Departamento de Sistemas Informáticos y Computación
Escuela Técnica Superior de Ingeniería Informática
Universitat Politècnica de València

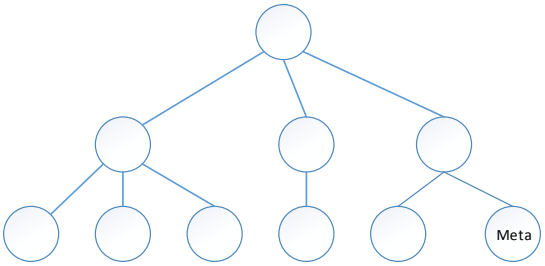
SOLUTIONS TO COLLECTION OF EXERCISES
INTELLIGENT SYSTEMS

Block 1: Solving problems by Search

September 2019

MULTIPLE CHOICE QUESTIONS

1) Given the space state of the figure, if we apply a DEPTH-FIRST search with *backtracking* (expanding first the leftmost node), which is the maximum number of nodes kept simultaneously in memory (nodes in both OPEN and CLOSED)?



A. 7
B. 5
C. 10
D. 4

---

2) Consider the 8-puzzle problem, where h1=number of misplaced tiles (tiles in the wrong place), and h2=Manhattan distance. Let h3=min(h1,h2) and h4=abs(h1-h2), which of these two heuristic functions, h3 and h4, is admissible?

A. Only h3.
B. Only h4.
C. Both of them.
D. None of them.

---

3) Consider a search problem with a branching factor b=10 and depth of the solution d=5. Let's assume we apply a BREADTH-FIRST search (BFS), a DEPTH-FIRST search (DFS) where the maximum depth limit set by the user is 5, and an ITERATIVE DEEPENING algorithm (ID). Regarding the number of generated nodes, which of the following orderings (from the most costly to the least costly) is correct?

A. BFS > ID > DFS
B. BFS > DFS > ID
C. ID > BFS > DFS
D. DFS > BFS > ID

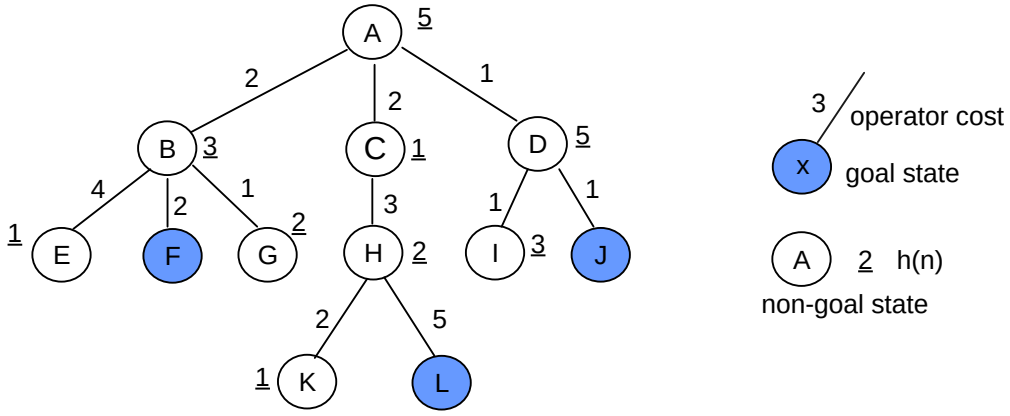
---

4) If we use a consistent heuristic function in an A algorithm, which assertion is **FALSE**?

A. f(n) is a non-decreasing function.
B. It guarantees the optimal solution in a graph-search version even without re-expanding the nodes.
C. The search process will never generate a node 'n1' equal to another already generated node 'n2' where f(n1) < f(n2).
D. The h-value of a child node can be lower than the h-value of a parent node.

---

5) Given the state space of the figure, if we apply a ***greedy*** search, which of the goals states is first selected as a solution?



A. J
B. L
C. I
D. F

---

6) Given the state space of figure 5, if we apply an Iterative Deepening (ID) algorithm (expanding first the leftmost node), how many nodes would be generated altogether?

A. 8
B. 7
C. 10
D. 12

---

7) Given a search problem in which *all the operators have the same cost*, show which of the following assertions is ***correct***:

A. A search algorithm that uses an admissible heuristic will return the shortest solution
B. A depth-first search (DFS) will always return the lowest cost solution
C. A breadth-first search (BFS) will return the shortest solution but not the lowest cost solution
D. A uniform-cost search strategy will return the lowest cost solution but not the shortest solution

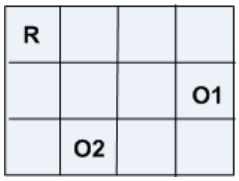
---

8) Given four search algorithms: M1 applies a breadth-first search (BFS); M2 applies a uniform-cost search strategy; M3 applies an A algorithm and uses an admissible heuristics; M4 applies an A algorithm and uses a non-admissible heuristic. Show the assertion that is ***incorrect***:

A. M1, M2 and M3 guarantee the optimal solution regardless the cost of the operators
B. M2 and M3 guarantee the optimal solution regardless the cost of the operators
C. M3 will expand fewer nodes than M2
D. M4 could find the optimal solution

---

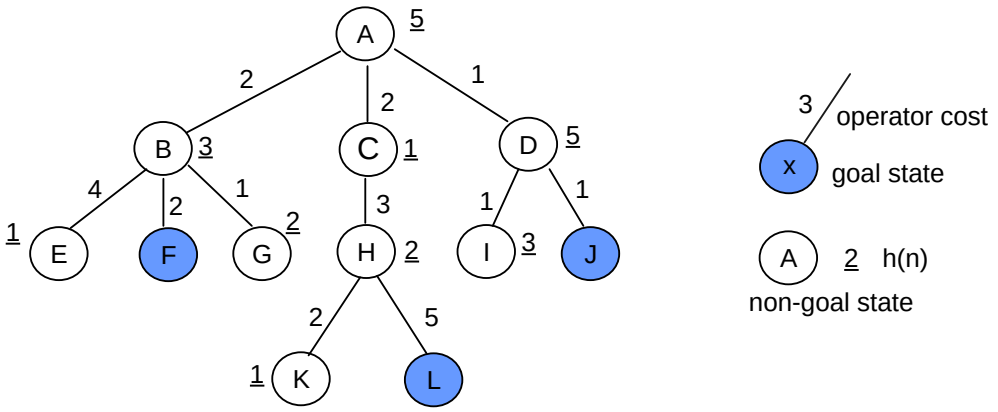
9) The figure below shows a grid where R is a robot which wants to reach the position where the object *O1* is located and then reach the position of the object *O2*. The robot only moves horizontally and vertically. R, *O1* and *O2* can be placed in any cell of the grid, the figure just shows a particular instance of the problem. Let n be a node of a search tree that represents a particular placement of R, *O1* and *O2*, and let manh(x,y) be the Manhattan distance between x e y, where x, y ∈ {R,*O1*,*O2*}. Which is the ***correct*** assertion?



A. h(n)=manh(R,*O1*)+manh(R,*O2*) is an admissible heuristic for this problem
B. h(n)=manh(R,*O1*)+manh(*O1*,*O2*) is an admissible heuristic for this problem
C. h(n)=manh(R,*O1*)*2 is an admissible heuristic for this problem
D. It is not possible to define an admissible heuristic for this problem

---

10) Given the search space of the figure and assuming we apply an algorithm of type A (f(n)=g(n)+h(n)), how many nodes need to be generated to find the solution?



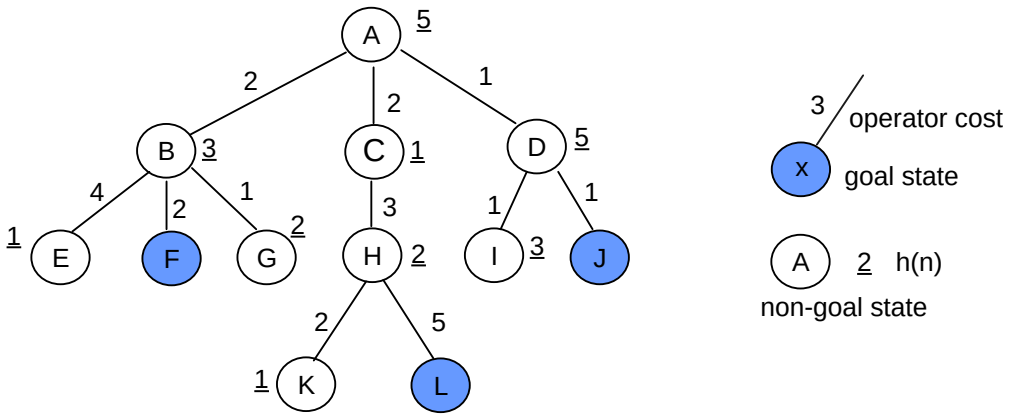
A. 6
B. 8
C. 10
D. 12

11) Regarding the search space of question 10, mark the ***correct*** assertion:

A. The application of the algorithm of type A returns the optimal solution
B. The function h(n) is consistent (monotone)
C. A uniform-cost search returns the same solution as the one returned by the algorithm of type A
D. None of the above.

---

12) For the search space of the figure and given a search of type A ($f(n)=g(n)+h(n)$), which of the following assertions is **CORRECT**:



A. The solution found by the search of type A is node J.
B. The search of type A generates 10 nodes to find the solution.
C. The heuristic function h(n) is not admissible.
D. None of the above.

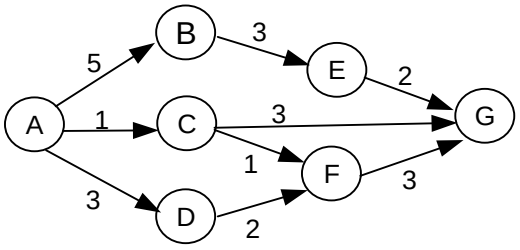
---

13) Given four search methods: M1 applies Breadth-first, M2 applies Uniform Cost, M3 applies Depth-first and M4 is an Iterative-Deepening algorithm; assuming all the operators have the same cost, mark the assertion that is **INCORRECT**:

A. M1, M2, M3 and M4 will find the optimal solution if it exists
B. M1 and M2 guarantee the optimal solution.
C. M4 will find the optimal solution.
D. The memory requirements are bigger for M1 than M4.

---

14) In the graph below, the numbers on the edges represent the operator cost to go from one node to the other. Mark the assertion that is **CORRECT**.



A. Breadth-first will find the path A-D-F-G
B. The cost of the solution found by a Uniform cost algorithm is 5
C. Breadth-first and Uniform cost will find the same solution
D. None of the above.

---

15) If we apply an Iterative Deepening algorithm over the search space of figure 10, how many iterations are necessary to find a solution?

A. 2
B. 3
C. 4
D. None of the above.

---

16) Let be a search algorithm of type A ($f(n)=g(n)+h(n)$) where $h(n)$ is admissible and consistent. The algorithm returns a solution path from the initial state *A* to the goal state *G* through a node n1. Which of the following assertions is **INCORRECT?**:

A. $f(A) <= f(n1) <= f(G)$
B. $f(G)=h^*(A)$
C. $h^*(A) < h(n1)$
D. $f(G)=g(G)$.

---

17) Let $f1(n)=g(n)+h1(n)$ and $f2(n)=g(n)+h2(n)$ be two evaluation functions for a problem such that $\forall n\ h1(n) \leq h2(n) \leq h^*(n)$. Given a search algorithm of type A that utilizes these functions, mark the assertion that is **TRUE**:

A. Only one of the two evaluation functions will find the optimal solution
B. The algorithm that uses $f1(n)$ will expand fewer nodes than the algorithm with $f2(n)$
C. The algorithm that uses $f1(n)$ will expand more nodes than the algorithm with $f2(n)$
D. None of the two algorithms will develop a complete search

---

18) Given three search methods: M1 applies a uniform-cost search, M2 is an A* algorithm and M3 applies a greedy search, which assertion is INCORRECT?

A. M1 and M2 will find the optimal-cost solution
B. There is guarantee that M3 will find the solution more rapidly than M1 and M2
C. There is no guarantee that M3 will find the optimal-cost solution
D. M1 will expand more nodes than M2

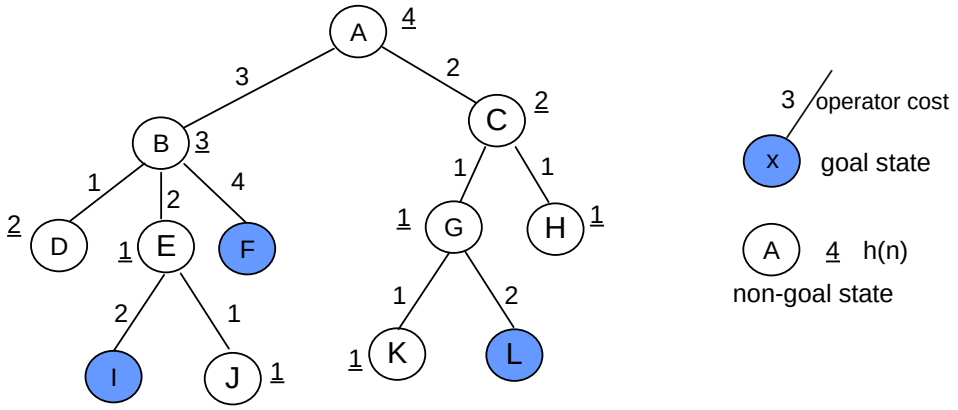
---

19) Given the evaluation functions $f1(n)=g(n)+h1(n)$ and $f2(n)=g(n)+h2(n)$, such that $h1(n)$ is admissible and $h2(n)$ is not, mark the CORRECT answer:

A. Both functions guarantee they will find the optimal-cost solution
B. There is guarantee that $f2(n)$ will generate fewer nodes than $f1(n)$
C. Only in the case that h1(n) is a consistent heuristic function, $f1(n)$ will generate fewer nodes than $f2(n)$
D. There exists some node n for which it holds $h2(n)>h^*(n)$

---

20) Given the state space of the figure and assuming a breadth-first search (expanding the leftmost node first), which assertion is TRUE?



A. The algorithm returns node I
B. The algorithm generates 8 nodes
C. The algorithm expands 4 nodes
D. None of the above

---

21) Given again the state space of question 20 and assuming now a search algorithm of type A ($f(n)=g(n)+h(n)$), which assertion is FALSE?

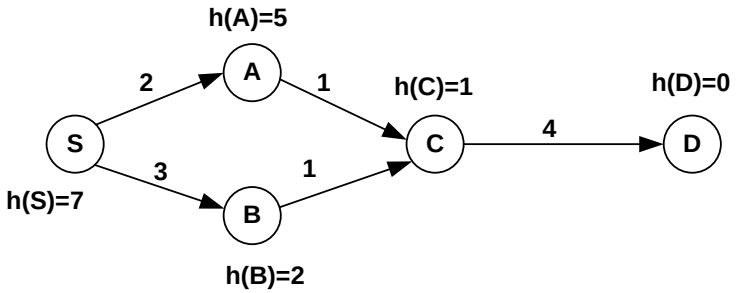
A. $h(n)$ is admissible
B. The algorithm returns the node L
C. The algorithm expands 3 nodes
D. The algorithm generates 7 nodes

22) Assuming that all of the nodes in a search space have more than one child, in which of the following search strategies the order of generation of the nodes is **never** the same as the order of expansion of the nodes?

A. Breadth-first search
B. Uniform cost search
C. Depth-first search
D. Greedy search

---

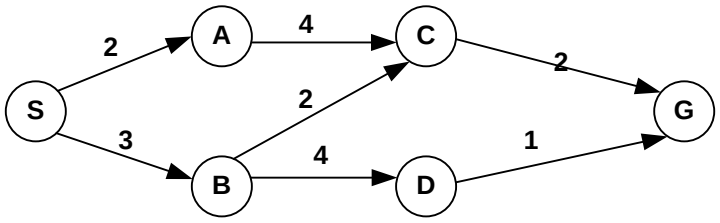
23) In the state space of the figure, S is the initial state, D is the goal state, h(n) is the heuristic estimation for each node and the numeric values on the edges are the costs of the arcs. Indicate the **CORRECT** statement.



A. An A algorithm (TREE SEARCH with control of repeated states in the OPEN list) does not return the optimal solution.
B. An A algorithm (GRAPH SEARCH with control of repeated states in the CLOSED list such that a newly generated node is discarded if it exists in CLOSED) returns the optimal solution
C. The answer A is not TRUE since h(n) is not admissible
D. The answer B is not TRUE since h(n) is not consistent

---

24) Given the state space of the figure, the number of nodes generated by a Uniform Cost search (TREE SEARCH) is (in case of two nodes with the same f-value, expand first the node that comes alphabetically before):



A. Higher than the number of nodes of a BFS
B. Lower than the number of nodes of a BFS
C. Lower than the number of nodes of a DFS
D. None of the above answers is TRUE

25) Given a consistent heuristic function h(n) in a search of type f(n)=g(n)+h(n), which of the following assertions is CORRECT?

A. It does not return the optimal solution
B. The heuristic value of the parent can be the same as the heuristic value of the child
C. It never generates a new node n1, equal to another already generated node n2, such that f(n1) < f(n2).
D. It never generates a new node which is the same as one already in the CLOSED list

---

26) Let be a search f(n)=g(n)+h(n), where h(n) is admissible, and two solution nodes *G1* and *G2*. *G1* is an optimal solution and *G2* is not. Let n1 be a node that belongs to the solution path to *G1*. Mark the **INCORRECT** statement:

A. $g(G1) \leq f(G2)$
B. $f(n1) \leq g(G2)$
C. $h^*(n1)+g(n1)=f(G1)$
D. None of the above

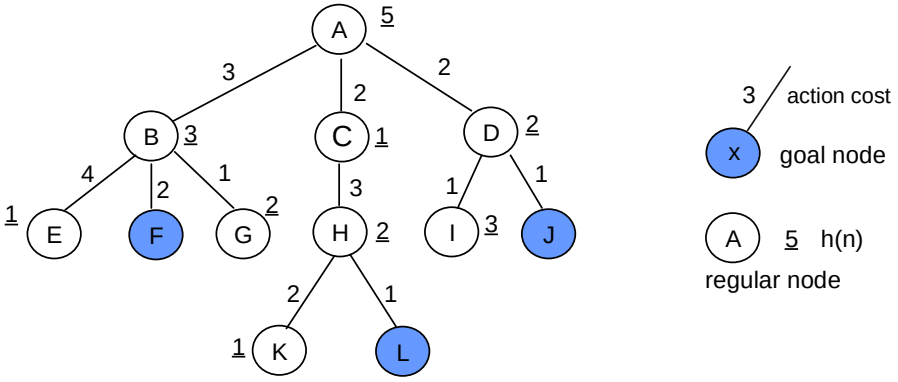
---

27) Regarding the number of nodes in the worst-case scenario for an Iterative Deepening search that finds the solution at level d and a limited Depth-first search with m=d for one same problem, which of the following assertions is CORRECT?

A. DFS generates more nodes than ID
B. ID generates more nodes than DFS
C. DFS and ID generate the same number of nodes
D. None of the above

---

28) Given the search space of the figure, if we apply a search of type A (f(n)=g(n)+h(n)) how many nodes are generated to find the solution?



A. 7
B. 8
C. 10
D. 12

29) If we apply a A* search in CLIPS, rules must not have a `retract` command in the RHS because:

A. Facts are retracted and so we would not be able to compute the value of $g(n)$
B. We would not be able to explore paths other than the path selected at first place
C. We would not be able to compute the optimal solution
D. None of the above

---

30) Given a search algorithm of type A, (f(n)=g(n)+h(n), indicate the **CORRECT** statement:

A. If $h(n)$ is consistent (and admissible), the algorithm will always expand fewer nodes than an uninformed search
B. If $h(n)$ is consistent (and admissible), the algorithm will always expand fewer nodes than when $h(n)$ is not consistent
C. Whether or not $h(n)$ is admissible, the same solution will always be found
D. None of the above

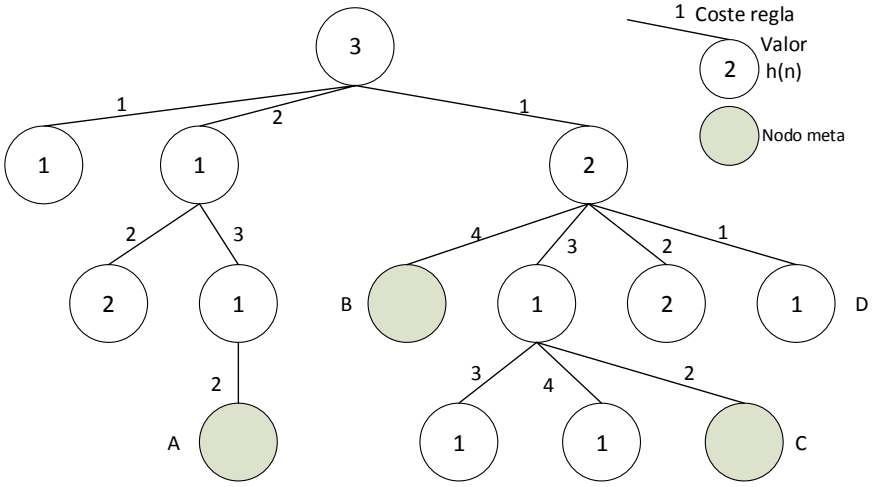
---

31) We have a search problem where operators have different cost. The application of the GRAPH-SEARCH version of a Uniform Cost algorithm with control of repeated states returns a solution of cost 'c' at the depth level 'd'. Mark the **CORRECT** answer.

A. The user does not need to set a maximum depth limit in order to prevent the algorithm from getting stuck in an endless loop
B. A solution of cost c' such that $c' > c$ can only be found at a level d' such that $d' > d$.
C. The application of an Iterative Deepening algorithm on the same problem will always find the optimal solution.
D. The application of a Breadth search on the same problem will always find the optimal solution.

---

32) If we apply an A algorithm over the state space of the figure below, which goal node (shadowy node) will be found?



A. A
B. B
C. C
D. D

---

33) Given the search space of the above question, an A algorithm that uses the shown values is:

A. Admissible and consistent
B. Admissible and not consistent
C. Non-admissible but consistent
D. Non-admissible and not consistent

---

34) In the application of an A algorithm with heuristic $h(n)$ to a particular initial state, the number of nodes generated to obtain the solution (show the **CORRECT** answer):

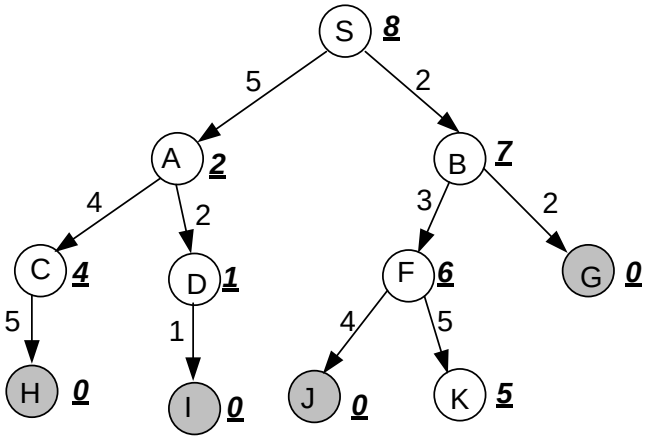
A. It will depend on the effective branching factor of the heuristic function $h(n)$ and the depth of the optimal solution.
B. It will depend on the cost of the operators.
C. If $h(n)$ is admissible, the number of nodes will never be higher than the number of nodes generated with a heuristic $h'(n)$ such that $h'(n)=h^*(n)$
D. None of the above.

---

35) The application of the GRAPH-SEARCH version of a search of type A has found the optimal path to each expanded node if:

A. The heuristic function is admissible.
B. We apply a control of repeated states in the OPEN list.
C. The heuristic function is consistent.
D. None of the above.

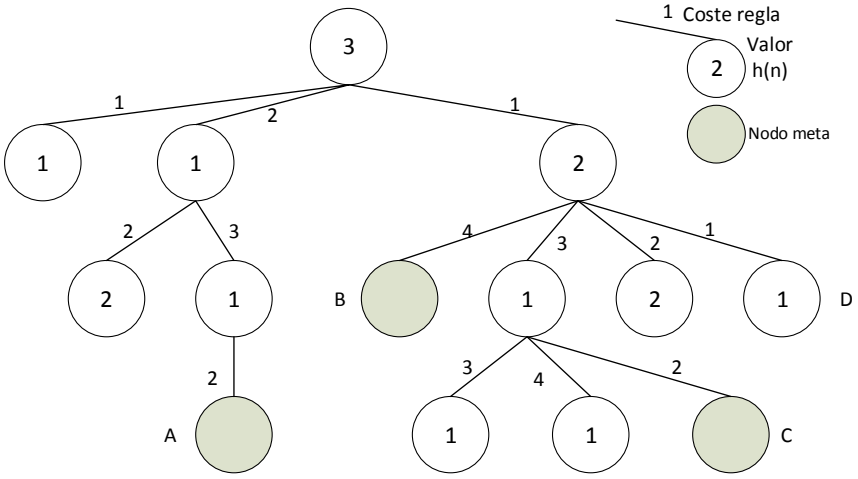
36) Given the search space of the figure where the value of the edges are the cost of the operators and the number on the right of each node is its heuristic value, mark the **CORRECT** answer:



A. An algorithm of type A will generate fewer nodes than Uniform Cost
B. An algorithm of type A will find the optimal solution.
C. A Breadth search, expanding first the leftmost nodes, will generate the same or a lower number of nodes than Uniform Cost.
D. None of the above is true.

