St. Francis Xavier University

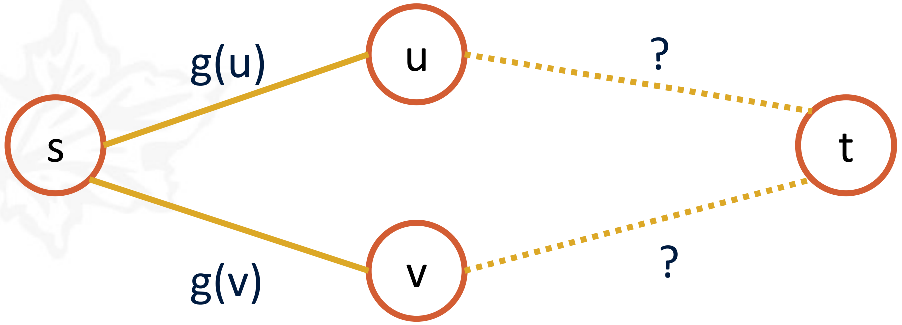
CSCI-564 CONSTRAINT PROCESSING AND HEURISTIC SEARCH

LECTURE 6 - INFORMED SEARCH

Dr. Jean-Alexis Delamer

Recap

- Uninformed search
  - Know the value of one action/edge.
  - No information on the estimated cost to reach the goal.



- Algorithms:
  - Algorithms explore all the nodes.
  - BFS, DFS, Dijkstra, Dynamic programming

Informed search

- Heuristic search
  - An estimate of the remaining distance/cost to reach the goal.
  - Use the estimate to prioritize the node expansion.

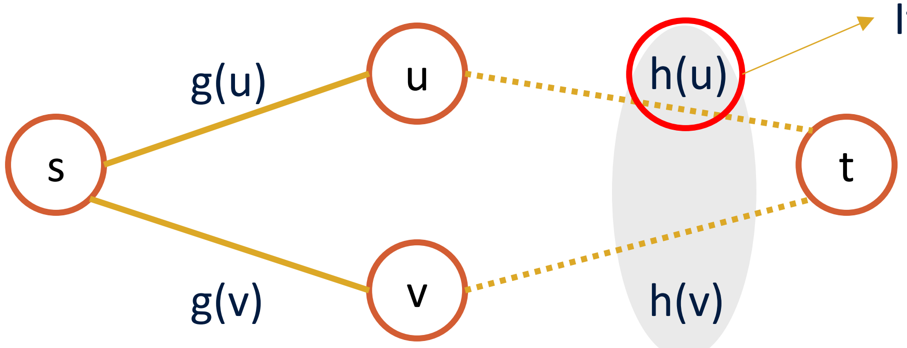
- It’s a way to exploit domain knowledge to prune the search tree.
  - Informed search

Informed search

- The estimate of a node u is noted $h(u)$.

But it's not always the exact distance!

