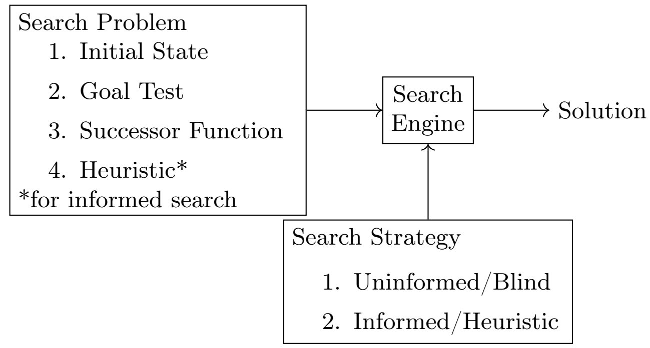
We can now expand on our earlier search problem model:



We will begin our discussion of heuristic searches by discussing best-first search.
This is uninformed, but will lead us naturally into heuristics.

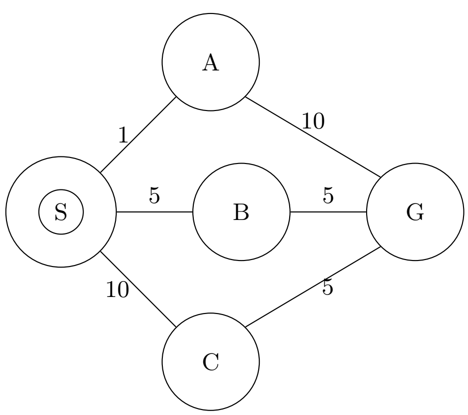
Uniform-Cost Search (UCS)

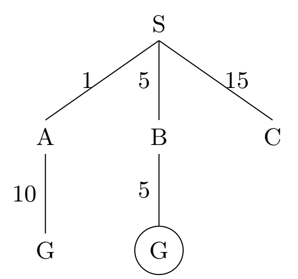
- uninformed generalization of BFS to weighted graphs
- expands based on contours of uniform cost and shape
- we goal check on expansion and not generation or our solution may be suboptimal
- we expand according to the function g(n) := cost of actions from initial state to node n

Properties:

1. complete? YES
2. optimal? YES
3. time? $O(b^{ceil(C*/E)})$
4. space? $O(b^{ceil(C*/E)})$

where E = minimal action cost & C* = optimal solution cost





Unfortunately, this algorithm tends to wander a bit; we want a smarter algorithm!

Greedy Search

- a modified UFS based on an estimate of the distance to the goal state h(n)
- h is an **admissible** function $\equiv h(G) = 0$
- h is a **consistent** function $\equiv h(n) \leq$ actual cost
  This is stricter than the admissible requirement, but is hard to avoid in practice.

Properties:

1. complete? NO (for example: A —1— B —2— C)
2. optimal? NO
3. time? $O(b^m)$
4. space? $O(b^m)$

Let's compare our two algorithms:
UCS: g(n): actual cost to get to n from initial state

- optimal
- conservative
- slow

Greedy: h(n) estimate of cost to get to final state from n

- non-optimal
- aggressive
- fast

Neither of these has all the properties we want; can we get the best properties of both?
It turns out YES: we just have to add them directly!
f(n) = g(n) + h(n) estimates the total cost & is admissible (provided h(n) is admissible)
This leads us to a very important algorithm:

A^* Search

- UCS based on f(n) = g(n) + h(n)
- forms contours which slim approaching goal
- prunes nodes outside contours
- we can prove that this algorithm is optimal:

Let's discuss the choice of heuristic: can we find a good heuristic for every problem?
h(n) = 0 works for every problem, but this makes the algorithm UCS!
Therefore, we want the maximum admissible heuristic for a given problem.

We will now consider a concrete example: 8 PUZZLE
We have two heuristics we would like to consider:
$h_1(n) = \#$ pieces out of place
$h_2(n) =$ Manhattan Distance $\equiv$ horizontal + vertical distance
$h_2(n) \leq h_1(n)$ for all n, so we say h_2 dominates h_1
Clearly h_2 is the better choice, but how much better can it really be?

We use two properties to evaluate search $A*$ search algorithms

1. effective branching factor $(b^*) :=$ avg branches per node
2. node count $(N) = 1 + (b^*)^0 + (b^*)^2 + \ldots + (b^*)^d$

d	IDS	$A^*(h_1)$	$A^*(h_2)$
4	N = 112; $b^* = 2.35$	N = 13; $b^* = 1.48$	N = 12; $b^* = 1.45$
6	N = 680; $b^* = 2.87$	N = 20; $b^* = 1.34$	N = 18; $b^* = 1.30$
8	N = 6384; $b^* = 2.73$	N = 39; $b^* = 1.33$	N = 25; b^* 1.24
10	N = 47,127	N = 93	N = 39
12	N = 3,644,035	N = 227	N = 73

Clearly, A^* is much more efficient, and h2 is much more efficient than h1;
how do we formally evaluate this?
time cost $= O(b^{\Delta})$, for absolute error $\Delta = h(n) - h^*$
$\implies$ hypothetical min $O(h^{\epsilon})$, for fractional error $\epsilon = (h * -h)/h^*$ is the