---

37) If we apply a *Greedy* search on the search space of the figure, which goal node (shadowy node) will be selected first as a solution and how many nodes are generated to find such a solution?



A. Goal node A generating 7 nodes
B. Goal node B generating 8 nodes
C. Goal node B generating 11 nodes
D. Goal node C generating 14 nodes

38) Given the search space of the above figure, mark the **INCORRECT** statement:

A. The heuristic function h(n) is admissible
B. The heuristic function h(n) is consistent
C. A Breadth-first algorithm will find the same solution as an algorithm of type A
D. A Depth-first algorithm will find the same solution as a greedy search algorithm

---

39) Let d1, d2 and d3 be three depth levels of a search tree where d1 < d2 < d3, such that there exists one solution at level d1, another solution at level d2 and another solution at level d3 (there is only one solution at each level). Mark the **CORRECT** statement:

A. The time complexity of a Breadth-first algorithm is $O(b^{d2})$ and the time complexity of an Iterative Deepening algorithm is $O(b^{d1})$.
B. The time complexity of a Depth-first tree search with maximum depth limit m=d1, is $O(b^{d1+1})$.
C. Assuming the user selects m=d3 as maximum depth limit, a Depth-first tree search will always find first the solution at level d1 or at level d2.
D. Assuming the user selects m=d1 as maximum depth limit, the time complexity of a Depth-first tree search and an Iterative Deepening algorithm is $O(b^{d1})$ for both.

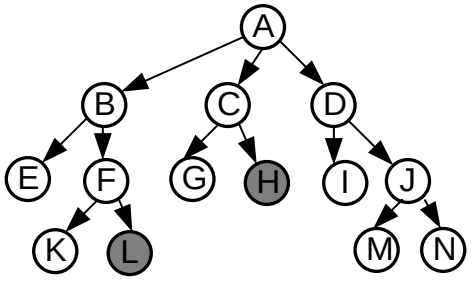
---

40) Let's assume an algorithm A* is applied to solve a problem and that *G* is the solution node found. Show the **FALSE** statement:

A. If h(n) is consistent then ∀n1, n2 such that n2 is a child node of n1, it always holds h(n2)>=h(n1)
B. ∀n1, n2, such that n1 and n2 are two nodes on the optimal path to *G*, it always holds g(n1)+h*(n1)=g(n2)+h*(n2)
C. ∀n, such that n is a node on the optimal path to *G*, it always holds f(n) <= g(*G*)
D. It always holds f(*G*)=g(*G*).

---

41) Let be the search tree of the figure, mark the answer that shows the correct order in which nodes would be generated when applying Iterative Deepening and the reached goal node.



A. ABCDEFKL and finds goal state L
B. ABCDEFGHIJKL and finds goal state H
C. AABCDABCDEFGH and finds goal state H
D. ABCDEFGH and finds goal state H

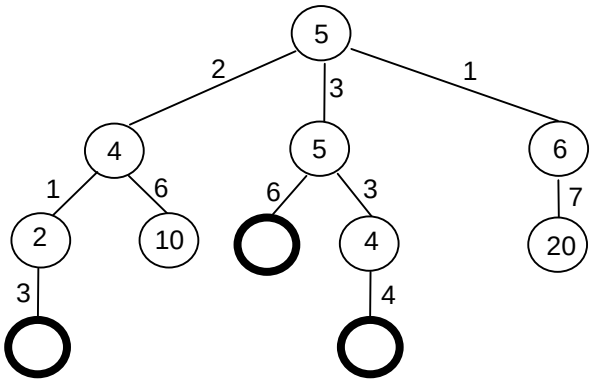
---

42) Given two A* algorithms for one same problem, where algorithm A1 uses heuristic $h1(n)$ and algorithm A2 uses $h2(n)$, such that $\forall n$, $h^*(n) \geq h2(n) > h1(n)$, show the **CORRECT** answer:

A. It is guaranteed that A1 will take less time than A2
B. It is guaranteed that A1 will expand fewer nodes than A2
C. The solution found by A2 will be better than the one found by A1
D. None of the above answers is correct

---

43) Let be the search tree of the figure, where bold-circled nodes are goal states, the value inside a node is the heuristic value of the node and the numeric value on the arcs is the operator cost. Show the **CORRECT** answer:



A. The heuristic is admissible and consistent
B. The heuristic is not admissible nor consistent
C. The optimal solution is found when an algorithm of type A is applied to the tree
D. None of the above answers is correct

---

44) Let be a search problem where operators have different costs. We know the search tree contains a solution node $G1$ at level $d1$, and a solution node $G2$, which is an optimal solution, at level $d2$. We also know that $d2 > d1$. Show the **CORRECT** answer:

A. The time complexity of a breadth-first strategy with respect to the number of generated nodes is $O(b^{d1})$
B. A depth-first strategy will never return the solution $G1$
C. An iterative deepening strategy will never find the solution $G1$
D. A uniform-cost strategy will always find the solution $G2$

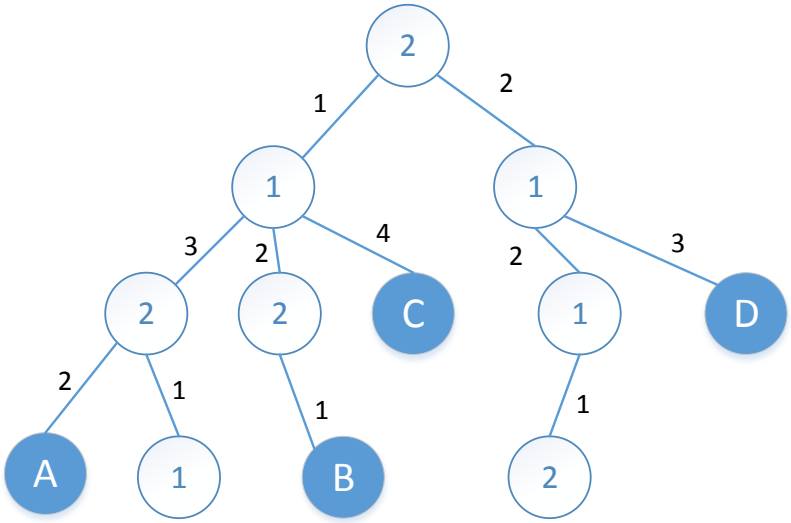
---

45) Let be the search tree generated with an A* algorithm. The tree contains two nodes, **n1** and **n2**, which correspond to two repeated states. We also know that **n1** is a node on the optimal path to a solution node *G*, whereas **n2** is not on the optimal path to *G*. Show the **INCORRECT** answer:

A. Se cumple siempre $f(n1) \leq f(G)$
B. Se cumple siempre $g(n1) < g(n2)$
C. Se cumple siempre $h(n1) < h(n2)$
D. Se cumple siempre $h(n2) \leq h^*(n2)$

---

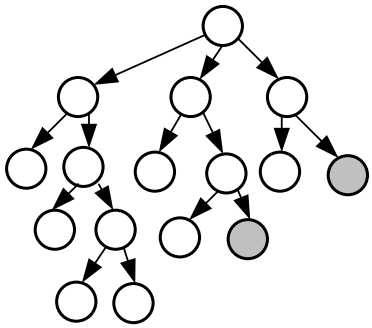
46) Let be the search tree of the figure, where the value inside the node is the heuristic value of the node and the numeric value on the arcs is the operator cost. The nodes which are labeled as A, B, C and D denote goal nodes. If we apply an A-type search strategy, which goal node is found at first place?



A. A
B. B
C. C
D. D

---

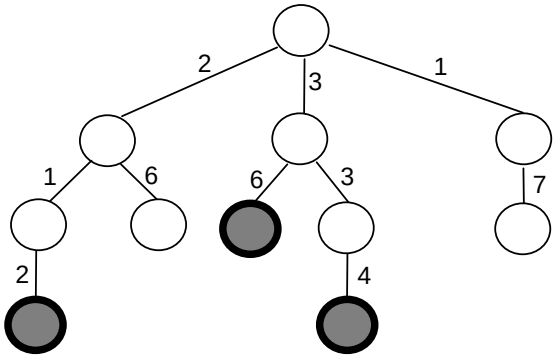
47) If we apply an Iterative Deepening (ID) search strategy to the tree shown below, which is the maximum number of nodes kept in memory? (Assume we expand first the leftmost node among the nodes that are at the same depth level)



A. 6
B. 3
C. 4
D. 5

---

48) The figure shows a search tree where the shadowy nodes are goal nodes. Show the **CORRECT** answer:



A. The application of a Breadth-first search returns the same solution node as the application of Uniform-cost search.
B. The application of a Breadth-first search returns the same solution node as the application of Depth-first search with depth limit m=2.
C. The application of a Breadth-first search returns the same solution node as the application of Depth-first search with depth limit m=3.
D. The application of a Uniform-cost search returns the same solution node as the application of Iterative Deepening search.

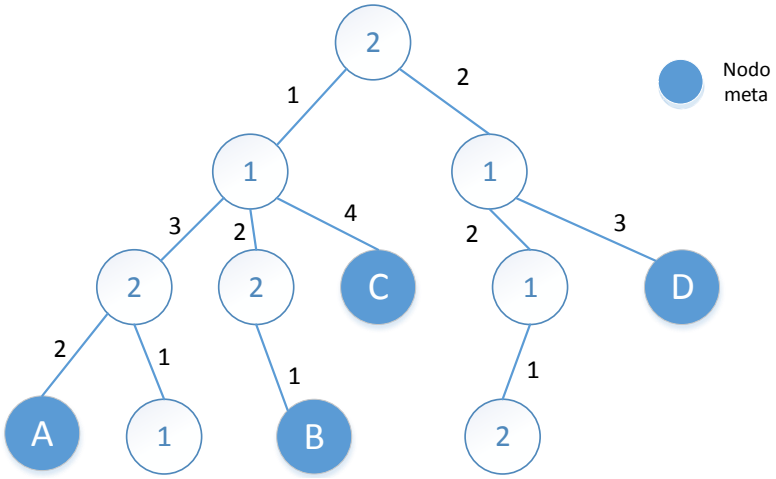
---

49) The application of an admissible heuristic, $h1$, to a problem returns a solution node $G1$ and the number of expanded nodes is $n1$. The application of an admissible heuristic, $h2$, to the same problem, where $h2$ dominates $h1$, returns a solution node $G2$ and expands $n2$ nodes. Mark the **CORRECT** answer:

A. It holds that $g(G1) < g(G2)$
B. It holds that $h1(G1) < h2(G2)$
C. It holds that $n1 < n2$
D. None of the above answers is correct.

---

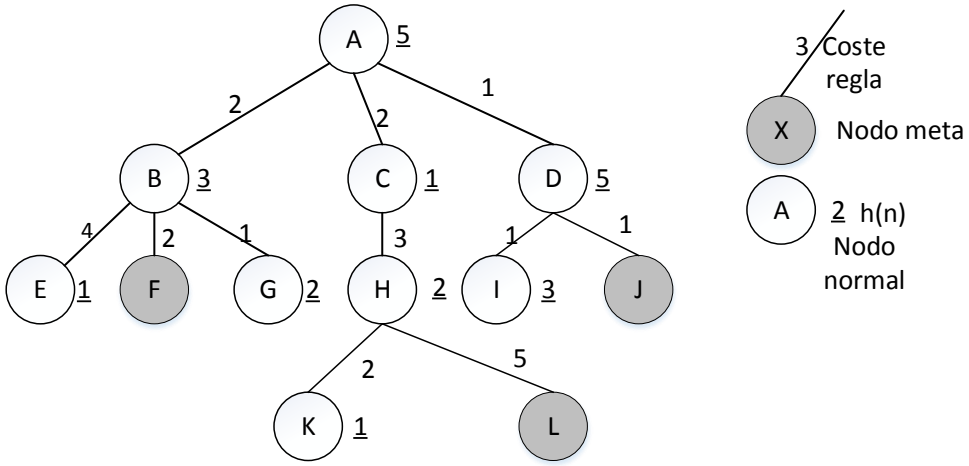
50) Given the below tree, how many nodes would be generated (including the root node) if we apply an A algorithm? (in case of nodes with the same value of f(n), expand the leftmost node).



A. 6
B. 8
C. 9
D. 10

---

51) Given the state space of the figure, if we apply a greedy search, which is the goal node (shadowy node) that will be returned as first solution? (in case of nodes with the same value of f(n), expand the leftmost nodes)



A. L
B. J
C. I
D. F

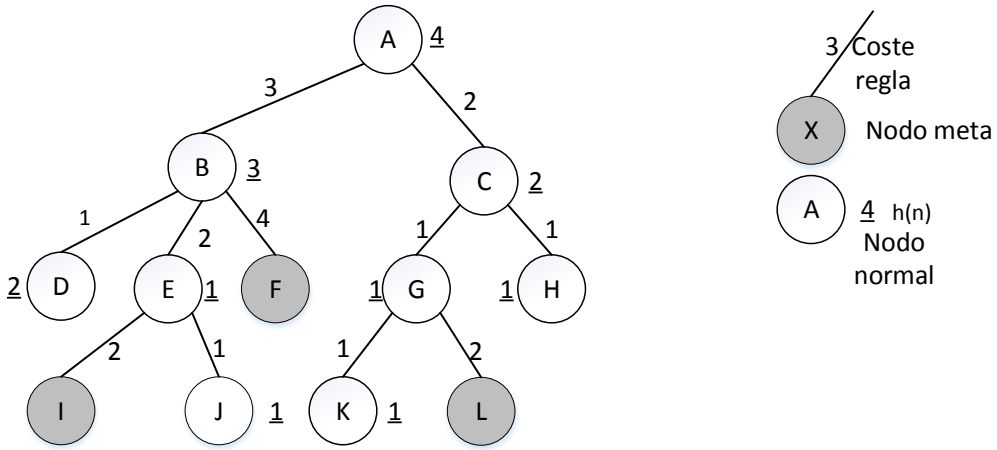
---

52) Given two evaluation functions $f1(n)=g(n)+h1(n)$ and $f2(n)=g(n)+h2(n)$, such that $h1(n)$ is admissible and $h2(n)$ is not admissible. Show the **CORRECT** answer:

A. If both functions are used in an algorithm of type A, they guarantee the optimal solution
B. Only in the case that $h1(n)$ is a consistent heuristic, $f1(n)$ will generate a smaller search space than $f2(n)$
C. It exists a node n for which it holds $h2(n)>h^*(n)$
D. We can affirm that $f2(n)$ guarantees to find a smaller search space than $f1(n)$

---

53) Assuming we apply a search of type A ($f(n)=g(n)+h(n)$), in the state space of the figure, which of the following assertions is **FALSE?**:



A. $h(n)$ is admissible
B. The search process expands 3 nodes
C. The search process generates 7 nodes
D. The solution returned is node L

---

54) Given a search problem where all operators have the same cost, mark the **CORRECT** statement:

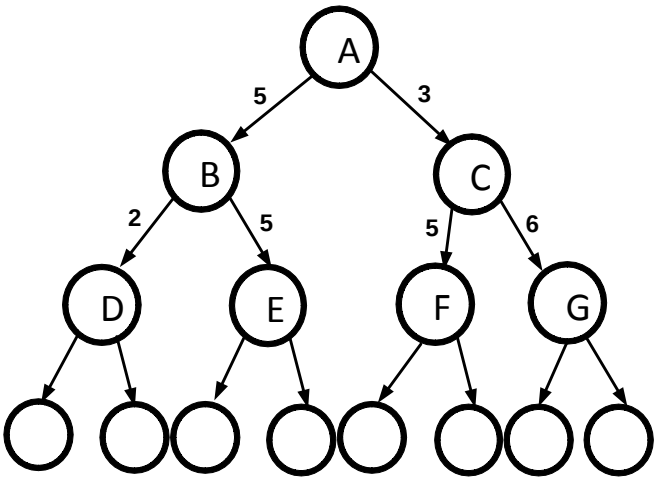
A. An algorithm of type A that uses an admissible heuristic will return the shortest solution
B. A breadth-first strategy will return the shortest solution but not the least cost solution
C. A uniform cost strategy will return the least cost solution but not the shortest solution
D. A depth-first search strategy will always return the least cost solution

EXERCISES
(open answer questions)

Exercise 1

Given the search space in the figure, say the order in which nodes will be expanded according to the following search strategies. Leaf nodes are not expanded. Values in the branches represent the action cost. The heuristic values of the nodes are as follows: h(B)=17, h(c)=14, h(D)=13, h(E)=11, h(F)=10, h(G)=7. Show the OPEN and CLOSED lists.

- **a)** Breadth-first
- **b)** Uniform cost
- **c)** Depth-first (expands the leftmost node first)
- **d)** Iterative deepening (one depth-level deeper at each iteration)
- **e)** Best-first (greedy)
- **f)** A algorithm



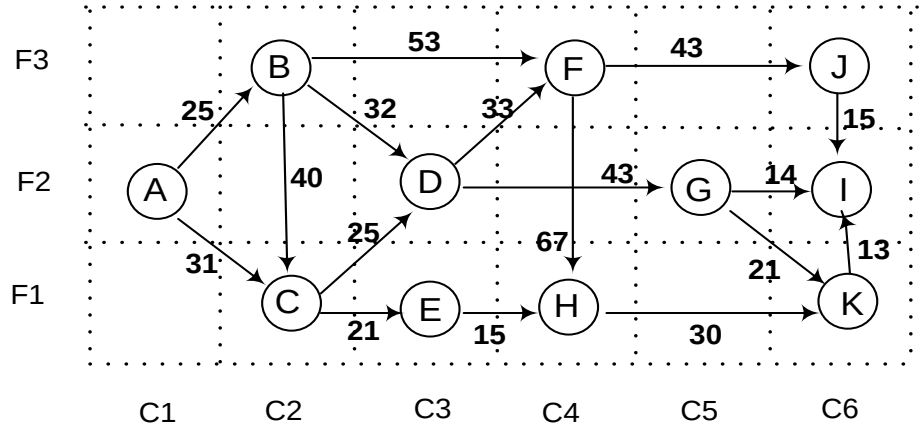
Solution:

Depth-first: {A, B, D, E, C, F, G}
Breadth-first: {A, B, C, D, E, F, G}
Iterative Deepening (increase the maximum depth in one at each iteration:
{A, A, B, C, A, B, D, E, C, F, G}
Uniform cost: {A, C, B, D, F, G, E}
Greedy best-first: {A, C, G, F, B, E, D}
A algorithm: {A, C, G, F, B, D, E}

Exercise 2

In the graph below, nodes represent cities and the edge labels are the real distance (in km) between pairs of cities. The graph is displayed in a map where each city is placed in a quadrant map. The initial state is A and the goal state is I.

Let's suppose we use the heuristic Manhattan-distance (same application as in the 8-puzzle problem from a starting node to a final node through the quadrants in the map; for example, the Manhattan-distance between city A and city E is 3; the Manhattan-distance between city E and I is 4). Let h(n)=Distancias_Manhattan(n)*10; the operators cost is the real distance in km. between pair of cities (edges labels). Answer the following questions by providing **APPROPRIATE JUSTIFICATIONS** to all your answers. Show clearly the creation of the OPEN list in each question.



1) We want to find a solution to go from city A to city I that traverses the minimum possible number of cities. Pick the appropriate strategy and draw the tree to find the solution path. In case of two nodes with the same value of the evaluation function, expand first the node that comes alphabetically before. Avoid repeated states.

2) We want to find a solution to go from city A to city I. Show the search tree that results from applying an algorithm type A (f(n)=g(n)+h(n)). Which solution will this algorithm find? Is it an A* search? Why? Avoid repeated states.

3) Consider again the same problem of going from city A to city I. Show the search tree which results from applying a depth-first expansion up to maximum depth m=4. Assume that nodes expanded first are those that come alphabetically before. Which solution will this search find? How many nodes are generated and how many expanded? Avoid repeated states (discard deeper nodes).

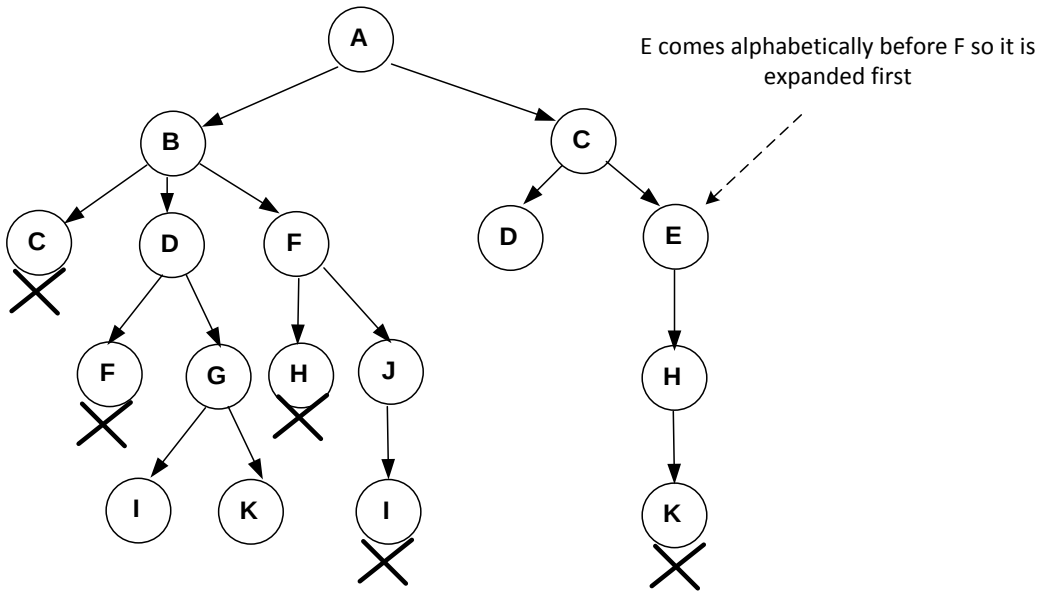
4) Consider now the problem of going from city A to any city in column C6. Show the search tree which results from applying a uniform cost search. Which city in C6 is reached first? What is the solution cost? How many nodes are generated and how many nodes are expanded? Avoid repeated states.

5) Consider now the problem of going from city A to any city in column C6. Show the search tree which results from applying a greedy best-first search. Which city in C6 is reached first? What is the solution cost? Avoid repeated states.

Solution:

Question 1

The appropriate strategy to solve this question is breadth-first.



Breadth-first always expands the shallowest node (f(n)=level(n)). Since breadth-first search is not particularly guided by cost, we should be given a criteria to avoid repeated states. It could be keeping the node with the lowest cost, or keeping the shallowest node, or any other criteria. We will apply the following criteria to avoid repeated states: discard always the most recent node.

The path returned by breadth-first is A-B-D-G-I

Question 2

Iteraciones del algoritmo. Entre paréntesis se muestra el valor f(n) del nodo.

1) OPEN ={A(50}) se extrae y expande el nodo A
2) OPEN={B(75), C(81)} se extrae y se expande B
3) OPEN={C(81), D(87), F(108)}

El nodo C(115) no se introduce en la lista porque ya existe un camino mejor hasta dicho nodo y solo nos interesa encontrar una solución. Además, al tratarse de un algoritmo A* monótono cuando se expanda C(81) sabemos que no puede existir camino mejor desde el nodo raíz hasta el nodo C.
Se extrae y se expande el nodo C(81).

a) OPEN={D(86), E(92), F(108)}
En esta iteración del algoritmo se genera un nuevo camino hasta el nodo D(86) con un coste menor que el que existe en la lista OPEN; eliminamos el nodo D(87) e insertamos el nodo D(86) ya que solo nos interesa encontrar una solución.
Se extrae y expande el nodo D(86).

b) OPEN={E(92), F(108), G(109)}
Al expandir D se genera un nuevo camino hasta F que no es mejor que el ya existente. No se introduce este nuevo camino encontrado por las razones expuestas anteriormente.
Se extrae y expande el nodo E(92).

c) OPEN={H(97), F(108), G(109)} Se extrae y expande el nodo H(97).

d) OPEN={K(107), F(108), G(109)} Se extrae y expande el nodo K(107).

e) OPEN={F(108), G(109), I(110)} Se extrae y expande el nodo F(108).

