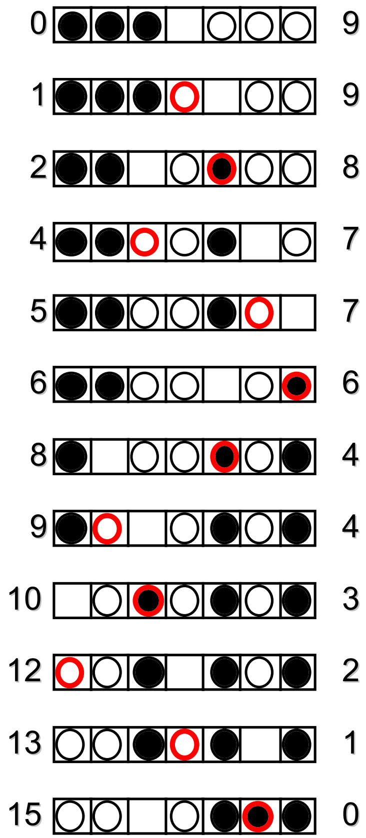
```
search_bstf([Goal|Rest],Goal):-
  goal(Goal).
search_bstf([Current|Rest],Goal):-
  children(Current,Children),
  add_bstf(Children,Rest,NewAgenda),
  search_bstf(NewAgenda,Goal).

% add_bstf(A,B,C) <- C contains the elements of A and B
%                    (B and C sorted according to eval/2)
add_bstf([],Agenda,Agenda).
add_bstf([Child|Children],OldAgenda,NewAgenda):-
  add_one(Child,OldAgenda,TmpAgenda),
  add_bstf(Children,TmpAgenda,NewAgenda).

% add_one(S,A,B) <- B is A with S inserted acc. to eval/2
add_one(Child,OldAgenda,NewAgenda):-
  eval(Child,Value),
  add_one(Value,Child,OldAgenda,NewAgenda).
```

Best-first search

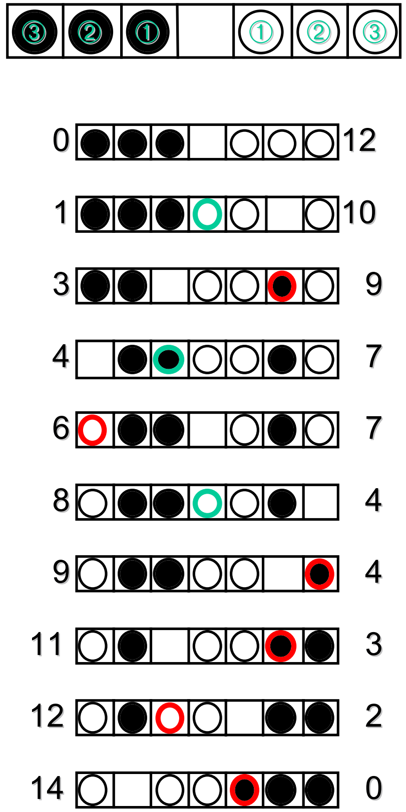
```
% tiles_a(A,M,V0,V) <- goal position can be reached from
%                      one of the positions on A with last
%                      move M (best-first strategy)
tiles_a([v(V,LastMove)|Rest],LastMove,Visited,Visited):-
 goal(LastMove).
tiles_a([v(V,LastMove)|Rest],Goal,Visited0,Visited):-
 show_move(LastMove,V),
 setof0(v(Value,NextMove),
       ( move(LastMove,NextMove),
         eval(NextMove,Value) ),
       Children),                    % Children sorted on Value
 merge(Children,Rest,NewAgenda),     % best-first
 tiles_a(NewAgenda,Goal,[LastMove|Visited0],Visited).
```

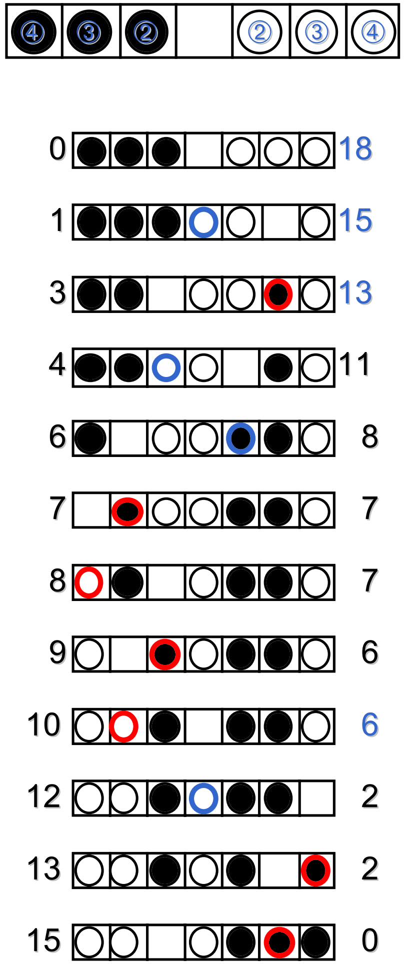
Solving a puzzle

bLeftOfw



outOfPlace





Comparing heuristics

☞ An ***A algorithm*** is a best-first search algorithm that aims at minimising the **total cost** along a path from start to goal.

$$f(n) = g(n) + h(n)$$

- $f(n)$: estimate of total cost along path through n
- $g(n)$: actual cost to reach n
- $h(n)$: estimate of cost to reach goal from n

A algorithm

Global and local optimism

- A heuristic is (globally) ***optimistic*** or ***admissible*** if the estimated cost of **reaching a goal** is always less than the actual cost.

