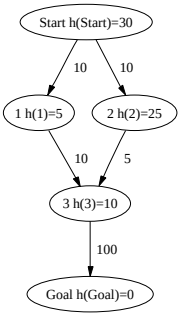
An example where graph search based on A* fails because of an inconsistent heuristic

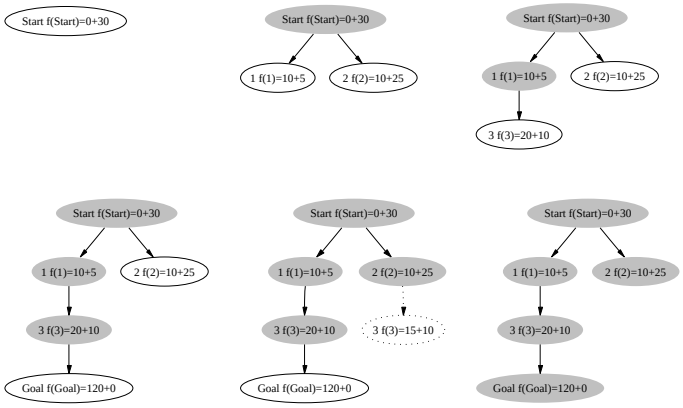
The following gives an example where graph search based on A* fails because an inconsistent heuristic is used. I went through a slightly different variant of this example on the whiteboard in class.

The graph that we want to process using A* looks like this:



The graph is a directed one to indicate that the actions are not reversible (to simplify the example). The states show the values of the heuristic function corresponding to them. Note that the heuristic is admissible—it never overestimates the cost of getting to the goal—so applying the tree search algorithm (as opposed to graph search) is guaranteed to give us the optimum route.

If we use the graph search algorithm to process this problem using the A* evaluation function to sort the fringe, then the following happens in the different steps of the main loop (each node in the search trees shown below shows the value of the evaluation function for that node; gray nodes have been expanded or identified as a goal; the node with the dotted line is never considered for expansion because it refers to a state that has been visited before and thus resides in the "closed" data structure in the graph search algorithm):



In this example, the route that is found is not optimum because the node corresponding to the optimum route is blocked from being expanded because it refers to a state that has been visited before. Obviously, this problem does not occur in the tree search variant of the algorithm, where we do not check whether a node refers to a state that has been visited before.

One way to fix this problem is to modify the graph search algorithm to use a more sophisticated method for dealing with states that have been visited before, replacing sub-optimum paths if necessary. This makes the code for graph search quite a bit more complicated. A more elegant solution is to ensure that the heuristic function $h(n)$ is consistent. The problem in the example only occurs because $10+h(3) \leq h(2)$. If the heuristic function were consistent (check the definition of consistency) then this could not happen because consistency would require that $h(2) \leq 5 + h(3)$, which would imply that $h(2) < 10 + h(3)$. This would mean that node 2 would be expanded before the sub-optimum node refering to state 3, and the optimum node corresponding to state 3 would be added to the fringe, subsequently selected for expansion, and thus the optimum solution would be found.

As mentioned in class, it can be shown that A* based on the graph search algorithm is guaranteed to be optimal if a consistent heuristic is used.