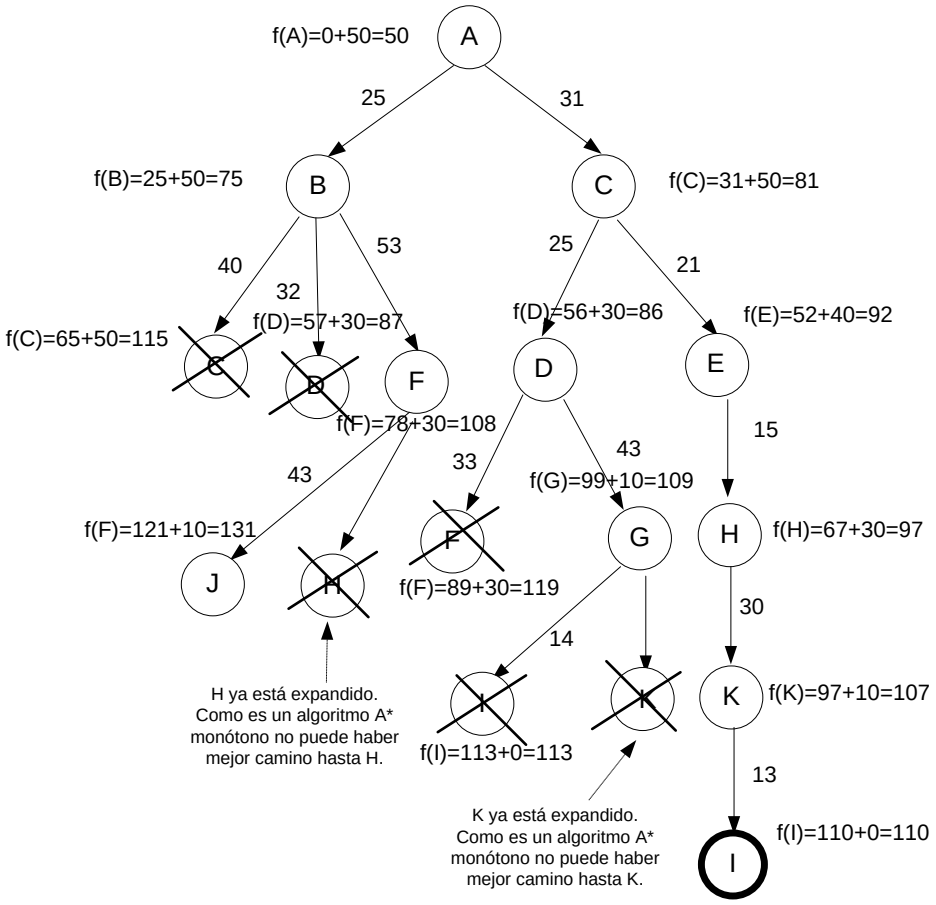
f) OPEN={G(109), I(110), J(131)}

El nodo H ya está expandido por tanto no es necesario considerar su evaluación ya que se trata de un algoritmo A* monótono.
Se extrae y expande el nodo G(109)

g) OPEN={I(110), J(131)}
El nodo K ya está expandido por tanto no es necesario considerar su evaluación ya que se trata de un algoritmo A* monótono. Se encuentra un nuevo camino hasta el nodo I(113) que no es mejor que el que existe en la lista OPEN y, por tanto, no se introduce en la lista. Se extrae y expande el nodo I.

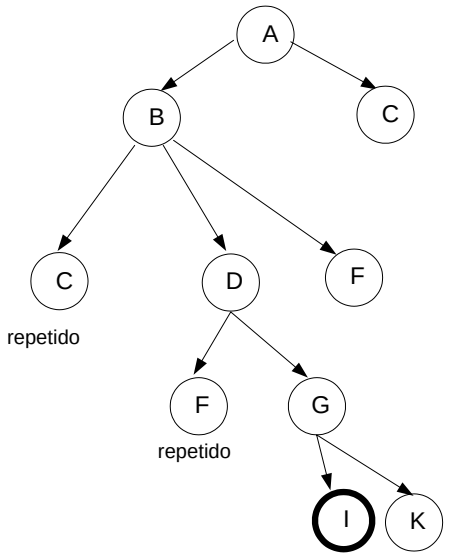
SOLUCIÓN: A-C-E-H-K-I
Coste: 110

Se trata de un algoritmo A* porque todas las estimaciones entre pares de ciudades son menores que el coste real entre ellas. También es un A* monótono porque se cumple que h(padre) <= h(hijo)+ coste(padre, hijo). Por esta razón, no es necesario considerar la expansión del nodo D(87) ni la evaluación del nodo K como hijo del nodo G.



Question 3

Avoiding repeated states



OPEN={A} se extrae y se expande el nodo A

OPEN={B, C} se extrae y se expande el nodo B. La expansión de B genera de nuevo el nodo C, que ya está en la lista OPEN. Como g(C)=1 para el nodo que está en la lista OPEN, y g(C)=2 para el nodo C que es hijo de B, nos quedamos con el nodo C de coste 1.

OPEN={D,F,C} se extrae y se expande el nodo D y se generan F y G. Como ya existe un nodo F en OPEN de menor coste, nos quedamos con éste nodo y no insertamos el hijo F de D.

OPEN={G,F,C} se extrae y se expande el nodo G hijo de D

OPEN={I,K,F,C}
Se extrae el nodo I, se expande y se comprueba que es SOLUCION.

SOLUCIÓN: A-B-D-G-I
Coste: 114
Se generan 10 nodos y se expanden 5 nodos.

Question 4

Uniform cost f(n)=g(n)
Avoiding repeated states in OPEN and CLOSED lists

CLOSED={} OPEN={A(0)}

CLOSED={A(0)} OPEN={B(25),C(31)}

CLOSED={A(0),B(25)} OPEN={C(31),D(57),F(78)} ;; we don't insert the node C child of B because we already have in OPEN list a better instance of C with cost 31

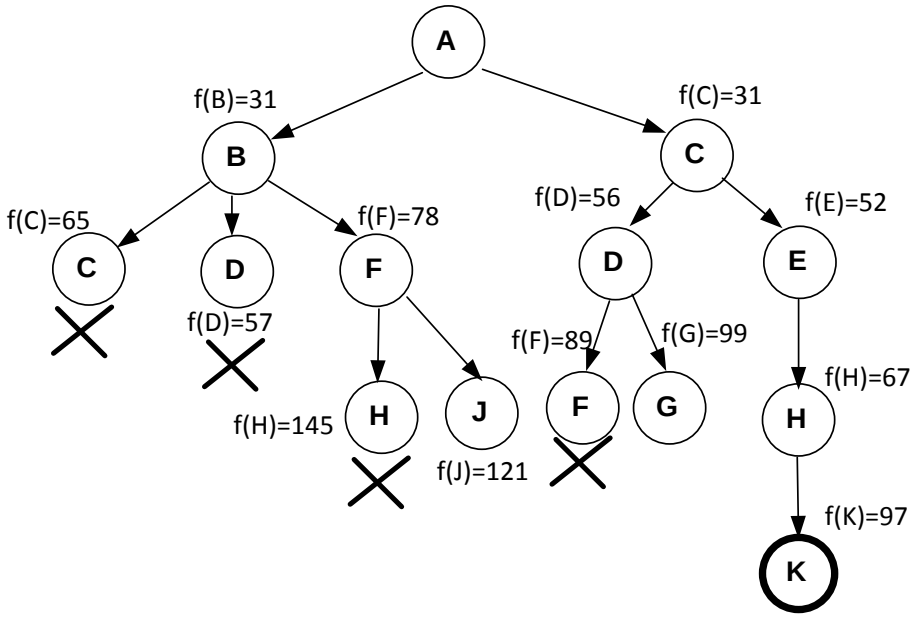
CLOSED={A(0),B(25),C(31)} OPEN={E(52),D(56),F(78)} ;; when expanding C we find another instance of node D with a better cost (56). We thus replace the instance of node D in the open list (D(57)) by this new finding.

CLOSED={A(0),B(25),C(31),E(52)} OPEN={D(56), H(67),F(78)}
CLOSED={A(0),B(25),C(31),E(52),D(56)} OPEN={H(67),F(78), G(99)} ;; when expanding node D(56) we also reach node F with f-value=89; since we have a better instance of node F in the OPEN list, we don't insert the newly found node.

CLOSED={A(0),B(25),C(31),E(52),D(56),H(67)} OPEN={F(78), K(97),G(99)}

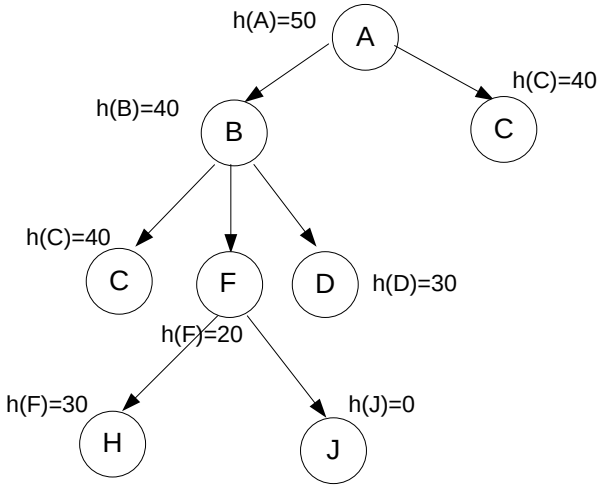
CLOSED={A(0),B(25),C(31),E(52),D(56),H(67),F(78) } OPEN={K(97),G(99),J(121)}
When we expand F(78) we reach H again, which has already been expanded because it is in the CLOSED list. The path cost to the already expanded node is better than the new node.

CLOSED={A(0),B(25),C(31),E(52),D(56),H(67),F(78) } OPEN={K(97),G(99),J(121)}
We remove K(97) from the OPEN list and we reach a solution.



Question 5

Avoiding repeated states



OPEN={A(50)} Se extrae y expande el nodo A
OPEN={B(40), C(40)} Podríamos expandir cualquiera de los dos. Extraemos y expandimos B.
OPEN={F(20), D(30), C(40)}
Ya existe un nodo C con valor 40, así que no se vuelve a insertar. Se extrae y expande el nodo F.
OPEN={J(0), H(30), D(30),C(40)} Se extrae y expande el nodo J. SOLUCIÓN.

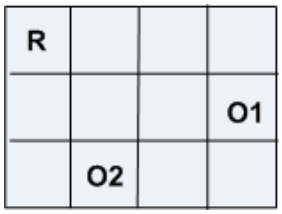
SOLUCIÓN: A-B-F-J Coste: 121

Si se expande antes el nodo C las soluciones podrían ser:

1) A-C-D-G-I Coste: 113
2) A-C-E-H-K Coste: 97

Exercise 3

We wish to find the route followed by a robot (R) to pick up two objects located in a room (O1 and O2) as indicated in the figure. The robot can only move vertically and horizontally. In each movement it only moves one square. Whenever the robot arrives to a square that contains an object, he takes it. The cost of each movement is 1.



Let $h(n)=\Sigma d_j$ be a heuristic function where d_j is the distance in movements of the robot to the object 'j'. That is, the heuristic function is the sum of the Manhattan distance of the robot 'R' to each remaining object in the room. Consider that, when the robot takes an object, the distance to that object becomes 0 and that the object moves with the robot once collected. The cost of each movement is 1.

1) Draw the search tree resulting of applying a breadth-first algorithm. Depict only the first three levels. Avoid repeated states.

2) Draw the search tree resulting of applying an A-algorithm (f(n)=g(n)+h(n)) to find the route the robot must follow to gather the two objects. Show clearly the value f(n) of each node, the order of the node expansion, and the OPEN list at each iteration. Which solution, and with which cost, is found with this algorithm? The heuristic function 'h', is it an A* heuristic? Why? Justify your answers.

3) Following the tree of section 2, the application of a greedy search, which solution will it find? Is it an optimal solution? Why? Justify your answers.

NOTES for the node expansion:

a. *Note 1: Apply movements in this order to create the search trees: first, move UP; second, move DOWN; third, move RIGHT; finally, move LEFT.*
b. *Note 2: If two nodes have the same f-value, expand first the **deepest** node. If both are at the same level, expand first the **oldest** node.*
c. *Note 3: Avoid repeated nodes.*

Solution:

Question 2

Se muestra el estado de la lista OPEN en cada iteración de acuerdo a la figura que se puede ver más abajo. Los valores entre paréntesis muestran el valor f(n) del nodo correspondiente.

1) OPEN={S0(7)}
2) OPEN={S1(6), S2(6)}
3) OPEN={S4(5), S2(6), S3(7)}
4) OPEN={S5(6), S6(6), S2(6), S3(7)}
5) OPEN={S7(6), S8(6), S6(6), S2(6), S3(7), S9(8)}

6) OPEN={S11(6), S8(6), S6(6), S2(6), S3(7), S10(8), S12(8), S9(8)}
7) OPEN={S14(6), S8(6), S6(6), S2(6), S3(7), S13(8), S10(8), S12(8), S9(8)}
8) Se extrae de la lista S14 que es un nodo solución

La solución que encuentra el algoritmo A es:

ABAJO-DERECHA-ABAJO-ARRIBA-DERECHA-DERECHA

que tiene un coste de 6 pasos.

Como se puede observar, el algoritmo ha encontrado la solución óptima (6 pasos). Sin embargo, la ***heurística no es admisible*** porque se puede encontrar algún caso en el que la función 'h' sobreestima el coste real. Véase por ejemplo el estado inicial S0, donde la heurística estima una distancia de 7 siendo el coste óptimo 6. Otro ejemplo se puede ver en S3. La heurística estima una distancia total de 5 cuando el coste óptimo a partir del estado S3 es 4 (un paso para recoger O2 y tres pasos más luego para recoger O1).

La razón de que la heurística no sea A* es que asume que recoger un objeto es independiente de recoger el otro objeto, es decir que el plan de acciones para recoger un objeto es independiente del plan de acciones para recoger el otro objeto, razón por la cual suma la distancia del robot R a cada uno de los objetos. Sin embargo, esto no es así, y la heurística no está teniendo en cuenta que R irá a recoger un segundo objeto a partir de la posición donde se haya quedado tras recoger el primer objeto.

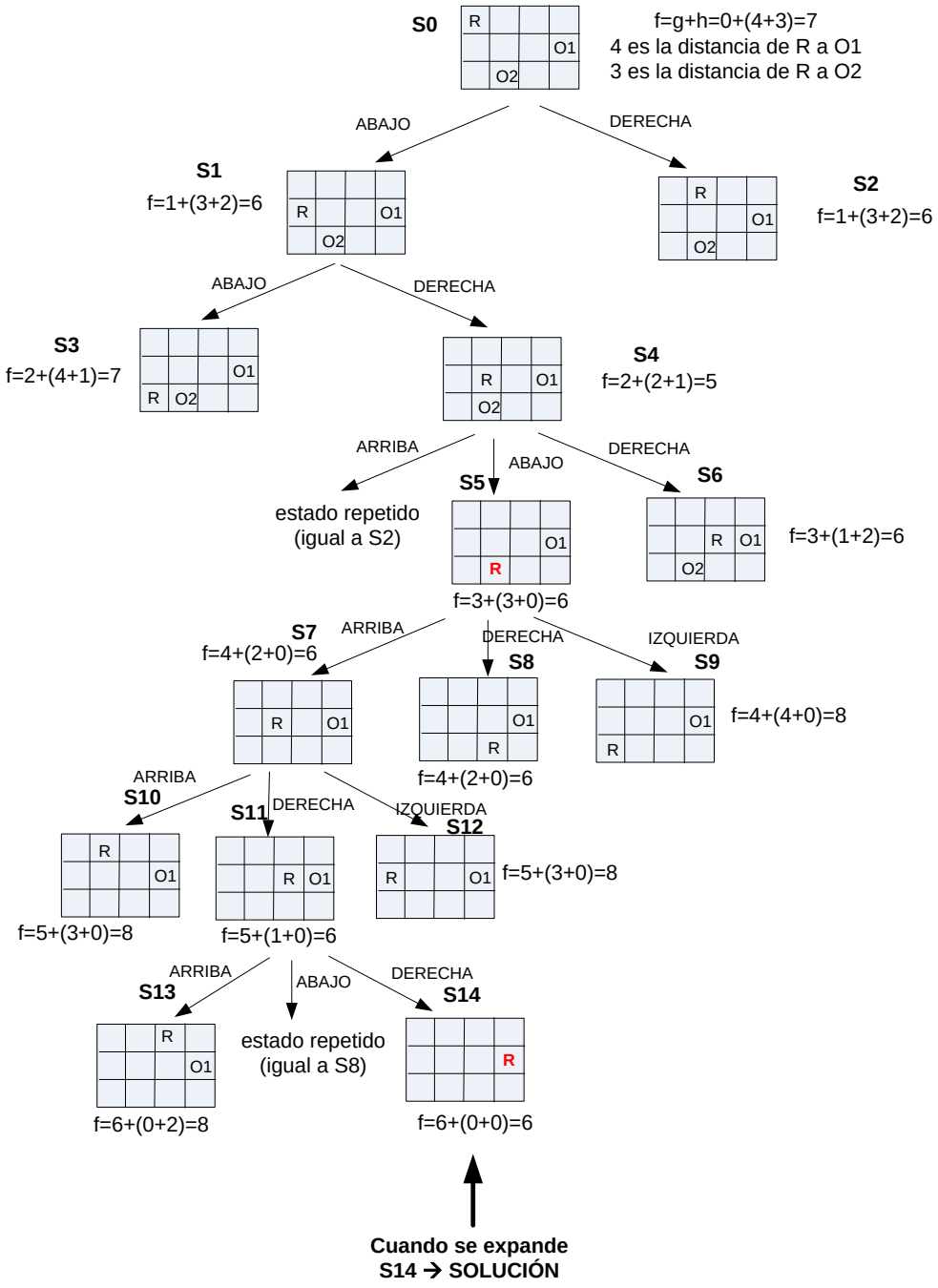
Question 3

Un algoritmo de búsqueda primero el mejor se guía únicamente por el valor h(n), y encontrará exactamente la misma solución que en el apartado A). Por tanto, misma solución y mismo coste. Concretamente, el proceso de búsqueda sería del siguiente modo:

a) Se expande S0. Se generan S1 y S2. Ambos tienen el mismo valor heurístico. Se escoge, por tanto, S1.
b) Se expande S1. Se generan S3 y S4. S4 es el nodo que tiene menor valor heurístico de todos los nodos abiertos (h(S4)=3).
c) Se expande S4. Se generan S5, S6. h(S5)=h(S6)=3 es el menor valor heurístico entre todos los nodos abiertos. Se escoge S5.
d) Se expande S5. Se genera S7, S8 y S9. h(S7)=h(S8)=2. Se escoge S7.
e) Se expande S7. Se genera S10, S11 y S12. h(S11)=1. Se escoge S11.
f) Se expande S11. Se genera S13 y S14. h(S14)=0. Se escoge S14.
g) Se expande S14 → SOLUCIÓN

NOTAS:

a. No se han dibujado los estados repetidos que implican un ciclo en la misma rama. Se han indicado algunos estados repetidos con nodos de otras ramas, aunque no es necesario pintarlos.
b. Nótese que el estado S7 no es igual que el S4, y por tanto no es repetido. Aunque la posición del robot es la misma, los estados son distintos porque en S4 el robot no tiene el objeto O2, y en el estado S7 sí.



Exercise 4

Given two admissible heuristic functions (h1,h2), compare the following compound functions and respond which of them will generate the lowest number of nodes: (i) max (h1,h2), (ii) min (h1,h2), (iii) h1+h2.

Solution:

La heurística h1+h2 no hay garantías de que sea admisible y será la que genere menos nodos porque está mucho más informada. Las otras dos heurísticas seguirán siendo admisibles y max (h1,h2) generará menos nodos ya que devuelve la función más informada entre h1 y h2. Ordenando las funciones de menor a mayor número de nodos generados tenemos: h1+h2, max (h1, h2), min (h1, h2).

Exercise 5

Consider an admissible A* algorithm with f(n)=g(n) + h(n) where the cost of each operator is 1. If g(n) changes and operators have now different cost (non-uniform cost), is the algorithm still A*?

Solution:

Si siendo el coste de cada operador 1 se cumple que h(n) <= h*(n), entonces para que se siga cumpliendo esta relación el coste de los operadores debería ser >=1, ya que sino no hay garantía de que siga siendo menor la estimación. Dicho de otro modo, si cambia el coste de los operadores, cambia entonces h*(n) (coste real de la solución), y para asegurar que se sigue cumpliendo h(n) <= h*(n), h*(n) debe crecer, y eso sólo sucede si el coste de los operadores es mayor o igual que el coste anterior.

Exercise 6

In order to solve a problem as a search process we have used three different operators O1, O2 and O3 whose costs are 10, 20 and 30 respectively. The search program asks the user for the maximum depth level of the tree expansion and the program returns the level where the solution has been found, the number of generated nodes and the solution cost. We know that for a maximum depth level equal to five (D=5), breadth-first strategy has found a solution at depth level 2 and the cost of the solution is 50 (there only exists a solution at level 2). According to these data:

a) Is the found solution the optimal one?. Otherwise, which strategy would you use to find the optimal solution?
b) If the user selects D=8 and breadth-first strategy, what can you say about the solution returned by the program? Compare this new solution with the one presented above in the statement.
c) If the user selects D=8 and iterative-deepening strategy, what can you say about the solution returned by the program? Compare this new solution with the one presented above in the statement.

Solution:

a) No, no se puede garantizar que sea la solución de menor coste ya que la estrategia de búsqueda en anchura sólo es admisible si todos los operadores tienen el mismo coste. En este caso, se puede decir que la solución es la de menor longitud pero no necesariamente la de menor coste.
En el caso de utilizar una estrategia no informada utilizaría BCU, ya que es admisible; en caso de una estrategia con información, utilizaría una búsqueda A*.

b) Será la misma solución ya que la estrategia de búsqueda en anchura encuentra la solución de menor profundidad en el árbol y por tanto se parará cuando encuentre la solución en el nivel 2.

c) También será la misma porque la estrategia de búsqueda por profundización iterativa parará cuando, desarrollando el árbol de nivel 2, encuentre la solución que está en dicho nivel.

Exercise 7

A particular search problem is solved with the following search strategies: breadth-first, depth-first, an A* algorithm with h1(n), another A* algorithm with h2(n) > h1(n) and a non-A* algorithm. The search program asks the user to select the depth level for the search tree and the obtained results are those shown in the table. Explain and reason which search strategy has been executed for the results obtained at each row in the table.

User depth level	Sol. depth level	Number of nodes
15	13	264
18	13	860
18	18	565

20	13	325
25	20	205

Solution:

Claramente la solución óptima está en el nivel 13. Hay por tanto tres estrategias que devuelven la sol. óptima y que deberán corresponderse con anchura y los dos algoritmos A*. Atendiendo al número de nodos tenemos:

Fila 2: anchura (mayor número de nodos, información heurística nula)
Fila 4: algoritmo A* con h1(n) (325 nodos, menos informada que h2(n))
Fila 1: algoritmo A* con h2(n) (264 nodos, más informada que h1(n))

Fila 3: podría corresponderse con profundidad o el algoritmo no A*. Atendiendo al número de nodos, es más probable que sea la estrategia profundidad ya que se trata de un número elevado de nodos.

Fila 5: podría corresponderse con profundidad o el algoritmo no A*. Atendiendo al número de nodos, es más probable que sea el algoritmo no A* ya que no encuentra la solución óptima y el número de nodos generados es inferior al del mejor algoritmo A*, lo que indica que la función heurística está sobre informada.

Exercise 8

We have a search problem whose operators have different cost (non-uniform cost). If we apply an Iterative Deepening (ID) search strategy to solve the problem, what can you say about the solution found?. Explain if ID will find a solution and if so define the solution found in terms of optimality, time complexity, space complexity, etc.

Solution:

Sí, ID encuentra solución ya que se trata de una estrategia completa. ID es una estrategia que obtiene las mismas soluciones que anchura pero con un coste espacial equivalente al de la estrategia de profundidad. Por tanto no se puede garantizar que la solución encontrada sea la óptima ya que los operadores tienen diferente coste; al igual que la estrategia de anchura, se puede garantizar que la solución encontrada será la más corta pero no necesariamente la óptima. El coste temporal será igual que el de la estrategia de anchura (b^d) y el espacial igual que el de la estrategia de profundidad (b.d) donde *b* es el factor de ramificación y *d* el nivel máximo de profundidad alcanzado.

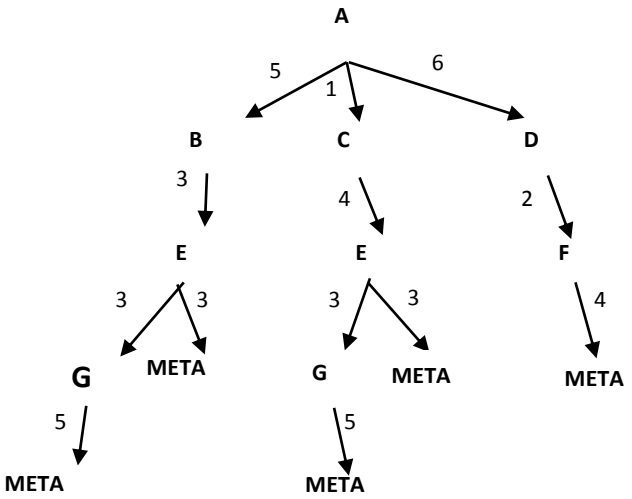
Exercise 9

Given the following search tree where values in the edges represent the cost of applying operators and the values of h(n) for each node are:

A	B	C	D	E	F	G
5	4	3	5	2	1	3

Show the search process if we apply:

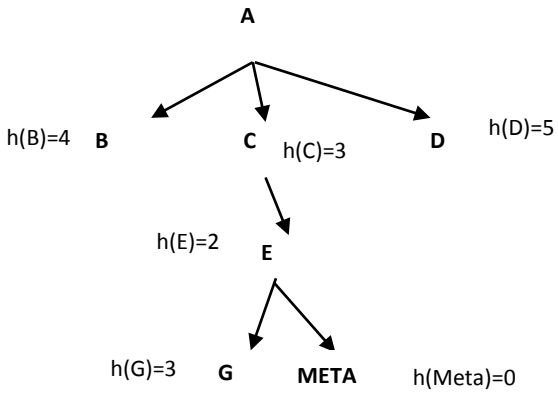
- A greedy best-first method
- An algorithm A
- According to the given values, is it an A* search?



