Artificial Intelligence Problems **(2017–2018)**
Third year – Grado en Ingeniería Informática,
Universidad de Sevilla

PLANNING

Questions

Question 1. Consider a problem formulated in the states space framework, having constant branching factor b and having a unique solution located at depth d. Calculate, both for the best and the worst cases, the number of nodes that need to be analyzed in order to find the solution when running a breadth-first search. Do the same for depth-first search.

Question 2. Provide an example of a state space problem such that the `VISITED` list is not needed in depth-first search, explaining why it is so.

Question 3. Consider a state space problem such that all the solutions, in case they exist, must have a fixed number of actions d. If we need to choose between depth-first search and breadth-first search, which one is more adequate? The answer should be justified.

Problems

Exercise 1. The Missionaries and Cannibals problem can be stated as follows: "M missionaries and C cannibals need to cross a river, and the only boat available has a capacity for P people. The problem is to find the way to transport everyone across the river, taking into account that when the boat leaves a bank, the missionaries remaining there cannot be outnumbered by the cannibals (if there are missionaries). Formulate the problem in the state space framework

More info at: `http://en.wikipedia.org/wiki/River-crossing_problems`

Exercise 2. Consider the following puzzle: we have a linear board having 3 black pieces (B), 3 white pieces (W) and an empty space (E), in the following initial configuration:

B B B W W W E

The goal of the puzzle consists on placing all the white pieces to the left of all the black pieces, the position of the empty space is irrelevant. The legal moves are the following:

- A piece can be moved to an adjacent empty cell.
- A piece can go to the empty space by jumping over at most two adjacent pieces.

You should:

- Formulate this problem in the states space framework, describing precisely all the elements required for the representation.
- Define an heuristic function for this problem.

Exercise 3. Explain, without getting into implementation details, how to formulate the following problem in the states space framework. That is, provide the representation for the states (specifying initial state and final states) and the actions (applicability conditions and how the actions are applied to a state). Define also a heuristic function.

The problem can be stated as follows: given four natural numbers n, m, r and T, find a sequence of basic arithmetic operations (addition, subtraction, multiplication and division) starting from 0 and getting T as final result, in such a way that using only numbers n and m are used in the operations, having the additional constraint that neither n or m can be used more than r times.

For example, let $n = 2$, $m = 3$, $r = 3$ and $T = 28$, a possible solution is $(((((0+3)\times 3)\times 2)-3)\times 2)-2)$, since the total result is 28 and neither 2 nor 3 have been used more than three times each.

Exercise 4. Consider the problem of the *3-puzzle*, a reduced version of the 8-puzzle, where there are three tiles (tagged with 1, 2, 3) in a 2×2 board (hence there is a blank space). Initial and final states are, respectively:

	1
3	2

Initial state

1	2
	3

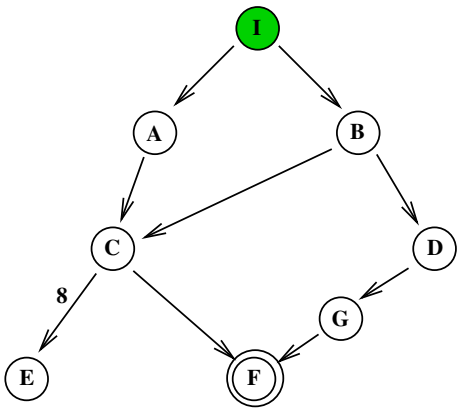
Final state

The actions to be considered are: moving the space up, moving the space down, moving the space left, and moving the space right. Represent graphically the search trees corresponding to:

1. Depth-first search, considering the actions in the order specified above.
2. Depth-first search search using heuristic h_1 = Manhattan distance from the current position of the space to its position in the final state.
3. Depth-first search search using heuristic h_2 = number of tiles which are not in the same position as in the final state.

Exercise 5. The following graph represents a states space for a problem. Nodes of the graph are the states of the problem, the edges connect the states to their successors.

