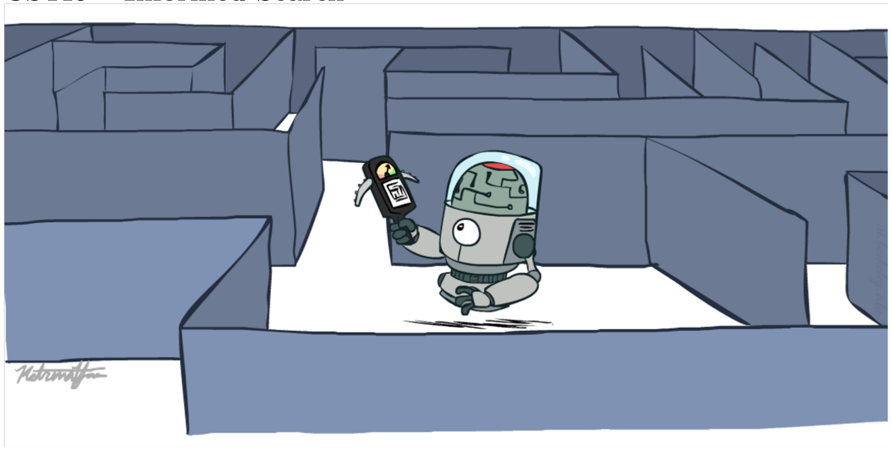
CS440 – Informed Search



Purpose: We have been not using all the information we could in performing our searches. It is possible to choose the next node in the fringe using knowledge about the problem and general rules of thumb, or heuristics, on how which direction we should take. We will see that properly chosen heuristics will lead to optimal solutions.

The purpose of this module is to present algorithms for informed searches which use heuristics.

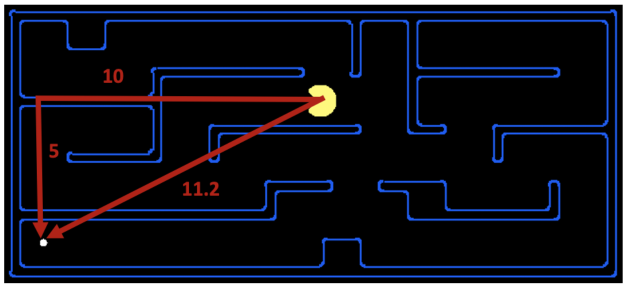
Knowledge: This module will help you become familiar with the following content knowledge:

- The properties of an admissible heuristic
- The A* algorithm

Activity1 - Heuristics and the A* algorithm:
With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 30 minutes.

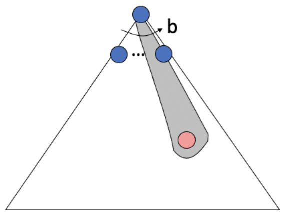
1. A **heuristic** is a function that *estimates* how close a state is to a goal state and is designed for a particular problem.

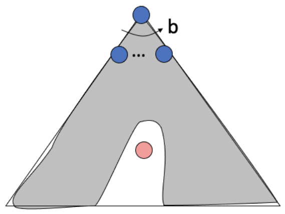
   To be *admissible*, a heuristic should never over estimate the cost.

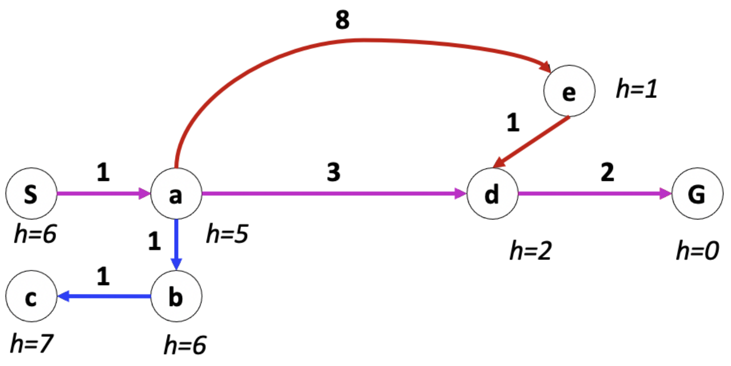
   Give a couple possible admissible heuristics for the pathing problem.

