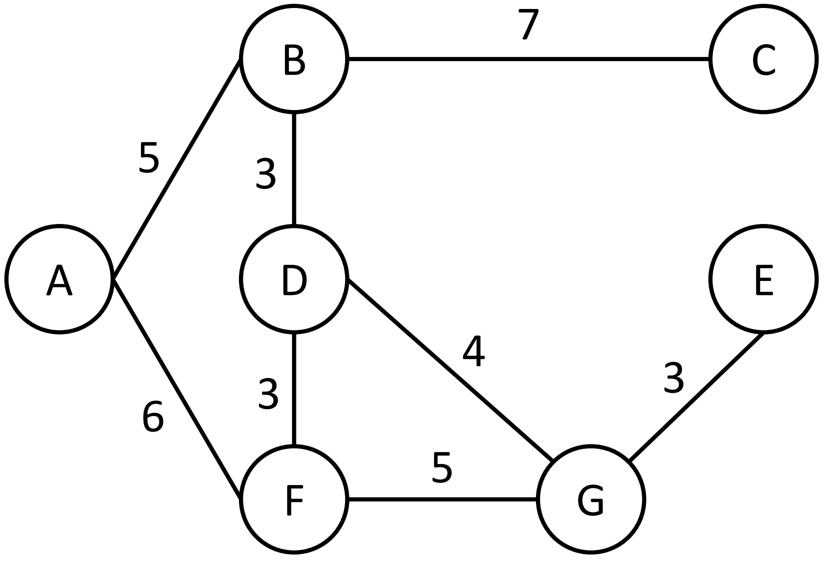
*Sistemi Intelligenti*
*Corso di Laurea in Informatica, A.A. 2019-2020*
*Università degli Studi di Milan*

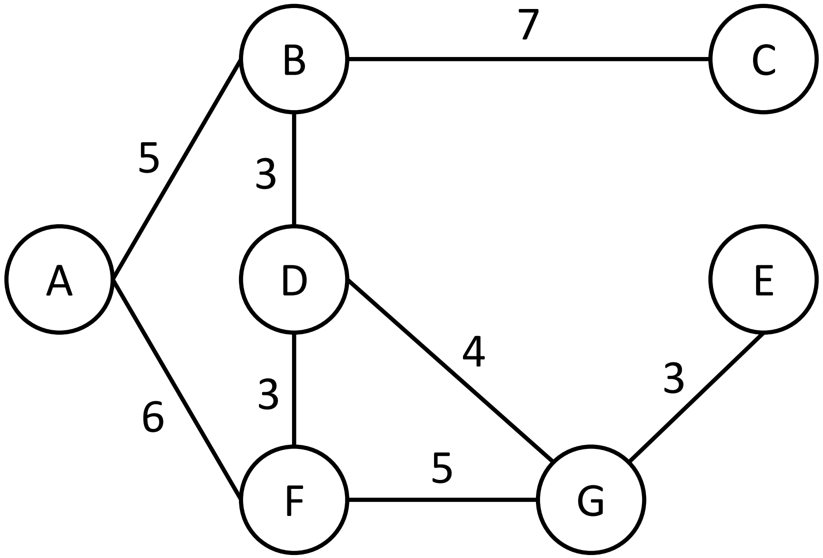
# Search algorithms for planning

**Nicola Basilico**
Dipartimento di Informatica
Via Celoria 18- 20133 Milano (MI)
Ufficio 4008
nicola.basilico@unimi.it
+39 02.503.16289

