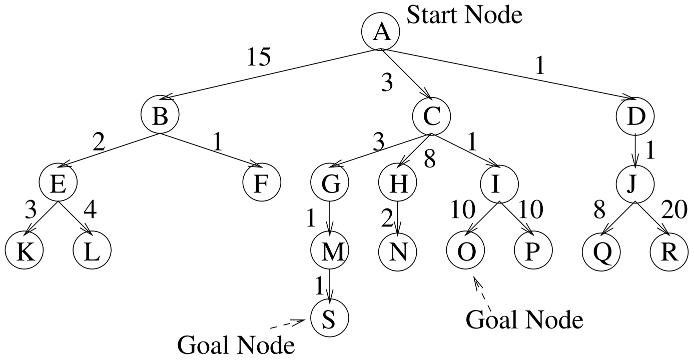
More Search Strategies

1 Uninformed Search: Least-Cost Search

The two strategies we have looked at so far do not take path cost into account. Breadth-first is capable, in general, of finding the *shortest-path*. But neither is, in general, capable of finding the *cheapest* path. (Of course, if all action costs are uniform, e.g. if all actions have a cost of 1, then the cheapest path and the shortest path amount to the same thing. We want to consider here the more general case where different actions can have different costs, and so the shortest path may not be the cheapest path.)

In *least-cost* search, the agenda is a *priority-ordered queue*, ordered by cost. When nodes are added to the agenda, they are added in such a way as to keep the agenda sorted by cost, with the cheapest node at the front. Hence the node that comes off the front of the agenda is always the cheapest unexplored node. (The path that is extended is always the one that currently has the least cost.)

In the lectures, we'll use this state space for our example:



Class Exercise. Is least-cost search complete? Optimal?

2 Informed Search Strategies

In least-cost search, the agenda is a priority-ordered queue, ordered by path cost. By always taking nodes from the front of the queue, the path that we select to extend is always the cheapest so far. In *informed search* (also called *directed search* and *heuristic search*), we continue to use a priority-ordered queue. The ordering is now determined by an *evaluation function*, which for each node on the agenda returns a number that signifies the 'promise' of that node. Perhaps counter-intuitively, we use the convention that smaller numbers designate higher 'promise', so the nodes on our queue will be in ascending order.

One of the most important kinds of knowledge to use when constructing an evaluation function is an estimate of the cost of the cheapest path from the state to a goal state. Functions that calculate such estimates are called *heuristic functions*. (In AI, the word 'heuristic' is not used only in the context of 'heuristic functions'. It is also used for any technique that might improve average-case performance but does not necessarily improve worst-case performance.)

It's important to be clear that a heuristic function is used to evaluate the promise of a *state*. We choose which node to expand next using the heuristic value of its state. Heuristic functions do not evaluate *operators*, i.e. if several operators can be used to expand a node, heuristic functions do not say which is the most promising action.

Heuristic functions are problem-specific. We must design (or learn) different functions for different problem domains. Here are examples of heuristic functions for the 8-tiles puzzle:

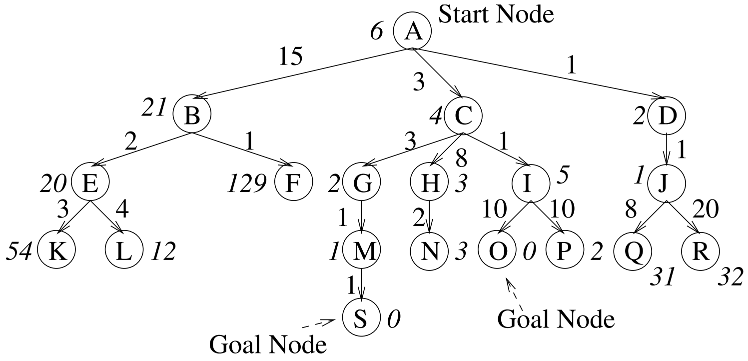
$$h_1(n) =_{\text{def}} \text{the number of tiles out of place in this state relative to the goal state}$$

$$h_2(n) =_{\text{def}} \text{the sum, for each tile, of the Manhattan distance between its position in this state and its position in the goal state}$$

Both functions estimate the number of moves we'll have to make to modify the current state into a goal state. In fact, both h_1 and h_2 *underestimate* the costs of the cheapest paths in this state space, and this turns out to be a significant property (see later).