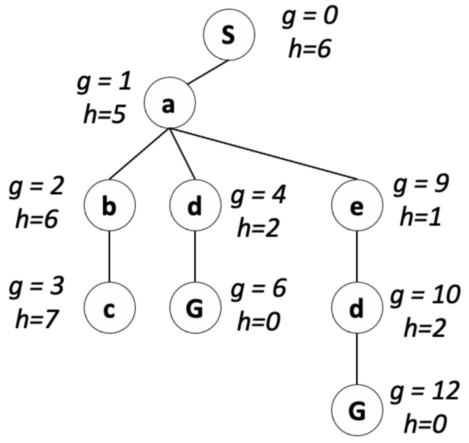
2. We could use the heursitic in a Greedy Search by expanding the node in the fringe that you think is closest to the goal. What could possibly go wrong?

3. Uniform Cost Search orders the nodes by past cost or *backwards cost*, $g(n)$.
Greedy Search orders the nodes by goal proximity or *forwards cost*, $h(n)$
A* Search orders the nodes by $f(n) = g(n) + h(n)$

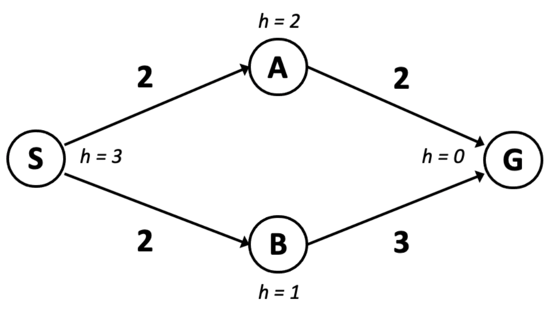




Verify the paths for all three methods.

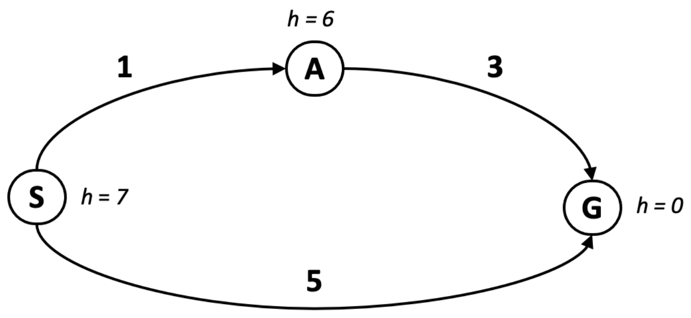
Activity2 - Analysis of the A* algorithm:
With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 30 minutes.

1. Draw the search tree with the f costs.



   What path did you get?
   Does A* terminate when you enqueue a goal or when you dequeue a goal?

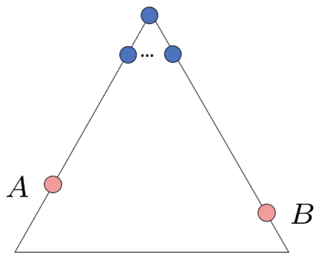
2. Draw the search tree with the f costs.



   What went wrong?

3. We will show that A* algorithm will always find the optimal goal, provided the heuristic is admissible.

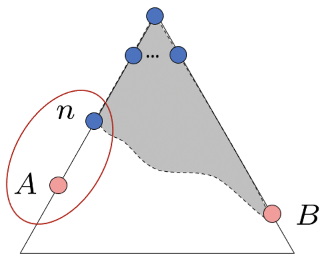
   Assume A is an optimal goal node, B is a suboptimal goal node, and h is an admissible heuristic. We will show that A will exit the fringe before B with the A* algorithm.



Imagine B is on the fringe. Some ancestor n of A is on the fringe, too (maybe A!).

I claim that n will be expanded before B. This will take a few steps.