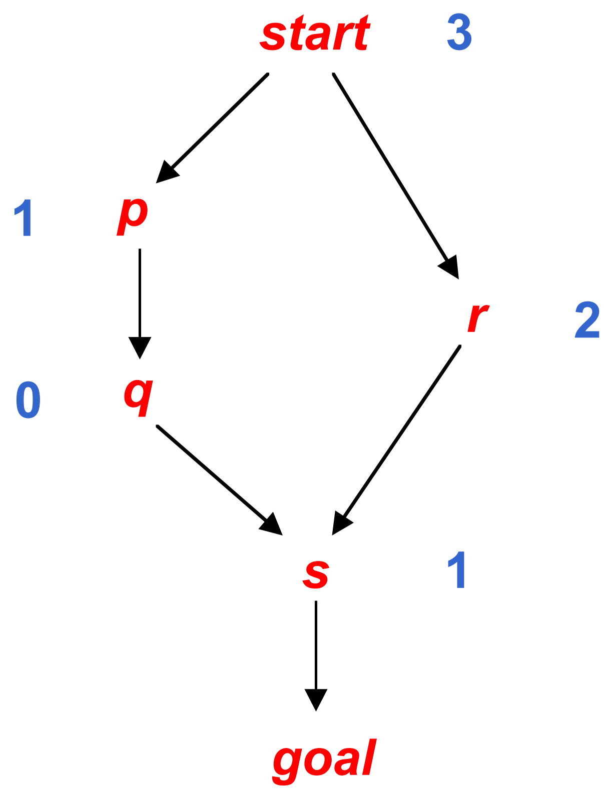
$$h(n) \leq h^*(n)$$

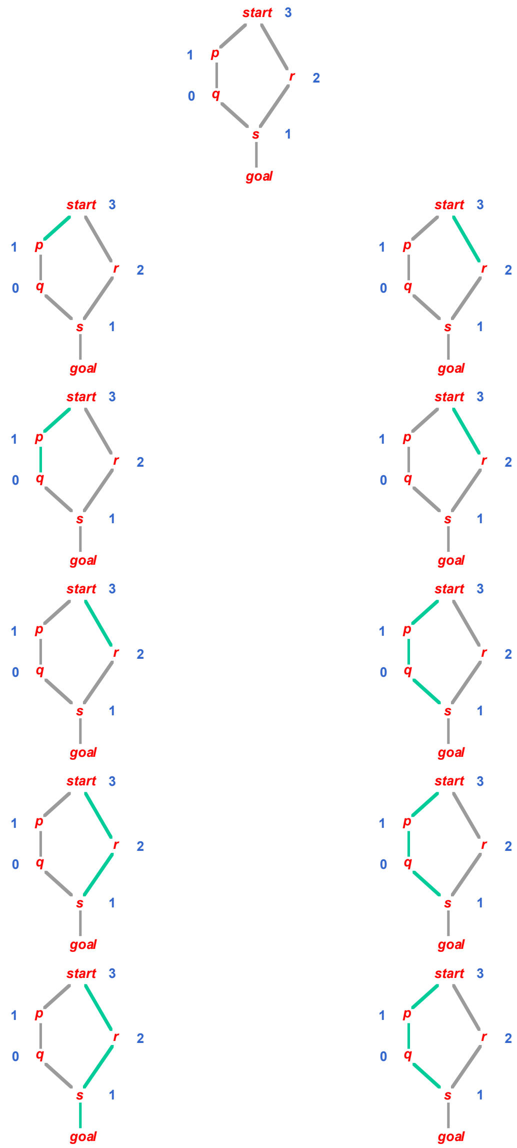
estimate of cost to reach goal from n

actual (unknown) cost to reach goal from n

- A heuristic is ***monotonic*** (locally optimistic) if the estimated cost of **reaching any node** is always less than the actual cost.

$$h(n_1) - h(n_2) \leq h^*(n_1) - h^*(n_2)$$





```
[start-3]

[p-2,r-3]

[q-2,r-3]

[r-3,s-4]

[s-3,s-4]

[goal-3,s-4]
```

Non-monotonic heuristic

```
search_beam(Agenda,Goal):-
  search_beam(1,Agenda,[],Goal).

search_beam(D,[],NextLayer,Goal):-
  D1 is D+1,
  search_beam(D1,NextLayer,[],Goal).
search_beam(D,[Goal|Rest],NextLayer,Goal):-
  goal(Goal).
search_beam(D,[Current|Rest],NextLayer,Goal):-
  children(Current,Children),
  add_beam(D,Children,NextLayer,NewNextLayer),
  search_beam(D,Rest,NewNextLayer,Goal).
```

☞ Here, the number of children to be added to the beam is made dependent on the depth **D** of the node

- ✓ in order to keep depth as a ‘global’ variable, search is layer-by-layer

Beam search

```
search_hc(Goal,Goal):-
 goal(Goal).
search_hc(Current,Goal):-
 children(Current,Children),
 select_best(Children,Best),
 search_hc(Best,Goal).
```

```
% hill_climbing as a variant of best-first search
search_hc([Goal|_],Goal):-
   goal(Goal).
search_hc([Current|_],Goal):-
   children(Current,Children),
   add_bstf(Children,[],NewAgenda),
   search_hc(NewAgenda,Goal).
```

Hill-climbing