One general property of a heuristic function is that $h(n) = 0$ if n is a goal.

We'll be illustrating the informed strategies in the lecture on the following simple state space. Path costs are shown along the edges in this graph. The results of applying the heuristic function are shown alongside the nodes.



2.1 Informed Search: Greedy Search

In *greedy search*, the agenda is a priority-ordered queue, ordered only using a heuristic function, h. This means that the node that we select to expand will always be the one whose state is judged, by the heuristic function, to be the one that is nearest to a goal.

Class Exercise. Is greedy search complete? Optimal?

2.2 Informed Search: A^* Search

We've looked at least-cost search and seen that, while complete and optimal, it may not focus the search enough and can therefore be inefficient. And we've looked at greedy search and seen that it may prove too focused (which is a problem if it focuses effort on the wrong parts of the state space). The idea in A^* search is to combine these two approaches.

In A^* *search*, we treat the agenda as a priority-ordered queue but the evaluation function, f, that we use to determine the ordering is:

$$f(n) =_{\text{def}} g(n) + h(n)$$

where $g(n)$ is the cost of the path to node n and $h(n)$ is a heuristic function that estimates the cost of the cheapest path from n to a goal.

We'll therefore be ordering the agenda based on estimates of full path costs (not just the cost so far, and not just the estimated remaining cost, but the two together).

But A^* search places a requirement on function h. (So, a search strategy that uses g and h as above should not be described as A^* search unless h satisfies this condition).

The condition is that h *never overestimates* the cost of the cheapest path to the goal. (Informally, we might say that it must be an optimistic function!) Heuristic functions that meet this condition are called *admissible heuristic functions*.

The heuristics h_1 and h_2 that we gave for the 8-puzzle (number of tiles out of place and sum of Manhattan distances) were both admissible.

If you were doing route planning, your heuristic function could be straight-line distances between two points ('as the crow files'), since this will never overestimate the true distance following roads/rail links/etc. on the ground.

Class Exercise. Is A^* search complete? Optimal?

Exercises (Past exam questions)

1. A^* search uses an evaluation function f:

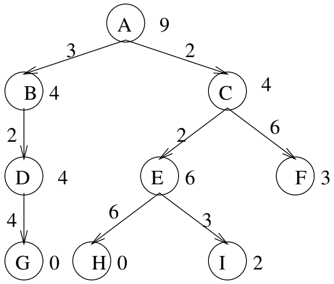
   $$f(n) =_{\text{def}} g(n) + h(n)$$

   where $g(n)$ is the cost of the path from the start state to node n and $h(n)$ is an estimate of the cost of the cheapest path from node n to a node labelled by a goal state.

   Breadth-first search and least-cost search are actually special cases of A^* search.

   - How could you define g and h so that the search carried out is actually a breadth-first search?
   - How could you define g and h so that the search carried out is actually a least-cost search.

2. Consider the following state space in which the states are shown as nodes labelled A through I. A is the start state, and G and H are the goal states. The numbers alongside the edges represent the costs of moving between the states. To the right of every state is the estimated cost of the path from the state to the nearest goal.



   Show how each of the following search strategies finds a solution in this state space by writing down, in order, the names of the nodes removed from the agenda. Assume the search halts when a goal state is removed from the agenda. (In some cases, multiple answers are possible. You need give only one such answer in each case.)

   (a) Breadth-first;

   (b) Depth-first;

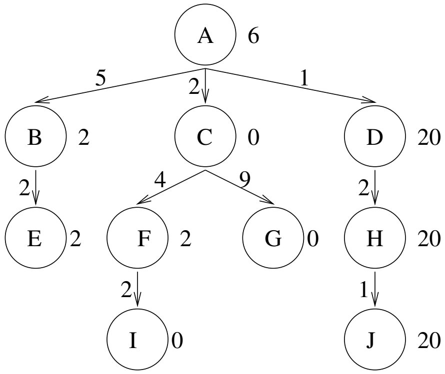
   (c) Least-cost search; and

   (d) Heuristic search using $f(n) = g(n) + h(n)$ as the heuristic function, where $g(n)$ is the cost of the path to node n, and $h(n)$ is the estimated cost of the path from node n to the nearest goal.