It's your expert knowledge

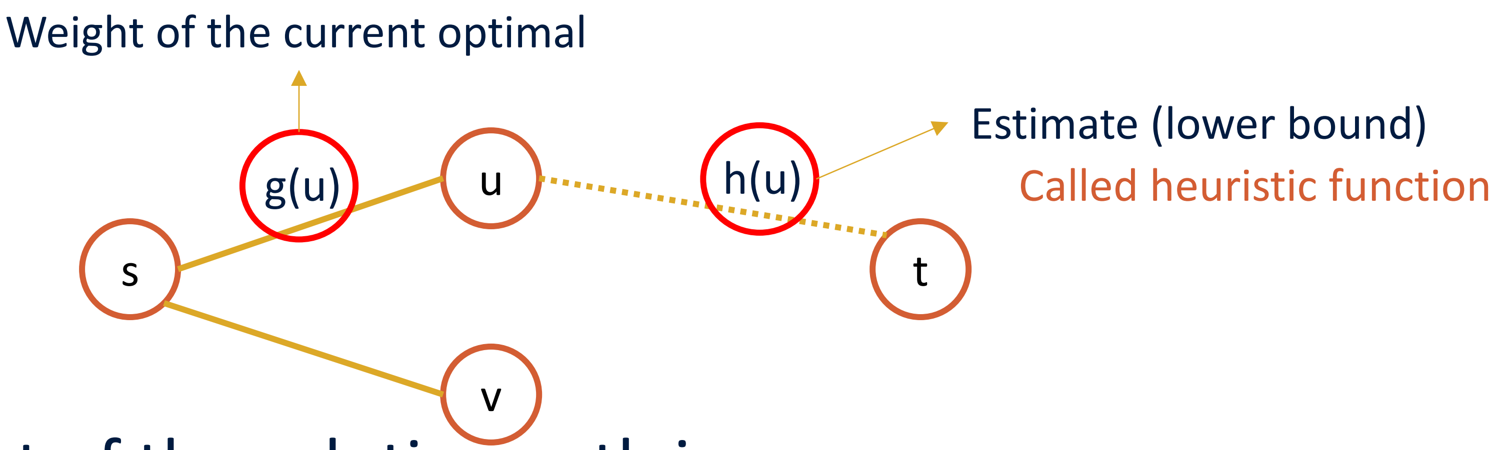


Not the same value

Do you know any algorithm that use heuristic search?

A* algorithm

- The most prominent heuristic search algorithm is A*.



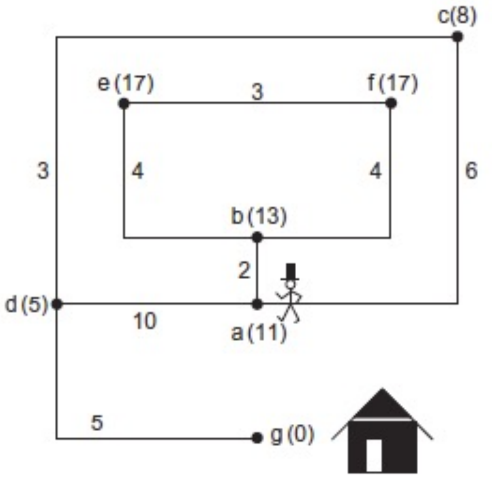
- The estimated cost of the solution path is:

$$f(u) = g(u) + h(t)$$

A* algorithm

```
Closed ← ∅ ;Open ← {s}; f(s) ← h(s)
While (Open ≠ ∅)
     Remove u from Open with minimum f(u)
     Insert u into Closed
     If u ∈ T return Path(u)
     Else
          Succ(u) ← Expand(u)
          Foreach v in Succ(u)
               If v in Open
                    If g(u)+ w(u,v) < g(v)
                         parent(v) ← u
                         f(v) ← g(u)+w(u,v)+h(v)
               Else if v in Closed
                    If g(u)+ w(u,v) < g(v)
                         parent(v) ← u
                         f(v) ← g(u)+w(u,v)+h(v)
                         Remove v from Closed
                         Insert v in Open
               Else
                    parent(v) ← u
                    Initialize f(v) ← g(u)+w(u,v)+h(v)
                    Insert v in Open
```

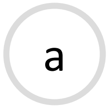
A* Algorithm



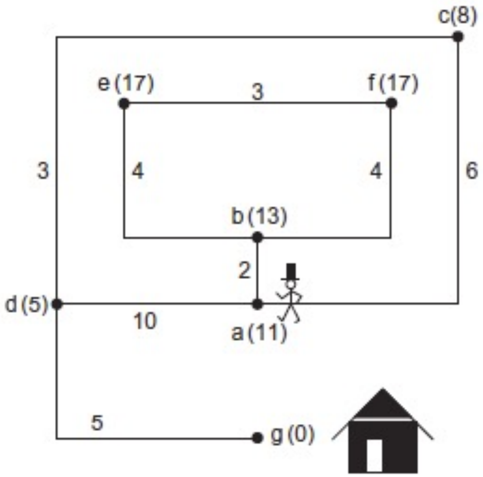
g(a)

