Frontier search (manage choices)

```
frontierSearch([Node|Rest]) :- goal(Node);
   (findall(Next, arc(Node,Next), Children),
    add2frontier(Children, Rest, NewFrontier),
    frontierSearch(NewFrontier)).
```

Frontier search (manage choices)

```
frontierSearch([Node|Rest]) :- goal(Node);
  (findall(Next, arc(Node,Next), Children),
   add2frontier(Children, Rest, NewFrontier),
   frontierSearch(NewFrontier)).
```

Depth first: `append(Children, Rest, NewFrontier)`

Breadth-first: `append(Rest, Children, NewFrontier)`

Frontier search (manage choices)

```
frontierSearch([Node|Rest]) :- goal(Node);
   (findall(Next, arc(Node,Next), Children),
    add2frontier(Children, Rest, NewFrontier),
    frontierSearch(NewFrontier)).
```

Depth first: `append(Children, Rest, NewFrontier)`

Breadth-first: `append(Rest, Children, NewFrontier)`

For `add2frontier(Children, Rest, NewFrontier)`, require

`NewFrontier` merges `Children` and `Rest`

where a list `L` is defined to *merge* lists `L1` *and* `L2` if

(a) every member of `L` is a member of `L1` or `L2`

(b) every member of `L1` or of `L2` is a member of `L`.

Exercise (Prolog)

Suppose a positive integer `Seed` links nodes `1,2,...` in two ways

```
arc(N,M,Seed) :- M is N*Seed.
arc(N,M,Seed) :- M is N*Seed +1.
```

e.g. `Seed`=3 gives arcs `(1,3), (1,4), (3,9), (3, 10) ...`

Exercise (Prolog)

Suppose a positive integer `Seed` links nodes `1,2,...` in two ways

```
arc(N,M,Seed) :- M is N*Seed.
arc(N,M,Seed) :- M is N*Seed +1.
```

e.g. `Seed`=3 gives arcs `(1,3), (1,4), (3,9), (3, 10) ...`

Goal nodes are multiples of a positive integer `Target`

```
goal(N,Target) :- 0 is N mod Target.
```

e.g. `Target`=13 gives goals `13, 26, 39 ...`

Exercise (Prolog)

Suppose a positive integer `Seed` links nodes `1,2,...` in two ways

```
arc(N,M,Seed) :- M is N*Seed.
arc(N,M,Seed) :- M is N*Seed +1.
```

e.g. `Seed`=3 gives arcs `(1,3), (1,4), (3,9), (3, 10) ...`

Goal nodes are multiples of a positive integer `Target`

```
goal(N,Target) :- 0 is N mod Target.
```

e.g. `Target`=13 gives goals `13, 26, 39 ...`

Modify frontier search to define predicates

```
breadth1st(+Start, ?Found, +Seed, +Target)
depth1st(+Start, ?Found, +Seed, +Target)
```

that search breadth-first and depth-first respectively for a `Target`-goal node `Found` linked to `Start` by `Seed`-arcs.

Refining frontier search

For `add2frontier(Children, Rest, NewFrontier)`, require

`NewFrontier` merges `Children` and `Rest`

and for `NewFrontier = [Head|Tail]`, ensure

`Head` is “no worse than” any in `Tail`.

Refining frontier search

For `add2frontier(Children, Rest, NewFrontier)`, require

`NewFrontier` merges `Children` and `Rest`

and for `NewFrontier = [Head|Tail]`, ensure

`Head` is "no worse than" any in `Tail`.

What can it mean for `Node1` to be *no worse than* `Node2` ?

(A1) `Node1` costs no more than `Node2`

Refining frontier search

For `add2frontier(Children, Rest, NewFrontier)`, require

`NewFrontier` merges `Children` and `Rest`

and for `NewFrontier = [Head|Tail]`, ensure

`Head` is "no worse than" any in `Tail`.

What can it mean for `Node1` to be *no worse than* `Node2` ?

