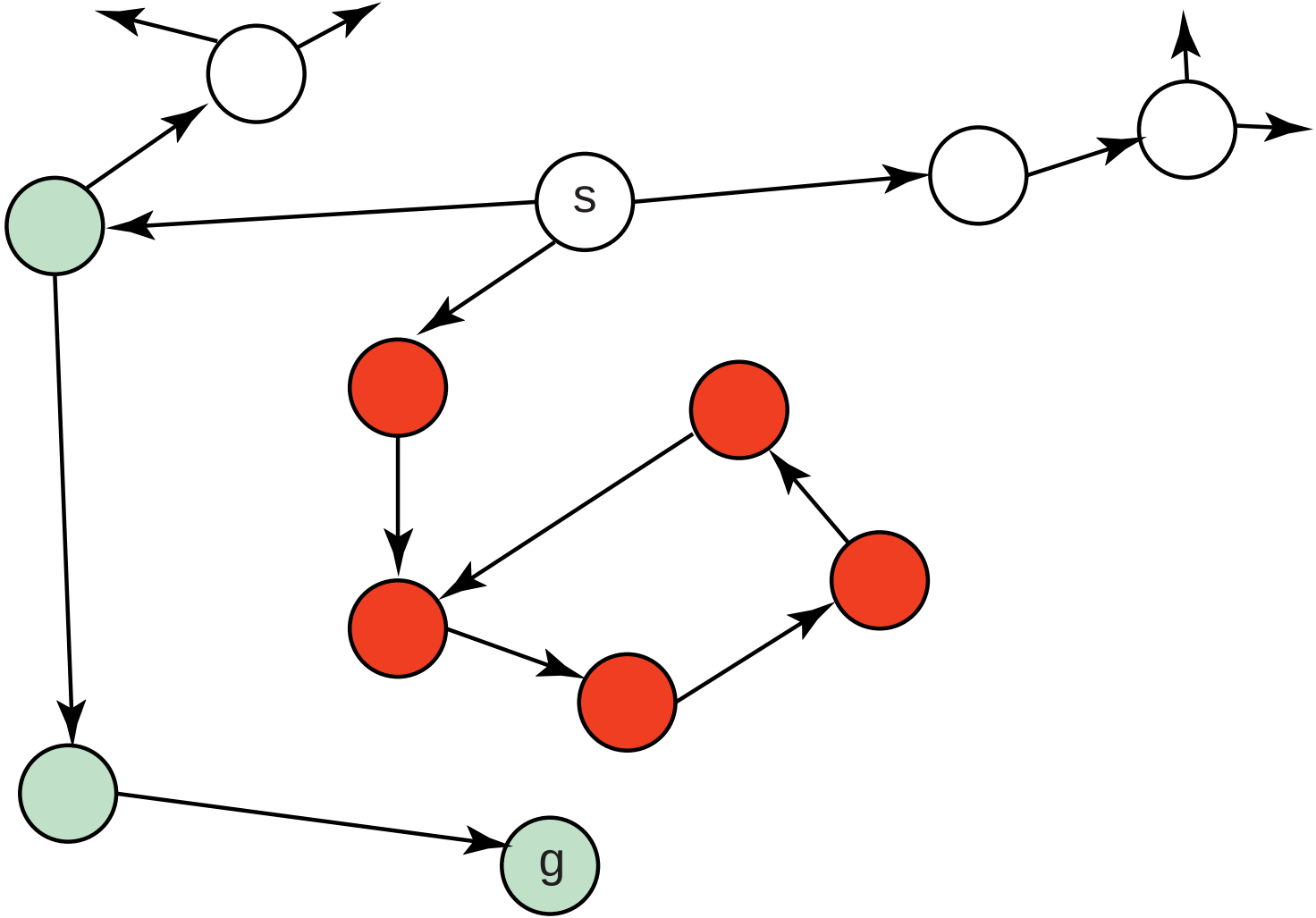
Heuristic Search

- Previous methods do not take into account the goal until they are at a goal node.
- Often there is extra knowledge that can be used to guide the search: heuristics.
- We use $h(n)$ as an estimate of the distance from node n to a goal node.
- $h(n)$ is an underestimate if it is less than or equal to the actual cost of the shortest path from node n to a goal.
- $h(n)$ uses only readily obtainable information about a node.

Best-first Search

- Idea: always choose the node on the frontier with the smallest *h*-value.
- It treats the frontier as a priority queue ordered by *h*.
- It uses space exponential in path length.
- It isn't guaranteed to find a solution, even of one exists. It doesn't always find the shortest path.

Illustrative Graph — Best-first Search



Heuristic Depth-first Search

- It's a way to use heuristic knowledge in depth-first search.
- Idea: order the neighbors of a node (by *h*) before adding them to the front of the frontier.
- Locally chooses which subtree to develop, but still does depth-first search. It explores all paths from the node at the head of the frontier before exploring paths from the next node.
- Space is linear in path length. It isn't guaranteed to find a solution. It can get led up the garden path.

A^* Search

- A^* search takes the path to a node and heuristic value into account.
- Let $g(n)$ be the cost of the path found to node n.
- Let $h(n)$ be the estimate of the cost from n to a goal.
- Let $f(n) = g(n) + h(n)$. It is an estimate of a path from the start to a goal via n.

$$\underbrace{\underbrace{start \xrightarrow{\text{actual}} n}_{g(n)} \underbrace{\xrightarrow{\text{estimate}} goal}_{h(n)}}_{f(n)}$$

A^* Search Algorithm

- A^* is a mix of lowest-cost-first and best-first search.
- It treats the frontier as a priority queue ordered by $f(n)$.
- It always chooses the node on the frontier with the lowest estimated distance from the start to a goal node constrained to go via that node.

Admissibility of A^*

If there is a solution, A^* always finds an optimal solution —the first path to a goal selected— if

- the branching factor is finite
- arc costs are bounded above zero (there is some $\epsilon > 0$ such that all of the arc costs are greater than ϵ), and
- $h(n)$ is an underestimate of the length of the shortest path from n to a goal node.

Why is A^* admissible?

- The f-value for any node on an optimal solution path is less than or equal to the f-value of an optimal solution. (As h is an underestimate).
- The search never selects a node with a higher f-value than the f-value of an optimal solution. A non-optimal solution has a higher f value — so it will never be selected.
- It halts, as the minimum g-value on the frontier keeps increasing, and will eventually exceed any finite number.