# UCS with extended list

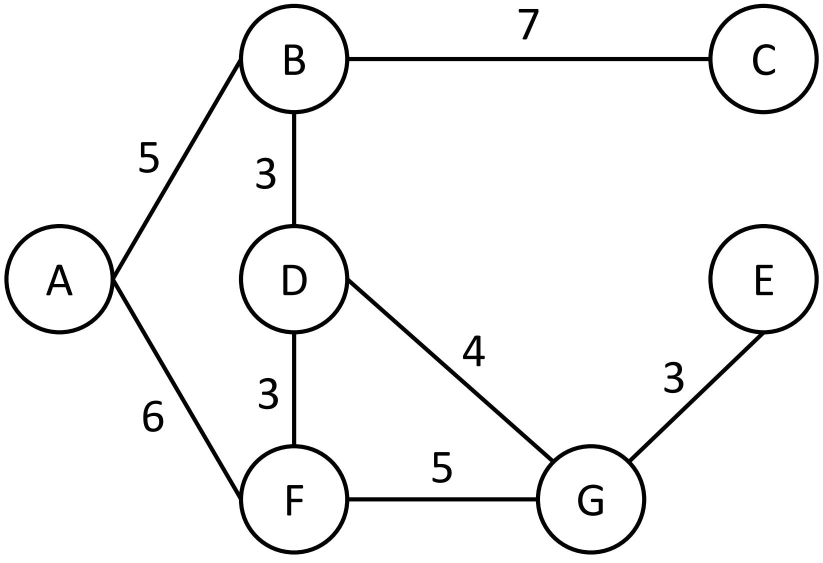


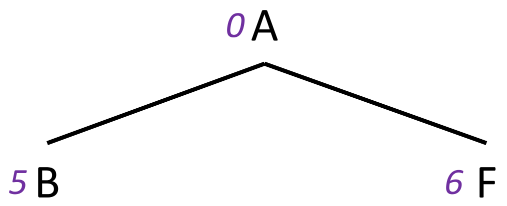
# UCS with extended list



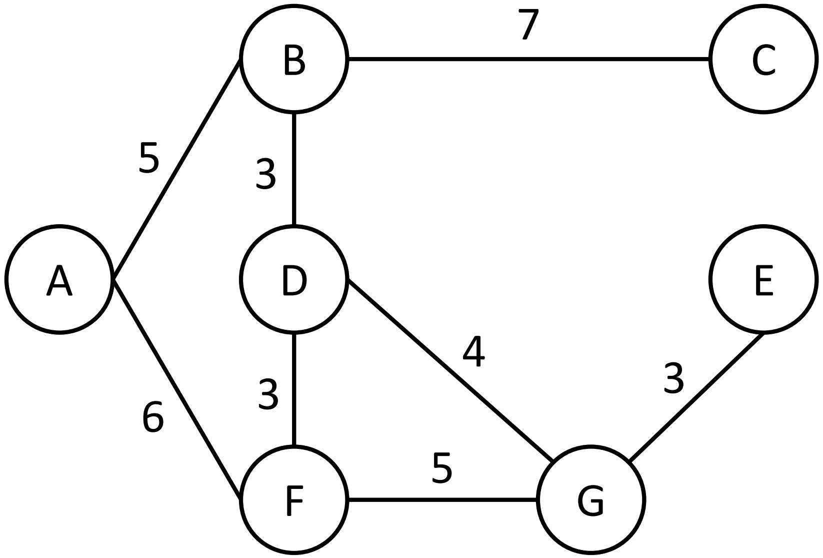

*0* A

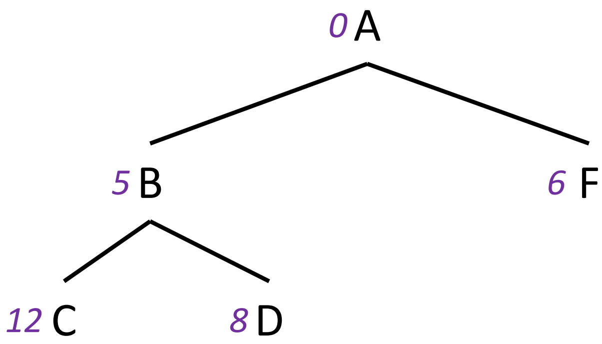
# UCS with extended list



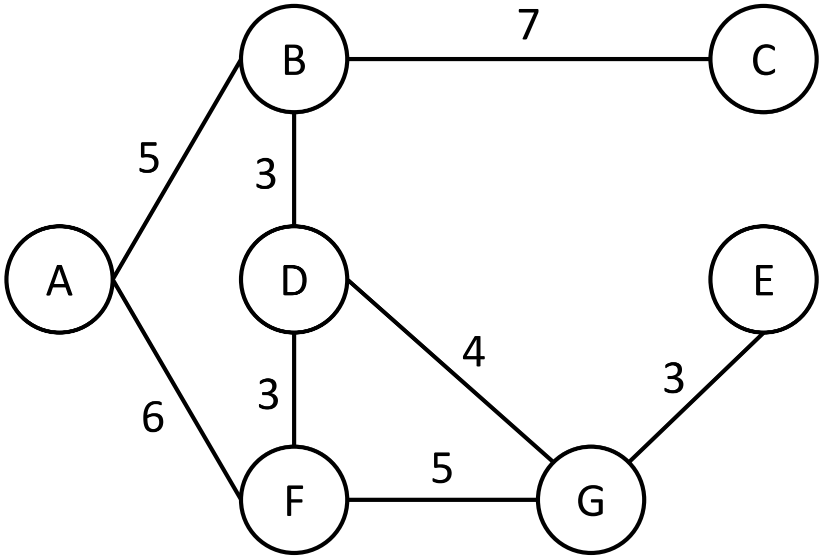



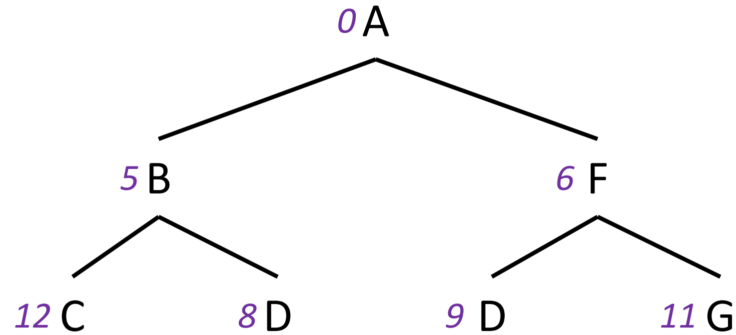