(A1) `Node1` costs no more than `Node2`

(A2) `Node1` is deemed no further from a goal node than `Node2`

Refining frontier search

For `add2frontier(Children, Rest, NewFrontier)`, require

`NewFrontier` merges `Children` and `Rest`

and for `NewFrontier = [Head|Tail]`, ensure

`Head` is "no worse than" any in `Tail`.

What can it mean for `Node1` to be *no worse than* `Node2` ?

(A1) `Node1` costs no more than `Node2`

(A2) `Node1` is deemed no further from a goal node than `Node2`

(A3) some mix of (A1) and (A2)

Refining frontier search

For `add2frontier(Children, Rest, NewFrontier),` require

`NewFrontier` merges `Children` and `Rest`

and for `NewFrontier = [Head|Tail]`, ensure

`Head` is "no worse than" any in `Tail`.

What can it mean for `Node1` to be *no worse than* `Node2` ?

(A1) `Node1` costs no more than `Node2`
$\rightsquigarrow$ minimum cost search (= breadth-first if every arc costs 1)

(A2) `Node1` is deemed no further from a goal node than `Node2`

(A3) some mix of (A1) and (A2)

Refining frontier search

For `add2frontier(Children, Rest, NewFrontier)`, require

`NewFrontier` merges `Children` and `Rest`

and for `NewFrontier = [Head|Tail]`, ensure

`Head` is "no worse than" any in `Tail`.

What can it mean for `Node1` to be *no worse than* `Node2` ?

(A1) `Node1` costs no more than `Node2`
$\rightsquigarrow$ minimum cost search (= breadth-first if every arc costs 1)

(A2) `Node1` is deemed no further from a goal node than `Node2`
$\rightsquigarrow$ best-first search (= depth-first for heuristic $\propto$ depth^{-1})

(A3) some mix of (A1) and (A2)

Refining frontier search

For `add2frontier(Children, Rest, NewFrontier)`, require

`NewFrontier` merges `Children` and `Rest`

and for `NewFrontier = [Head|Tail]`, ensure

`Head` is "no worse than" any in `Tail`.

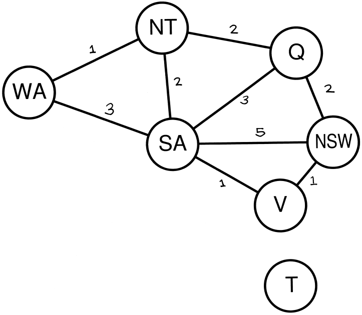
What can it mean for `Node1` to be *no worse than* `Node2` ?

(A1) `Node1` costs no more than `Node2`
$\leadsto$ minimum cost search (= breadth-first if every arc costs 1)

(A2) `Node1` is deemed no further from a goal node than `Node2`
$\leadsto$ best-first search (= depth-first for heuristic $\propto$ depth^{-1})

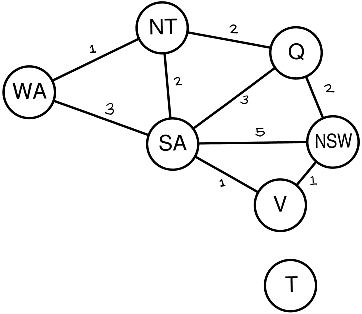
(A3) some mix of (A1) and (A2)
$\leadsto$ A-star

Arc costs (space, time, money, ...)




```
arc(wa,nt,1).   arc(nt,q,2).
arc(q,nsw,2).   arc(wa,sa,3).
arc(nt,sa,2).   arc(sa,q,3).
arc(sa,nsw,5).  arc(sa,v,1).
arc(v,nsw,1).
```

Arc costs (space, time, money, ...)

