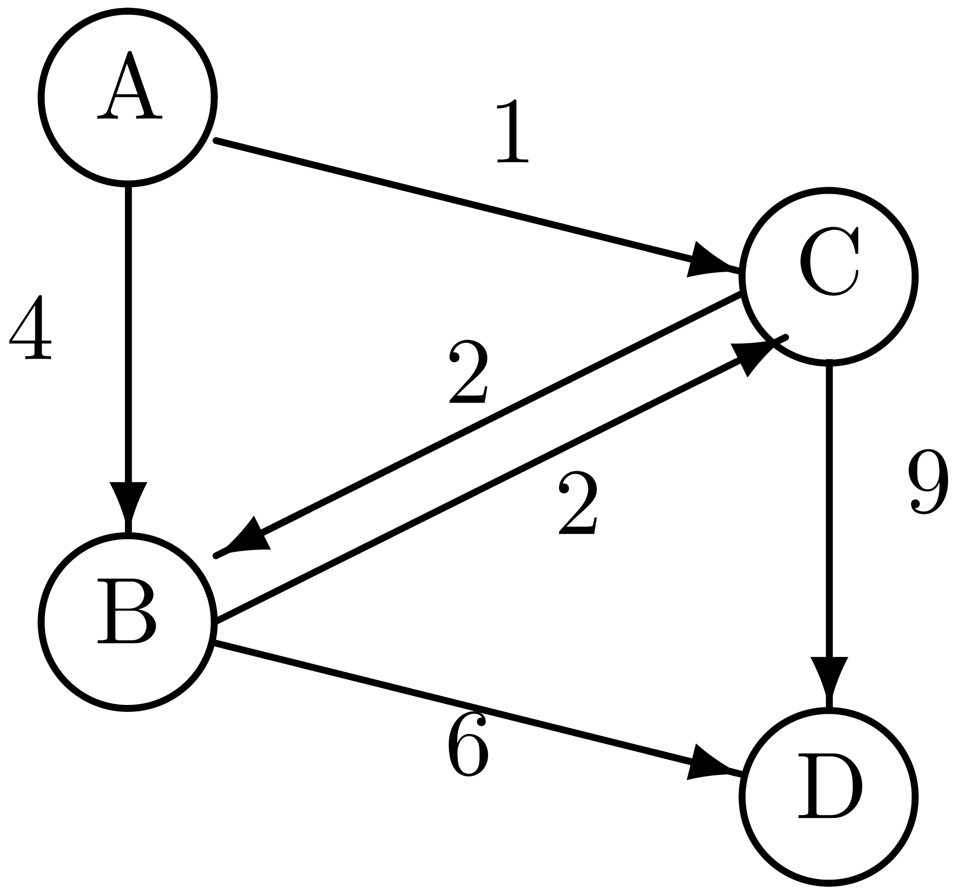
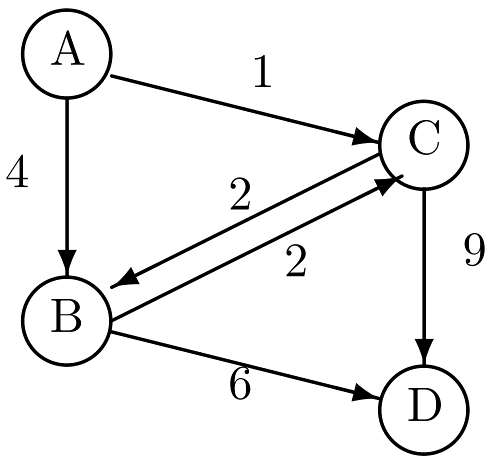
A* Example