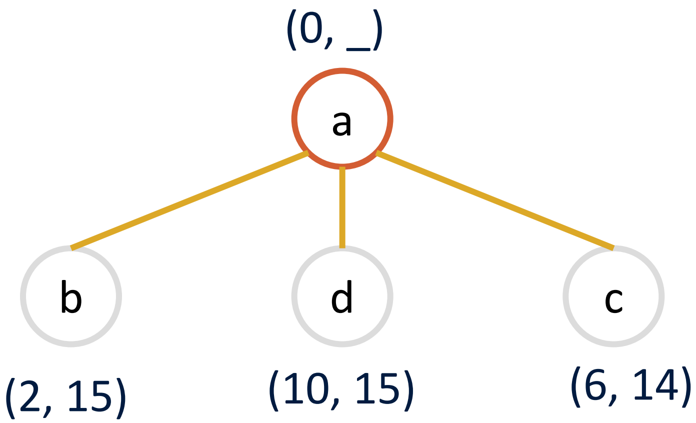
f(a)

```
Open = {a(0,11)}
Closed = {}
```



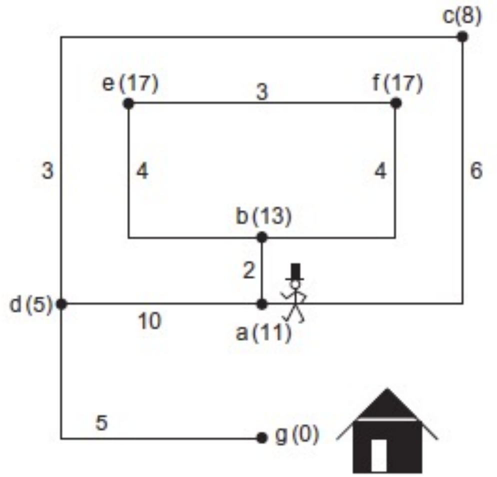
A* Algorithm



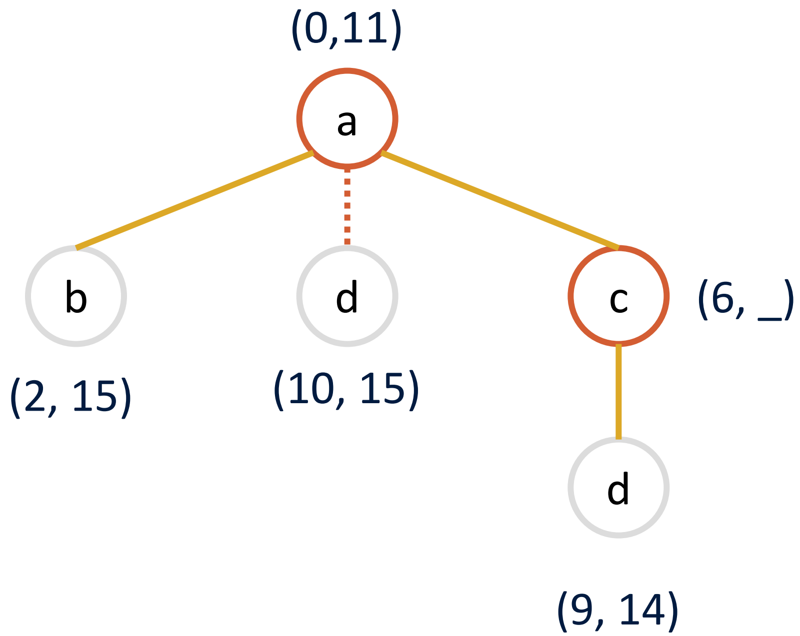



```
Open = {c(6,14), b(2,15),
    d(10,15)}
Closed = {a}
```