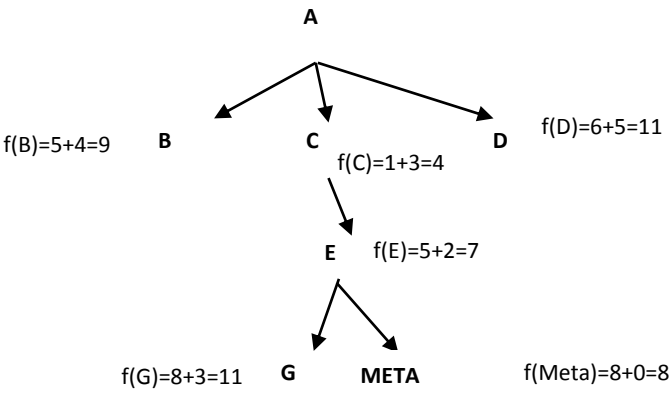
Solution:

La búsqueda greedy best-first se guía únicamente por el valor heurístico. f(n)=h(n). De acuerdo a esta función de evaluación, la expansión sería del siguiente modo:

OPEN={C,B,D} por orden creciente de h(n). Se saca C de la lista; no es solución y se expande
OPEN={E,B,D} Se saca E de la lista; no es solución y se expande
OPEN={Meta, G, B, D} se ordena primero Meta porque h(Meta)=0



Aplicando una búsqueda A, f(n)=g(n)+h(n)

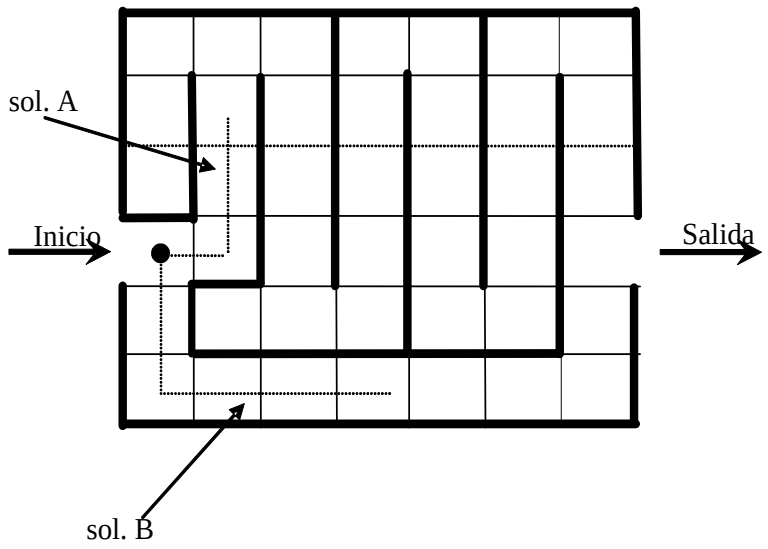


En primer lugar, la búsqueda anterior devuelve la solución óptima (hay varios caminos solución pero el camino de menor coste es el encontrado, coste = 8). En caso de no devolver la solución óptima, podríamos afirmar que no sería un algoritmo A*. Sin embargo, el hecho que devuelva la solución óptima no se garantía de que sea A*; para afirmarlo, hay que mirar los valores h(n) de cada uno de los nodos.

Se puede observar que los valores h(n) de cada uno de los nodos son siempre menores o iguales que el coste real hasta la solución. Por ejemplo, h(B)=4 y h*(B)=6, h(C)=3 y h*(C)=7, h(D)=5 y h*(D)=6, etc.

Exercise 10

Consider the maze in the figure where the goal is to find the finish location (Salida) starting from initial location (Inicio). The possible operators are UP, DOWN, LEFT and RIGHT which are only valid if there is not a thick line (bold line = obstacle) between the grid boxed in the map maze. As you can see there are two solutions for this problem (Sol. A and Sol. B).

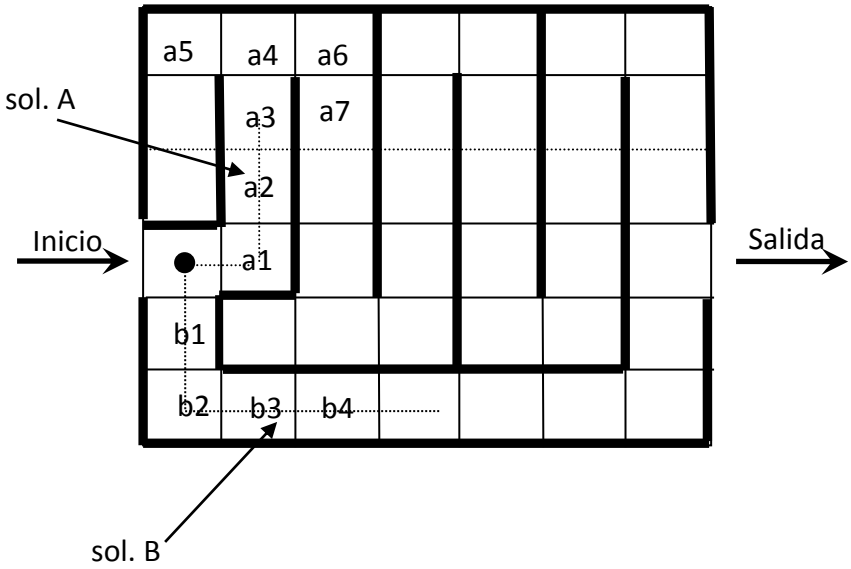


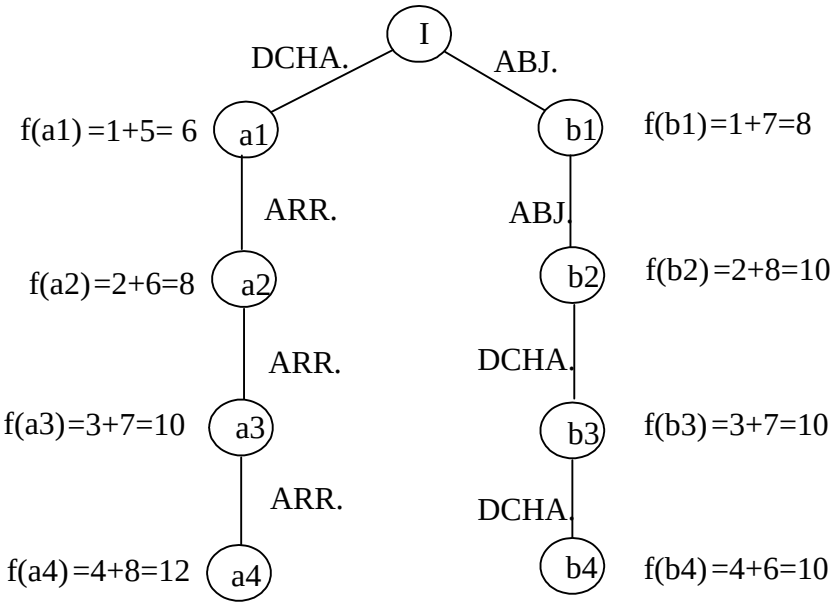
Consider that the heuristic used is Manhattan-distance (same application as in the 8-puzzle problem from the Inicio to the Salida). We assume a unit cost for each move. We assume the search process avoids direct loops. Answer the following questions by providing **APPROPRIATE JUSTIFICATIONS** to all your answers.

1) Draw the search tree that results from applying an algorithm type A (f(n)=g(n)+h(n)) and number the 7 first expanded nodes (squares) in this maze. Which solution will this algorithm find? Is it an A* search? Show your reasoning.
2) If operator RIGHT costs 2 units instead of one unit, the solution found by the algorithm A in section 1 would be the same? same solution quality and same search cost? Why?
3) If we apply a greedy search, which solution is found? Justify your answer.
4) If we apply a breadth-first search, which solution is found? Justify your answer.
5) Assume we want to apply a depth-first search. Which parameters would we have to set up to run a depth-first search that guarantees to find a solution? Indicate which solution depth-first would find under those parameters.

Solution:

1) Asumimos que los nodos que constituyen la sol. A se numeran consecutivamente como a1, a2, a3, a4, …. etc. y los nodos que conforman la sol. B se numeran consecutivamente como b1, b2, b3 … etc.





Como se puede observar a partir del desarrollo del árbol, la solución que encontrará es la B. porque dicha rama se seguirá expandiendo con un valor f=10 para el resto de nodos (un punto más de g(n) y un punto menos de h(n)) hasta llegar a la Salida. Y justamente el valor de la sol. B es 10.
Efectivamente encuentra la sol. óptima (coste=10). Es un algoritmo A* porque la estimación de las distancias es siempre <= que el coste real ya que se ignoran los posibles obstáculos que se podrían atravesar en el camino. Concretamente:

a) la estimación para las casillas de la sol. A estará muy por debajo del coste real y se irá acercando a h* a medida que las casillas están más cerca de la Salida
b) la estimación para las casillas de la sol. B será h(n)=h*(n) excepto para la casilla b1 donde h(b1) < h*(b1)

2) Si el coste del operador DERECHA vale 2 cambiarán los valores de g(n) y h*(n) pero el valor de h(n) sigue siendo el mismo. Por tanto, la solución que encuentra sería la misma (sol. B) pero el coste de la búsqueda de dicha solución sería distinto, concretamente mayor que en el apartado 1. La razón es la siguiente:

- la sol. B tendría un coste total de 16.

- como la heurística es A* (y lo sigue siendo con un coste del operador DERECHA=2) el máximo valor de f(n) que tomarían las casillas de la ruta B es por tanto 16
- las casillas de la sol. A se irán expandiendo mientras su valor sea < 16, cosa que sucede hasta la casilla a11 (f(a11)=13+4=17). A partir de la casilla a11, los valores de f(n) serán superiores a 16 y nunca se expandirán, haciendo por tanto que se expanden las casillas b_i y encontrando la sol. B

Se generan por tanto 6 nodos más que en el apartado 1). Este incremento se justifica también porque al aumentar el coste del operador DERECHA, se incrementa el coste real de la solución (h*(n)) y por tanto h(n) << h*(n), es decir, h(n) proporcionará menos información en este caso que en el apartado 1) (función heurística menos informada) y por tanto se expanden más nodos.

3) Aquí ocurre un fenómeno curioso. Como se puede observar en el árbol del apartado 1) y fijándonos solo en el valor de h(n), llega un punto en el que tendríamos en la lista OPEN dos nodos con valor h(n)=8 (b2 y a4). A partir de este punto pueden ocurrir dos cosas:

- si se decide expandir b2 entonces la solución que encontrará será la sol. B porque la heurística a partir del nodo b2 siempre decrece
- si se decide expandir a4 sucede lo mismo pero en sentido contrario, que la solución que encontrará será la sol. A, justamente por la misma razón, porque a partir de la casilla a4 los valores heurísticos de las siguientes casillas siempre serán < 8.

4) Búsqueda anchura y con costes uniformes encontrará la solución más corta, es decir la B.

5) El parámetro más importante para una estrategia en profundidad es el nivel de profundidad. Por tanto si:

- nivel-prof $\in$ [10,27] encontrará la solución B
- nivel-prof >= 28 entonces podría encontrar cualquiera de las dos soluciones. En este caso depende de qué rama expande primero: si empieza por una casilla a1 entonces encontrará la sol. A, si empieza por la casilla b1 entonces encontrará la sol. B

Exercise 11

Given the following search methods in graphs:

M1) A* algorithm, with f(n)=g(n)+h1(n)
M2) A* algorithm, with f(n)=g(n)+h2(n) / h2(n) > h1(n)
M3) A algorithm, non A*

give an answer to the these questions:

a) Which method will expand more nodes?
  • M1 or M2?
  • M1 or M3?
b) Say, if possible, which method will return the best quality solution:
  • M1 or M2?
  • M1 or M3?
c) About the total search cost, make a comparison between:
  • M1 and M2
  • M1 and M3

Solution:

a)

a.1) M1 porque la función heurística está menos informada y por tanto se generará un árbol con más nodos, más similar a una búsqueda en anchura.

a.2) No se puede saber ya que se está comparando un algoritmo A* y uno de tipo A que no es admisible. M3 generará un árbol de búsqueda más estrecho que M1, pero se desconoce la profundidad de la solución encontrada; además, M3 no garantiza admisibilidad ni completitud.

b)

b.1) Igual, ambos son A*.
b.2) M1 obtiene la óptima, M3 no lo garantiza. En el caso particular que M3 tuviera una h(n) acotada (h(n) ≤h*(n)+e), tendría la garantía de encontrar una solución acotada por (Coste-Óptimo + e)

c)

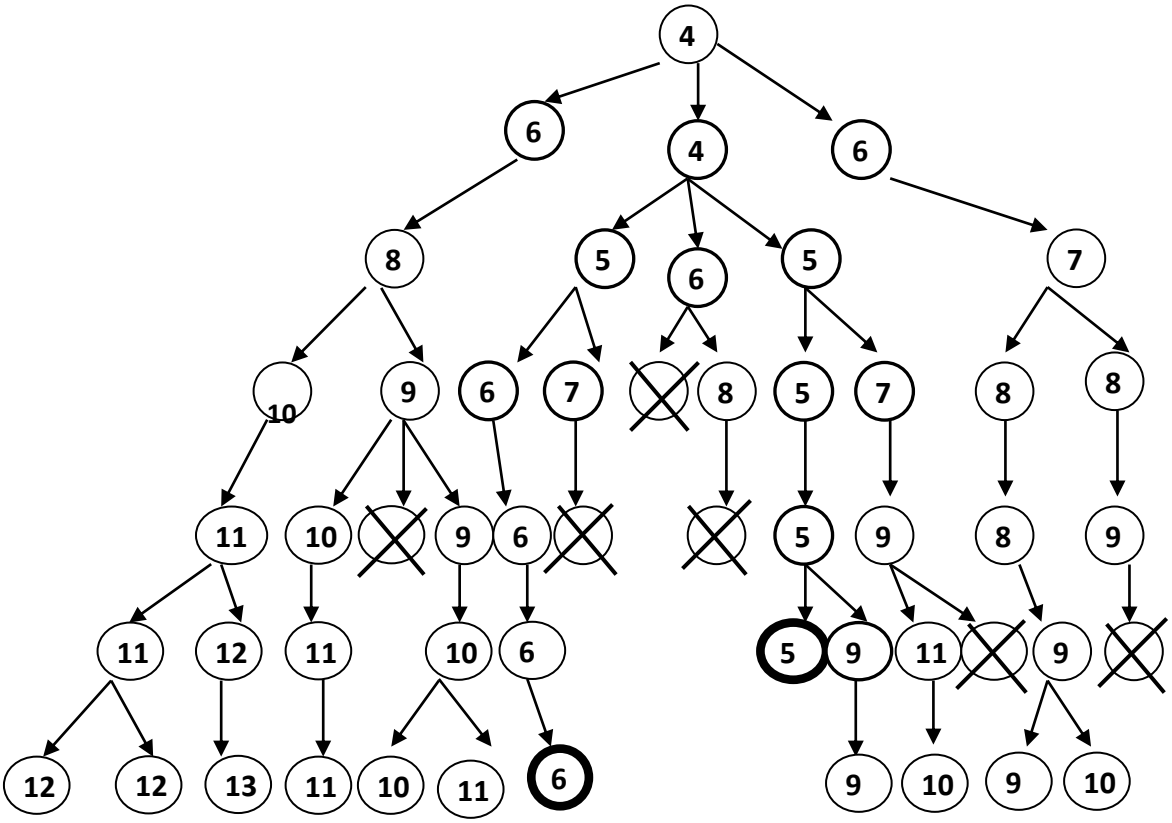
c.1) Se generan más nodos con M1 pero por el contrario h1(n) es más fácil de calcular que h(2). Dependerá por tanto del coste del cálculo de las funciones heurísticas.
c.2) No se puede saber. h(3) está muy informada, por ser un algoritmo de tipo A y no ser admisible, y por tanto costará más de calcular que h1(n). Por otra parte, no se conoce la relación exacta entre los nodos expandidos por M1 y M3 (ver a.2)

Exercise 12

The tree below represents the search space for a particular problem. Values inside the nodes are the result of applying f(n)=g(n)+h(n), h(n) <= h*(n). All actions cost are equal (1 unit cost). The two nodes in bold are goal states, and the nodes crossed out mean that no children are generated. Answer the following questions:

a) Which solution will be found if we apply the 'f' function? Which is the cost of the solution path? How many nodes have been generated to find such a solution path?

b) According to the f-values of each node, is the 'h' function consistent (A* monotone)? Why? Justify your answer.



Solution:

a) La solución que encontraría el proceso de búsqueda es la del nodo 5. Dado que el coste de cada operador es 1, el coste de dicha solución sería 5. El número de nodos generados sería 13 como máximo (incluyendo nodo meta pero no el inicial; los nodos son los señalados en negrita en el árbol). Como el nodo 4 del nivel 1 tiene dos sucesores con el mismo valor de f(n)=5, podría suceder que se expandiera antes el nodo sucesor de la derecha, en cuyo caso el número de nodos totales generados sería 11.

b) Todos los nodos del árbol satisfacen la condición h(n1) <= h(n2) + c(n1,n2) excepto el nodo 11 de nivel 6 que tiene un nodo sucesor con un valor de 10. En este caso, nodo11= g(n)+h(n)=5+6, y el nodo sucesor, nodo10=g(n)+h(n)=6+4, por tanto no se satisface 6 <= 4+1. Esto también puede observarse en que el valor f(n) en estos dos nodos no es monotónicamente no decreciente, ya que el valor de f(n) decrece. Por tanto, no se puede afirmar que sea A* monótono.

Exercise 13

Reason and give an answer to the following questions:

a) Given the following evaluation functions:

f1(n) =g(n) + h1(n)
f2(n) = g(n) + h2(n)
f3(n) = g(n) + h3(n)
f4(n) = g(n) (Uniform Cost)

and knowing that $h1(n) \leq h2(n) \leq h^*(n) \leq h3(n)$,

a.1) Order firstly the evaluation functions according to admissibility and then according to the number of generated nodes (increasing order).

a.2) Order firstly the evaluation functions according to the number of generated nodes (increasing order) and then according to admissibility.

b) Let's suppose we have a new function f5(n) = h2(n), can you ensure f5 guarantees the optimal solution generating fewer nodes than f2?

Solution:

a.1) Por admisibilidad se agruparían primero f1, f2 y f4 y después f3 (que no es admisible). Dentro de las admisibles, de mejor a peor por expansión de nodos, sería f2, f1 y f4. El orden total queda por tanto f2, f1, f4 y f3.

a.2) No se puede establecer un orden en cuanto a la generación de nodos. Sabemos que existe un orden de menor a mayor número de nodos generados: f2, f1, f4. Pero no sabemos donde estaría f3. Aunque genera un árbol de búsqueda más estrecho que los otros, no tiene la garantía de completitud.

b) En este caso sería un algoritmo búsqueda primero el mejor (voraz), donde no se aplica factor de coste, y por tanto no se garantiza la obtención de la solución óptima. De hecho, existe bastante más probabilidad de no encontrar la óptima y encontrar otra solución ya que el proceso de búsqueda va sólo guiado por la función heurística.

Exercise 14

We have two A* algorithms for the same problem with heuristic functions h1(n) and h2(n) respectively.

a) Assuming, ∀n h1(n) <= h2(n), which algorithm will find the best quality solution? Which method will generate the lowest number of nodes?

b) Assuming h1(n) is A* monotone (consistent) and h2(n) is not, which method will have the lowest branching factor?

Solution:

a) Ambos encontrarán la mejor solución porque son los dos A*. Y expandirá menos nodos el algoritmo de h2 ya que la función está más informada.

b) El algoritmo de h1 tendrá un factor de ramificación menor. La razón es que si A* es monótono se garantiza que cuando se expande un nodo se cumple g(n)=g*(n), o sea, que ese es el camino óptimo desde el nodo raíz hasta n (dicho de otro modo, si se encuentra otro nodo igual a n, no se expandirá porque ya se ha expandido el nodo de menor coste). Sin embargo, si el algoritmo no es A* monótono no se garantiza esta propiedad, por lo que podría volver a expandirse un nodo ya expandido para el cual se ha encontrado un valor de g(n) menor. Por tanto, como se pueden generar más nodos con un A* no monótono, el factor de ramificación también será mayor.

Exercise 15

Answer the following questions:

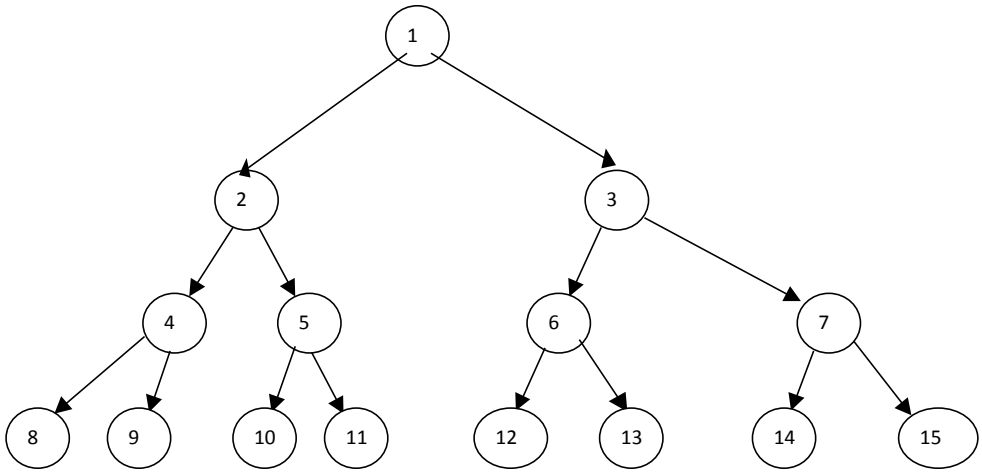
a) Consider a state space where the initial state is the number 1 and the successor function (operator) for any state returns two children: 2n and 2n+1.

1. Draw the state space for states from 1 to 15
2. Assuming that the goal state is number 11; depict the search space and the order in which nodes are expanded if we apply a breadth-first algorithm and an iterative-deepening algorithm.

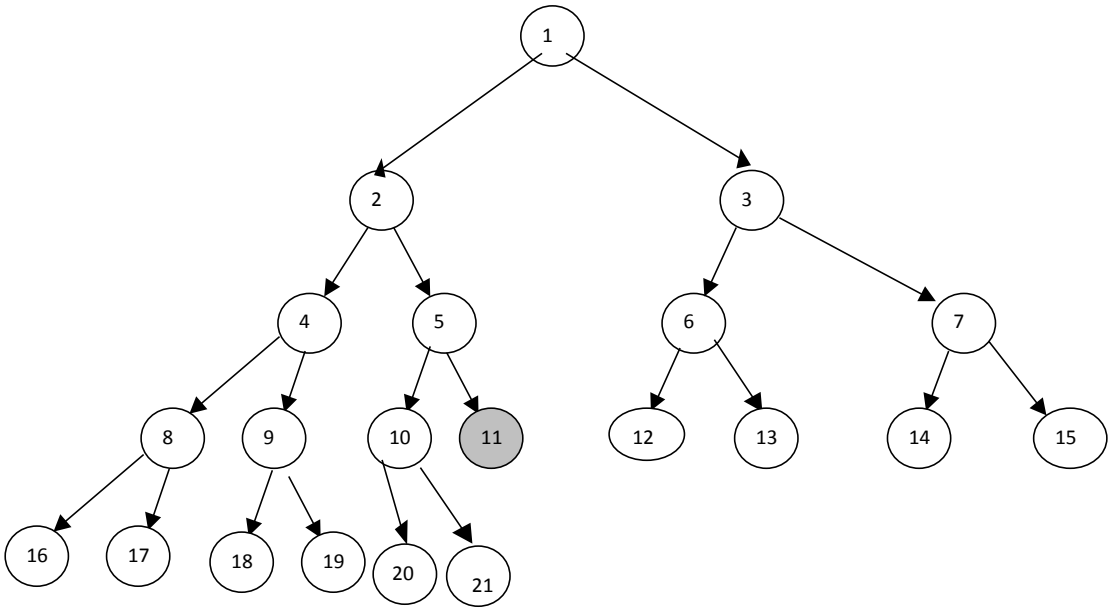
b) Given the evaluation function f(n)= (2-a).g(n)+h(n) such that h(n) ≤ h*, which values would variable 'a' take on if we wish to guarantee the optimal solution? Do you think the algorithm will find the optimal solution if a=2?

