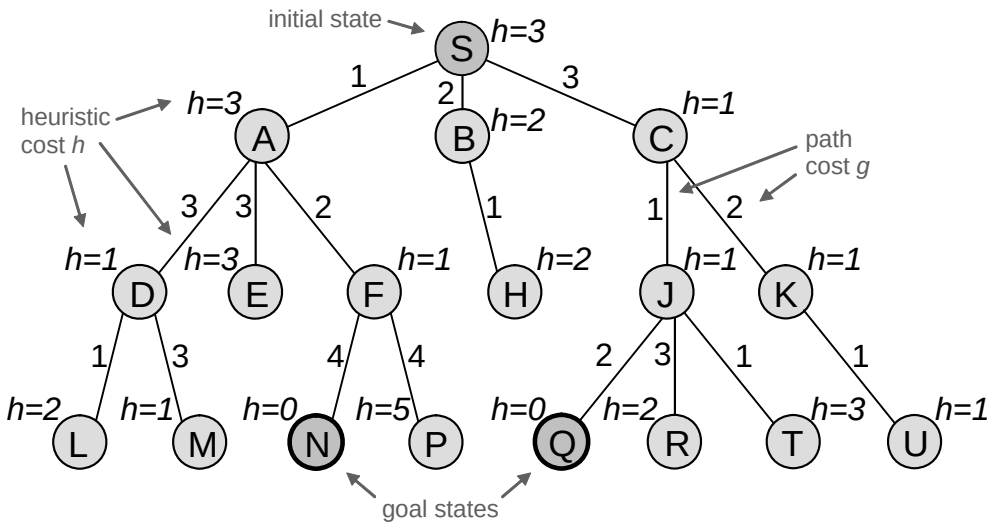
1.1 Explain which *search algorithm* is most appropriate in the following situations:

(a) We have a very large search space with a large branching factor and with possibly infinite paths. We have no heuristic function. We want to find a path to the goal with minimum number of states.

(b) We have a state space with lots of cycles and links of varying costs. We have no heuristic function. We want to find the shortest path.

(c) Our search space is a tree of fixed depth and all the goals are at the bottom of the tree. We have a heuristic function and we want to find any goal as quickly as possible.

1.2 Consider the search problem defined by the annotated search tree below.



(a) Apply the standard *A* search algorithm.* Draw all generated nodes, write their f-costs, and number expanded nodes in order of expansion. Assume that the children of a node are processed in alphabetical order, and that nodes of equal priority are extracted from the search queue in FIFO order.

(b) State how many nodes were generated and how many were expanded. Comment on the solution obtained and the *effectiveness* of the search. What do you think of the *heuristic function h* employed?

1.3 The *w-A** search algorithm is a *weighted* variant of *A** that places more emphasis on the heuristic function by using the *f*-cost $f_w(n) = g(n) + w \times h(n)$, for any $w > 1$.

(a) Similarly to question 1.2a, apply the *w-A** search algorithm for $w = 2$.

(b) Similarly to question 1.2b, comment on the *performance* and usefulness of the *w-A** search algorithm – in this case and in general.