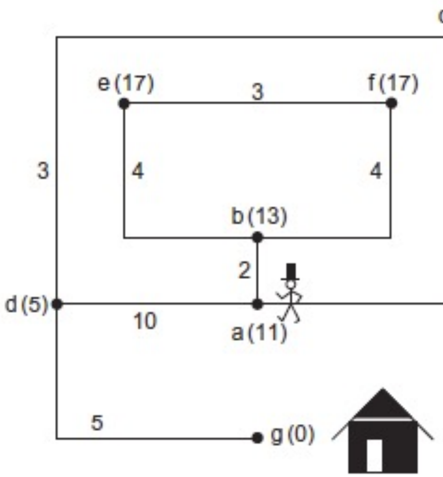
A* Algorithm



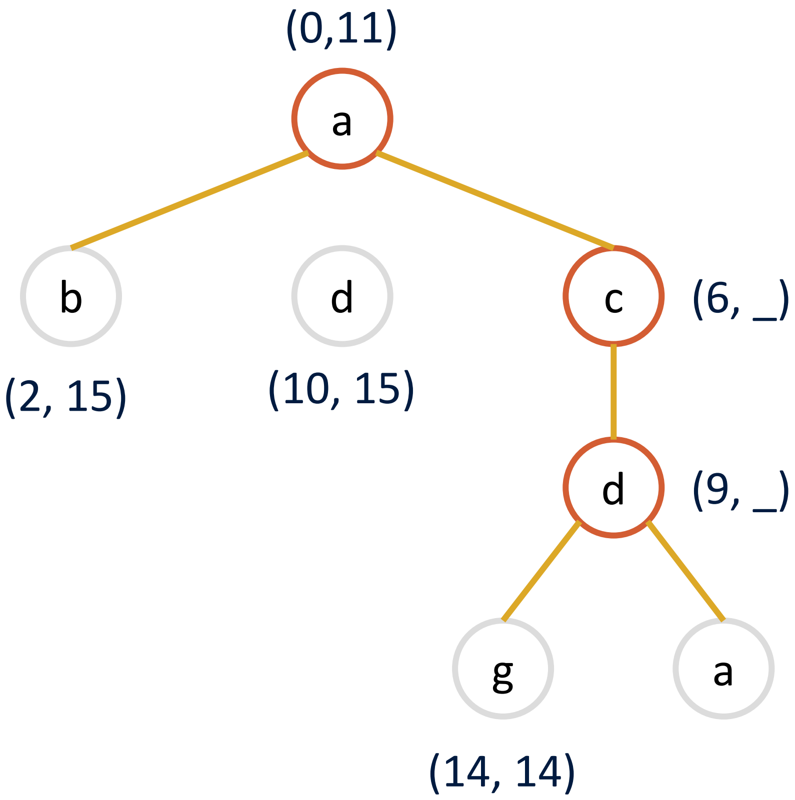

```
Open = {b(2,15),
    d(9,14)}
Closed = {a, c}
```