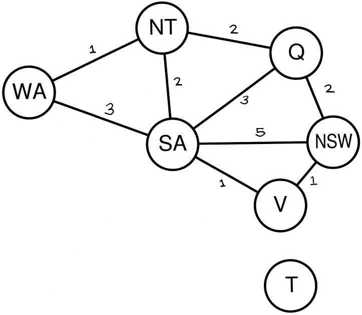



```
arc(wa,nt,1).    arc(nt,q,2).
arc(q,nsw,2).    arc(wa,sa,3).
arc(nt,sa,2).    arc(sa,q,3).
arc(sa,nsw,5).   arc(sa,v,1).
arc(v,nsw,1).
```

$$\mathrm{cost}(\texttt{wa,nt,q,nsw}) \ = \ 1+2+2 \ = \ 5$$

$$\mathrm{cost}(x_1, x_2, \ldots, x_{k+1}) \ := \ \sum_{i=1}^{k} \mathrm{cost}(x_i, x_{i+1})$$

Arc costs (space, time, money, . . .)




```
arc(wa,nt,1).   arc(nt,q,2).
arc(q,nsw,2).   arc(wa,sa,3).
arc(nt,sa,2).   arc(sa,q,3).
arc(sa,nsw,5).  arc(sa,v,1).
arc(v,nsw,1).
```

$$\text{cost}(\texttt{wa,nt,q,nsw}) \;=\; 1+2+2 \;=\; 5$$

$$\text{cost}(x_1, x_2, \ldots, x_{k+1}) \;:=\; \sum_{i=1}^{k} \text{cost}(x_i, x_{i+1})$$

$$\text{cost}(\texttt{wa,sa,nsw}) \;=\; 3+5 \;=\; 8$$

Heuristics

$$h(\texttt{Node}) = \text{estimate the minimum cost of}$$
$$\text{a path from } \texttt{Node} \text{ to a goal node}$$

Heuristics

$$h(\texttt{Node}) = \text{estimate the minimum cost of a path from } \texttt{Node} \text{ to a goal node}$$

EXAMPLES

- Fsm accept where node $=$ `[Q,String]` and every arc costs 1

$$h([\texttt{Q}, \texttt{String}]) = \text{length}(\texttt{String})$$

Heuristics

$$h(\texttt{Node}) = \text{estimate the minimum cost of a path from } \texttt{Node} \text{ to a goal node}$$

EXAMPLES

- Fsm accept where node $=$ `[Q,String]` and every arc costs 1

$$h([\texttt{Q}, \texttt{String}]) = \text{length}(\texttt{String})$$

- Prolog search where node $=$ list of propositions to prove, and every arc costs 1

$$h(\texttt{List}) = \text{length}(\texttt{List})$$

Heuristics

$$h(\texttt{Node}) = \text{estimate the minimum cost of a path from } \texttt{Node} \text{ to a goal node}$$

EXAMPLES

- Fsm accept where node = [Q,String] and every arc costs 1

$$h([\texttt{Q}, \texttt{String}]) = \text{length}(\texttt{String})$$

- Prolog search where node = list of propositions to prove, and every arc costs 1

$$h(\texttt{List}) = \text{length}(\texttt{List})$$

- Node = point on a Euclidean plane, cost = distance between nodes, goal is a point G

$$h(\texttt{Node}) = \text{straight-line distance to } \texttt{G}$$

Heuristics

$$\begin{aligned} h(\texttt{Node}) = {} & \text{estimate the minimum cost of} \\ & \text{a path from } \texttt{Node} \text{ to a goal node} \end{aligned}$$

Examples

- Fsm accept where node = [Q,String] and every arc costs 1

$$h([\texttt{Q}, \texttt{String}]) = \text{length}(\texttt{String})$$

- Prolog search where node = list of propositions to prove, and every arc costs 1

$$h(\texttt{List}) = \text{length}(\texttt{List})$$

- Node = point on a Euclidean plane, cost = distance between nodes, goal is a point G

$$h(\texttt{Node}) = \text{straight-line distance to } \texttt{G}$$

- estimate assuming lots of arcs (simplifying the problem)