The initial state is **I**, and the only final state is **F**. We consider the heuristic functions H_1 and H_2 given by the following table:

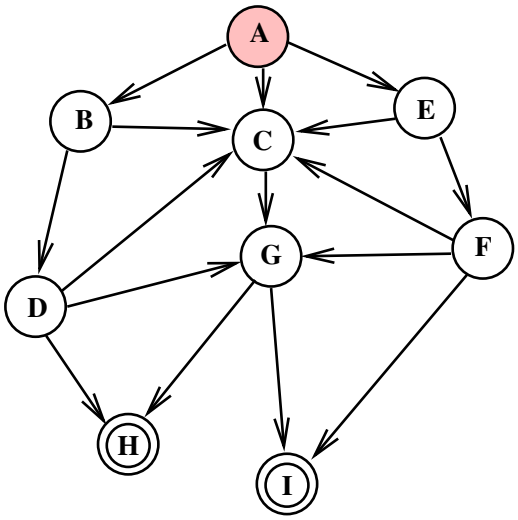


State	H_1	H_2
I	16	20
A	8	8
B	10	6
C	3	12
D	2	2
E	9	9
F	0	0
G	1	1

1. Build the search trees generated by the breadth-first and depth-first search, considering succesors in alphabetical order
2. Build the search trees generated by depth-first search with heuristics H_1 and H_2 , respectively.

Exercise 6. The following graph represents a states space for a problem. Nodes of the graph are the states of the problem, the edges connect the states to their successors, and the number on each edge stands for the cost of going from a state to its successor.

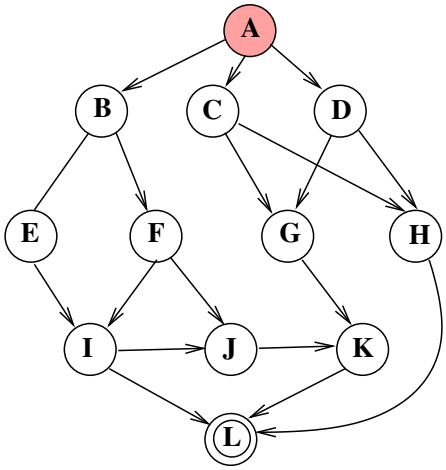
The initial state of the problem is **A**, and the final states are **H** and **I**. We consider the heuristic function h given by the following table:



Node	Heuristic
A	7
B	7
C	2
D	7
E	4
F	9
G	1
H	0
I	0

1. Build the search trees corresponding to breadth-first and depth-first search, considering the successors in alphabetical order.
2. Build the search tree corresponding to depth-first search, using the heuristics above

Exercise 7. The following graph represents a states space for a problem. Nodes of the graph are the states of the problem, the edges connect the states to their successors, and the number on each edge stands for the cost of going from a state to its successor. The initial state is **A**, and the only final state is **L**.



1. Find the shortest path (in number of actions) to go from A to L, by using either *Depth-first* search or *Breadth-first* search algorithm. Explain your choice. Show the search tree generated.
2. Run a depth-first search using the heuristic of the following table:

	A	B	C	D	E	F	G	H	I	J	K	L
h	10	8	10	6	6	5	3	4	5	3	2	0

Exercise 8. Consider the following set of predicates describing the world for a planning problem where a truck `T` needs to deliver packages to several cities:

- `PACKAGE(x)`: object `x` is a package.
- `CITY(x)`: object `x` is a city.
- `HIGHWAY(c1,c2)`: the cities `c1` and `c2` are connected by a highway.
- `IN(x,c)`: object `x` (the truck or a package) is in the city `c`.
- `IN-TRUCK(x)`: package `x` is loaded in the truck.
- `UNLOADED()`: the truck is unloaded.

The actions that can be executed are the following:

- `LOAD(p,c)`: the truck (that should be unloaded) loads the package `p` in the city `c`. Once loaded, the package is no longer considered to be in the city `c`.
- `UNLOAD(p,c)`: the truck unloads the package `p` in the city `c`.
- `GO(c1,c2)`: the truck travels along the highway from city `c1` to city `c2`.

Suppose that we are in an initial state where there is a package `P1` in Barcelona, another package `P2` in Madrid, and the truck `T` is in Sevilla. We want to find a sequence of actions such that the package `P1` is delivered to Sevilla, the package `P2` to Barcelona and the truck is unloaded. There are two available highways, one of them connecting Barcelona and Madrid, and the other one connecting Madrid and Sevilla.