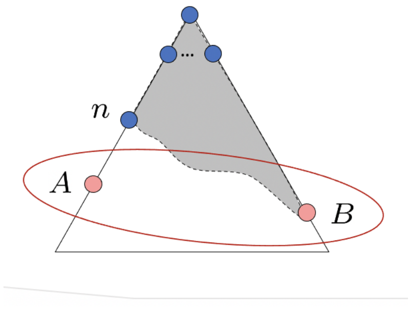
We will first show that $f(n) \leq f(A)$



Explain each of the three steps:

$f(n) = g(n) + h(n)$
$f(n) \leq g(A)$
$g(A) = f(A)$

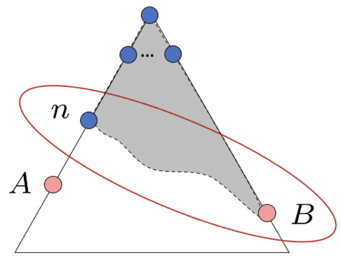
We will now show that $f(A) < f(B)$



Explain these steps:

$g(A) < g(B)$
$f(A) < f(B)$

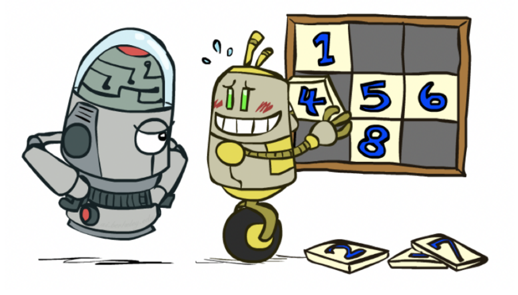
Finally we will put all the pieces together to show that n expands before B:



Given what we have shown so far, $f(n) \leq f(A) < f(B)$. Why does this show that A* is optimal?

Activity3 - Creating Heuristics and what happens with Graph Search
With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 20 minutes.

1. You may be wondering about how to come up with heuristics. One way to do that is to *relax* the constraints of a problem. Consider the 8-puzzle problem again. We know we can only move tiles by sliding them into the blank but just suppose in a relaxed version of the problem the tiles could be moved any which way.

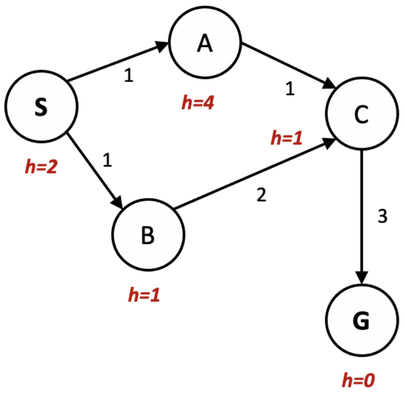
   

   With this relaxed version the cost to get to a goal state would simply be the number of mismatched tiles. Why would the heuristic counting the number of mismatched tiles be admissible?

   Another possible relaxed version of the 8-puzzle is where any tile could slide any direction at any time ignoring any tiles that might be in the way. What heuristic is derived from this relaxed problem? Would this heuristic be better or worse than the number of misplaced tiles?

   How about using the *actual* cost as the heuristic. Would this be admissible? What is wrong with this idea?

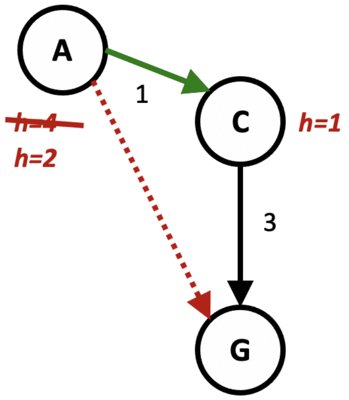
2. Draw the search tree for A* and see what happens when we eliminate duplicates. What is up with that?



The problem is that the heuristic is not *consistent*. Consistency requires that for all edges in the graph $h(A) - h(C) \leq \text{cost(A to C)}$.

The consequences of consistency is that the f value along a path never decreases.

We can make the heuristic consistent in our previous example by making this change:



Draw the search tree again with this new consistent heuristic. Do we get optimal search?