Solution:

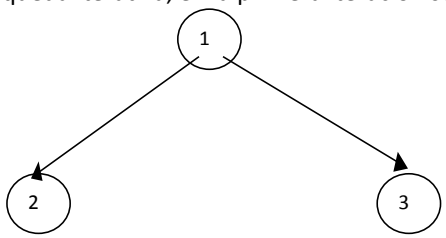
1. El espacio de estados, para los estados del 1 al 15, sería el siguiente:



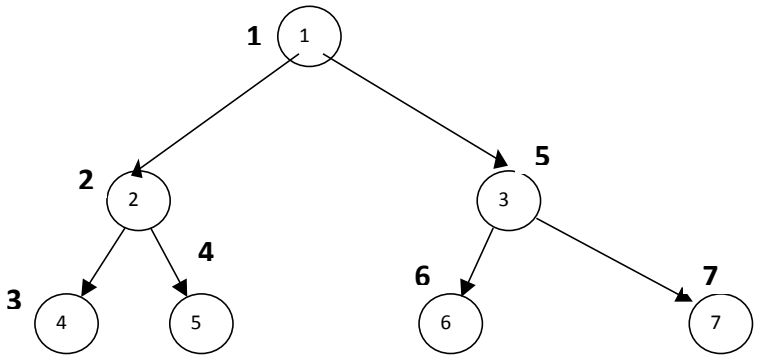
2. En el caso de desarrollar una búsqueda primero en anchura el espacio de búsqueda será:



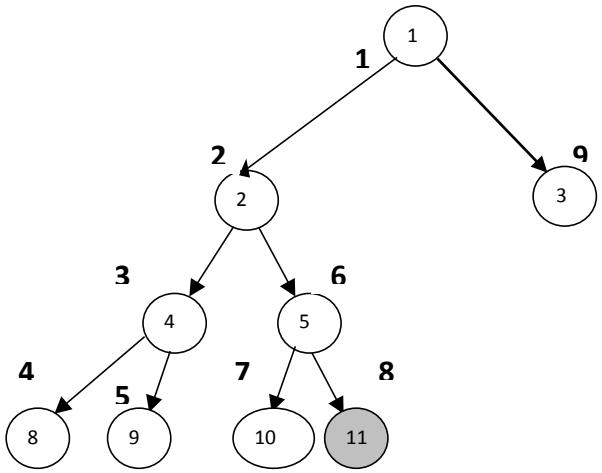
El orden de expansión de los nodos coincide con el etiquetado de los mismos. Concretamente, antes de expandir el nodo 11 y comprobar si es el nodo meta, se habrán expandido previamente los nodos 8, 9 y 10.
En el caso de desarrollar una búsqueda iterativa, en la primera iteración se generará el siguiente árbol de búsqueda:



En la segunda iteración el espacio de búsqueda será:



La solución se encuentra al desarrollar la tercera iteración del árbol de BPI. Nótese que, a diferencia de anchura, no se expanden los nodos del nivel 3 del árbol.



b) Podremos garantizar que se encuentra la solución óptima para valores de 'a' que cumplan a<2, ya que en este caso tendremos una búsqueda de tipo A* que es admisible, lo que hacemos es multiplicar el coste real g(n) en cualquier nodo por el mismo factor, por lo que al final encontraremos un camino óptimo aunque el coste que obtengamos al aplicar f(n) al nodo hoja de ese camino esté multiplicado por (2-a).

Para valores negativos de (2-a) no encontraríamos la solución óptima ya que al aplicar la función de evaluación al nodo hoja del camino encontrado obtendríamos el valor más pequeño posible al aplicar la función que, en realidad se correspondería con el mayor valor de g(n) al estar multiplicado por un factor negativo.

Para a=2 no se encontrará la solución óptima. Al no incluir en la función de evaluación el coste real del camino desde el nodo raíz al nodo valuado (g(n)), tendremos una búsqueda voraz primero el mejor que no es admisible.

Exercise 16

Given the following A algorithms where h*(n) represents the cost from node n to the optimal state:

1) f1(n) = g(n) + h1(n) / h1(n)≤h*(n)
2) f2(n) = g(n) + h2(n) / h2(n)>h1(n)
3) f3(n) = g(n) + h3(n) / h1(n) ≤ (h3(n) ≤ (1+a) h*(n), being *a* a positive constant.
4) f4(n) = g(n) + h4(n) / h4(n) ≤h1(n)

Respond briefly to the following questions:

a) Which of the above methods is admissible?
b) Which of the two methods, f1 or f4, will have a larger profoundness?
c) What can you say about the quality of the solution returned by f3(n)? And about the solution returned by f2(n)?
d) Given a very complex problem, which method is the most appropriate in each of the following cases:

   d.1) When we want a response in a VERY SHORT TIME. In this case, importance is given to promptness rather than the solution quality or even the guarantee to eventually find a solution.
   d.2) When we want a response in a SHORT TIME and with a good quality
   d.3) When we want the optimal solution

e) In general, is it possible to determine for a given problem ...

   e.1) the value of h*(n)?
   e.2) if an heuristic function satisfies the condition h(n) ≤h*(n)?
   e.3) if an heuristic function satisfies the monotonicity condition?

Solution:

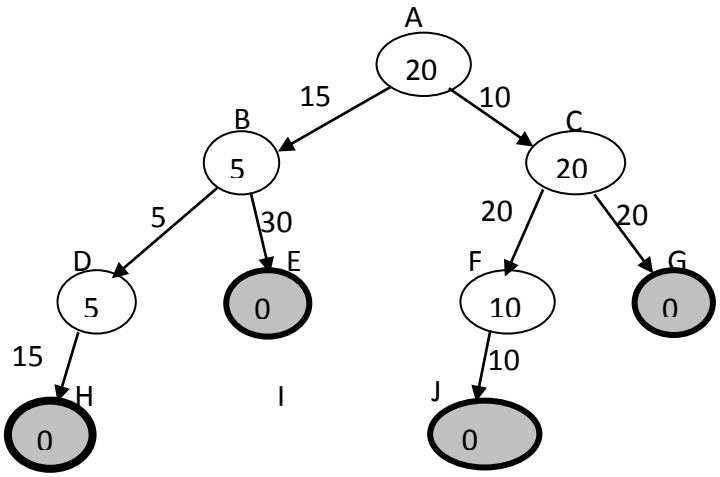
a) f1 y f4 son ambas admisibles por tener un h(n)<= h*(n). De las otras dos funciones no se puede garantizar la admisibilidad.
b) f1 ya que está más informada que f4 y por tanto generará menos nodos, por lo que el factor de penetrabilidad será mayor.
c) El coste de la solución que obtenga f3 estará acotado respecto al coste de la solución óptima por un factor (1+a). Sobre la solución que obtenga f2 no puede decirse nada, puede ser la óptima, una solución no óptima o no encontrar la solución.
d) d.1 – Mejor f2, ya que puede estar sobreinformada y obtener soluciones muy rápidas
d.2 - Mejor f3, está sobreinformada pero acotada por un factor (1+a)
d.3 - Mejor f1, ya que es admisible y está más informada que f4

e) e.1- No, no es posible conocer el coste real de la solución porque en dicho caso se conocería la solución. h(n) es una estimación de h*(n).
e.2- Se puede hacer un análisis general del problema y comprobar si la heurística es admisible. Además si se encuentra un contraejemplo en el que para un nodo h(n) > h*(n), entonces ya se puede decir que no es admisible. También sería posible si se conociera una cota inferior del coste de la solución óptima y h(n) ser inferior a dicha cota.
e.3- En algunos casos sí, si se puede comprobar que h(n1) <= h(n2) + coste (n1,n2)

Exercise 17

The following tree represents the complete search space of a particular problem. Edges are labelled with the cost of operators ($g(n_i->n_j)$). Values inside the nodes are the estimations to a goal state (h(n)). Shadowed nodes represent goal states. Apply an A-type algorithm from the initial node (A) and indicate clearly the order of expansion of nodes (for instance, step i: node X is expanded and nodes Y and Z are generated). Answer the following questions:

a) Is this an A* search? Why?
b) Show the goal state returned by the search process
c) Does the algorithm return the optimal solution? Why?
d) Is the f(n)=g(n)+h(n) a monotone function? Why?
e) Apply a greedy best-first search indicating the order of expansion of nodes and the goal state that would be reached.



Solution:

a) Es una búsqueda de tipo A*, ya que se cumple que $\forall n, h(n) \leq h^*(n)$.

b) Paso-1 Se expande A, se generan B, con f(B)=20, y C, con f(C)=30

Paso-2 Se expande B, se generan D, con f(D)=25, y E, con f(E)=45
Paso-3 Se expande D, se genera H, con f(H)=35
Paso-4 Se expande C, se generan F, con f(F)=40, y G, con f(G)=30
Paso-5 Se expande G. Se detecta que nodo meta. FIN.

c) El proceso devuelve G como nodo meta.

d) Claramente, devuelve la senda óptima, como cabría esperar de una búsqueda tipo A*

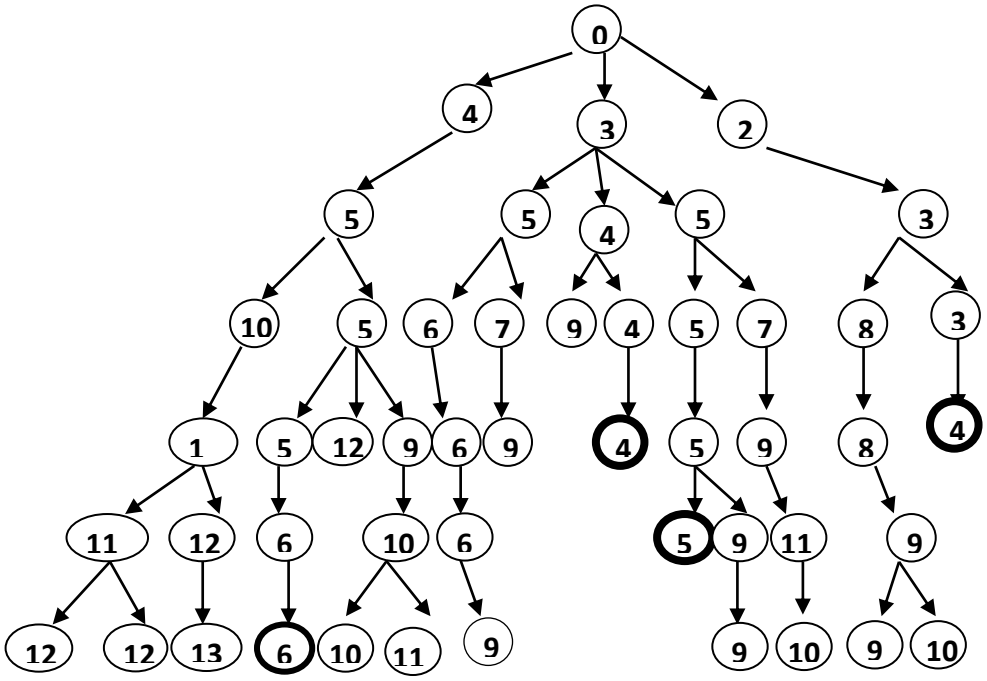
e) Claramente, es una función f(n) monótona, al cumplirse en todos los casos:
$\forall n, h(n2) \geq h(n1) - Coste(n1 \rightarrow n2)$

g) Con búsqueda 'primero el mejor', el orden de expansión sería: "A, B, E". El estado meta que devolvería sería E.

Exercise 18

The following search tree represents the solution state space of a particular problem. Values in each node are the result of applying a function f(n)=g(n)+h(n), h(n) <= h*(n) (admissible heuristic or A*); the cost of all operators is 1. Nodes circled in bold represent solution states for the problem. Answer the following questions:

a) If we apply a heuristic search with function f(n), which solution would this method find? how many nodes would be necessary generate to find such a solution?
b) If we apply an iterative deepening search strategy, which solution would this method find? Is this the same solution as the one found in section a)? Otherwise, show which solution is better.
c) If we apply a depth-first search, which solution would this method find?
d) If we apply a heuristic search with function f1(n)=g(n)+h1(n), such that h(n)< h1(n)<= h*(n), is the solution path found with f1(n) better/worse/the same as the one obtained with f(n)? Which evaluation function generates the minimum number of nodes?



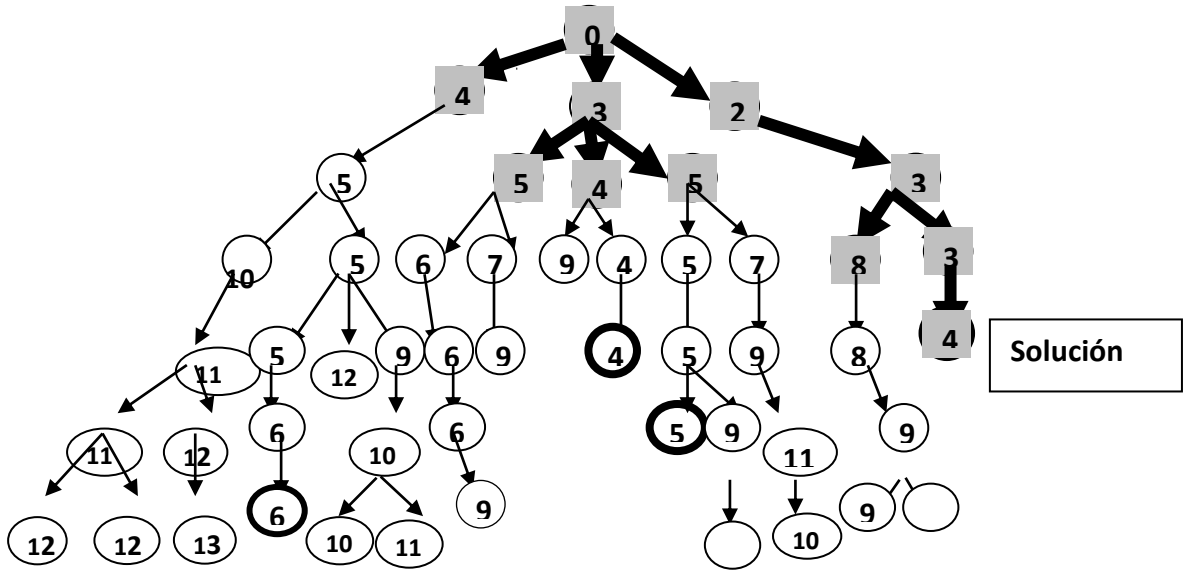
Solution:

Exercises Block 1- Search

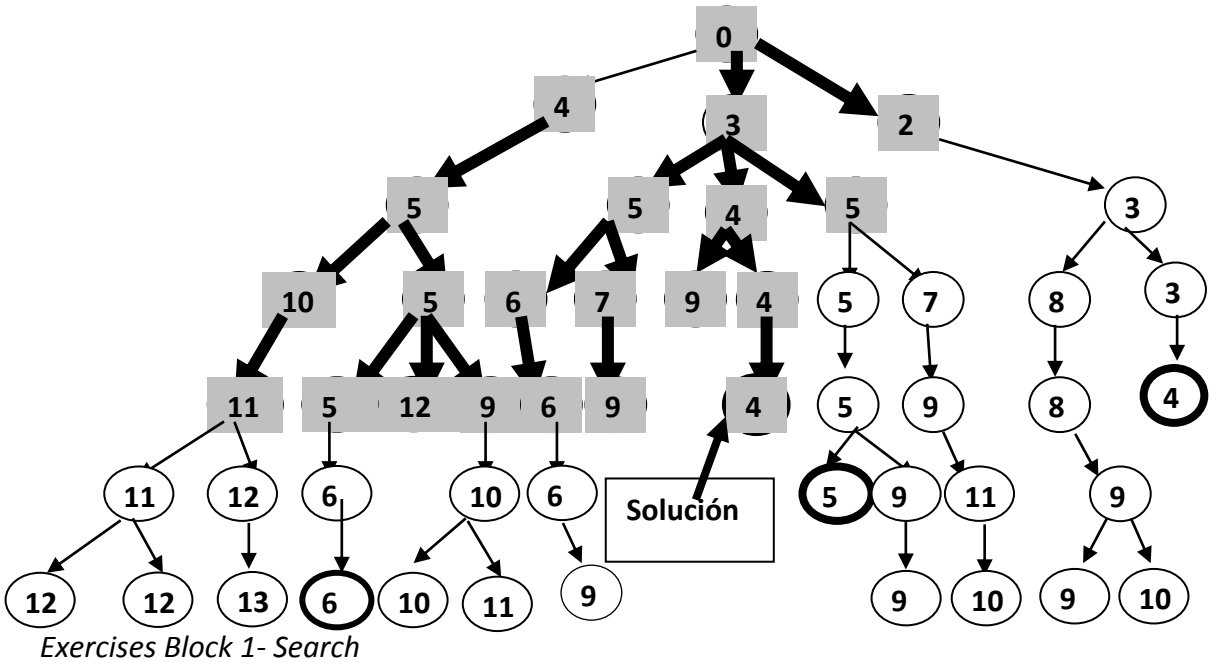
a) En trazo grueso se indica el espacio de búsqueda generado. En caso de un mismo valor de la función de evaluación, asumiremos que se expande en primer lugar el nodo más reciente, o sea el nodo más profundo.

OPEN={2,3,4} se saca el nodo de valor 2 de la lista; no es solución y se expande
OPEN={3 (nivel2),3 (nivel1),4} se saca el nodo de valor 3 de nivel 2 de la lista; no es solución y se expande
OPEN={3 (nivel3), 3 (nivel1), 4, 8} se saca el nodo de valor 3 de nivel 3 de la lista; no es solución y se expande
OPEN ={3 (nivel1), 4 (nivel4), 4 (nivel1), 8} se saca el nodo el nodo de valor 3 de nivel 1 de la lista; no es solución y se expande.
OPEN= {4 (nivel4), 4 (nivel2), 5 (nivel2), 5 (nivel2), 8} se saca el nodo de valor 4 de nivel 4 de la lista; es solución y el algoritmo termina.

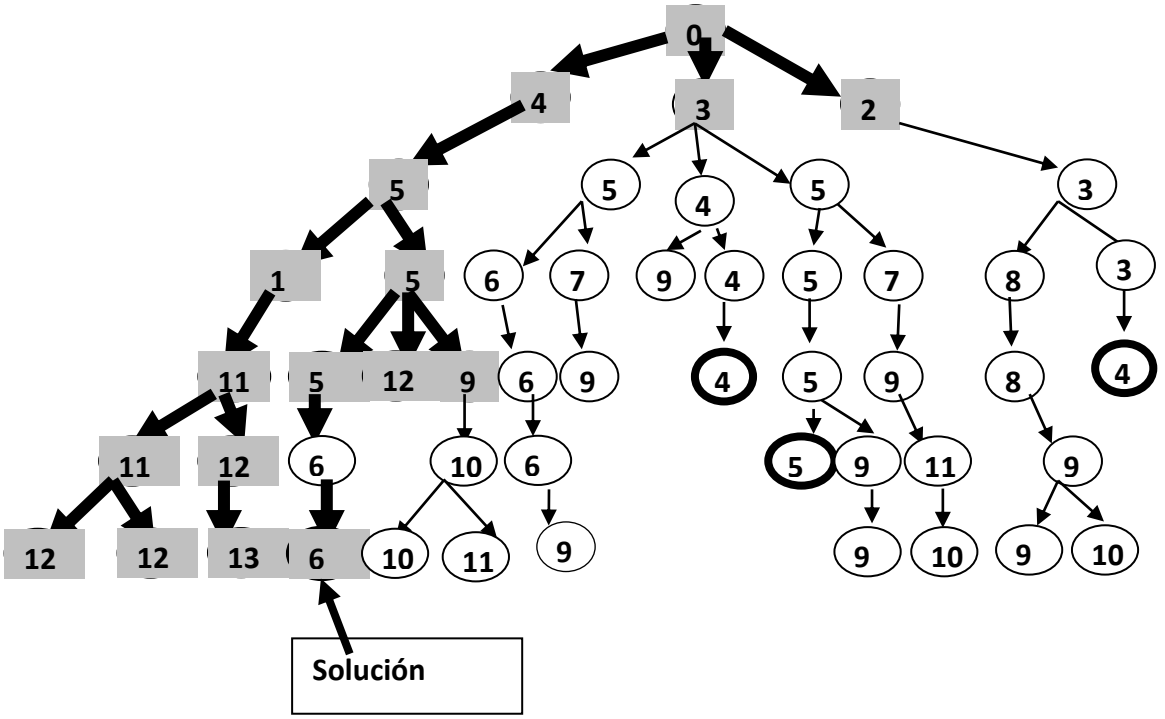
En total se han generado 10 nodos (incluyendo nodo meta pero no el inicial) que equivale a 10 nodos expandidos.



b) En el caso de realizar una búsqueda por profundización iterativa, se generarían diversos árboles de búsqueda, para profundidad 1, 2, y 3 que irán explorando exhaustivamente el espacio de soluciones, a dichas profundidades, ya que en estas profundidades no se encuentra ninguna solución. Será en la siguiente iteración, 4, donde se encuentre la solución que se indica en la siguiente figura donde se representa, mediante trazo grueso y marcando los nodos, el espacio de búsqueda en profundidad a dicha profundidad.



c) En la figura está representado el espacio de búsqueda que se generará en este caso. Como se puede observar el coste de la solución es mayor que en los casos anteriores, como ya sabemos esto puede suceder ya que, a diferencia de las estrategias de los apartados anteriores, la búsqueda primero en profundidad no es admisible.



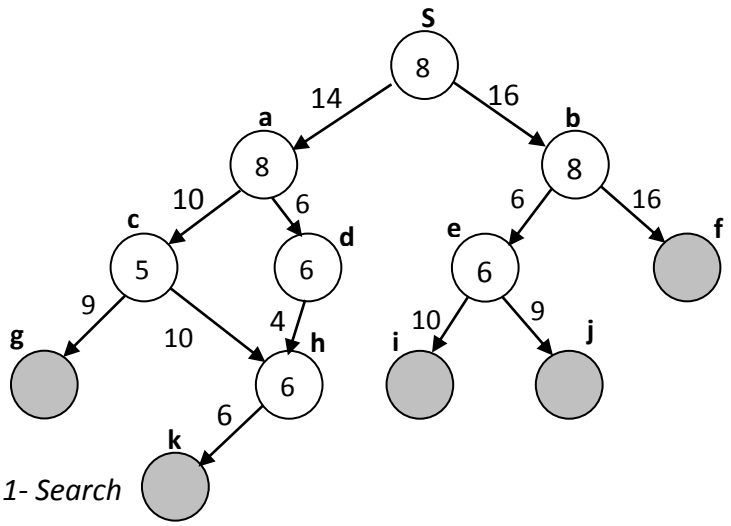
d) Como en ambos casos tendremos una búsqueda A*, que como ya sabemos es admisible, la solución encontrada será la mejor solución. Lo que si sucederá es que la función más informada, h1(n), producirá potencialmente más poda por lo tanto el espacio de búsqueda generado por f1(n) será menor.

Exercise 19

The figure below shows a search process in a state space where the values labelling the edges are the cost of operators (g(n)), the values inside the nodes are the heuristic estimation (h(n)), and the shadowed nodes are the goal nodes. Over this search space apply i) ***an A algorithm*** and ii) a ***uniform cost search process*** and answer the following questions for each of these two methods:

- The order in which nodes are expanded and which of the goal states is reached
- Does the strategy find the optimal solution? Why?

In case of two nodes with the same value of the evaluation function, expand first the node at the shallowest level



Solution:

Búsqueda Coste Uniforme:

Expande: S;	Genera: A, B;	Lista Abiertos: (A-14, B-16)
Expande: A;	Genera: C, D;	Lista Abiertos: (B-16, C-24, D-20)
Expande: B;	Genera: E, F;	Lista Abiertos: (C-24, D-20, E-22, F-32)
Expande: D;	Genera: H;	Lista Abiertos: (C-24, E-22, F-32, H-24)
Expande: E;	Genera: I, J;	Lista Abiertos: (C-24, F-32, H-24, I-32, J-31)
Expande: C;	Genera: G, H;	Lista Abiertos: (F-32, H-24, I-32, J-31, G-33)
Expande: H;	Genera: K;	Lista Abiertos: (F-32, I-32, J-31, G-33, K-30)
Expande: K;	NODO META	

Búsqueda A:

Expande: S;	Genera: A, B;	Lista Abiertos: (A-22, B-24)
Expande: A;	Genera: C, D;	Lista Abiertos: (B-24, C-29, D-26)
Expande: B;	Genera: E, F;	Lista Abiertos: (C-29, D-26, E-28, F-32)
Expande: D;	Genera: H;	Lista Abiertos: (C-29, E-28, F-32, H-30)
Expande: E;	Genera: I, J;	Lista Abiertos: (C-29, F-32, H-30, I-32, J-31)
Expande: C;	Genera: G, H;	Lista Abiertos: (F-32, H-30, I-32, J-31, G-33)
Expande: H;	Genera: K;	Lista Abiertos: (F-32, I-32, J-31, G-33, K=30)
Expande: K;	NODO META	

Al ser h(n)<h*(n), es por tanto A*.

En ambos casos, obtienen la senda óptima, Tanto A* como coste uniforme son admisibles.

