee D$

$[A \wedge (B \wedge C)] \vee [A \wedge (\neg B \wedge \neg C)] \vee D$

$(A \wedge B \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee D$

$[(A \wedge B \wedge C) \vee A] \wedge [(A \wedge B \wedge C) \vee \neg B] \wedge [(A \wedge B \wedge C) \vee C] \vee D$

$[A \wedge (A \vee B) \wedge (A \vee C) \wedge (A \vee \neg B) \wedge (C \vee \neg B) \wedge (B \vee C) \wedge C] \vee D \equiv$

Δ : $\{A, D\}$ $\{A, B, D\}$ $\{A, C, D\}$ $\{A, \neg B, D\}$ $\{C, \neg B, D\}$ $\{B, C, D\}$ $\{C, D\}$

b) $\forall x \exists y ((F(x) \wedge G(x,y)) \rightarrow \exists z H(x,z)) \equiv$

$\forall x ((F(x) \wedge G(x, S(x))) \rightarrow \exists z H(x,z)) \equiv$

$\forall x (\exists z H(z) \vee \neg F(x) \vee \neg G(x, S(x))) \equiv$

$\forall x \exists z (H(z) \vee \neg F(x) \vee \neg G(x, S(x))) \equiv$

$\forall x [H(g(x)) \vee \neg F(x) \vee \neg G(x, S(x))]$

The sentence is

1) $\forall x \left[Child(x) \rightarrow \left(Happy(x) \leftrightarrow \exists y \, MakesH(y,x) \right) \right]$

2) $\forall x \forall y \left[(Child(x) \wedge Parent(y,x)) \rightarrow MakesH(y,x) \right]$

3) $Parent(anna, paola)$

in CNF:

1) $\forall x \left[Child(x) \rightarrow \left(Happy(x) \leftrightarrow \exists y \, MakesH(y,x) \right) \right]$

$\forall x \left[\neg Child(x) \vee \left(Happy(x) \leftrightarrow MakesH(S(x),x) \right) \right]$

$\forall x \left[\neg Child(x) \vee [Happy(x) \wedge MakesH(S(x),x)] \vee [\neg Happy(x) \wedge \neg MakesH(S(x),x)] \right]$

$\forall x \left[(\neg Child(x) \vee Happy(x)) \wedge (\neg Child(x) \vee Makes(S(x),x)) \vee [\neg Happy(x) \wedge \neg MakesH(S(x),x)] \right]$

$\forall x \left[(\neg Child(x) \vee Happy(x)) \wedge (\neg Child(x) \vee Makes(S(x),x) \vee \neg Happy(x)) \wedge (\neg Child(x) \vee Makes(S(x),x) \vee \neg MakesH(S(x),x)) \right]$

true

$\Rightarrow \{\neg Child(x_1), Happy(x_1)\}_{1A}$

$\{\neg Child(x_1), MakesH(S(x_1),x_1), \neg Happy(x_1)\}_{1B}$

2) $\forall x \forall y \left[(Child(x) \wedge Parent(y,x)) \rightarrow MakesH(y,x) \right]$

$\forall x \forall y \left[\neg Child(x) \vee \neg Parent(y,x) \vee Makes(y,x) \right]$

$\{\neg Child(x_2), \neg Parent(y_2,x_2), Makes(y_2,x_2)\}_2$

We want to prove:

$Happy(paola) \wedge Child(paola)$, negated: $\neg Happy(paola) \vee \neg Child(paola)$

So we consider the KB:

$\{\neg Child(x_1), Happy(x_1)\}_{1A}$

$\{\neg Child(x_1), MakesH(S(x_1),x_1), \neg Happy(x_1)\}_{1B}$

$\{\neg Child(x_2), \neg Parent(y_2,x_2), Makes(y_2,x_2)\}_2$

$\{Parent(anna, paola)\}_3$

$\{\neg Happy(paola), \neg Child(paola)\}_4$

This is not sufficient since we need to know that paola is a child. we add also: $\{Child(paola)\}_5$.

From 4, 5 $\Rightarrow \{\neg Happy(paola)\}_6$

From 1A, 6 with $s = \left\{ \frac{x_1}{paola} \right\} \Rightarrow \{\neg Child(paola)\}_7$

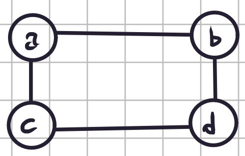
From 5 and 7 we get $\Rightarrow \{\square\}$ so the implication that paola is a happy child is true.