Represent the problem using the PDDL formalism. That is, describe the initial state, the goal and the actions.

Exercise 9. Robi the robot is in a house, and there are objects that he can carry inside some of the rooms. Basically, Robi can: move from a room to another one; grab an object if it is in the same room (with the restriction that it cannot take more than one object); and release an object in the room where it currently is. If the goal is to take all the objects to the kitchen, formulate the problem as a planning problem, using the PDDL formalism (that is, defining the logic language to be used, the initial state, the actions and the goal).

Exercise 10. Consider a domain where there is a robot able to transport boxes between connected rooms.

This domain is represented using the following language:

- `C1`, `C2` stand for the two boxes; `H1`, `H2`, `H3` for the three rooms; `P1`, `P2` for the two doors.
- `OPEN(x)`: door `x` is open
- `IN(x,y)`: box `x` is in room `y`
- `ROBOT-IN(x)`: the robot is in room `x`
- `CONNECTS(x,y,z)`: door `x` connects rooms `y` and `z`

Provide a representation, within the PDDL formalism, of the following four actions:

- `GO-THROUGH(x,y,z)`: robot moves from room `y` to `z` through the door `x`
- `CARRY-THROUGH(c,h1,h2,p)`: robot carries the box `c` and moves it from `h1` to `h2` through the door `p`

- `CLOSE(x)`: robot closes door `x`
- `OPEN(x)`: robot opens door `x`

Exercise 11. Consider the problem of the kid and the candy.

There is a kid in a room who wants to pick a candy tied to the ceiling with a string. The kid cannot reach the candy unless he gets on top of a bench.

Assume that, initially, the kid is in room `A`, the candy is in room `B` and the bench is in room `C`.

The available actions are the following: GO from one room to another; TRANSPORT an object from one room to another; GET-ON-TOP of an object; and CATCH an object (the kid can catch an object if they are both in the same room and he can reach the object).

The goal is, of course, that the kid catches the candy.

You should:

1. Declare the constant symbols and the predicates needed to represent the domain of the problem.
2. Provide a representation, within the PDDL formalism, of the previous actions.
3. Define the initial state and the goal as conjunctions of literals.

Exercise 12. Consider the following planning problem:

- Actions:

```
A             B             C         D
-----------   -----------   -------   -------
Prec: P1      Prec: P1      Prec: P4  Prec: P1
Effe: -P1,P2  Effe: -P1,P3  Effe: P1  Effe: P4
```

- Initial state: {`P1`}
- Goal: {`P2,P3`}

Apply depth-first search to get a solution to this problem. Consider the applicable actions in alphabetical order

Exercise 13. Consider the following planning problem, where the initial state is $\{p_1\}$ and the goal is $\{p_4, p_5\}$:

Action	Preconditions	Effects
A	p_6	$\neg p_6$, p_2
B	p_3	$\neg p_4$, p_5
C	p_3	$\neg p_3$, p_2
D	p_2	$\neg p_2$, $\neg p_5$, p_4
E	p_1	p_3
F	p_3	$\neg p_1$, p_5

Solve this problem applying depth-first search. Note: order the applicable actions alphabetically.

Exercise 14. Consider the following planning problem, with actions:

```
A              B              C
-----------    ----------     ----------
Prec: P3       Prec: P1       Prec: P3
Effe: P6,-P2   Effe: P2,P3    Effe: -P2,P5, P8

D              E              F
------------   ---------      -----------
Prec: P1,P2    Prec: P5,P6    Prec: P2, P4
Effe: P5,-P8   Effe: P7       Effe: P7
```

and with initial state $e =$ {P1} and goal $g =$ {P7, P8}. Compute $\Delta_0(e, g)$.

Exercise 15. With the following set of actions, compute $\Delta_0(\{p_1\}, \{p_3, p_4\})$.