$$h(A) = 8$$
$$h(B) = 3$$
$$h(C) = 7$$
$$h(D) = 0$$

START = A
GOAL = D

A* Example



$h(A) = 8$
$h(B) = 3$
$h(C) = 7$
$h(D) = 0$

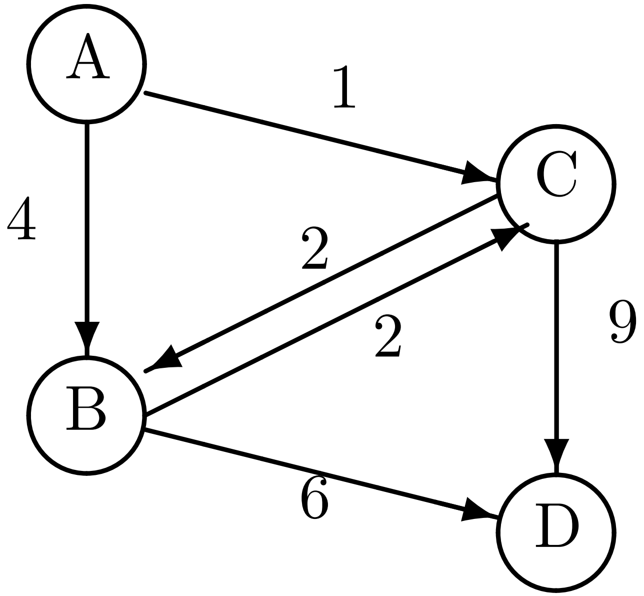
START = A
GOAL = D

h*(A) = 9
h*(B) = 6
h*(C) = 8
h*(D) = 0

- This heuristic is admissible
- Typically we know admissibility by some argument. E.g., Manhattan distance in the 8-puzzle. Euclidian distance in Romanian travel
- Here the graph is small enough to check that h(s) ≤ h*(s) for all states in the state space.

A*

- First **no cycle checking of either type.**
- Successive states of OPEN:
  Items on OPEN are paths
  Where each path <s0, s2 …> a sequence of states
- We also keep track of the g-value, h-value, and f-value of each path.



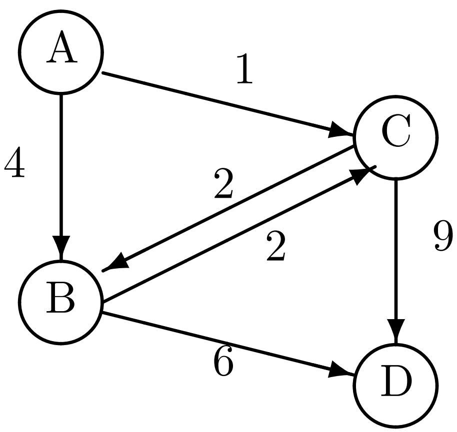
$h(A) = 8$
$h(B) = 3$
$h(C) = 7$
$h(D) = 0$

A*

OPEN:

1. {<A> (g-val+h-val=f-val)} == {<A> (0+8= 8)}
2. {<A,B> (4+3=7), <A,C> (1+7=8)}
3. {<A,C> (1+7=8), <A,B,D> (10+0=10), <A,B,C> (6+7=13),}
4. {<A,C,B> (3+3=6), <A,B,D> (10+0=10), <A,C,D> (10+0=10), <A,B,C> (6+7=13)}
5. {<A,C,B,D> (9+0=9), <A,B,D> (10+0=10), <A,C,D> (10+0=10), <A,C,B,C> (5+7=12), <A,B,C> (6+7=13)}
6. Return solution path <A,C,B,D>

Green = next node expanded



$h(A) = 8$
$h(B) = 3$
$h(C) = 7$
$h(D) = 0$

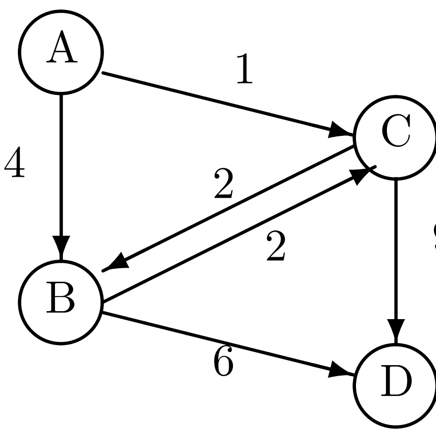
A* Path Checking

OPEN:

1. {<A> (g-val+h-val=f-val)} == {<A> (0+8= 8)}
2. {<A,B> (4+3=7), <A,C> (1+7=8)}
3. {<A,C> (1+7=8), <A,B,D> (10+0=10), <A,B,C> (6+7=13),}
4. {<A,C,B> (3+3=6), <A,B,D> (10+0=10), <A,C,D> (10+0=10), <A,B,C> (6+7=13)}
5. {<A,C,B,D> (9+0=9), <A,B,D> (10+0=10), <A,C,D> (10+0=10), ~~<A,C,B,C> (5+7=12),~~ <A,B,C> (6+7=13)}
6. Return solution path <A,C,B,D>