# UCS with extended list



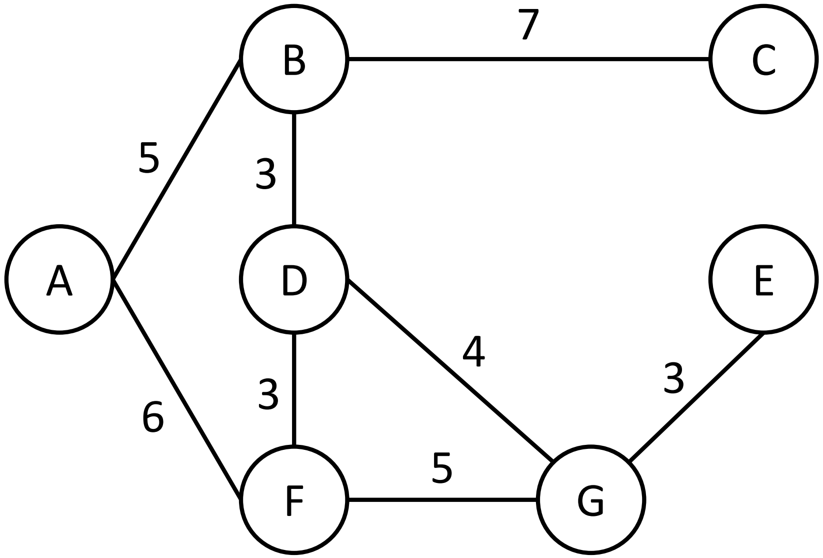



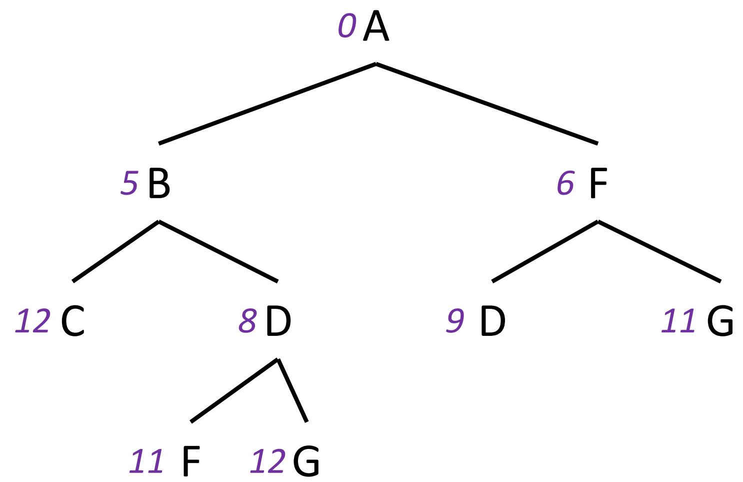
# UCS with extended list



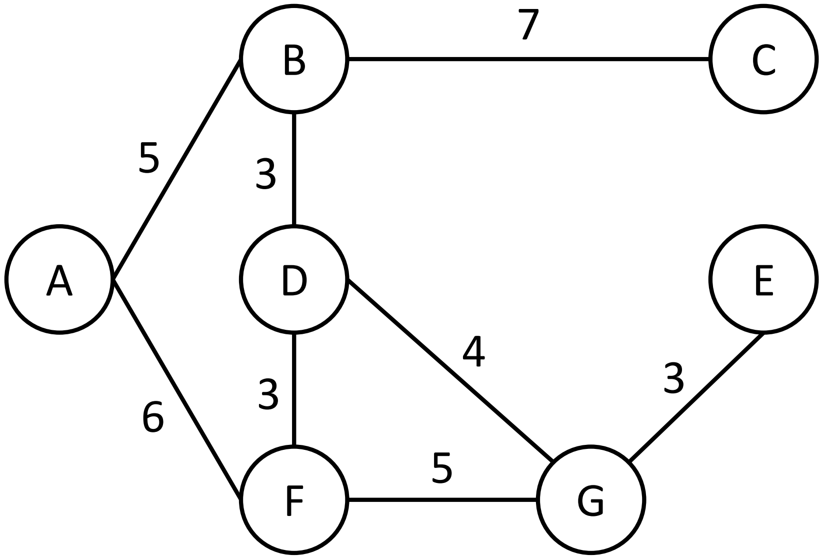



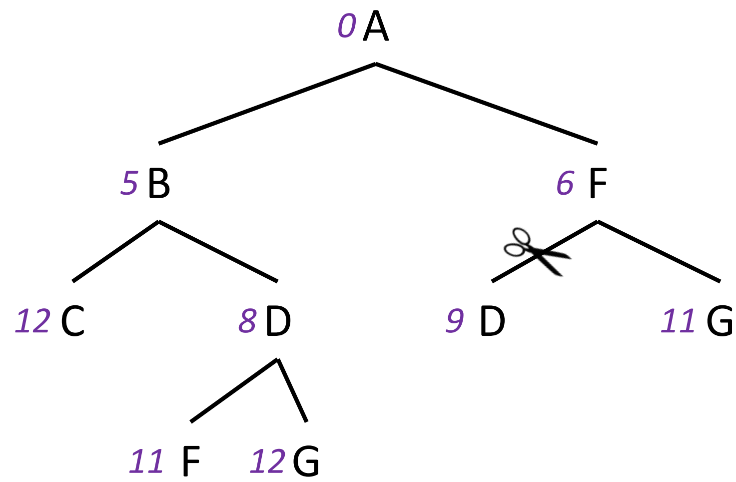
# UCS with extended list



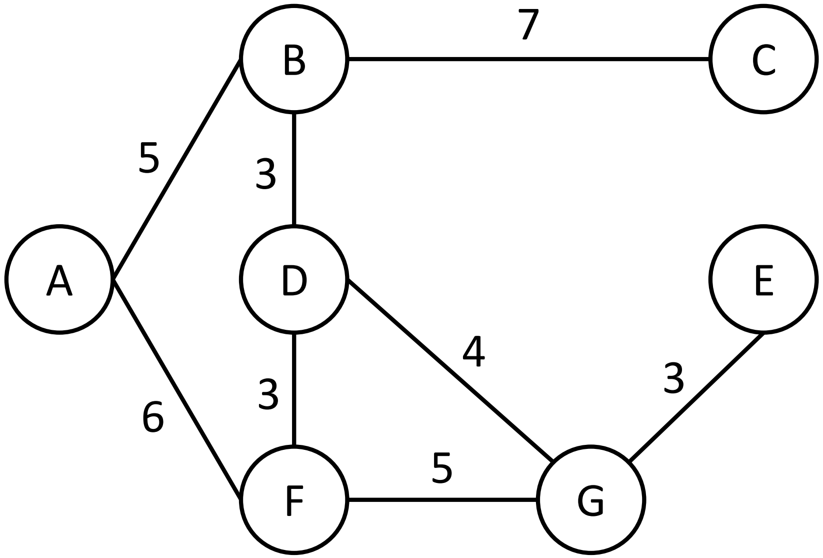



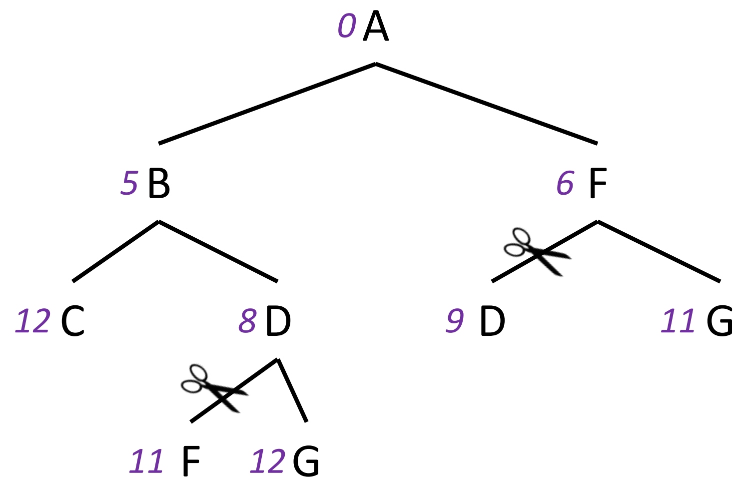
# UCS with extended list



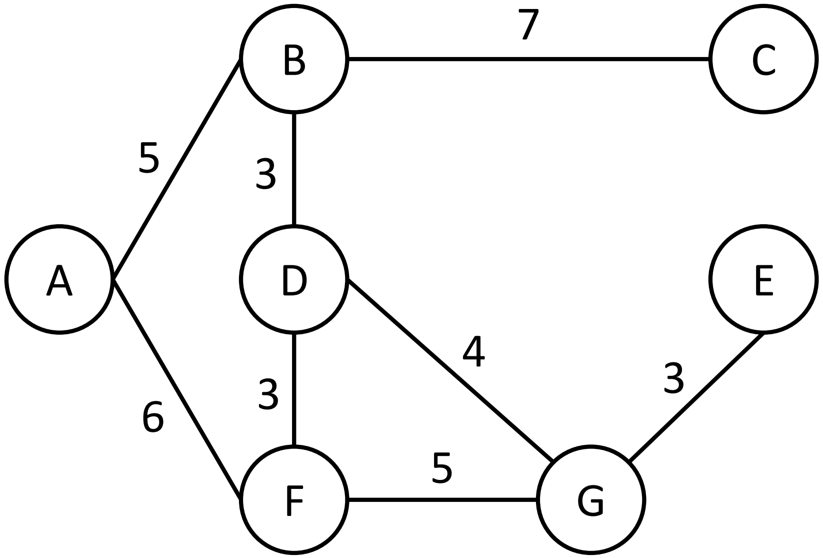



