Informed search

The uninformed methods are very inefficient due to the combinatorial explosion, on the other hand, **informed** or **heuristic methods** use domain knowledge to guide the search. For this, information about the proximity of each state to a target state is provided, also with this information we reduce the complexity of the combinatorial explosion while exploring, we call this information **heuristic**. But it has some limitations such as that it doesn't prevent the combinatorial explosion, if the heuristic is not reliable, the efficiency gets worse and in some cases a solution is not guaranteed.

In this type of search we must determine a **Heuristic function h(n)**, the value that the function returns is evaluated as a number that provides an estimate of how "promising" the state is in reaching a target state. We can interpretate the function in 2 ways:

- By estimating the "quality" of a state.
- By estimating the cost of a state.

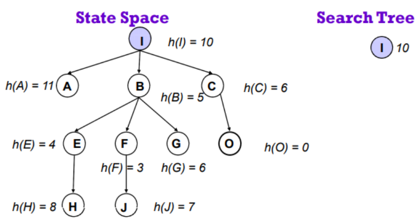
There's an agreement in which we cannot have negative heuristic values (the lower the value the better) and the state that has assigned the value 0 for the heuristic is the target state.

Best-first search

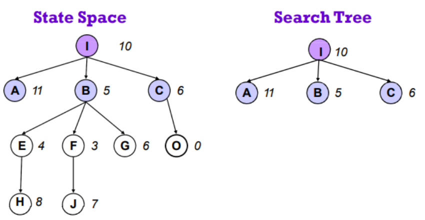
This type of search uses an **evaluation function f(n)** that gives the cost of the cheaper path from the initial state to the target. It uses a graph-search with a priority queue for the list of candidates to be expanded (open-list). The algorithm is **optimal**, **complete**, it has the **least possible complexity**, but it is not a search.

For the evaluation function we have two types:

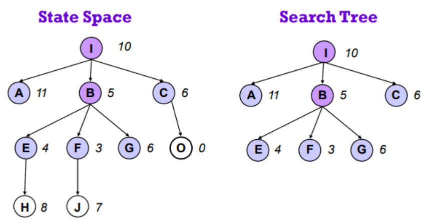
1. f(n) = h(n) ***Greedy best-first search***
   It uses just the heuristic function. It is **not optimal nor complete**, in the worst case, if the heuristic is poor it may have a worse performance than uninformed $O(b^m)$, but if the heuristics are good the performance could be better.



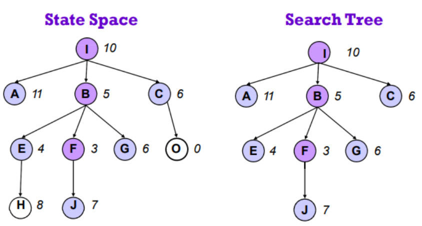
opened-list (priority queue): (I_{10})
closed-list : ( )



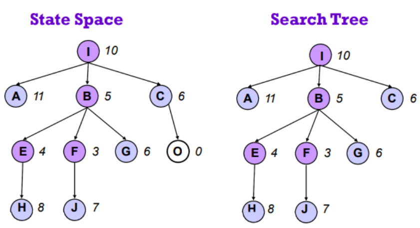
opened-list (priority queue): (B_5 C_6 A_{11})
closed-list : (I)



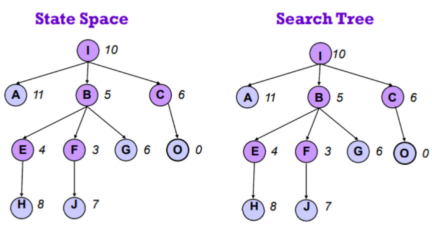
opened-list (priority queue): (F_3 E_4 C_6 G_6 A_{11})
closed-list : (B I)



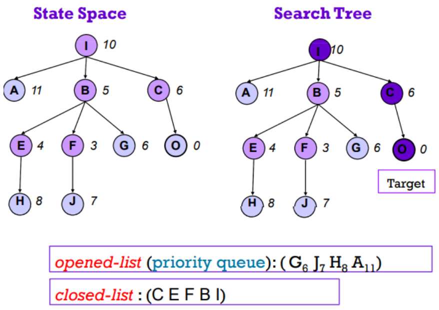
opened-list (priority queue): (E_4 C_6 G_6 J_7 A_{11})
closed-list : (F B I)



opened-list (priority queue): (C_6 G_6 J_7 H_8 A_{11})
closed-list : (E F B I)



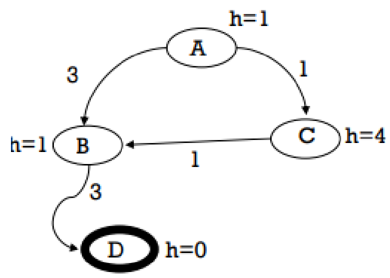
opened-list (priority queue): (O_0 G_6 J_7 H_8 A_{11})
closed-list : (C E F B I)



2. f(n) = g(n) + h(n) ***A* search***
g(n) is the real cost of the path to n, and h(n) is the heuristic function. Here breadth-first and depth-first **are combined**. We say that **h** is an admissible heuristic if $h(n) \leq g*(n)$ for all n. So, the A* search without elimination of repeated states and with an admissible heuristic is **optimal**.
A* may be implemented with graph-search, if this is the case, even if h is admissible, A* **may not be optimal**, but if we use a tree-search (without elimination of repeated states) and if **h** is admissible, A* **is complete and optimal**.
A heuristic function is **monotonic** if it satisfies the following triangular equality:

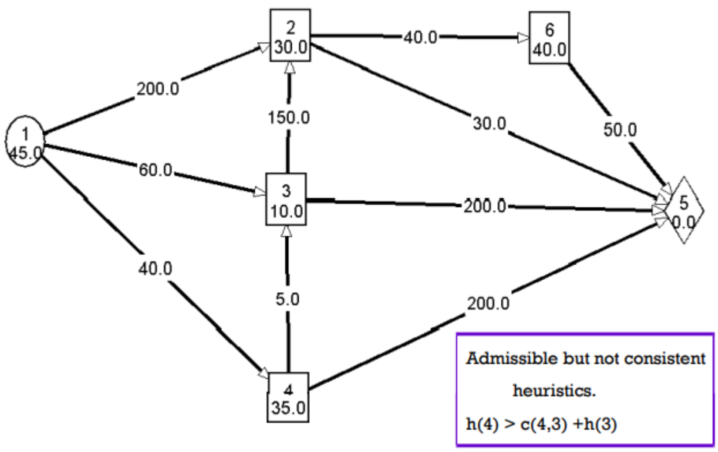
$$h(n) \leq cost(n \rightarrow n') + h(n)$$

If h is monotonic, it is also admissible.
For example, h is monotonic because in the node A we have that:
h(A) = 1, cost(A→B) = 3, h(B) = 1. 1 ≤ 3 + 1, the condition is fulfilled.



Another property of a heuristic is **consistency**, so h is consistent if, having a node **n** and a successor **n'**, if the estimated cost of reaching the target from **n** is not greater than the real cost of reaching **n'** plus the estimated cost of reaching the target from **n'**, then it is **consistent**.

$$h(n) \leq c(n, n') + h(n')$$



This is an example of a non-consistent heuristic.

So, **A* using a tree-search (without elimination of repeated states) is optimal if the heuristic is admissible**, and **A* using a graph-search (with elimination of repeated states) is optimal if the heuristic is consistent**.

IDA* search

IDA refers to an iterative deepening A* search.