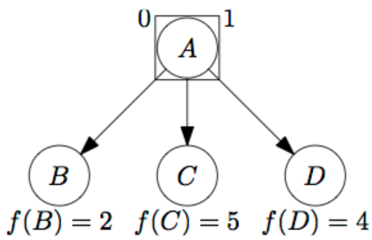
Lecture 4.

Informed Search

To obtain a solution more quickly, we use *additional information* to guide the node expansion.

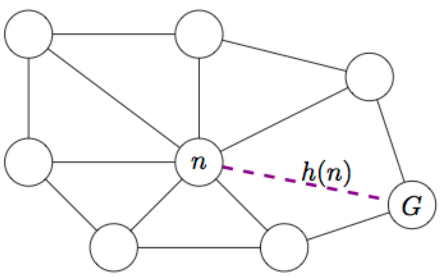
4.1. Evaluation Function

One way to include the additional information is to used an **evaluation function** $f(n)$ where n is a node. This evaluation function estimates the cost when a node is selected to be a part of solution. The node with the lowest evaluation value is chosen first.



4.2. Heuristic Functions

Heuristic function, $h(n)$ estimates the cost of the cheapest path from node n to a goal node. It is a problem-specific function with one constraint: *if* n *is a goal node, then* $h(n) = 0$.

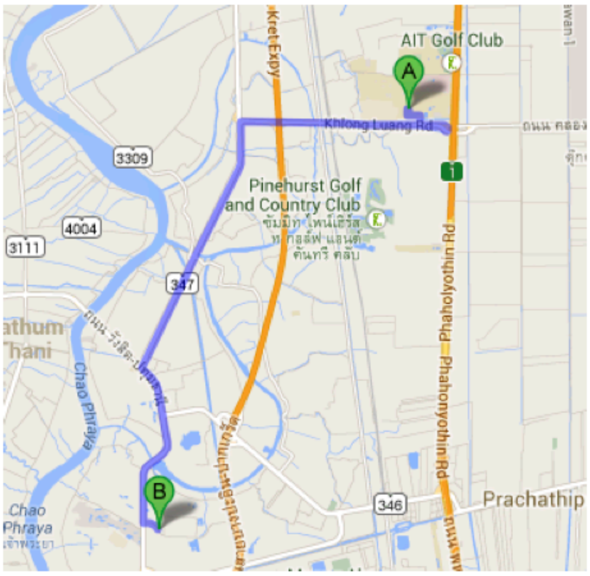


4.3. Greedy Best-first Search

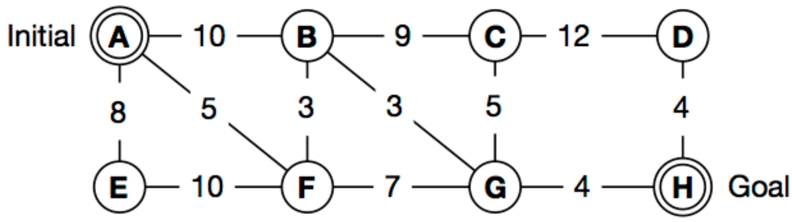
Greedy Best-first search chooses to expand the node expected to be the closest to the goal since it is likely to lead to a solution quickly. GBFS always chooses the node with the *smallest* $h(n)$ from the nodes in the frontier. Thus, we

$$f(n) \stackrel{\text{def}}{=} h(n)$$

In the route-finding problem, we can use *straight-line distance* as a heuristic function. The straight-line distance is basically shorter than the actual distance, but it roughly shows the distance between two cities.



Exercise 4.1 Use the *greedy best-first tree search* to find a route from A to H.

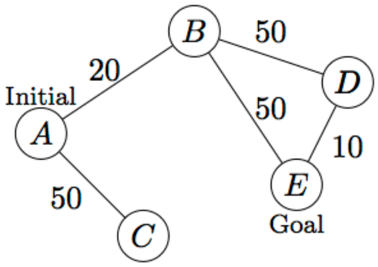


Node	Heuristic	Node	Heuristic	Node	Heuristic	Node	Heuristic
A	10	B	4	C	6	D	4
E	18	F	9	G	4	H	0

4.3.1. Evaluating Greedy Best-first Search

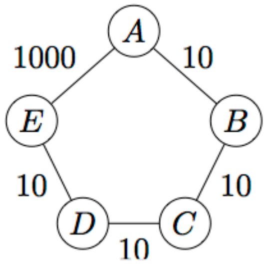
Complete

Let's conduct the greedy best-first tree search and graph search using the following state space and heuristic function.



n	$h(n)$	n	$h(n)$
A	40	B	50
C	20	D	5
E	0		

Optimality



n	$h(n)$	n	$h(n)$
A	10	B	20
C	20	D	5
E	0		

4.4. A* Search

A* search minimizes the total cost from the initial node to a goal node.