Exercise 20

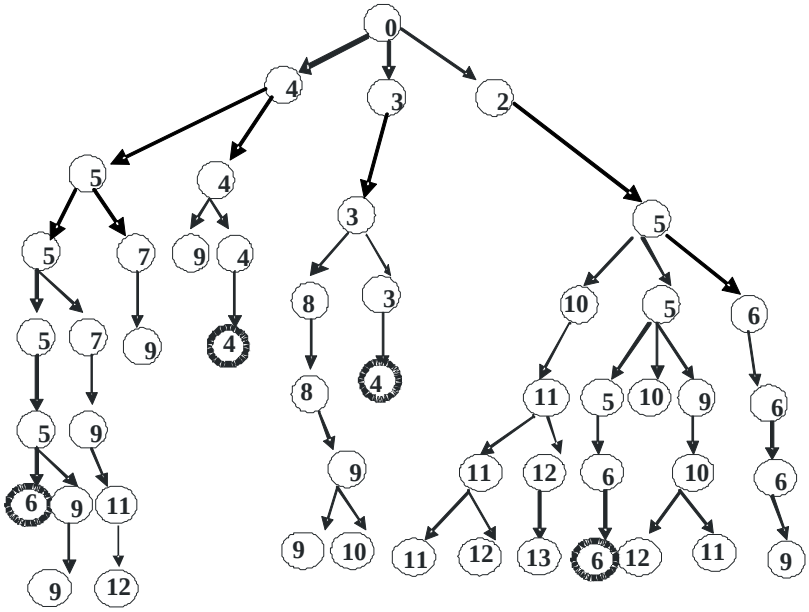
The tree below represents the search space for a particular problem. Values inside the nodes are the result of applying the function f(n)=g(n)+h(n), where h(n) is a heuristic function such that h(n) ≤ h*(n). All actions have the same cost (1). Nodes in bold are goal states. Answer the following questions:

a) Which solution will the A* algorithm find by applying the 'f' function? How many nodes are generated to find the solution path?

b) If we apply an iterative deepening strategy in the search tree, which solution is found? Is this solution better than the one found by A*? Why?

c) If we use an evaluation function f1(n)= g(n)+h1(n), such that h(n) < h1(n) ≤h*(n), the solution found with 'f1', is it better/worse/equal to the solution found with function 'f'? which strategy will require less search cost?

d) If we apply a uniform-cost search, which solution is found? How many nodes have to be generated to find a solution? Is this solution better or worse than the solution found in question a)? Why?



Solution:

a) La solución que se encuentra es el nodo solución de valor 4 (nivel 4 del árbol) de la rama central del árbol. Se generan un total de 9 nodos, incluido el nodo raíz.

b) En este caso, asumiendo una expansión de izquierda a derecha, la solución que se encontraría es el nodo solución de valor 4 (nivel 4 del árbol) de la rama izquierda del árbol. Son soluciones distintas pero ambas son óptimas.

c) Ambas son admisibles por lo que encontrarán una solución óptima. Presumiblemente, la función heurística que está más informada (h1(n)) realizará más poda y por tanto generará menos nodos.

d) En este caso en que el coste de aplicación de cada operador es uniforme, la búsqueda por coste uniforme coincide con una búsqueda en anchura, que a su vez coincidirá con la solución encontrada por la estrategia de profundización iterativa. Por tanto, asumiendo de nuevo un orden de expansión de izquierda a derecha, la solución encontrada es la misma que la del apartado b). Para encontrar la solución se generarán todos los nodos de los cuatro primeros niveles del árbol y los dos primeros nodos del nivel 5 hasta que se extrae de la lista de nodos abiertos el nodo 4 de nivel 4 para ser expandido (total: 30 nodos generados incluido el nodo raíz). Respecto a la comparativa con la solución del apartado a), la solución es distinta pero ambas son óptimas ya que las dos estrategias son admisibles.

Exercise 21

The figure (a) shows the initial state of a sliding tile puzzle that consists of a row of 5 squares, with 2 white tiles (W) at the left, 2 black tiles (B) at the right and an empty space in the middle. The figure (b) shows the final state we want to reach.

1	2	3	4	5
W	W		B	B

(a) Initial state

1	2	3	4	5
B	B		W	W

(b) Final state

The legal moves or actions, with costs, are as follows:

A1. If the empty space is next to a tile, the tile may move into the empty space (cost=1)
A2. A tile may hop over 1 tile (of either colour) into the empty space (cost=1)
A3. A tile may hop over 2 tiles (of either colour) into the empty space (cost=2)

Additionally, tiles W can only move towards the RIGHT, and tiles B can only move towards the LEFT.

Consider the following heuristic based on the distance of a tile to its "finishing zone". We define the "finishing zone" of a tile to the two squares into one of which that tile has to move; that is, the finishing zone for each of the W tiles are the squares at positions 4 and 5, and the finishing zone of the B tiles are the squares 1 and 2. The heuristic function 'h' applied over a tile that is at one position (1, 2, 3, 4 or 5) returns an estimation of moving that tile to its finishing zone. The h-values for each tile and position are:

POSITIONS	1	2	3	4	5
function h(W_i):	2	1	1	0	0
function h(B_i):	0	0	1	1	2

For example, the estimation of a tile W that is at position 1 is 2 (h(W1)=2), or the h-value of a tile B that is at position 1 is 0 because the tile is already in its finishing zone (h(B1)=0). The application of the h-function over a problem state is the sum of the heuristic values for each of the tiles in that state. For instance, given the state WW_BB, the result of applying the heuristic function is:

$$h(WW\_BB)=h(W1)+h(W2)+h(B4)+h(B5)=2+1+1+2=6$$

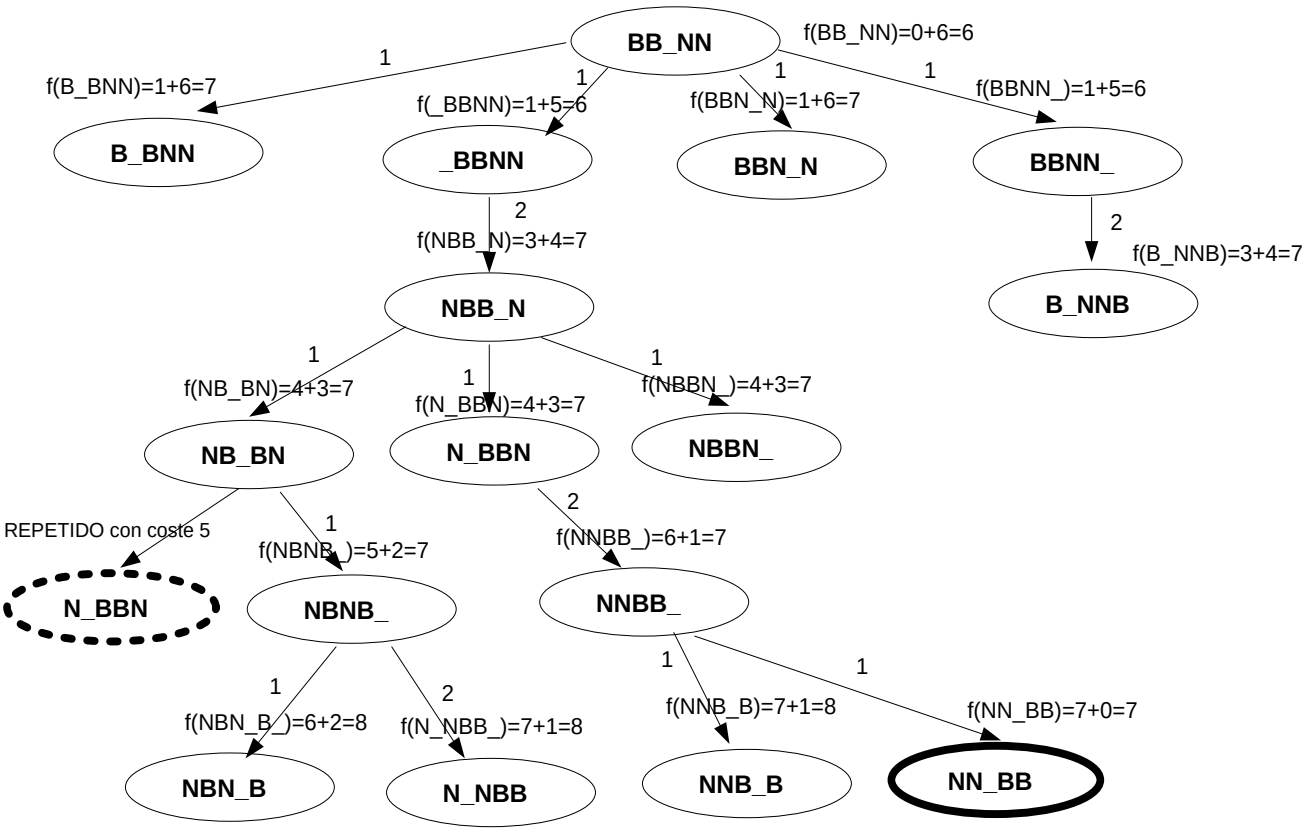
Answer the following questions by providing **APPROPRIATE JUSTIFICATIONS** to all your answers.

1) Draw the search tree resulting of applying an A-algorithm (f(n)=g(n)+h(n)). Show clearly the value f(n) of each node, the order of nodes expansion and the OPEN list at each iteration. Which solution, and with which cost, will this algorithm find? Is it an A* search? Why?
    a. *Note 1: Apply actions in this order to create the search tree: first, action A1 over a W tile; then A2 over a W tile; then A3 over a W tile; then A1 over a B tile; then A2 over a B tile and, finally, A3 over a B tile.*
    b. *Note 2: If two nodes have the same f-value, expand first the **deepest** node. If both are at the same level, expand first the **oldest** node.*
    c. *Note 3: Avoid repeated nodes.*
2) Consider the application of a greedy best-first search. Which solution will it find? Is it an optimal solution? Why?
3) In case of a depth-first expansion, is it necessary to set a maximum depth level of the tree to guarantee a solution is found? Why? If so, show the value of such a depth limit.

Solution:

Pregunta 1

1) OPEN= {BB_NN (6)}
2) OPEN= {_BBNN (6), BBNN_ (6), B_BNN (7), BBN_N (7)}
3) OPEN= { BBNN_ (6), NBB_N (7), B_BNN (7), BBN_N (7)}
4) OPEN= { NBB_N (7), B_NNB (7), B_BNN (7), BBN_N (7)}
5) OPEN= { NB_BN (7), N_BBN (7), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7)}
6) OPEN= { NB_NB (7), N_BBN (7), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7)}
7) OPEN= { N_BBN (7), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7), NBN_B (8), N_NBB (8)}
8) OPEN= { NNBB_ (7), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7), NBN_B (8), N_NBB (8)}
9) OPEN= { NN_BB (7), NNB_B (8), NBBN_ (7), B_NNB (7), B_BNN (7), BBN_N (7), NBN_B (8), N_NBB (8)}
10) Se extrae NN_BB => Solución (5 movimientos, coste =7)



Es un algoritmo A* porque con los valores h de la tabla se satisface siempre la relación h(n)<= h*(n) para todo nodo del árbol dado que:

- el coste mínimo de mover B1 o N5 a su zona objetivo es 2 (salto sobre 2 fichas)
- el coste mínimo de mover B2 o N4 a su zona objetivo es 1 (salto sobre 1 ficha)
- el coste mínimo de mover B3 o N3 a su zona objetivo es 1 (desplazar al espacio blanco)

Por ejemplo, h(BB_NN)=6 y h*(BB_NN)=7.

Pregunta 2

Fijándonos SOLO en los valores h(n) del árbol de la pregunta 1, podemos ver que el orden de expansión sería:

1. nodo inicial (h=6)
2. _BBNN (h=5)
3. NBB_N (h=4)
4. NB_BN (h=3)
5. NBNB_ (h=2)
6. N_NBB (h=1)
7. Ahora expandiría el nodo N_NBB llegando al nodo solución NN_BB (h=0)

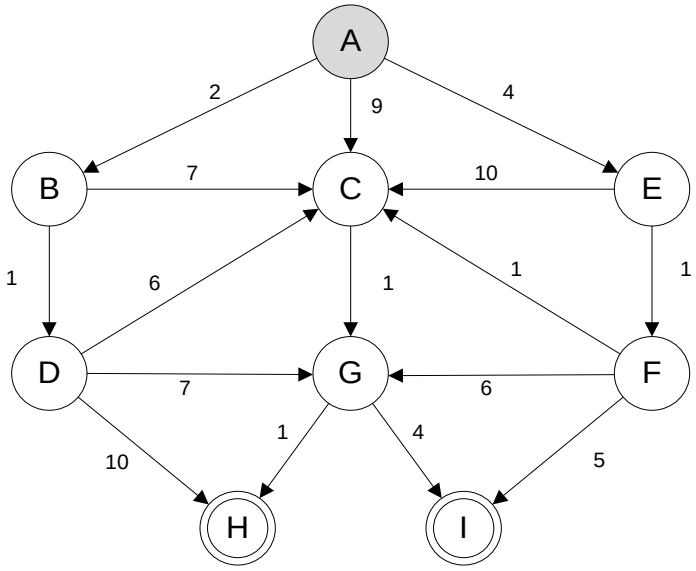
La solución que encontraría este algoritmo se compone de 6 movimientos y tiene un coste=8. Por tanto, no es la óptima.

Pregunta 3

No, en este problema no es necesario limitar el árbol porque no hay operadores reversibles ya que las fichas B solo se pueden mover hacia la derecha y las fichas N solo se pueden mover a la izquierda. Por tanto, no hay riesgo de quedarse estancado en un ciclo. Una búsqueda en profundidad expandiría a partir del hijo B_BNN y eventualmente encontraría una

Exercise 22

The graph below represents the state space of a particular problem. The nodes are the states of the problem, the edges connect a state with its successors, and the values in the edges represent the cost of the operators to move from one state to the corresponding successor. The initial state of the problem is the node A, and the final states are the nodes H and I. The heuristic estimates of the nodes are shown in the table.



Nodo	h(n)
A	7
B	7
C	2
D	7
E	4
F	9
G	1
H	0
I	0

Apply an ***uniform cost search*** process and an ***A algorithm*** over the graph. In both cases, show the lists of OPEN and CLOSED nodes, the obtained solution and the cost of such a solution. Avoid repeated states. Do both strategies find the optimal solution? Is the function h(n) admissible? Why?

Solution:

Coste uniforme: f(n)=g(n)
OPEN={ A (0), B (2), E (4), C (9),
CLOSED={A, B,

Al expandir B, se generan D con coste 3 y C con coste 9 (no se añade a OPEN por tener el mismo coste que el nodo que ya se había añadido al expandir A – esto sucede en otras ocasiones, pero ya no se comentará, simplemente no se añadirá el nodo correspondiente a la lista OPEN).

OPEN={ E (4), C (9), D (3), G (10), H (13), F (5),
CLOSED={A, B, D, E, F,

Al expandir F, se genera C con coste 6, que sí se añade a OPEN al tener un coste menor que el que ya figura en la lista, que se elimina; se genera G con coste 11, que no se añade e I con coste 10:

OPEN={ G (10), H (13), C (6), I (10)
CLOSED={A, B, D, E, F, C,

Al expandir C, se genera G con coste 7, que sustituye al nodo que está ahora en OPEN.

OPEN={ ~~H (13),~~ I (10), G (7), H (8),
CLOSED={A, B, D, E, F, C, G, H }

Al extraer H de la lista OPEN, se comprueba que es un nodo solución -> FIN

Solución: A -> E -> F -> C -> G -> H Coste: 8

Algoritmo A: f(n)=g(n)+h(n)

OPEN={ A (0+7), B (2+7), C (9+2), E (4+4), F (5+9), D (3+7), G (10+1), H (13+0)
CLOSED={ A, E, B, D, G,

Al expandir G, se genera H con coste 11, por lo que sustituye al nodo que figura ahora en OPEN:

OPEN={ C (9+2), F (5+9), H (11+0), I (14+0)
CLOSED={ A, E, B, D, G, H }

Solución: A -> B -> D -> G -> H Coste: 11

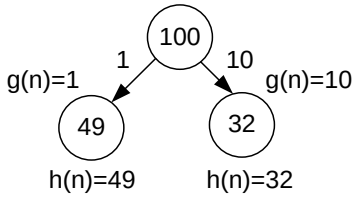
Se obtiene la solución óptima solamente en el caso de coste uniforme. h(n) no es admisible porque h(F)=9 > h*(F)=3 (F-C-G-H).

Exercise 23

Consider a search space where the initial node is labelled with the number 100 (n=100), and expanding any node 'n' always results in two successor nodes:

1) one of the successors is a node labelled as int(n/2)-1, and the cost of applying this operator is 1,
2) the other successor is a node labelled as int(n/3)-1, and the cost of applying this operator is 10

Let's assume a heuristic function 'h(n)' such that h(n)=n. The tree below shows the first level of the tree.

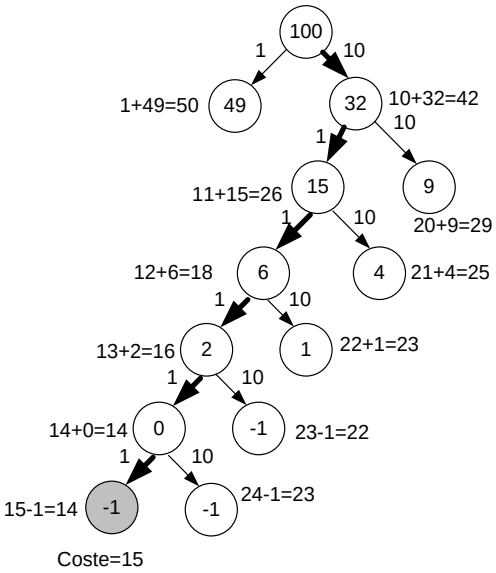


Taking into account that a goal state in this problem is any node 'n' such that n < 0 (negative labelled node):

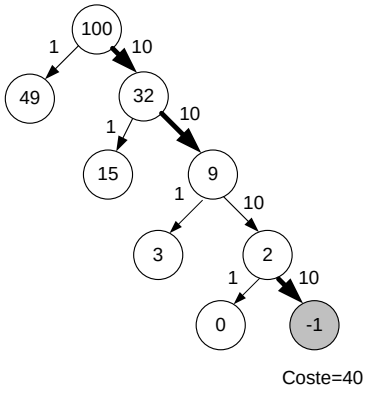
a) Draw the search tree resulting from applying an A algorithm (f(n)=g(n)+h(n)). Show clearly the g-values and h-values of each node. Show the solution path and its cost.
b) Draw the search tree resulting from applying a greedy best-first search. Show the solution path and its cost.
c) Is the A-algorithm an A* search? Why? Justify your answer.

Solution:

Búsqueda A:



Búsqueda greedy best-first:

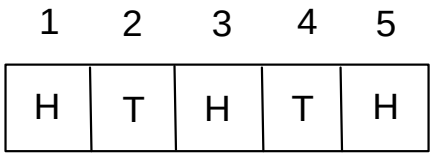


Claramente, la función h(n) utilizada **NO es admisible**. Por ejemplo, estima para el nodo inicial un coste h(n)=100, cuando se ha obtenido una senda de coste menor.

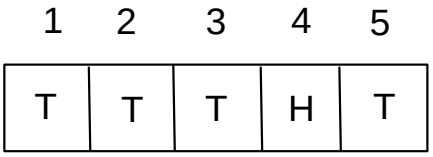
Exercise 24

We have five coins placed in a row as indicated in figure (a). The coin head is represented by **H** and the tail by **T**.

The only legal movement in this problem is to turn over (from **H** to **T** or from **T** to **H**) two contiguous (adjacent) coins; that is, turn over coin1-coin2 (coin at position 1 and coin at position 2), coin2-coin3, coin3-coin4 or coin4-coin5. This operation costs 1. We wish to get the situation shown in figure (b).



(a) Initial state



(b) Final state

Heuristic function. The heuristic function **h(n)** checks the coins in the row in two steps, first step from left to right, and second step from right to left. At each step, the contiguous coins that are analyzed are:

Step1 (from left to right): check coin 1 with coin2, and coin 3 with coin 4
Step2 (from right to left): check coin 5 with coin4, and coin 3 with coin 2

h-values for pairs of contiguous coins are as follows:

h(coin X in its right place, coin Y in its right place) = 0 points
h(coin X out of place, coin Y out of place)= 1 point
h(coin X in its right place, coin Y out of place)= 2 points
h(coin X out of place, coin Y in its right place)= 2 points

Thus, if two contiguous coins are at their right place then the h-value for the pair of coins is 0. If the two contiguous coins are both out of place then the h-value for the pair is 1. If one of the two coins is out of place and the other is in its right place then the h-value for the pair of coins is 2.

The h(n) function for a state 'n' of the problem is **h(n)= min(h_left_right(n), h_right_left(n))** (minimum value between the h-value of step1 and step2) where:

h_left_right(n)= h(coin1,coin2) + h(coin3,coin4)
h_right_left(n)= h(coin5,coin4) + h(coin3,coin2)

For instance, for the initial configuration in figure (a), the value of h(n) is:

h(HTHTH)=min(2+1,1+2)=3

Answer the following questions by providing **APPROPRIATE JUSTIFICATIONS** to all the questions:

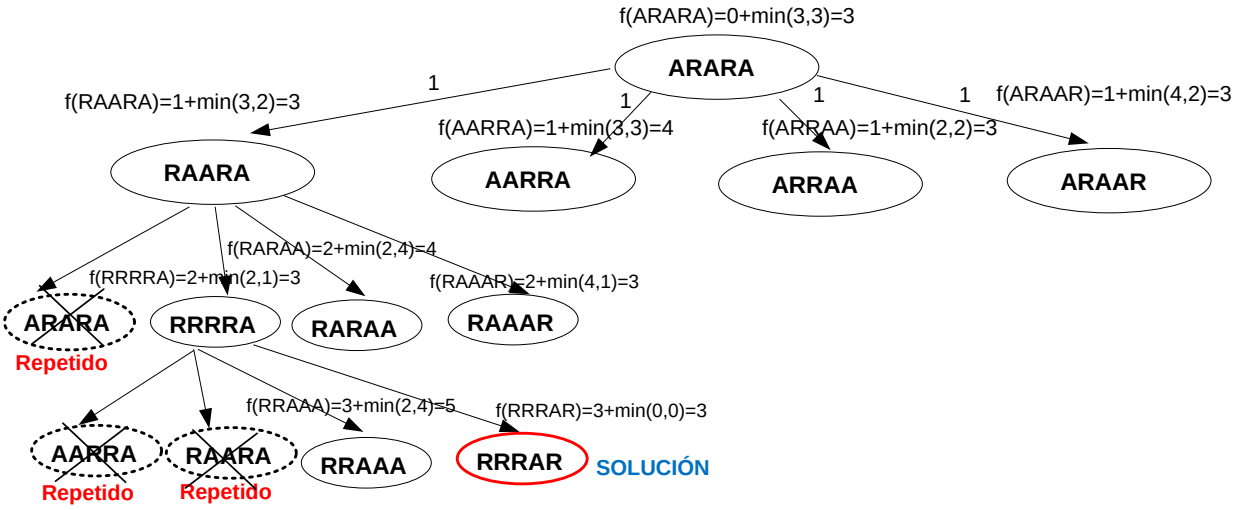
1) Draw the search tree resulting of applying an A-algorithm (f(n)=g(n)+h(n)) over the initial state in figure (a). Show clearly the f-value of each node, the node expansion order, and the OPEN list at each iteration. Which solution, and with which cost, is found with this algorithm? Is it an optimal solution?
2) The heuristic function 'h', is it an A* heuristic? Why? Justify your answer.
3) Assuming the user selects m=5 as maximum depth level for a depth-first search, which solution will this algorithm return and which is the cost of this solution?

IMPORTANT NOTES for the node expansion:

a. *Note 1: Apply movements in this order to create the search trees: first, coin1-coin2, then coin2-coin3, then coin3-coin4 and finally coin4-coin5.*
b. *Note 2: If two nodes have the same f-value, expand first the **deepest** node. If both are at the same level, expand first the **oldest** node.*
c. *Note 3: Avoid **ALWAYS** repeated nodes.*

Solution:

1)

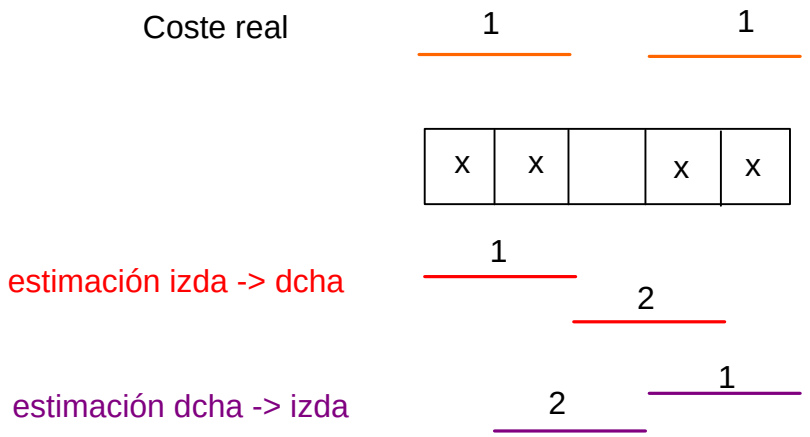


1) OPEN= { ARARA (5)}
2) OPEN= { RAARA (3), ARRAA (3), ARAAR (3), AARRA (4)}
3) OPEN= { RRRRA(3), RAAAR (3), ARRAA (3), ARAAR (3), RARAA (4), AARRA (4)}
4) OPEN= { RRRAR (3), RAAAR (3), ARRAA (3), ARAAR (3), RARAA (4), AARRA (4), RRAAA (5)}

5) Se extrae RRRAR => Solución 3 movimientos. Coste = 3.