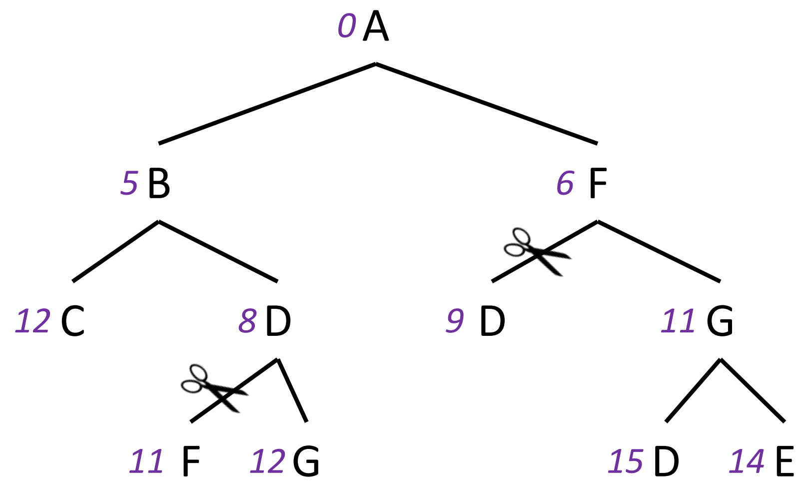
# UCS with extended list



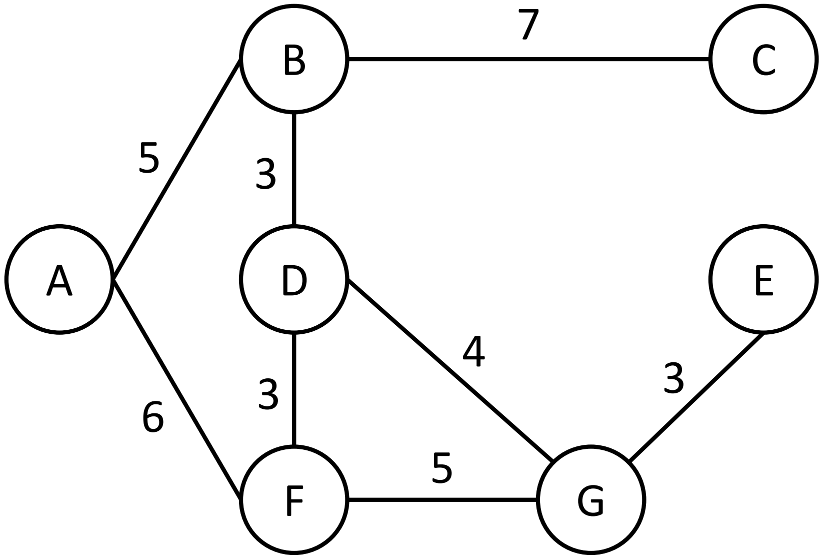



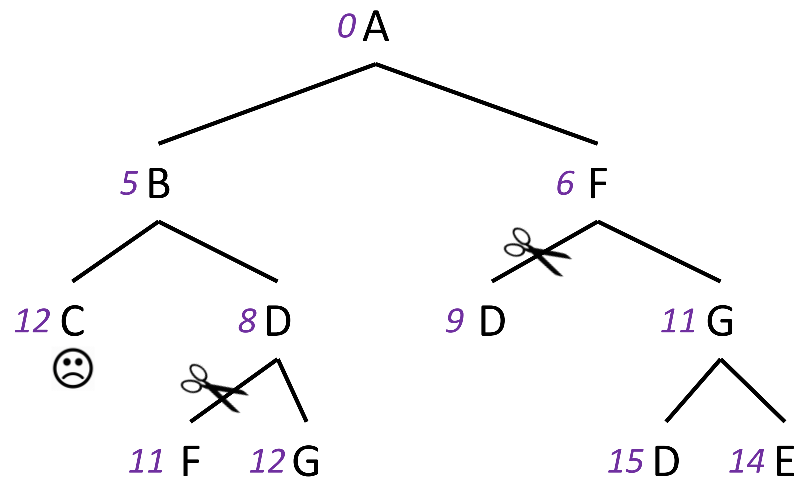
# UCS with extended list



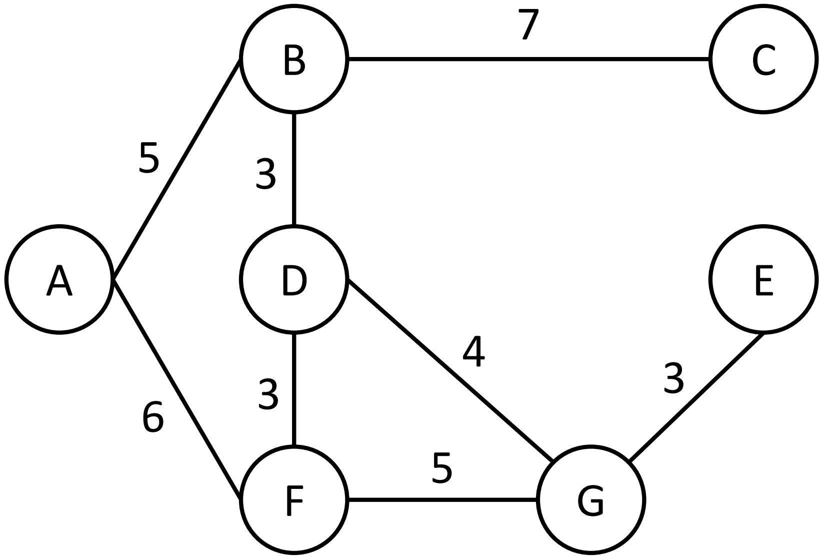



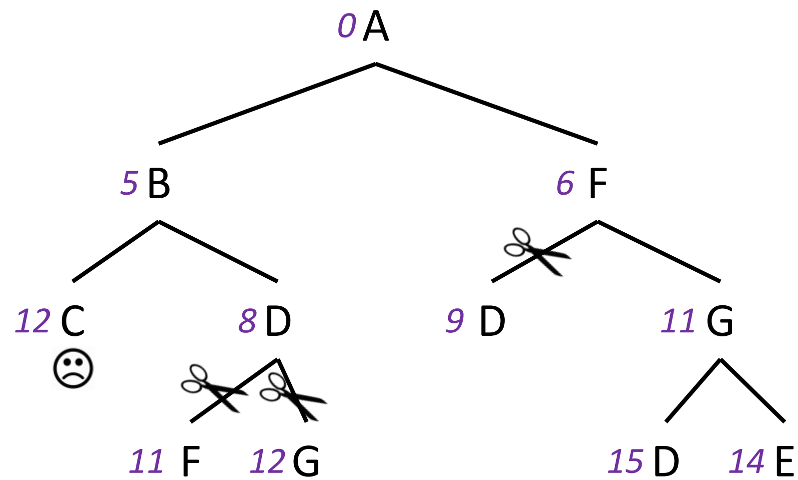
# UCS with extended list



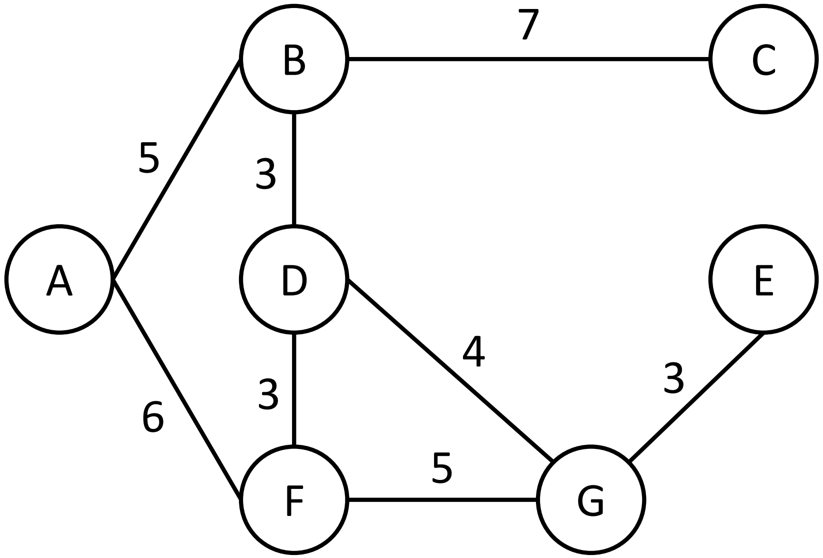