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

where $g(n)$ is the actual cost to reach node n from the initial node, and $h(n)$ is a heuristic function representing the estimated cheapest cost from the node n to a goal node. $f(n)$ represents the estimated cost of the cheapest solution through n.

Example 4.1 Use A* search for the route-planning problem.

4.4.1. Evaluating A* Search

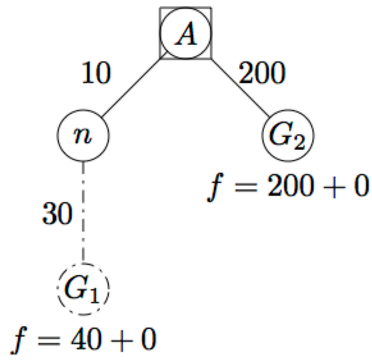
Optimality

Tree Search A* using Tree Search is optimal if $h(n)$ is an *admissible* heuristic.

A heuristic function is *admissible* if it *never overestimates* the cost to reach the goal.

$$\forall n, h(n) \leq C(n)$$

where n is a node, $h(n)$ is an estimated cost to reach a goal from n, and $C(n)$ is the actual cost to reach a goal from n.



Suppose a suboptimal goal node G_2 is appended to the *frontier*, and let the optimal cost be C^*, we have

$$\begin{aligned} f(G_2) &= g(G_2) + h(G_2) \\ &= g(G_2) > C^* \end{aligned}$$

We also have a node n that is on an optimal path, we have

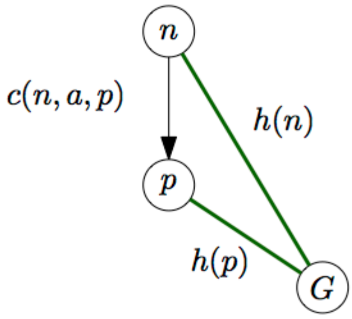
$$f(n) = g(n) + h(n) \leq C^*$$

Then, $f(n) \leq C^* < f(G_2)$. G_2 will not be selected until the end of the search. A* returns an optimal solution.

Graph Search A* using Graph Search is optimal if $h(n)$ is a *consistent* (or *monotone*) heuristic.

A heuristic function is consistent if, for every node n and every successor n' of n generated by any action a, the estimated cost to the goal from n is no greater than the step cost of getting to p plus the estimated cost of reaching the goal from p.

$$h(n) \leq c(n, a, p) + h(p)$$

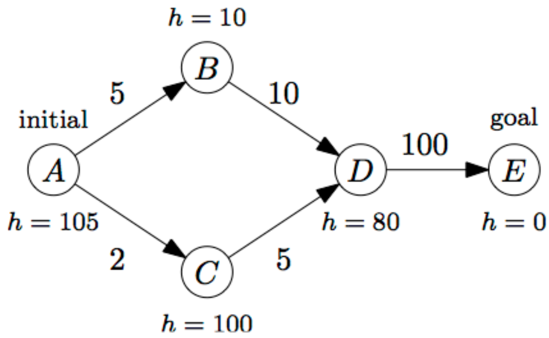


triangle inequality

From the consistency of $h(n)$, we have

$$\begin{aligned} f(p) &= g(p) + h(p) \\ &= g(n) + c(n, a, p) + h(p) \\ &\geq g(n) + h(n) = f(n) \end{aligned}$$

Exercise 4.2 Check the admissibility and consistency of the heuristic function shown in the following figure.



References

Russell, S. and Norvig, P. (2010). Artificial Intelligence: A Modern Approach (3rd edition). Pearson/Prentice Hall.
Michalewicz, F. and Fogel, D. B. (1998). How to Solve It: Modern Heuristics. Springer.