Best-first search

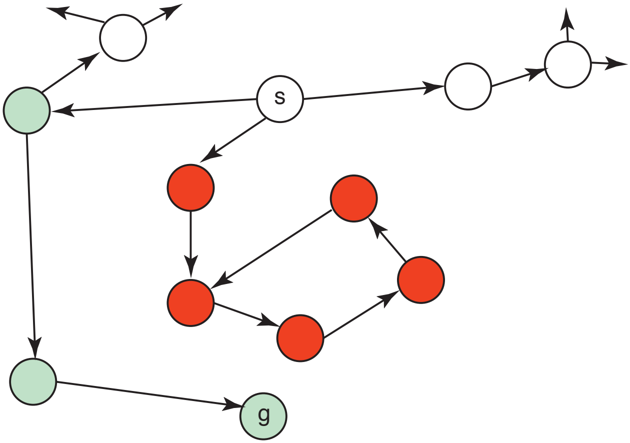
Form `NewFrontier = [Head|Tail]` such that

$$h(\texttt{Head}) \leq h(\texttt{Node}) \text{ for every } \texttt{Node} \text{ in } \texttt{Tail}$$

Best-first search

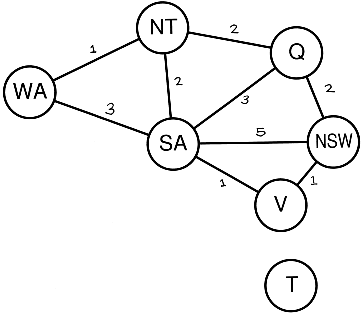
Form `NewFrontier = [Head|Tail]` such that

$$h(\texttt{Head}) \leq h(\texttt{Node}) \text{ for every } \texttt{Node} \text{ in } \texttt{Tail}$$

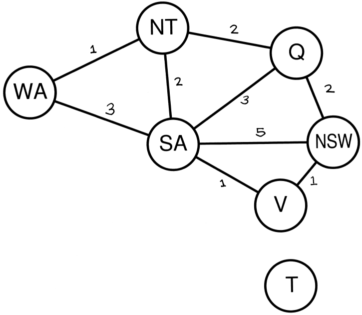


Poole & Mackworth

Min-cost



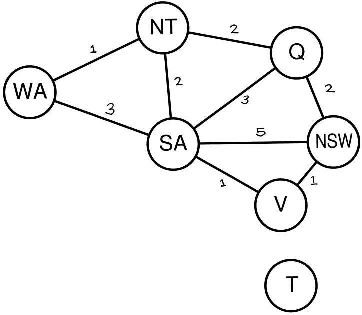
Min-cost $\neq$ breadth-first



$\text{cost}(n_1 \cdots n_k) = \sum_{i=1}^{k-1} \text{cost}(n_i, n_{i+1})$

$\text{cost}(\texttt{wa nt q nsw}) = 5$
$\text{cost}(\texttt{wa sa nsw}) = 8$

Min-cost $\neq$ breadth-first



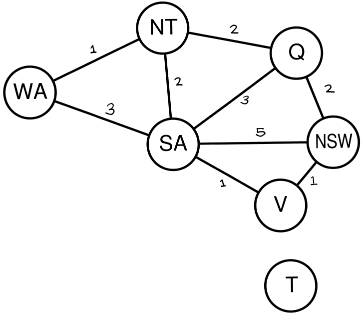
cost$(n_1 \cdots n_k) =$
$\sum_{i=1}^{k-1} \text{cost}(n_i, n_{i+1})$

cost(`wa nt q nsw`) $= 5$
cost(`wa sa nsw`) $= 8$

`add2frontier(Children, Rest, [Head|Tail])`

cost(`Start`$\cdots$`Head`) $\leq$ cost(`Start`$\cdots n$) for each n in `Tail` ?