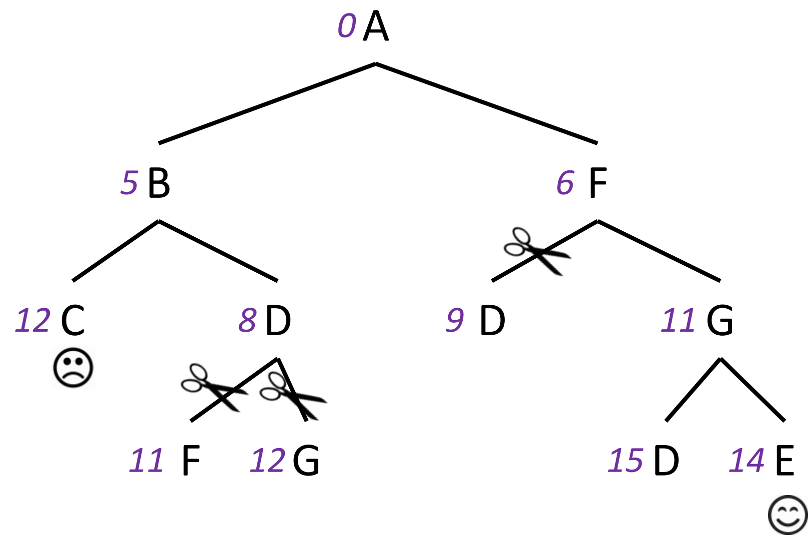
# UCS with extended list



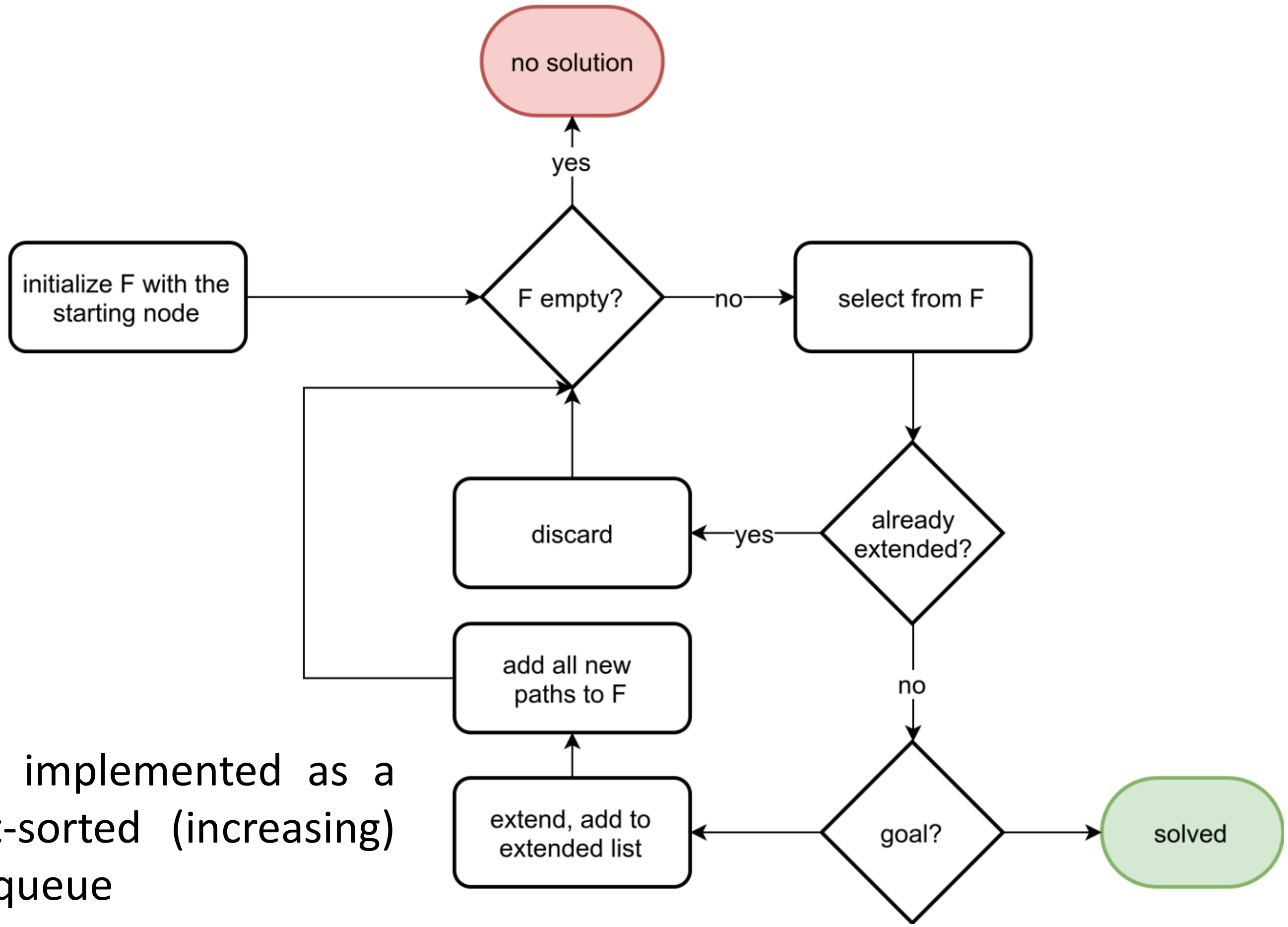
# UCS with extended list




- Thanks to the extended list we can prune two branches

# Implementation



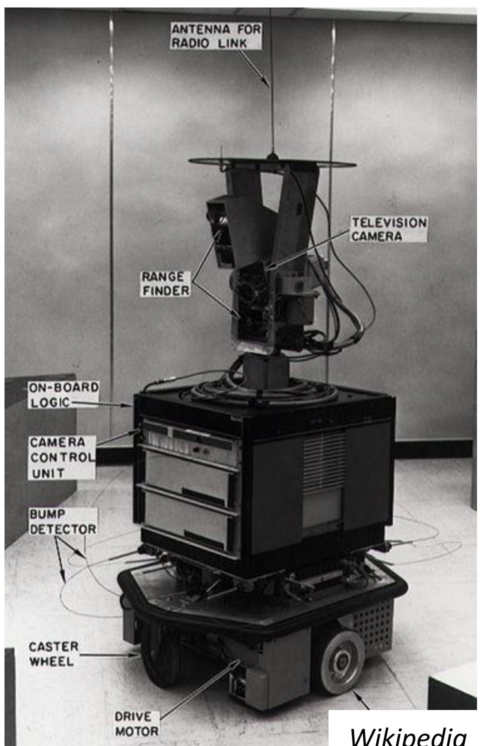

F is implemented as a cost-sorted (increasing) list queue