La solución que obtiene es cambiar **moneda1-moneda2, moneda2-moneda3, moneda4-moneda5**. Esta solución es de coste óptimo (3 movimientos).

2) En general la heurística estima bastante bien pero dependiendo de cómo estén las parejas contiguas de piezas descolocadas el análisis podría sobreestimar el coste. Dado que los análisis de izquierda a derecha y de derecha a izquierda siempre toman parejas fijas y no hacen todas las posibles combinaciones de parejas, podría darse el caso que las parejas estudiadas tuvieran siempre en cuenta una ficha que no está descolocada. Por ejemplo, supongamos que todas las piezas están mal colocadas excepto la del medio.



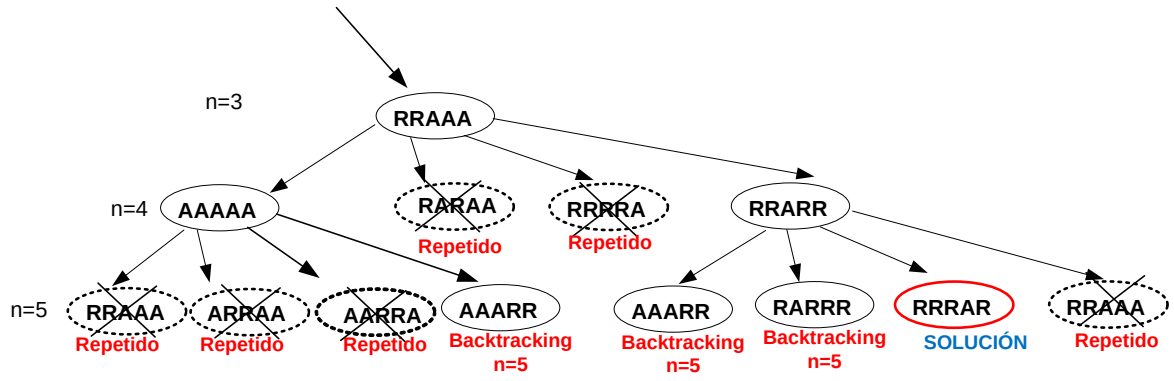
El mínimo de la estimaciones devolverá 3 porque siempre tienen en cuenta la moneda del medio en las estimaciones pero el coste real son 2 porque la moneda del medio no hace falta moverla.

Un caso claro de esto se puede observar en el segundo hijo (empezando por la izquierda) del nodo raíz.

Por tanto, la heurística no es A*.

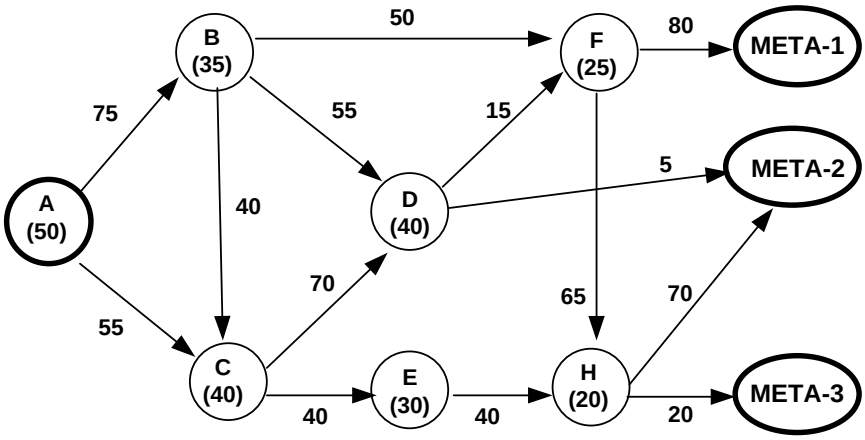
3) El árbol de la figura 1 sería equivalente al de una búsqueda en profundidad hasta m=3 por la primera rama del nodo raíz. Para continuar la búsqueda en profundidad hasta m=5 tendríamos que seguir por el nodo RRAAA, manteniendo los nodos OPEN del árbol de la figura 1.

La solución que encuentra está en el nivel n=5 (moneda1-moneda2, moneda2-moneda3, moneda3-moneda4, moneda4-moneda5, moneda3-moneda4)



Exercise 25

The graph shown below represents a search space where the initial state is node A, the goal states are META-1, META-2 and META-3, and edges denote the accessible states. The values on the edges represent the cost of the arc to go from one node to the other, and the numbers inside the nodes denote the value of a heuristic function h(n).



a) Draw the search tree for each of the three following search strategies. Show clearly the f-value of each node and the node expansion order. Show also the solution path found, the reached goal state and the cost of the solution.

a.1) An A-algorithm (f(n)=g(n)+h(n))
a.2) A greedy best-first algorithm
a.3) A uniform cost strategy

Note: If two nodes have the same f-value, expand in alphabetical order

b) Does any of the methods in points a) and b) return the optimal solution? Is h(n) an A* heuristic function? Why? Justify your answer.

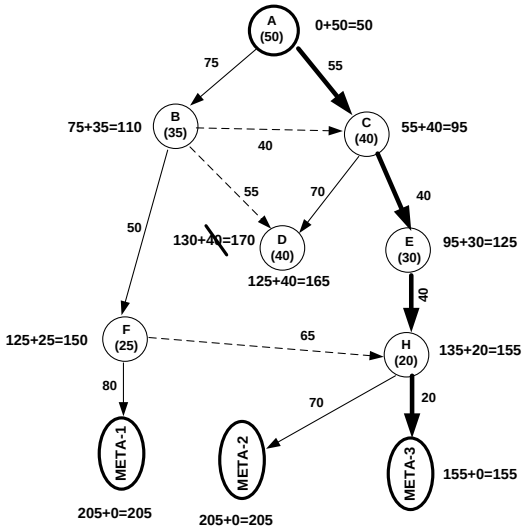
Solution:

a.1) Algoritmo A

Orden de expansión de nodos: A, C, B, E, F, H, Meta3

Estado Objetivo alcanzado: Meta-3,

Coste Senda solución = 155

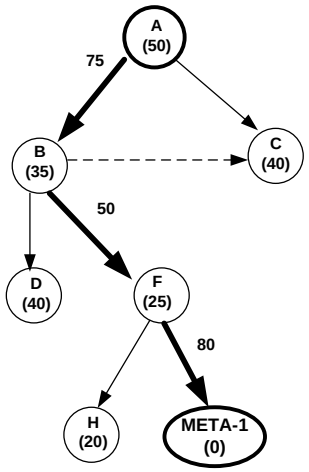


a.2) Greedy best-first

Orden de expansión de nodos: A, B, F, Meta-1.

Estado Objetivo alcanzado: Meta-1,

Coste Senda solución = 205

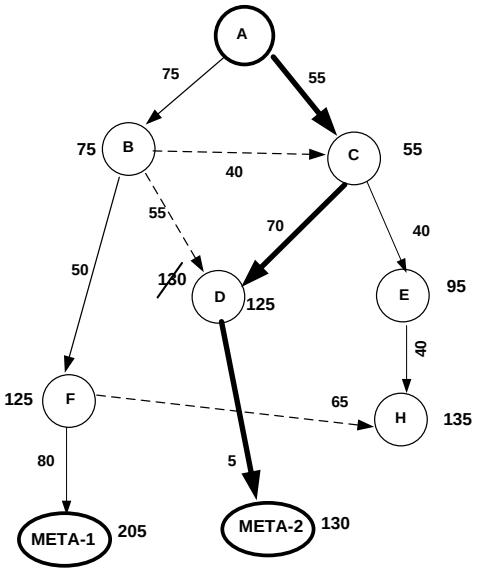


a.3) Coste Uniforme

Orden de expansión de nodos: A, C, B, E, D, F, Meta2

Estado Objetivo alcanzado: Meta-2,

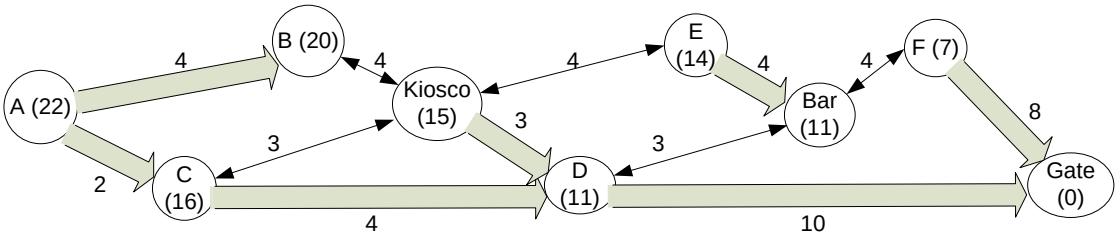
Coste Senda solución = 130



b) La senda óptima es la encontrada por la búsqueda de coste uniforme (coste 130). Claramente, la función h(n) no es admisible, ya que el algoritmo A no ha encontrado la solución óptima.

Exercise 26

A passenger in an airport is located at spot A and wants to reach the gate (Gate) passing by the kiosk (Kiosco) to buy the newspaper and the bar (Bar) to have a coffee (see figure below). The thick lines represent passenger conveyors that run in only one direction whereas the thin lines represent corridors to move from one spot to another in both directions. Numbers labelling the edges (conveyors or corridors) denote the time to move from spot X to spot Y. Values inside the nodes show the estimated time to move from spot X to the Gate.



a) Calculate the solution path from spot **A** to **Gate** by applying an A algorithm and avoiding repeated states. A state in this problem includes the following information: the spot at which the passenger is, whether the passenger has already dropped by the **Kiosko** or not, and whether the passenger has already passed by the **Bar** or not. Consequently, two states are repeated if they contain exactly the same information regarding these three items, no matter the path to reach the states. For instance, the path A→B→Kiosco→D and the path A→C→Kiosco→D reach the same state as in both states the passenger is at spot D and the passenger has already passed by the Kiosco but not yet by the Bar. Put another way, a solution path is not a path that simply reaches the Gate but a path that also traverses the Kiosco and the Bar.

Show clearly:

- The tree that results from the search process. Show the solution path.
- The order of the nodes expansion.
- The cost of the solution path.
- The number of generated and expanded nodes.

Note: *If two nodes have the same f-value, expand first the* ***deepest*** *node.*

b) The heuristic function 'h', is it an A* heuristic? Why? Justify your answer.

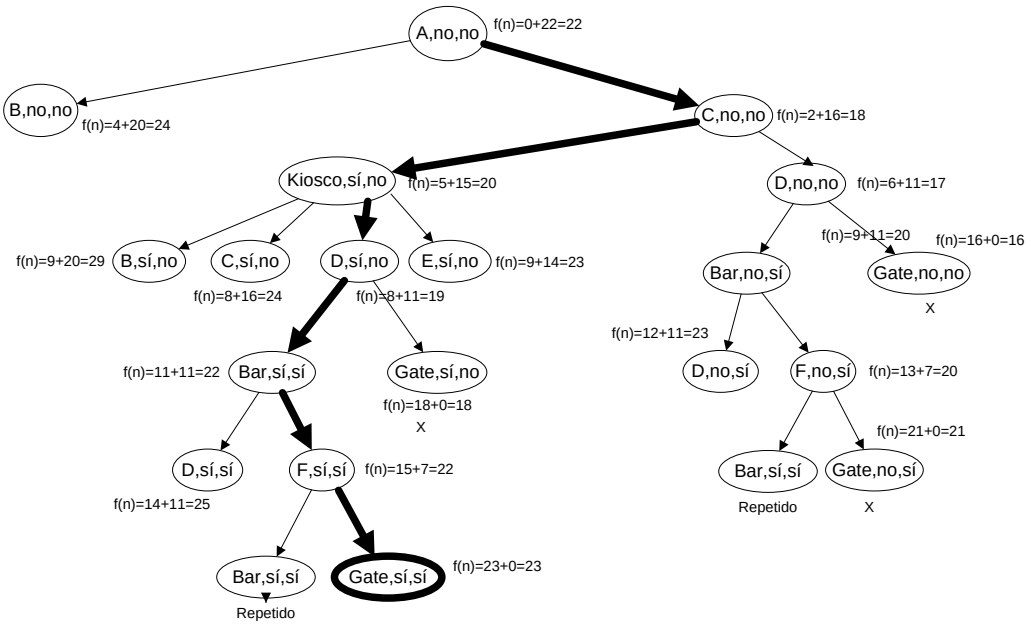
c) Apply again an A algorithm with the following heuristic function: given a state 's', where the current location of the passenger is 'n', the estimated cost to reach the Gate is: the h(n) value indicated in the figure above, plus 10 if the passenger has not passed by the Kiosco yet, plus 10 if the passenger has not passed by the Bar yet. For example, given the state 's' resulting from the path A→B→Kiosco→D, h(s)=11 (because the passenger is at spot D) + 10 (because the passenger has not visited the Bar yet)= 21.

Show clearly:

- The tree that results from the search process. Show the solution path, the order of expanding the nodes, the cost of the solution path, and the number of generated and expanded nodes.
- Compare this heuristic with the one in section a). Is this new heuristic admissible? Justify your answer.

Solution:

a) Árbol de búsqueda:
Representamos cada estado con la tripleta (posición del pasajero, kiosco visitado (sí/no), bar visitado (sí/no)).

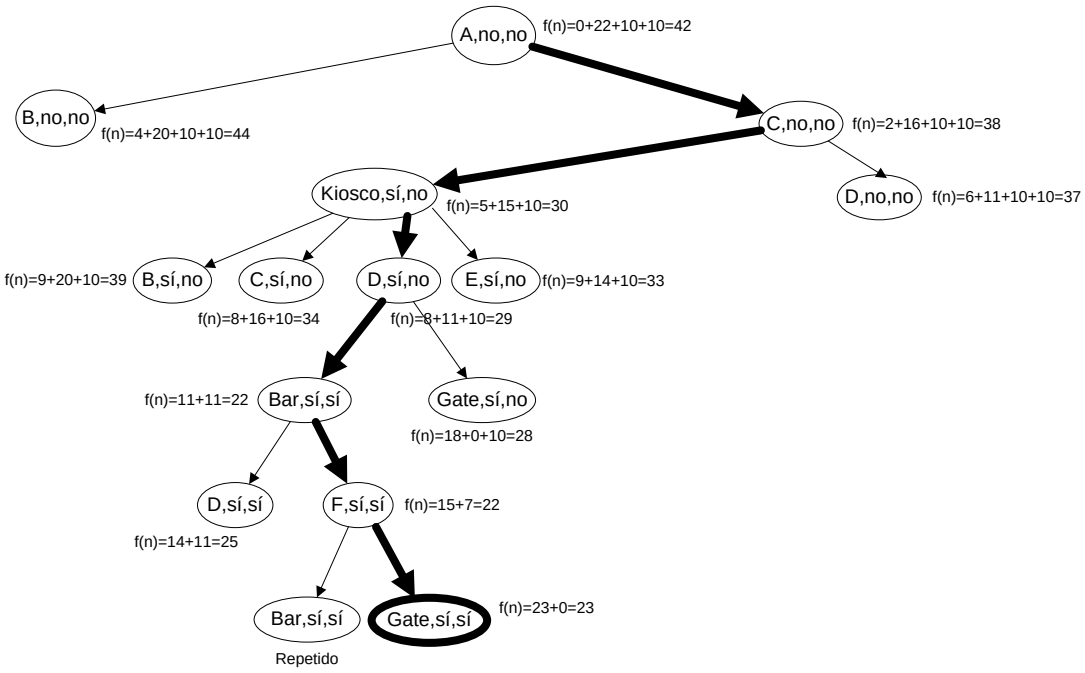


1. OPEN={(A,no,no,22)}. Se extrae el nodo A y se expande. Se añaden B y C.
2. OPEN={(C,no,no,18), (B,no,no,24)}. Se extrae el nodo C y se añaden Kiosco y D.
3. OPEN={(D,no,no,17), (Kiosco,sí,no,20), (B,no,no,24)}. Se extrae D y se generan Bar y Gate, que se añaden a OPEN).
4. OPEN={(Gate,no,no,16), (Bar,no,sí,20), (Kiosco,sí,no,20), (B,no,no,24)}. Se extrae Gate. No es meta, ya que no se han visitado ni el kiosco ni el bar. Desde éste, no se puede generar ningún otro nodo. Se extrae Bar y se generan D y F, que se añaden a OPEN. F se ordena antes que Kiosco, por estar en un nivel de profundidad mayor en el árbol.
5. OPEN={(F,no,sí,20), (Kiosco,sí,no,20), (D,no,sí,23), (B,no,no,24)}. Se extrae F y se genera Gate.
6. OPEN={(Kiosco,sí,no,20), (Gate,no,sí,21), (D,no,sí,23), (B,no,no,24)}. Se extrae Kiosco, y se generan B, C, D y E (ninguno es un nodo repetido).
7. OPEN={(D,sí,no,19), (Gate,no,sí,21), (D,no,sí,23), (E,sí,no,23), (C,sí,no,24), (B,no,no,24),(B,sí,no,29)}. Se extrae D y se generan Bar y Gate.
8. OPEN={(Gate,sí,no,18), (Gate,no,sí,21), (Bar,sí,sí,22), (D,no,sí,23), (E,sí,no,23), (C,sí,no,24), (B,no,no,24),(B,sí,no,29)}. Se extrae Gate y no es un estado meta porque no se ha visitado el bar. Se extrae el siguiente nodo Gate, que tampoco es un estado meta porque no se ha visitado el kiosco. Se extrae Bar y se generan D y F.
9. OPEN={(F,sí,sí,22), (D,no,sí,23), (E,sí,no,23), (C,sí,no,24), (B,no,no,24), (D,sí,sí,25), (B,sí,no,29)}. Se extrae F y se genera Bar (repetido) y Gate.
10. OPEN={(Gate,sí,sí,23), (D,no,sí,23), (E,sí,no,23), (C,sí,no,24), (B,no,no,24), (D,sí,sí,25), (B,sí,no,29)}. Se extrae Gate y SÍ es un nodo meta, ya que se han visitado tanto el kiosco como el bar.

El coste de esta solución es 23. Se han generado 21 nodos y se han expandido 13.

b) La heurística NO es admisible, ya que el valor indicado en cada nodo (estimación del coste desde ese nodo hasta Gate) NO es siempre menor o igual que el coste real de ir desde ese nodo hasta Gate. Por ejemplo, en el caso del punto D, la estimación es 11 cuando se podría llegar hasta Gate en 10.

c) Árbol de búsqueda con la nueva heurística:



1. OPEN={(A,no,no,42)}. Se extrae A y se generan B y C.
2. OPEN={(C,no,no,38), (B,no,no,44)}. Se extrae C y se generan Kiosco y D.
3. OPEN={(Kiosco,sí,no,30), (D,no,no,37), (B,no,no,44)}. Se extrae Kiosco y se generan B, C, D y E.
4. OPEN={(D,sí,no,29), (E,sí,no,33), (C,sí,no,34), (D,no,no,37), (B,sí,no,39), (B,no,no,44)}. Se extrae D y se generan Bar y Gate.
5. OPEN={(Bar,sí,sí,22), (Gate,sí,no,28), (E,sí,no,33), (C,sí,no,34), (D,no,no,37), (B,sí,no,39), (B,no,no,44)}. Se extrae Bar y se generan D y F.
6. OPEN={(F,sí,sí,22), (D,sí,sí,25), (Gate,sí,no,28), (E,sí,no,33), (C,sí,no,34), (D,no,no,37), (B,sí,no,39), (B,no,no,44)}. Se extrae F y se genera Bar (repetido) y Gate.
7. OPEN={(Gate,sí,sí,23), (D,sí,sí,25), (Gate,sí,no,28), (E,sí,no,33), (C,sí,no,34), (D,no,no,37), (B,sí,no,39), (B,no,no,44)}. Se extrae Gate y es un nodo meta.

La solución encontrada es la misma que aplicando la primera heurística. Se generan 15 nodos y se expanden 7. Esta heurística NO es admisible, ya que, por ejemplo, el valor de h(n) del nodo A es 42, cuando el coste de la solución es 23. Sin embargo, esta heurística permite enfocar la búsqueda claramente hacia la solución, porque asigna un menor valor a aquellos nodos que ya han visitado el kiosco y/o el bar y que, por tanto, están más próximos a la solución.

Exercise 27

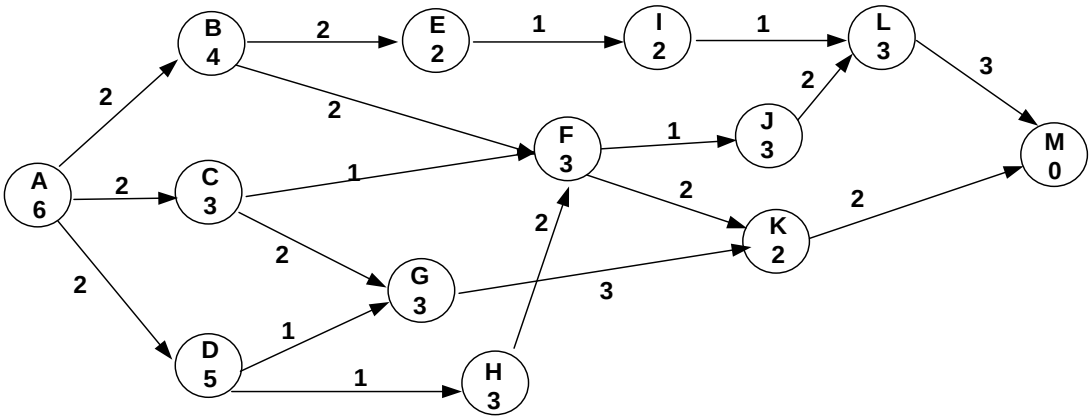
The graph below represents a search space where the initial state is node A and the goal state is M. The values on the edges represent the cost of the arc to go from one node to the other, and the numbers inside the nodes denote the value of a heuristic function h(n).

a) Apply an A algorithm to the graph of the figure (avoid repeated states). Draw the search tree, indicate clearly the f-values of the nodes and the order in which nodes are expanded. Say the cost of the solution found, the number of generated and expanded nodes. ***NOTE***: *Note: If two nodes have the same f-value, expand in alphabetical order.*

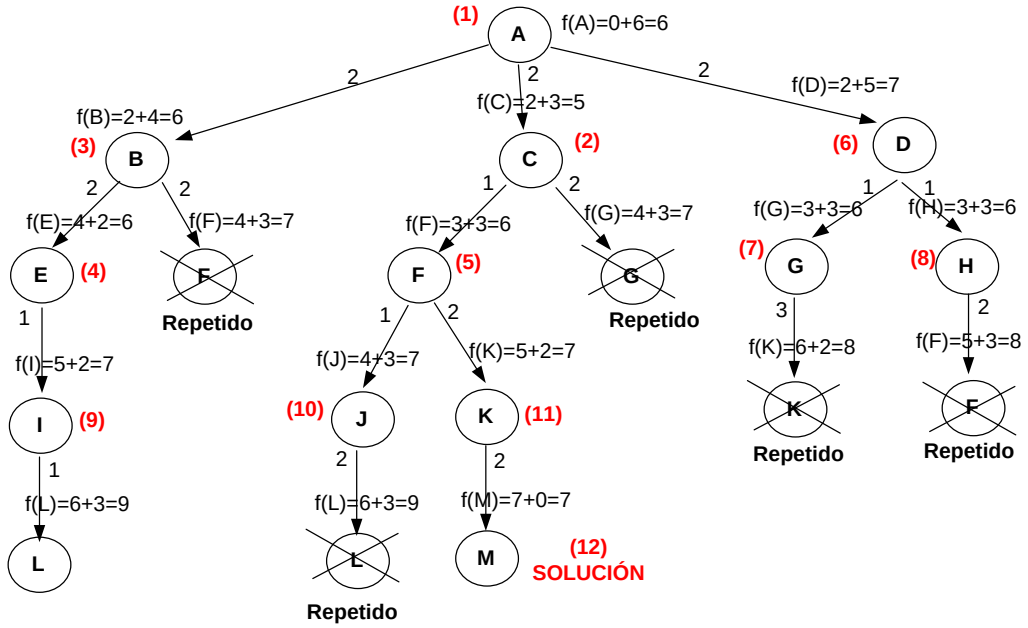
b) The solution found by the A algorithm, is it optimal? The heuristic function 'h', is it an A* heuristic? Why? Justify your answers.

c) The 'h' function is non-monotone (no consistent). Say the values in the graph and/or the tree that indicate the function 'h' is non-monotone and explain the consequences of the absence of this property in the developed tree.

d) Draw the search tree that results from the application of a depth-first strategy with maximum depth level m=4 (avoid repeated states). Show the OPEN and CLOSED lists in each iteration. Which solution returns DFS? Which is the cost of this solution? ***NOTE***: *Note: If two nodes are at the same depth level of the tree, expand in alphabetical order.*



Solution:



Se genera un total de 18 nodos. Se expande un total de 12 nodos.
La solución es A-C-F-K-M. El coste de la solución es 7.