Min-cost $\neq$ breadth-first



cost($n_1 \cdots n_k$) = $\sum_{i=1}^{k-1}$ cost(n_i, n_{i+1})

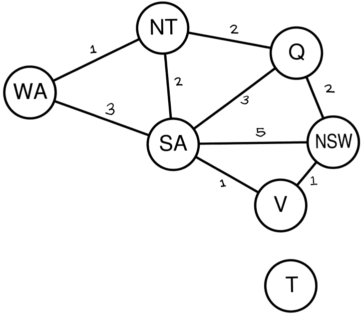
cost(`wa nt q nsw`) = 5
cost(`wa sa nsw`) = 8

`add2frontier(Children, Rest, [Head|Tail])`

cost(`Start`$\cdots$`Head`) $\leq$ cost(`Start`$\cdots n$) for each n in `Tail` ?

- node $\rightsquigarrow$ path

Min-cost $\neq$ breadth-first



$\text{cost}(n_1 \cdots n_k) = \sum_{i=1}^{k-1} \text{cost}(n_i, n_{i+1})$

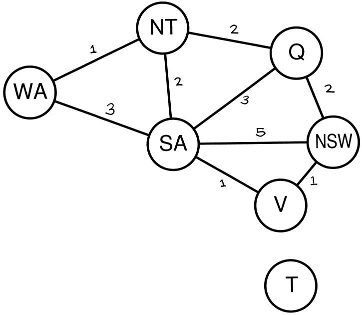
$\text{cost}(\texttt{wa nt q nsw}) = 5$
$\text{cost}(\texttt{wa sa nsw}) = 8$

`add2frontier(Children, Rest, [Head|Tail])`

$\text{cost}(\texttt{Start} \cdots \texttt{Head}) \leq \text{cost}(\texttt{Start} \cdots n)$ for each n in `Tail` ?

- node $\rightsquigarrow$ path or pair (n,cost($\texttt{Start} \cdots n$))

Min-cost $\neq$ breadth-first



$\text{cost}(n_1 \cdots n_k) = \sum_{i=1}^{k-1} \text{cost}(n_i, n_{i+1})$

$\text{cost}(\texttt{wa nt q nsw}) = 5$
$\text{cost}(\texttt{wa sa nsw}) = 8$

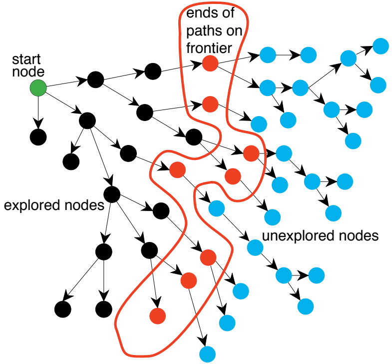
`add2frontier(Children, Rest, [Head|Tail])`

$\text{cost}(\texttt{Start}\cdots\texttt{Head}) \leq \text{cost}(\texttt{Start}\cdots n)$ for each n in `Tail` ?

- node $\rightsquigarrow$ path or pair $(n,\text{cost}(\texttt{Start}\cdots n))$
- what about proximity to goal?

$$h(n) = \text{estimate of min cost path } n \cdots \text{goal}$$

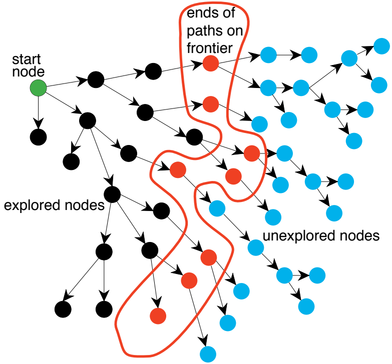
A*



$$\text{solution} = \underbrace{\text{start} \cdots n}_{\text{explored}} \cdots \text{goal}$$

$$f(\text{start} \cdots n) = \text{cost}(\text{start} \cdots n) + h(n)$$

A*



$$\text{solution} = \underbrace{\text{start} \cdots n}_{\text{explored}} \cdots \text{goal}$$

$$f(\text{start} \cdots n) = \text{cost}(\text{start} \cdots n) + h(n)$$

Ensure `Frontier = [Head|Tail]` where `Head` has minimal f

- $h(n) = 0$ for every n $\leadsto$ min-cost
- $\text{cost}(\text{start} \cdots n) = 0$ for every n $\leadsto$ best-first (disregarding the past)

Admissibility

A* is *admissible* (under cost, h) if it returns a solution of min cost whenever a solution exists.