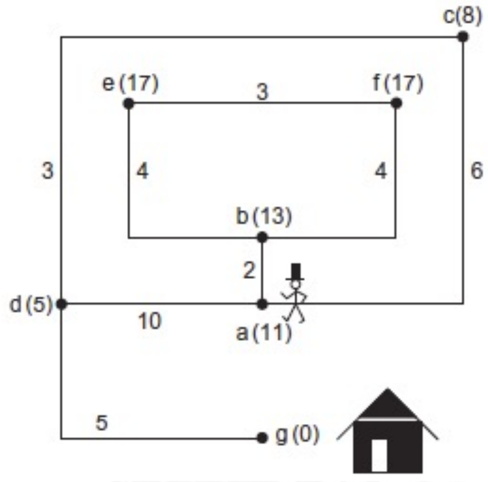
A* Algorithm

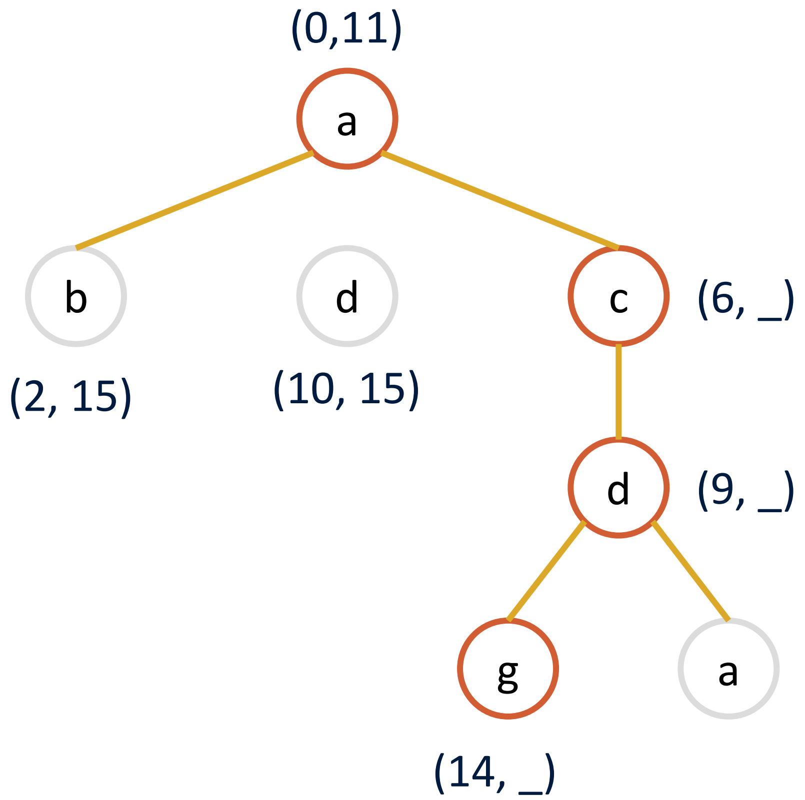



```
Open = {b(2,15), g(14,14)}
Closed = {a, c, d}
```



A* Algorithm



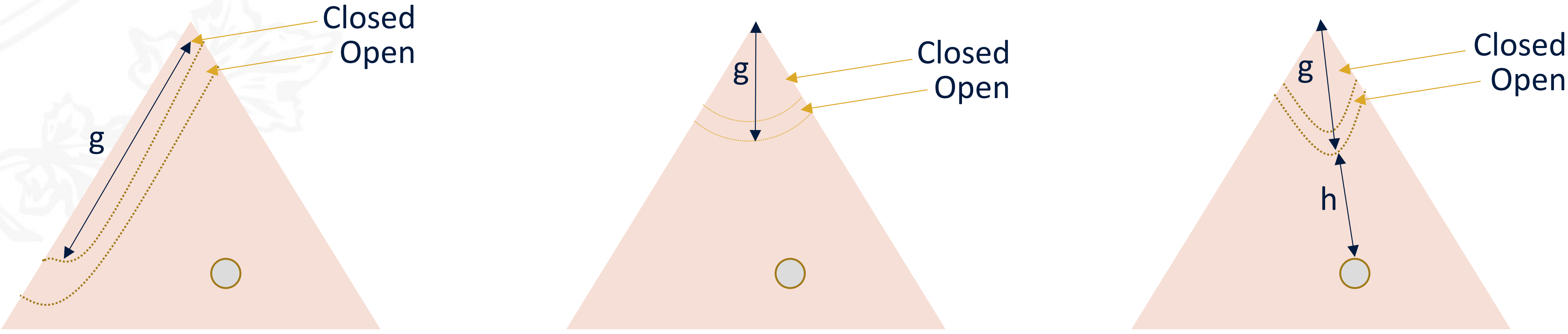



```
Open = {b(2,15)
Closed = {a, c, d, g}
```

We don’t need to visit every nodes (e and f are avoided).

A* Algorithm

- Difference between A* and other algorithms:



	DFS	BFS	A*
Expansion criterion	$u \in \text{Open}, \max g(u)$	$u \in \text{Open}, \min g(u)$	$u \in \text{Open}, \min g(u) + h(u)$

Optimality

- We often say that A* has optimal efficiency:
  - It gives an optimal solution.
  - Expands the minimal number of nodes.