Observaciones: se producen dos tipos de casos de **nodos repetidos**, aquellos nodos que cuando se generan no mejoran el coste del nodo existente, y aquellos nodos que cuando se generan, mejoran el coste del nodo existente. Por ejemplo:

a) el nodo repetido F sucesor de B se genera cuando ya existe en la lista OPEN el nodo F sucesor de C. Por tanto, el nuevo nodo F generado no se inserta en OPEN

b) el nodo repetido G sucesor de C se detecta cuando se genera el nodo G sucesor de D, el cual tiene un coste mejor. Por tanto, en este caso, se elimina de OPEN el nodo G sucesor de C y se inserta el nuevo nodo G.

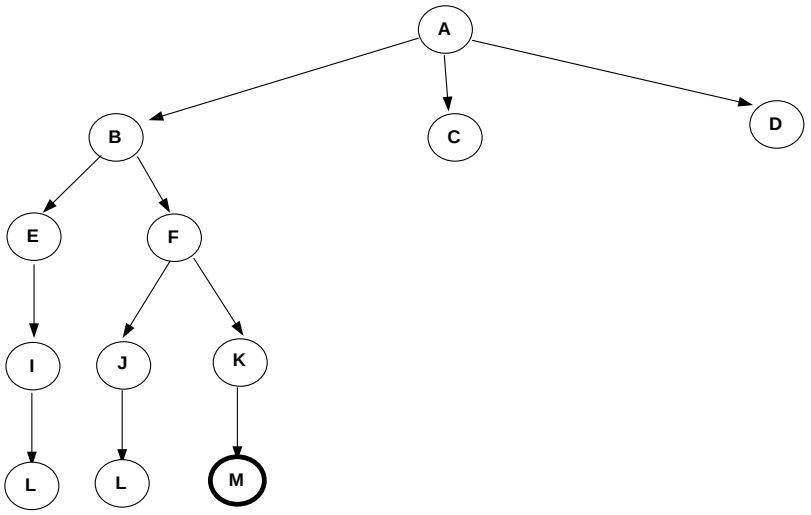
b) Sí, es la solución óptima. La función h(n) es admisible, todas las estimaciones de los nodos son menores ó iguales que el coste real. Por ejemplo, h(L)=3, h*(L)=3, h(K)=2, h*(K)=2, h(J)=3, h*(J)=5.

c) En el grafo. Se puede observar que la función h(n) no es consistente con los costes en 2 puntos: A –> C y D → G. Por ejemplo, de A a C, la heurística desciende en tres unidades, mientras que el coste de la arista es solo 2.

En el árbol. Se puede observar que la función f(n) no es monotónicamente no decreciente, es decir, los valores de f(n) pueden ahora decrecer a lo largo del árbol. Esto se observa en tres puntos: de f(A) a f(C) (el valor pasa de 6 a 5), y de f(D) a f(G) y f(H) (el valor pasa de 7 a 6).

Una de las consecuencias de la no monotonía es que podría expandirse un nodo y posteriormente encontrar otro camino de menor coste a dicho nodo. En el árbol ocurre que se expande el nodo F en el orden (5), y por consiguiente se introduce en la lista CLOSED, y posteriormente se vuelve a generar el nodo F como sucesor de H. En este caso, el camino a F a través de H (coste=5) no es mejor que el que ya se tiene (el coste de F a través de C es 3) y por tanto se descarta el nuevo nodo generado. Pero si el nuevo camino de A a F a través de D y H fuera mejor que el del nodo ya expandido e introducido en CLOSED, habría que re-expandir otra vez el subárbol a partir del nodo F con su nuevo valor de g(n).

d) PROFUNDIDAD m=4



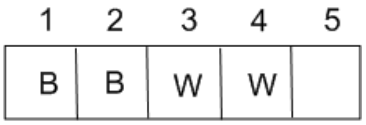
1. OPEN={A} se extrae el nodo A, no es solución y el nivel no es 4. Se inserta en CLOSED={A} y se expande A.
2. OPEN={B C D} se extrae el nodo B, no es solución y el nivel no es 4. Se inserta en CLOSED={A B} y se expande B.
3. OPEN={E F C D} se extrae el nodo E, no es solución y el nivel no es 4. Se inserta en CLOSED={A B E} y se expande E.
4. OPEN={I F C D} se extrae el nodo I, no es solución y el nivel no es 4. Se se inserta en CLOSED={A B E I} y se expande I.
5. OPEN={L F C D} se extrae el nodo L, no es solución y hemos llegado al nivel 4. Backtracking.
6. OPEN={F C D} Se extrae el nodo F, no es solución, y el nivel no es 4. Se inserta en CLOSED y se expande. Se actualiza la lista CLOSED de modo que el nodo anterior en CLOSED sea el nodo padre de F; CLOSED={A B F}
7. OPEN={J K C D} se extrae el nodo J, no es solución y el nivel no es 4. Se inserta en CLOSED={A B F J} y se expande.
8. OPEN={L K C D} se extrae el nodo L, no es solución y hemos llegado al nivel 4. Backtracking.

9. OPEN={K C D} se extrae el nodo K, no es solución, y el nivel no es 4. Se inserta en CLOSED y se expande. Se actualiza la lista CLOSED de modo que el nodo anterior en CLOSED sea el nodo padre de K; CLOSED={A B F K}
10. OPEN={M C D} se extrae M, es solución y lo insertamos en **CLOSED={A B F K M}**

La solución es el camino indicado en CLOSED. El coste de la solución es 8.

Exercise 28

Given the following configuration of a lineal puzzle:



Initial state

where there are two black pieces (B), two white pieces (W) and one empty position, and considering the following actions:

a) A piece can move to an empty adjacent position with cost 1.
b) A piece can jump over another one (only one) to be placed in the following empty position with cost 1. If the piece that jumps and the jumped one have different colours, the jumped one changes its colour.

The objective is to have all the pieces of the board **black coloured** in any position. Thus, the end position of the empty square is indifferent.

1) Consider the following heuristic function: h(n)=$\sum_{i=1}^{5} b(i) * c(i)$, where:

   - b(i)=1, if square (i) contains a W piece; b(i)=0, otherwise
   - c(i)=1, if the empty square is adjacent to square (i), i.e., if the empty space is in square (i-1) or square (i+1); c(i)=2, otherwise.

   In short, the heuristic only considers the W pieces. For each W piece, it counts 1 if the empty square is adjacent to the W piece or it counts 2 if the empty square is not adjacent to the W piece.

   a) Show the search tree generated by applying an A algorithm with f(n)=g(n)+h(n).
   b) Show the *order of expansion* of the nodes in the tree
   c) Show the solution found and the cost of the solution

NOTES: **(a)** Avoid repeated states. **(b)** If two nodes have the same f-value, expand first the **deepest node**.

2) The 'h' function described in 1), is it admissible? Justify your answer.
3) Without developing a new search tree, answer the following questions:
   a) How many iterations (different search trees) would Iterative Deepening do to find a solution? Justify your answer.
   b) If the cost of jumping over another piece were 2 instead of 1, would the above A algorithm find the same solution? Justify your answer.

Solution

Cuestión 1

Planteamos los movimientos en términos de la casilla vacía (mover vacío a la izda, mover vacío a la dcha, saltar vacío a la izda, saltar vacío a la dcha).

La solución encontrada es la que se muestra en negrita en la figura de abajo. El coste de la solución es 5.

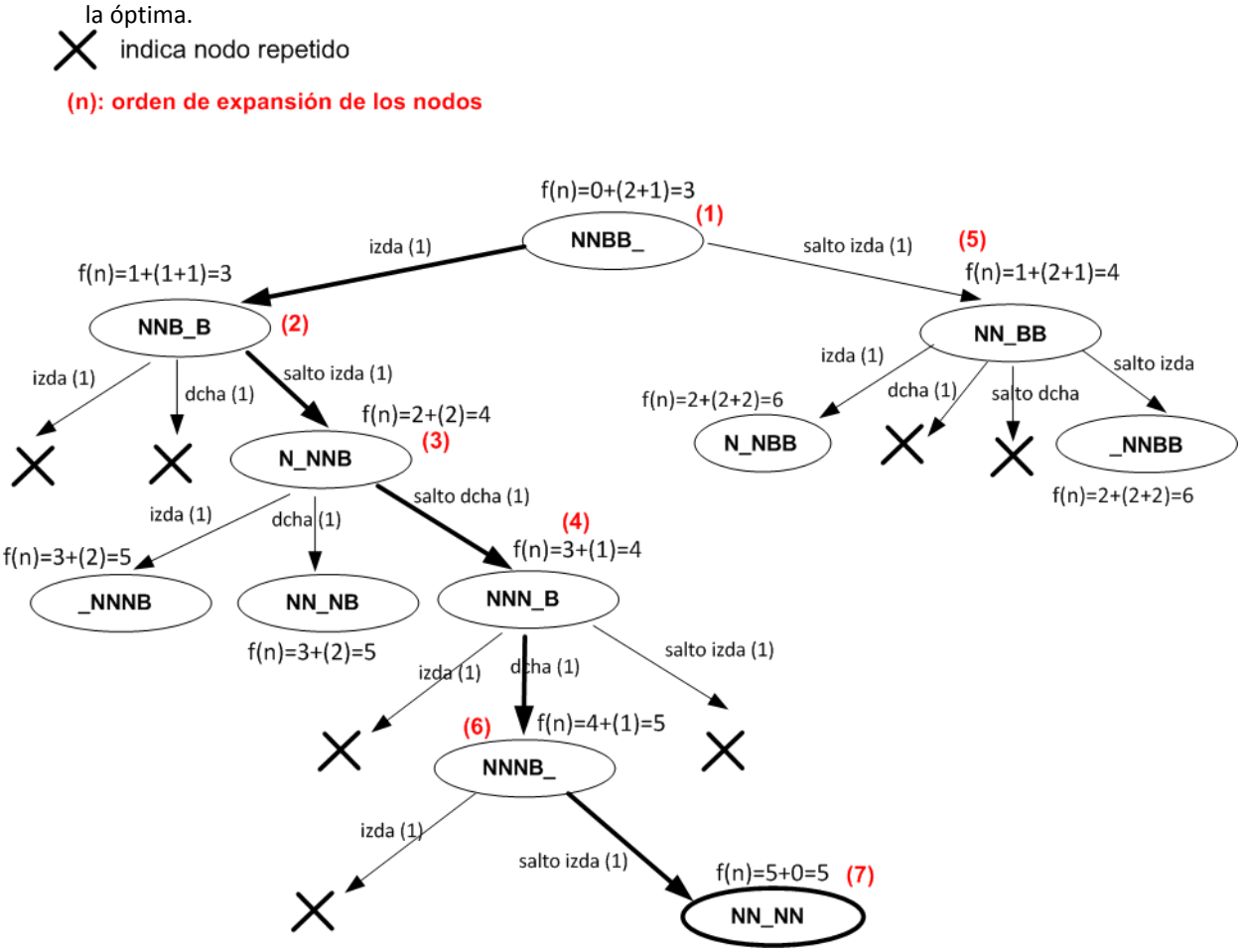
Cuestión 2

Como se puede observar en el árbol, las estimaciones de todos los nodos son menores que el coste real y la solución que ha encontrado tiene un coste de 5 pasos. En este caso, se puede comprobar que se trata de la solución óptima para el estado inicial dado pero no se puede garantizar que el algoritmo encuentre siempre la óptima porque la heurística no es admisible. Particularmente, se puede encontrar al menos un caso en el que la estimación supera el coste real: por ejemplo, para el nodo n= NBNB_ donde h(n)=3 y h*(n)=2.

Cuestión 3

a) Asumiendo que la solución requerirá un mínimo de 5 pasos para el estado inicial dado, una búsqueda por profundización iterativa tendrá que realizar iteraciones para nivel=0,1,..,5.

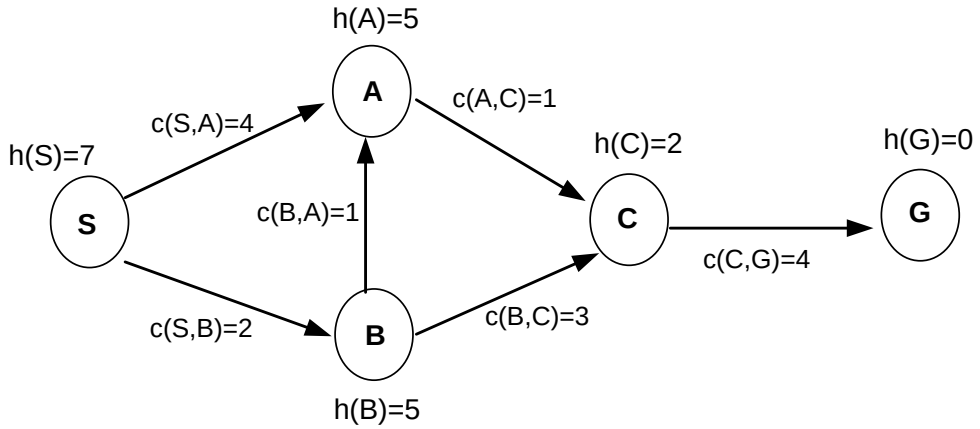
b) Como la heurística no cambia, para todos los nodos del árbol para los que se cumple h(n)<h*(n) se seguirá cumpliendo ahora h(n)<<h*(n). Particularmente, la sobreestimación que se ha comentado en la cuestión 2 ahora no se produciría con los nuevos costes (por ejemplo, para el nodo n= NBNB_ donde h(n)=3, ahora h*(n)=4), por lo que la heurística sería admisible en este caso. Podría por tanto encontrar la misma solución (ahora con coste=8) o bien otra solución distinta con el mismo coste que la óptima.



Exercise 29

Exercises Block 1- Search

Given the following state space where S is the initial state and G is the goal state:

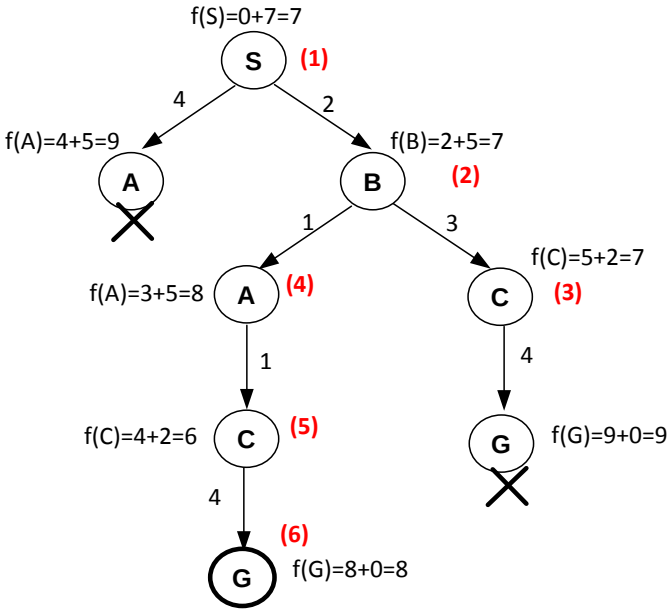


, h(X) denotes the value of applying the heuristic function 'h' in state X; c(X,Y) is the cost of the operator to move from state X to state Y. Answer the following questions:

a) Apply an A algorithm with f(n)=g(n)+h(n). Does it find the optimal solution?
b) Is the 'h' function admissible? Is the 'h' function consistent?

Solution

(n): orden de expansión de los nodos



a) Si se hace una búsqueda en grafo manteniendo la lista de nodos cerrados CLOSED entonces al encontrar el nodo repetido C con f(C)=6 podrían suceder dos cosas:

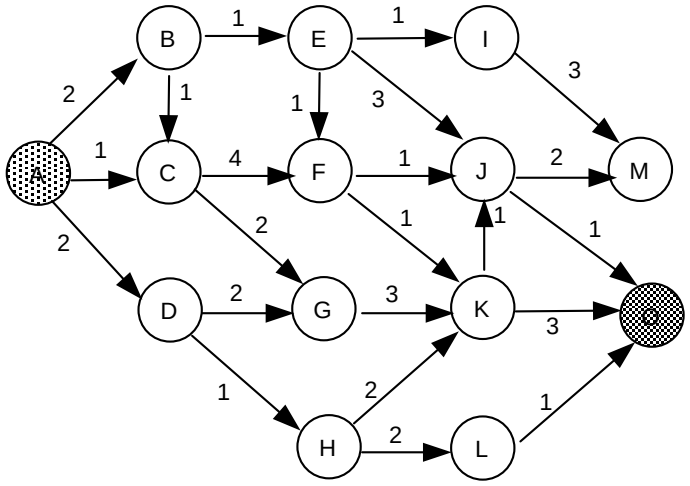
1. Se ignora este nodo por estar repetido en la lista CLOSED. En este caso expandiríamos a continuación el nodo G con valor f(G)=9 y no se encontraría la solución óptima.
2. Se comprueba que el coste de la senda del nodo S a C por el nodo A es mejor que la senda previa por B. Se rehacen los punteros de (C) desde (B) hacia el nodo (A). El nodo G que está en la lista OPEN actualiza su valor a f(G)=8+0=8, expandiéndolo a continuación y encontrando así la solución óptima.

b) la heurística es admisible porque se cumple para todos los nodos que h(n)<=h*(n). Por ejemplo: h(S)=7, h*(S)=8, h(A)=5, h*(A)=5, etc. Pero no es consistente. Esto se puede observar de dos modos distintos:

- bien en el árbol desarrollado donde se ve que el valor de 'f' es decreciente del nodo A al nodo C
- bien comprobando que la propiedad h(n)<=h(n')+c(n,n') no se cumple entre A y C porque h(A)>=h(C)+c(A,C);

Exercise 30 (Exam 2013)

Arcs in the graph below are labeled with the cost to go from one node to the other. The table shows the estimated cost 'h' to reach the solution. The initial state is the node 'A' and the final state is the node 'O'.



n	A	B	C	D	E	F	G	H	I	J	K	L	M	O
h(n)	6	5	6	6	3	5	5	4	8	2	2	1	5	0

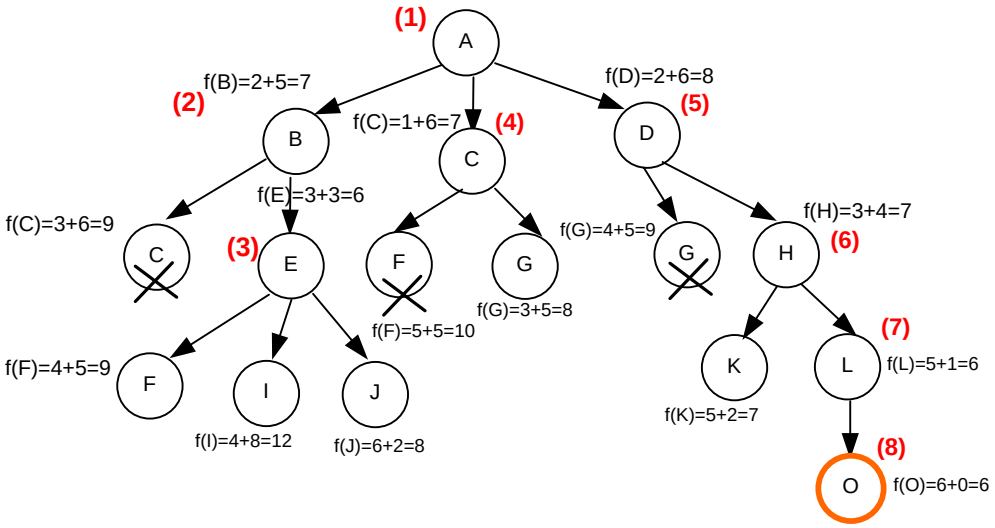
1. Show the search tree that would result from the application of an A algorithm (f(n)=g(n)+h(n)). Apply the graph-search version avoiding repeated states. Indicate the number of generated nodes and expanded nodes. Show clearly the value of the evaluation function (f(n)) in each node and the order of expansion of the nodes. If two nodes have the same f-value, expand first the node that comes alphabetically before.

2. According to the problem data and the tree developed in the previous section, does the algorithm return the optimal solution? Is the heuristic function admissible? And consistent (monotone)? Provide justifications for all your answers.

3. Answer the following questions without need of building a tree. Justify all your answers.
   a. Which strategy would you use if we want to find a solution that goes through the least number of nodes? Show a solution that this strategy would find.
   b. If we apply an iterative deepening depth-first algorithm, in which level would it find a solution? Why?
   c. If we apply a depth-first algorithm and the user does not set a maximum depth level, would the algorithm find a solution? Why?

Solution

1) 16 nodes generated, 8 nodes expanded



2)

Sí, el algoritmo devuelve la solución óptima. Se puede ver en el grafo que no hay solución de menor coste.

No, la función no es admisible. Esto se puede observar en algunos casos, por ejemplo: h(H)=4, h*(H)=3; h(J)=2, h*(J)=1.

No, si la función no es admisible no puede ser consistente ya que la consistencia es una propiedad más fuerte que la admisibilidad. Esto se puede observar, por ejemplo, en: h(H)=4, h(L)=1, c(H,L)=2, por tanto no se cumple que h(H)<= h(L)+c(H,L). Esto también se puede observar entre el nodo B y E, D y H ó H y L, prueba de lo cual es que la función f(n) es decreciente en el árbol desarrollado.

3)

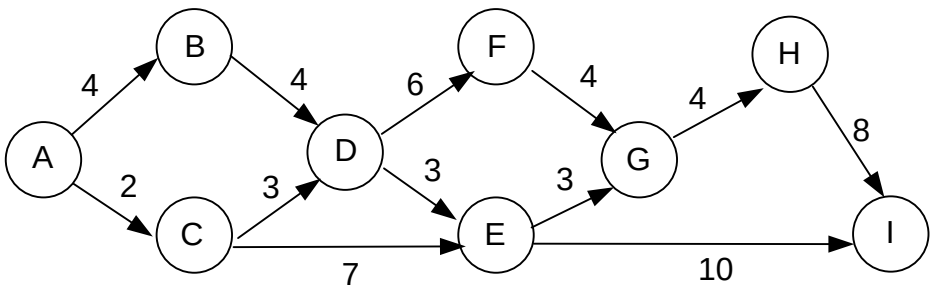
a) Para encontrar la solución más corta aplicaríamos la estrategia de anchura. Una solución que encontraría anchura es A-B-E-J-0, que es la solución más corta posible para este problema (4 pasos).

b) El algoritmo Profundización Iterativa encontraría la solución en el nivel 4 porque PI encuentra una solución de la misma calidad que anchura.

b) Sí, el algoritmo encontraría una solución porque el espacio de estados es finito y no contiene ciclos por tanto la búsqueda en profundidad no se quedaría estancada en un espacio de búsqueda infinito.

Exercise 31 (Exam 2013)

Consider the graph below where arcs are labeled with the cost to go from one node to the other. The table shows the estimated cost (h) to reach a solution. The initial state is node 'A' and the goal state is node 'I'.



n	A	B	C	D	E	F	G	H
h(n)	15	14	13	12	11	10	9	8

1) Assuming we apply a breadth-first algorithm (in case of the same value of the function 'f(n)', the node that comes alphabetically before is expanded first) with control of repeated states (discard deeper nodes or nodes that have been expanded before in case of two nodes at the same depth level).

   a) Write the nodes of the solution path from node A to node I.
   A C E I

   b) How many nodes have been generated and how many expanded in the breadth-first tree?
   12 generated nodes (3 of them are repeated nodes), and 8 expanded nodes (including node I)

2) Assuming we apply the graph version of an A algorithm with control of repeated states:

   a) Write the nodes of the solution path from node A to node I and the cost of the solution path. is the solution found optimal?
   A C D E I; the cost of the solution path is 18.
   Yes, it is the optimal solution because it does not exist a path of lower cost between node A and node I

   b) How many nodes have been generated and how many expanded in the developed tree?
   10 generated nodes (2 of them are repeated nodes), 6 expanded nodes (including node I)

   c) Show the nodes in the order in which they are expanded.
   A C D B E I

3) Answer briefly the following questions. Justify your responses:

   a) Is the heuristic function admissible? Why?
   No, because h(E)=11 and h*(E)=10

   b) Is the heuristic function consistent? Why?
   No, because h(E) <= h(I)+c(E,I) does not hold; i.e., it is not true that 11 <= 0 + 10

   c) If we apply an iterative deepening (ID) algorithm for this problem, which solution would ID find? Show the nodes of the solution path as well as the total number of generated nodes.
   Same solution as breadth-first, A C E I.

   4 depth-first trees are generated, from level 0 up to level 3. The total number of generated nodes is, counting the nodes of each tree: 1+3+6+12=22.

   We might also compute the number of generated nodes by counting the nodes generated at each depth level:

   4 times * 1 (the node at level 0) +
   3 times * 2 (two nodes at level 1) +
   2 times * 3 (three nodes at level 2) +
   once *6 (four nodes at level 3) = 22.