The goal check is done when the node is selected (not when is generated)

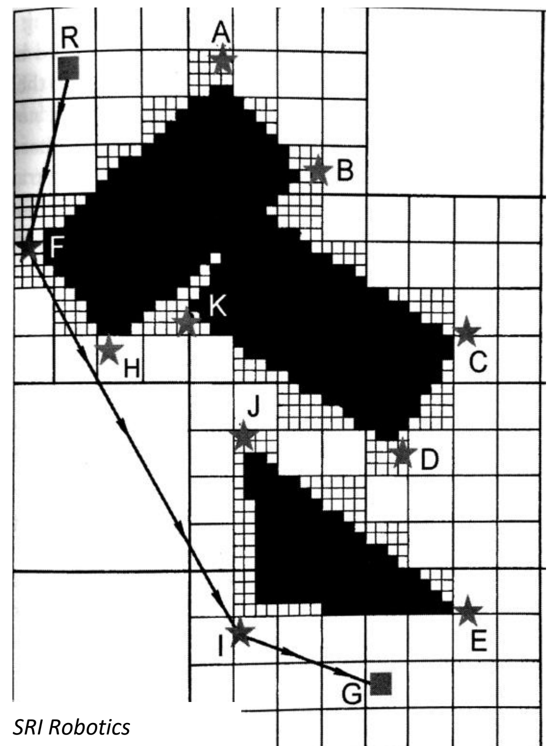
- Question: is this search informed?

# A*

- The informed version of UCS is called A*
- Very popular search algorithm
- It was born in the early days of mobile robotics when, in 1968, Nilsson, Hart, and Raphael had to face a practical problem with Shakey (one of the ancestors of today's mobile robots)