Consider the following constraint network: $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d\}$
- Domains: For all $v \in V$: $D_v = \{5, 10, 15, 20\}$
- Constraints: $a > b + 10; b + d \leq 15; 20 < d + c; a + c > 30;$

1. Draw the constraint graph.
2. Can you solve the problem using the AC-3 algorithm? Please explain why and if you can, proceed to solve the problem using the algorithm.
3. Can you solve the problem using the AcyclicCG algorithm? Please explain why and if you can, proceed to solve the problem using the algorithm.

the constraint graph is :



We try to make it Arc Consistent using AC3

$M = \{ac, ca, ab, ba, cd, dc, bd, db\}$

pick $ac \Rightarrow D_a : \{15, 20\}$

$M : \{ca, ab, ba, cd, dc, bd, db\}$

pick $ca \Rightarrow D_c : \{15, 20\}$

$M : \{ab, ba, cd, dc, bd, db\}$

pick $ab \Rightarrow D_a : \{20\}$, add ca

$M : \{ba, cd, dc, bd, db, ca\}$

pick $ba \Rightarrow D_b : \{5\}$

$M : \{cd, dc, bd, db, ca\}$

pick cd D_c unchanged
pick dc D_d unchanged

$M : \{bd, db, ca\}$

pick bd, D_b unchanged

$M : \{db, ca\}$

pick db, $D_d : \{5, 10\}$, add cd

$M : \{ca, cd\}$

pick ca unchanged

$M : \{cd\}$ unchanged

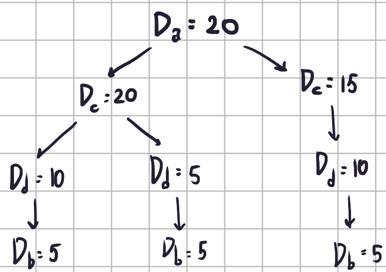
$\Rightarrow D_a : \{20\}$

$D_b : \{5\}$

$D_c : \{15, 20\}$

$D_d : \{5, 10\}$

We can't directly solve the problem.



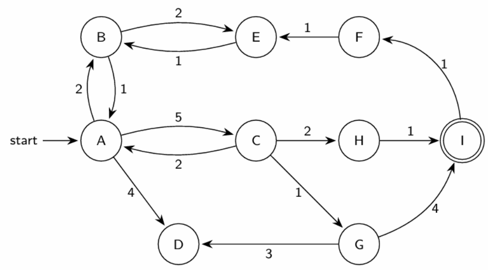
$\Rightarrow$ this AC network leads to three possible solution.

It is not possiple to apply the AcyclicCG algorithm since the constraint graph is not acyclic.

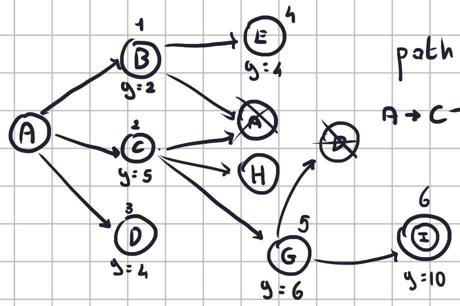
Consider the state space in the figure below, where A is the initial state and I the goal state. The transitions are annotated by their costs.

i. Run the uniform-cost search algorithm on this problem. Draw the search graph and annotate each node with its g value and the order in which states are selected for expansion. Draw duplicate nodes, and mark them accordingly by crossing them out. If the choice of the next state to be expanded is not unique, expand the lexicographically smallest state first (e.g., a before d). Give the solution found by uniform-cost search. Is this solution guaranteed to be optimal? Justify your answer.

ii. Run the iterative-deepening search algorithm on this problem until it finds a solution. For each depth, depict the corresponding search tree. Annotate each state with the order in which states are selected for expansion. If the choice of the next state to be expanded is not unique, expand the lexicographically smallest state first. Give the solution found by iterative-deepening search. Is this solution guaranteed to be optimal? Justify your answer.



Uniform Cost Search



notation

order
n
g = cost

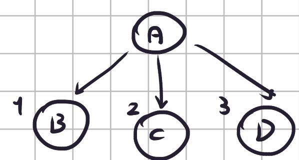
path found:

A → C → G → I   cost: 10

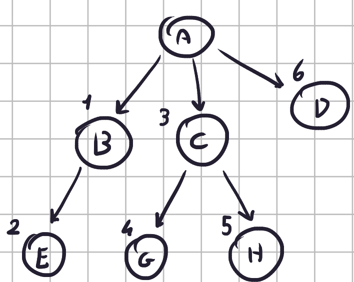
This solution is trivially not guaranteed to be optimal, the UCS assumes a uniform heuristic for all the nodes, and is clearly not a good solution when the cost of the arcs are different. It would be correct if the graph have uniform cost on arcs.

Iterative deepening search

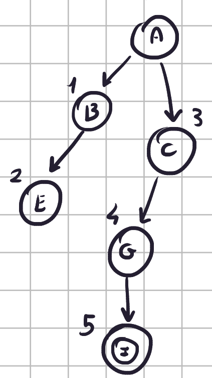
depth = 1



depth = 2



depth = 3



at depth 3 the solution A → C → G → I is found.

For the same reason of the UCS, the solution is not optimal, this method, like the BFS, find the shortest path in term of number of edges, without considering the cost. Only A^* can give the optimal solution A → C → H → I