Green = next node expanded

only one path
not added to
OPEN



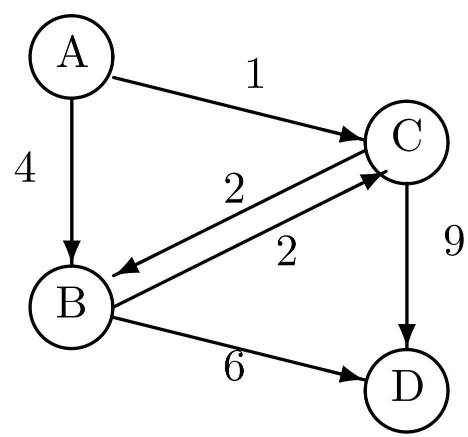
$h(A) = 8$
$h(B) = 3$
$h(C) = 7$
$h(D) = 0$

A* Cycle Checking

OPEN:

1. {<A> (g-val+h-val=f-val)} == {<A> (0+8= 8)}
2. {<A,B> (4+3=7), <A,C> (1+7=8)}
3. {<A,C> (1+7=8), <A,B,D> (10+0=10)}
4. {<A,C,B> (3+3=6), <A,B,D> (10+0=10), <A,B,C> (6+7=13)}
5. {<A,C,B,D> (9+0=9), <A,B,D> (10+0=10), <A,B,C> (6+7=13)}
6. Return solution path <A,C,B,D>

Green = next node expanded



$h(A) = 8$
$h(B) = 3$
$h(C) = 7$
$h(D) = 0$

A*

OPEN:

1. {<A> (g-val+h-val=f-val)} == {<A> (0+8= 8)}
2. {<A,B> (4+3=7), <A,C> (1+7=8)}
3. {<A,C> (1+7=8), <A,B,D> (10+0=10)}
4. {<A,C,B> (3+3=6), <A,B,D> (10+0=10), <A,B,C> (6+7=13)}
5. {<A,C,B,D> (9+0=9), <A,B,D> (10+0=10), <A,B,C> (6+7=13)}
6. Return solution path <A,C,B,D>

This is an optimal solution. So C* = 9

By Proposition 2. A* with an admissible heuristic never expands a path with f-value > C*

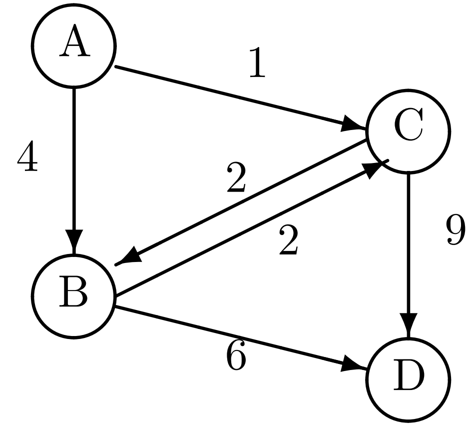
f-values of paths expanded: 8, 7, 8, 6, 9

A*

OPEN:

1. {<A> (g-val+h-val=f-val)} == {<A> (0+8= 8)}
2. {<A,B> (4+3=7), <A,C> (1+7=8)}
3. {<A,C> (1+7=8), <A,B,D> (10+0=10)}
4. {<A,C,B> (3+3=6), <A,B,D> (10+0=10), <A,B,C> (6+7=13)}
5. {<A,C,B,D> (9+0=9), <A,B,D> (10+0=10), <A,B,C> (6+7=13)}
6. Return solution path <A,C,B,D>

At every step a prefix of an optimal path must be on OPEN. Easy to see that this holds.