Admissibility

A* is *admissible* (under cost, h) if it returns a solution of min cost whenever a solution exists.

3 conditions sufficient for admissibility

under-estimate: for every solution $n \cdots \text{goal}$,
$$0 \leq h(n) \leq \text{cost}(n \cdots \text{goal})$$

Admissibility

A* is *admissible* (under cost, h) if it returns a solution of min cost whenever a solution exists.

3 conditions sufficient for admissibility

under-estimate: for every solution $n \cdots \text{goal}$,
$$0 \leq h(n) \leq \text{cost}(n \cdots \text{goal})$$
termination: for some $\epsilon > 0$, every arc costs $\geq \epsilon$

finite branching: $\{n' \mid \text{arc}(n, n')\}$ is finite for each node n

Admissibility

A* is *admissible* (under cost, h) if it returns a solution of min cost whenever a solution exists.

3 conditions sufficient for admissibility

under-estimate: for every solution $n \cdots \text{goal}$,
$$0 \leq h(n) \leq \text{cost}(n \cdots \text{goal})$$

termination: for some $\epsilon > 0$, every arc costs $\geq \epsilon$

finite branching: $\{n' \mid \text{arc}(n, n')\}$ is finite for each node n

Assuming the 3 conditions above, let p be a solution.

TO SHOW: A* returns a solution with min cost c.

Admissibility

A* is *admissible* (under cost, h) if it returns a solution of min cost whenever a solution exists.

3 conditions sufficient for admissibility

under-estimate: for every solution $n \cdots$ goal,
$$0 \leq h(n) \leq \text{cost}(n \cdots \text{goal})$$
termination: for some $\epsilon > 0$, every arc costs $\geq \epsilon$
finite branching: $\{n' \mid \text{arc}(n, n')\}$ is finite for each node n

Assuming the 3 conditions above, let p be a solution.

TO SHOW: A* returns a solution with min cost c.

Let $F_0 =$[Start], F_{n+1} be A*'s next frontier after F_n ([] if none), and c_n be the cost of the head of F_n (∞ if $F_n =$ []).

Admissibility

A* is *admissible* (under cost, h) if it returns a solution of min cost whenever a solution exists.

3 conditions sufficient for admissibility

under-estimate: for every solution $n \cdots \text{goal}$,
$$0 \leq h(n) \leq \text{cost}(n \cdots \text{goal})$$

termination: for some $\epsilon > 0$, every arc costs $\geq \epsilon$

finite branching: $\{n' \mid \text{arc}(n, n')\}$ is finite for each node n

Assuming the 3 conditions above, let p be a solution.

TO SHOW: A* returns a solution with min cost c.

Let $F_0 =$[Start], F_{n+1} be A*'s next frontier after F_n ([] if none), and c_n be the cost of the head of F_n (∞ if $F_n =$ []).

(i) for every $n \geq 0$ s.t. $c_n < c$, F_n has a prefix of p

Admissibility

A* is *admissible* (under cost, h) if it returns a solution of min cost whenever a solution exists.

3 conditions sufficient for admissibility

under-estimate: for every solution $n \cdots \text{goal}$,
$$0 \leq h(n) \leq \text{cost}(n \cdots \text{goal})$$
termination: for some $\epsilon > 0$, every arc costs $\geq \epsilon$

finite branching: $\{n' \mid \text{arc}(n, n')\}$ is finite for each node n

Assuming the 3 conditions above, let p be a solution.

TO SHOW: A* returns a solution with min cost c.

Let $F_0 =$[Start], F_{n+1} be A*'s next frontier after F_n ([] if none), and c_n be the cost of the head of F_n (∞ if $F_n =$ []).

(i) for every $n \geq 0$ s.t. $c_n < c$, F_n has a prefix of p

(ii) $c = c_n$ for some n s.t. the head of F_n is a solution.