Action	Precondition	Effects
A	p_1	$\neg p_6$,p_3
B	p_2,p_5	$\neg p_2$, p_3, p_4
C	p_1	p_2, p_5

Exercise 16. Consider a robot that can execute the following three actions

Action	Preconditions	Effects
GO(x,y)	IN(Robot,x)	-IN(Robot,x), IN(Robot,y)
TAKE(o)	IN(Robot,x), IN(o,x)	-IN(o,x), HOLDS(o)
RELEASE(o)	IN(Robot,x), HOLDS(o)	-HOLDS(o), IN(o,x)

Assume that we have the following problem:

- Initial state: IN(Robot,A), IN(O1,B), IN(O2,C)
- Goal: HOLDS(O1), HOLDS(O2)

Obtain a final partial plan by applying a sequence of refinement steps to the initial partial plan. After that, obtain a solution (ordered plan).

Exercise 17. Consider the actions:

Action	Preconditions	Effects
GO(x,y)	IN(x)	-IN(x), IN(y)
BUY(x)	SELLS(y,x), IN(y)	HAVE(x)

and assume that we have the following problem:

- Initial state: IN(Home), SELLS(Mercadona,Coffee), SELLS(Mercadona,Milk), SELLS(Carrefour, Sugar).
- Goal: HAVE(Coffee), HAVE(Milk), HAVE(Sugar), IN(Home).

Solve this problem applying the POP algorithm, including for each step the graphical representation of the partially ordered plan and the set of open preconditions.

Exercise 18. Consider the following planning problem, described using PDDL notation. Prove, by executing the POP algorithm, that **there is no solution** for that problem.

- Actions:

```
A              B
-------------  -------------
Prec:  P1      Prec:  P1
Effec: -P1,P2  Effec: -P1,P3
```

- Initial state: {P1}
- Goal: {P2,P3}

Exercise 19. Paul wants to give a surprise to Lucy, by preparing dinner for her and buying a gift. For that, he needs to get rid of the rubbish (to avoid its bad smell), have his hands clean, and wrap the gift. In order to get rid of the rubbish, he can either grab it and take it out (getting his hands dirty in the process), or he can use the noisy rubbish crusher (will keep his hands cleen, but the gift can get accidentally crushed). One last observation: Lucy is sleeping, so he has to wrap the gift in silence.

This situation can be described by the following PDDL representation:

- Actions:

```
CRUSH()                          TAKE-OUT()
-------                          --------------
Prec: {}                         Prec: {}
Effe: -QUIET, -GIFT, GOOD-SMELL  Effe: -CLEAN-HANDS,GOOD-SMELL

COOK()                           WRAP()
------                           ------
Prec: CLEAN-HANDS                Prec: QUIET,CLEAN-HANDS
Effe: -CLEAN-HANDS,DINNER        Effe: GIFT

WASH-HANDS
----------
Prec: {}
Effe: CLEAN-HANDS
```

- Initial state: {QUIET, CLEAN-HANDS}
- Goal: {DINNER,GIFT,CLEAN-HANDS,GOOD-SMELL}

Apply the POP algorithm to obtain a solution of this planning problem. When different actions can solve an open precondition, consider them in alphabetical order (except for action START, that should be considered always first). Also, consider always first simple establishment (if possible) and then step addition. To solve threats, first promotion and then demotion.

Exercise 20. Execute the POP algorithm to find a solution to the following planning problems. You should describe in detail the sequence of partial plans that are analyzed by the algorithm, indicating for each branching point the available refinement alternatives (which one is chosen first, and if the election has to be reconsidered). Finally, indicate the solution(s) finally obtained.

Note the following:

- The sequence of partial plans should be described in detail, by numbering them and indicating the points when the algorithm needs to go back and choose a different branch.
- Provide also the final partial plan obtained, as well as the solution that can be obtained from it. **Check that the plan obtained is in fact a solution**.
- Whenever there exist several actions solving an open precondition, consider the alternatives **in alphabetical order** of the names of the actions, except for action START, that should be considered always first.
- Besides, if an action can be used by *simple establishment* and also by *step addition*, try them in this order.
- Whenever there exists a threat or conflict to be solved, try first *promotion*, and then if needed, try also *demotion*.
- It is allowed (even encouraged) to apply several refinement steps together. However, it is advisable to draw a new graph every time there is a branching point to be eventually reconsidered.