3. What does it mean for a heuristic function to be *admissible*?

4. Is $h(n)$ in question 2 admissible?

5. Consider the following state space in which the states are shown as nodes labelled A through J. A is the start state, and G and I are the goal states. The numbers alongside the edges represent the costs of moving between the states. To the right of every state is the estimated cost of the path from the state to the nearest goal.



   Show how each of the following search strategies finds a solution in this state space by writing down, in order, the names of the nodes removed from the agenda. Assume the search halts when a goal state is removed from the agenda. (In some cases, multiple answers are possible. You need give only one such answer in each case.)

   (a) Breadth-first;

   (b) Depth-first;

   (c) Least-cost;

   (d) Greedy search, i.e. heuristic search using $f(n) = h(n)$ as the heuristic function, where $h(n)$ is the estimated cost of the path from node n to the nearest goal; and

   (e) Heuristic search using $f(n) = g(n) + h(n)$ as the heuristic function, where $g(n)$ is the cost of the path to node n, and $h(n)$ is as before.

6. Explain precisely and concisely why A^* search is complete and why it is optimal.

7. *Beam search* is a form of heuristic search using $f(n) = g(n) + h(n)$. However, it is parameterised by a positive integer k. Having computed the successors of a node, it only places onto the agenda the best k of those children. (The k with the lowest $f(n)$ values.)

(a) Beam search is *not* complete. Draw a small state space showing an admissible heuristic and a solution path, but give a value for k for which beam search on your state space would fail to find that solution path.

(b) Beam search is *not* optimal. Draw a small state space showing an admissible heuristic and more than one solution path, but give a value for k for which beam search on your state space would only find the more costly of the two solution paths.

8. A sliding tiles puzzle has room for seven tiles but contains only three black tiles (B) and three white tiles (W). The initial configuration is:

B	**B**	**W**		**B**	**W**	**W**

The goal configuration is:

B	**B**	**B**	**W**	**W**	**W**	

Tiles can move into the adjacent position if it is empty, with a cost of 1. They can also hop over *at most* two other tiles into an empty position, with a cost equal to the number of tiles hopped over (1 or 2).

(a) Give an analogical representation for the *states* of this puzzle.
Give the *start state*.
Give the *goal state*.
Give the *operators*. (Extra marks will be given for precision.)

(b) One possible heuristic function, $h_1(n)$, for the sliding tiles puzzle described above is: if n is a goal state, then $h_1(n) = 0$; for all other states, $h_1(n) = 1$.

- What is the value of $h_1(n)$ applied to the initial configuration above?
- Is $h_1(n)$ admissible? Carefully justify your answer.

(c) Another possible heuristic function, $h_2(n)$, for the sliding tiles puzzle described above is: for each tile position in the current state, excluding the one that is empty, if its current contents do not equal its goal contents, then add one to the estimate.

- What is the value of $h_2(n)$ applied to the initial configuration above?
- Is $h_2(n)$ admissible? Carefully justify your answer.

(d) An AI student claims that $h_2(n)$ is a better heuristic function for this puzzle than $h_1(n)$. Do you agree? Carefully justify your answer.

9. Suppose we have an *admissible* heuristic function h for a state space. Also, for all states n in the state space, $h(n) \geq 0$ and all action costs are positive.

Hence, state whether each of the following is *true* or *false*. To obtain credit, you must in each case **explain** your answer correctly and in detail.

(a) If n is a goal state, then $h(n) = 0$.

(b) If $h(n) = 0$, then n is a goal state.

(c) If n is a 'dead-end' (i.e. it is a non-goal state from which a goal state cannot be reached), then $h(n) = \infty$.

(d) If $h(n) = \infty$, then n is a 'dead-end'.