Is it true?

Optimality

- We often say that A* has optimal efficiency:
  - It gives an optimal solution.
  - Expands the minimal number of nodes.

- It is true for consistent heuristics.
- But not always for admissible heuristics.

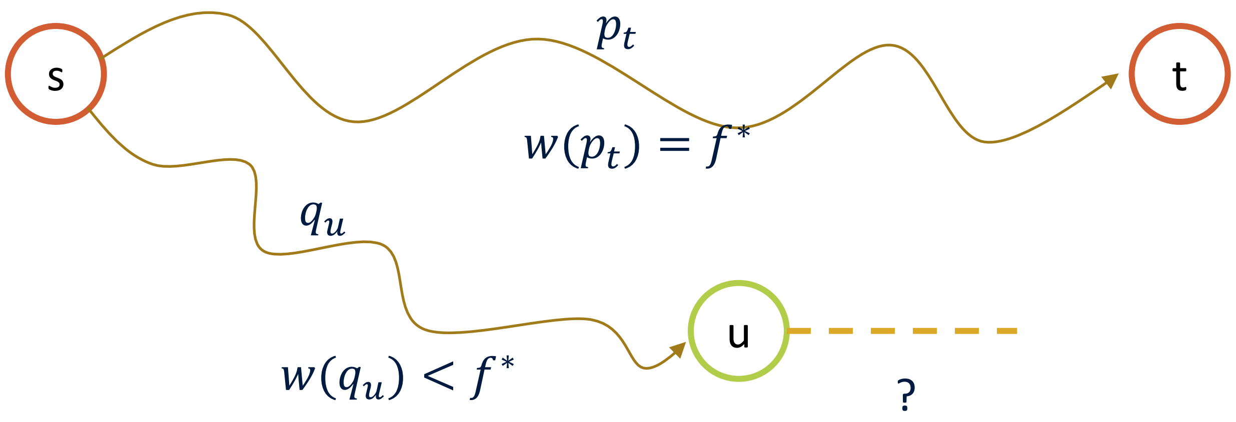
Consistent heuristics

- Definition (Consistent heuristic):
  - A goal estimate h is a consistent heuristic if $h(u) \leq w(u, v) + h(v)$ for all edges $e = (u, v) \in E$.

- Theorem (Efficiency Lower Bound):
  - Let G be a problem graph with nonnegative weight function, with initial node s and final node set T, and let $f^* = \delta(s, T)$ be the optimal solution cost. Any optimal algorithm has to visit all $u \in V$ nodes with $\delta(s, u) < f^*$.

Consistent heuristics

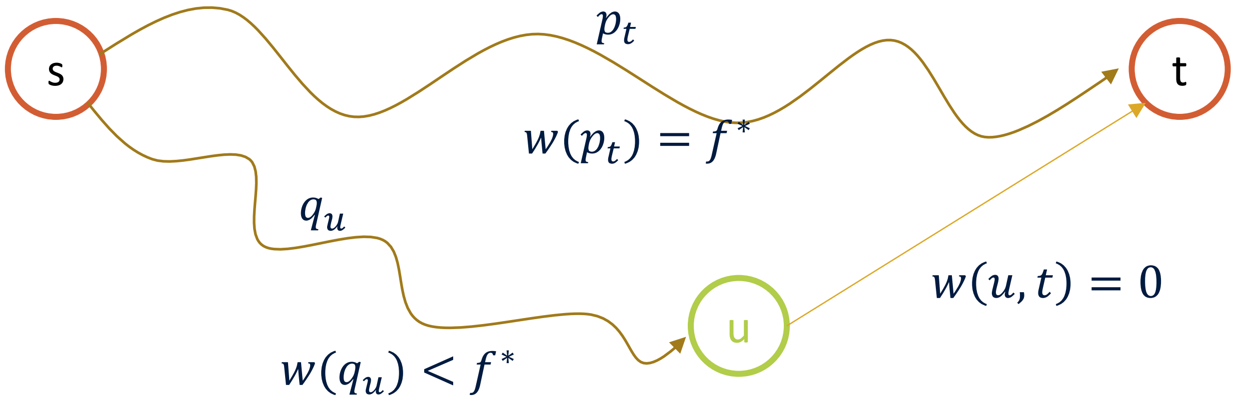
- Theorem (Efficiency Lower Bound):
  - Let G be a problem graph with nonnegative weight function, with initial node s and final node set T, and let $f^* = \delta(s, T)$ be the optimal solution cost. Any optimal algorithm has to visit all $u \in V$ nodes with $\delta(s, u) < f^*$.
- Proof:



Consistent heuristics

- Theorem (Efficiency Lower Bound):
  - Let G be a problem graph with nonnegative weight function, with initial node s and final node set T, and let $f^* = \delta(s,T)$ be the optimal solution cost. Any optimal algorithm has to visit all $u \in V$ nodes with $\boxed{\delta(s,u) < f^*}$.
- Proof:

$w(q) = w(q_u) + w(u,t) = w(q_u) = \delta(s,u) < f^*$



Consistent heuristics

- Definition (Consistent heuristic):
  - A goal estimate h is a consistent heuristic if $h(u) \leq w(u, v) + h(v)$ for all edges $e = (u, v) \in E$.

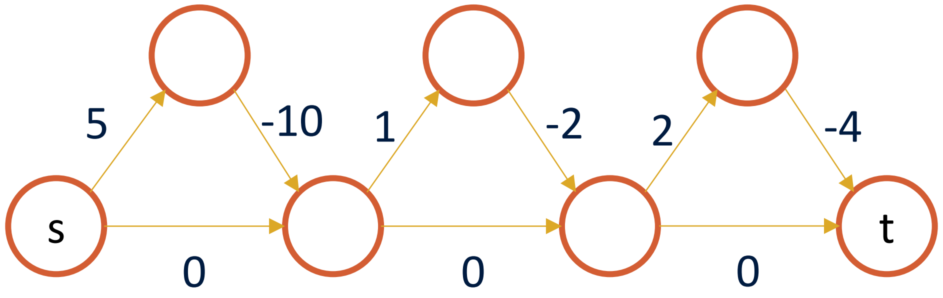
- Theorem (Efficiency Lower Bound):
  - Let G be a problem graph with nonnegative weight function, with initial node s and final node set T, and let $f^* = \delta(s, T)$ be the optimal solution cost. Any optimal algorithm has to visit all $u \in V$ nodes with $\delta(s, u) < f^*$.

- The number of nodes that any algorithms expands will have to be larger than or equal to the number of nodes that A* expands.

Admissible heuristics

- Definition (Admissible Heuristic):
  - An estimate h is an admissible heuristic if it is a lower bound for the optimal solution costs; that is, $h(s) \leq \delta(s, T)$ for all $s \in V$.

- If we have admissibility but not consistency, A* will reopen nodes.
- Worse! A* might reopen nodes exponentially many times.
  - This behavior does not appear frequently in practice.



Admissible heuristics

- In these cases, we use other algorithms:
  - Bellman-Ford algorithm, that deal with negative edge costs.

- But A* is not optimal with non consistent heuristics.

A*

- A* can find a shortest path even though it expands every state at most once. It does not need to reexpand states that it has expanded already.
- A* is at least as efficient as every other search algorithm (that has the same heuristic values as A*). They needs to expand at least the states that A* expands.

Exercise

- The Missionary and Cannibals:
  - At one side of a river there are three missionaries and three cannibals.
  - They have a boat that can transport at most two persons. The goal for all persons is to cross the river. The boat requires someone to cross the river.
  - At no time should the number of cannibals exceed the number of missionaries.

1. Draw the problem graph and provide its adjacency list representation.
2. Solve the problem via DFS and BFS by annotating the graph with numbers.
3. Consider a heuristic function that counts the number of people on the other side of the river. Do you observe an inconsistency?