*Wikipedia*




*SRI Robotics*

# A*

- The idea behind A* is simple: perform a UCS, but instead of considering accumulated costs consider the following:

Heuristic
("cost-to-go")

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

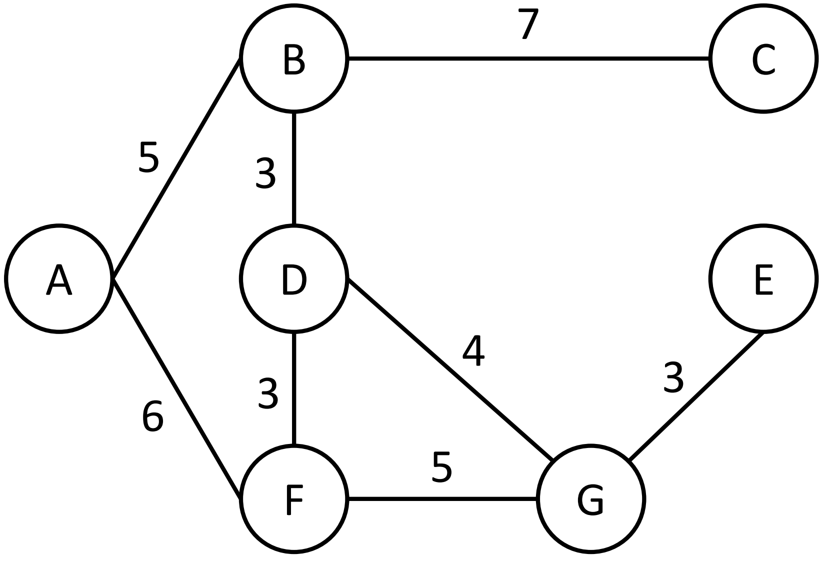
Cost accumulated
on the path to n
("cost-to-come")