$h(A) = 8$
$h(B) = 3$
$h(C) = 7$
$h(D) = 0$

Consequences of monotonicity

1. The sequence of f-values of the paths expanded by A* is non-decreasing. That is, if n_2 is expanded **after** n_1 by A* then f(n1) ≤ f(n2). (Not necessarily true for an admissible heuristic as that heuristic might be non-monotone)
2. With a monotone heuristic, the first time A* expands a path **n =** $<s_0, \ldots, s_n>$ that reaches the state s_n, **n** must be a minimum cost path to s_n
   - This means that cycle checking need not keep track of the minimum cost of getting to a state found so far. It can simply prune all future paths to a state if that state has already been reached by an expanded path.
   - Again this is not necessarily true for an admissible heuristic. So with admissible heuristics that are not monotone (or not known to be monotone) we have to keep track of the cost of getting to states as well as the states visited during cycle checking.

A* Sequence of f-values expanded

OPEN:

1. 8 {<A> (g-val+h-val=f-val)} == {<A> (0+8= 8)}
2. 7 {<A,B> (4+3=7), <A,C> (1+7=8)}
3. 8 {<A,C> (1+7=8), <A,B,D> (10+0=10)}
4. 6 {<A,C,B> (3+3=6), <A,B,D> (10+0=10), <A,B,C> (6+7=13)}
5. 9 {<A,C,B,D> (9+0=9), <A,B,D> (10+0=10), <A,B,C> (6+7=13)}
6. Return solution path <A,C,B,D>

8, 7, 8, 6, 9

The heuristic is not monotone!

Also we first reach state B via the path <A,B> that costs 4. Then later we reach state B via a cheaper path <A,C,B> that costs 3—again this can't happen with monotone heuristic

A* Short Questions

- If h(n) is admissible and s is the start node how is h(s) related to the cost of the solution eventually found by A*?

A* Short Questions

- What happens when h(n) = h*(n)
  a. A* only expands nodes that lie on an optimal path to the goal
  b. Does this mean that A* will find an solution in time linear in the length of an optimal solution?

A* Short Questions

- What happens when $h(n) = h^*(n)$
  a. A* only expands paths that are prefixes of an optimal path to the goal

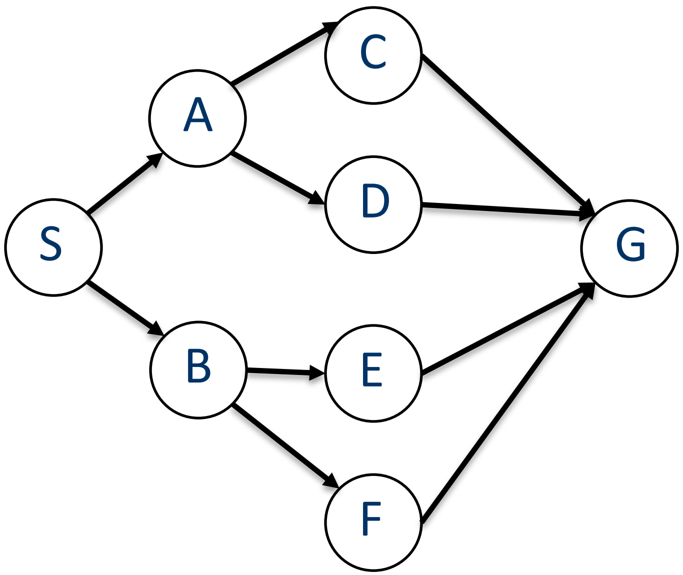
     Say that A* expands path p. $f(p) = g(p) + h(p)$
     $= g(p) + h^*(p) \leq C^*$ (proposition 1)

     But $g(p) + h^*(p)$ is the cost of a the path p plus some extension that reaches a goal state. So it is the cost of some solution path and so we must have
     $g(p) + h^*(p) \geq C^*$

     Hence $g(p) + h^*(p) = C^*$. And there is a way of extending p to reach the goal with total cost C^*. That is p is a prefix of an optimal path.

A* Short Questions

- What happens when h(n) = h*(n)
  b. Does this mean that A* will find an solution in time linear in the length of an optimal solution?



S is initial state of search
G is goal state

All actions cost one

h(s) = 3
h(A) = h(B) = 2
h(C) = h(D) = h(E) = h(F) = 1
h(G) = 0

A* Short Questions

- What happens when h(n) = h*(n)
  b. in this example every path to G is optimal, and if we add more layers we can obtain an exponential number of paths to G all of which are optimal. So the fact that A* when h(n) = h*(n) only expands prefixes of optimal paths is not sufficient to ensure that A* will only expand a linear number of paths: as their can be an exponential number of optimal paths.