---

1. Planning problem:
   - Actions:

```
A               B              D
-------------   -------------  -------------
Prec:  P6       Prec:  P3      Prec:  P3
Effec: -P6,P2   Effec: -P4,P5  Effec: -P3,P2


E                  F           G
-----------------  ---------   -------------
Prec:  P2          Prec:  P1   Prec:  P3
Effec: -P2,-P5,P4  Effec: P3   Effec: -P1,P5
```

   - Initial state: {P1}
   - Goal: {P4,P5}

---

2. Planning problem:
   - Actions:

```
A             B             C              D             E          H
------------  ------------  -------------  ------------  ---------  ---------
Prec:  P7     Prec:  P2,P5  Prec:  P3      Prec:  P1,P4  Prec:  P1  Prec:  P2
Effec: -P7,P3 Effec: P4     Effec: -P2,P5  Effec: P6     Effec: P3  Effec: P4
```

- Initial state: {P1,P2}
- Goal: {P5,P6}

---

3. Planning problem:

   - Actions:

```
A          B             C              D
---------  ------------  -------------  -------------
Prec:  P1  Prec:  P2     Prec:  P3,P4   Prec:  P3
Effec: P5  Effec: P4,P6  Effec: -P6,P7  Effec: -P2,P7
```

   - Initial state: {P1,P2,P3}
   - Goal: {P5,P6,P7}

---

4. Planning problem:

   - Actions:

```
A              B             C              D
-------------  ------------  -------------  -------------
Prec:  {}      Prec:  P0,P3  Prec:  P1      Prec:  {}
Effec: -P1,P3  Effec: P4     Effec: -P0,P2  Effec: P1,-P2
```

   - Initial state: {P0}
   - Goal: {P2,P4}

---

5. Planning problem:

   - Actions:

```
A          B                  C
---------  -----------------  --------------------
Prec:  P8  Prec:  P2,P7       Prec:  P1,P7
Effec: P4  Effec: -P6,-P7,P5  Effec: -P4,-P7,P3,P6
```

```
D                E              F             G
---------------  -------------  ------------  ---------
Prec:  {}        Prec:  P3,P7   Prec:  P1,P2  Prec:  {}
Effec: -P5,P2,P6 Effec: -P5,P4  Effec: P6     Effec: P7
```

   - Initial state: {P1}
   - Goal: {P4,P5,P6}

---

6. Planning problem:

   - Actions:

```
A                 B             C
----------------  ------------  ------------
Prec:  {}         Prec:  P1     Prec:  P7
Effec: -P5,-P4,P2 Effec: -P1,P5 Effec: -P7,P5


D                 E             F
----------------  ------------  ------------
Prec:  P5         Prec:  P6     Prec:  P6
Effec: -P2,-P7,P4 Effec: -P2,P3 Effec: -P6,P2
```

   - Initial state: {P6,P7}
   - Goal: {P2,P3,P4}

---

7. Planning problem:

   - Actions:

```
A             B                 C
-----------   ----------------  ------------
Prec:  P1     Prec:  P2,P4      Prec:  P3
Effec: P3,P4  Effec: -P3,-P5,P6 Effec: -P3,P5
```

   - Initial state: {P1,P2}
   - Goal: {P5,P6}

---

8. Planning problem:

   - Actions:

```
A              B             C           D           E
-------------  ------------  ----------  ----------  ----------
Prec:  P2      Prec:  P3     Prec:  P0   Prec:  P0   Prec:  {}
Effec: P1,     Effec: P2,P5  Effec: P4,  Effec: P3,  Effec: P4,
      -P2,-P5                     -P0         -P1         -P1
```

   - Initial state: {P0}
   - Goal: {P1,P2,P4}

---

9. Planning problem:

   - Actions:

A	B	C	D
Prec: P0	Prec: P4	Prec: P3	Prec: P0,P2
Effec: P3,P4	Effec: -P4,P5,	Effec: -P4,P2,	Effec: -P0,P5,

   - Initial state: {P0}
   - Goal: {P4,P5}