- To guarantee that the search is sound and complete we need to require that the heuristic is **admissible**: it is an optimistic estimate or, more formally:

$$h(n) \leq \text{ Cost of the minimum path from n to the goal}$$

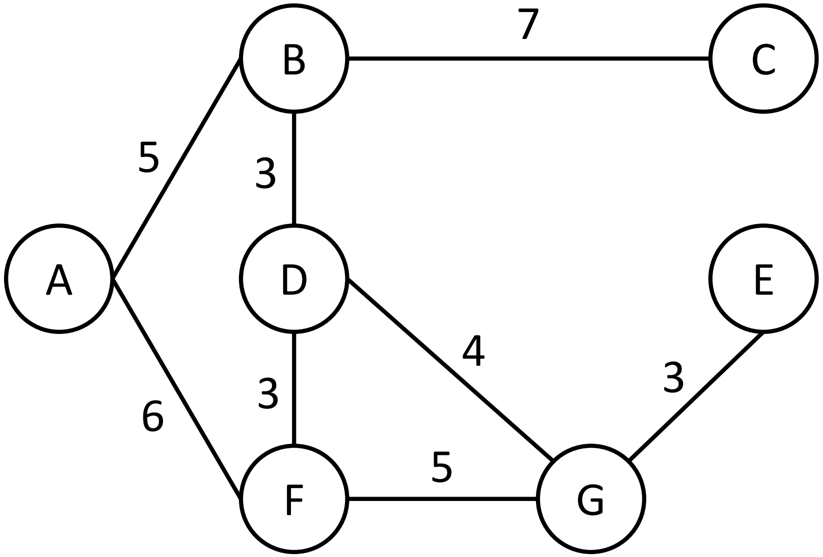
- If the heuristic is not admissible we might discard a path that could actually turn out to be better that the best candidate found so far

# A*




| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |

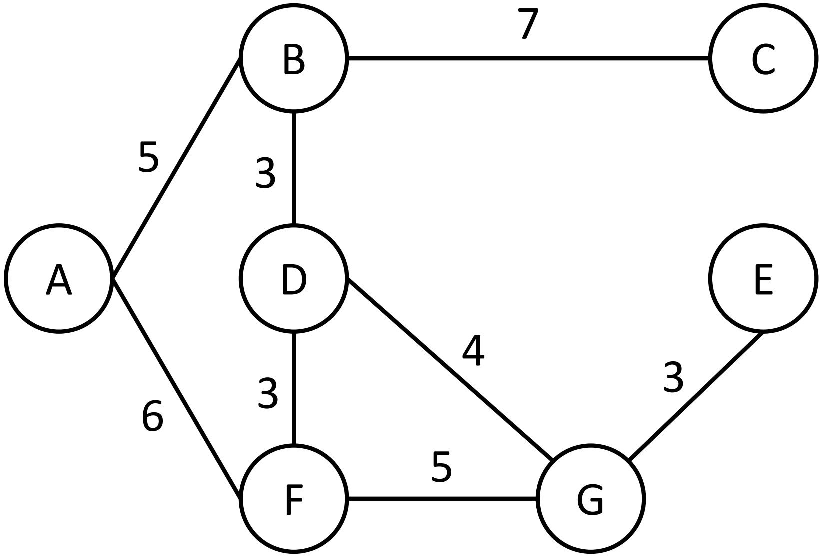
# A*

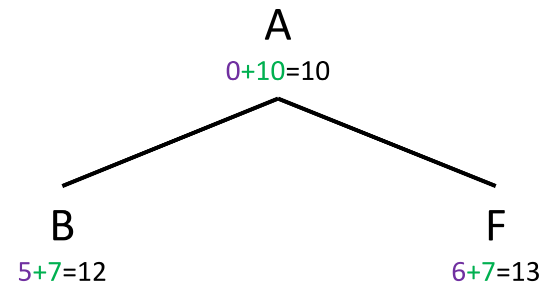



A
0+10=10

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |

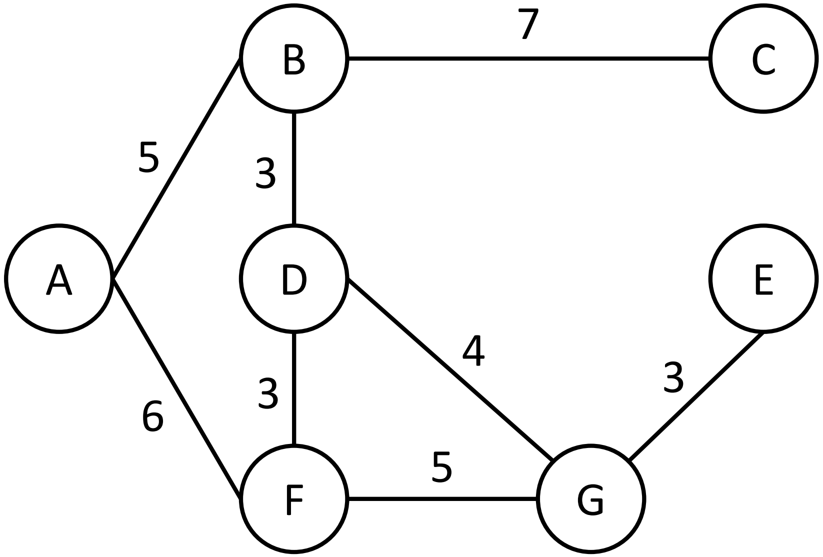
# A*

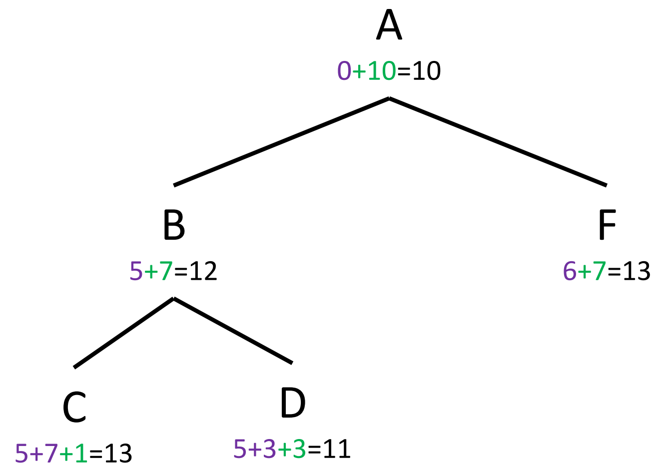



| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |

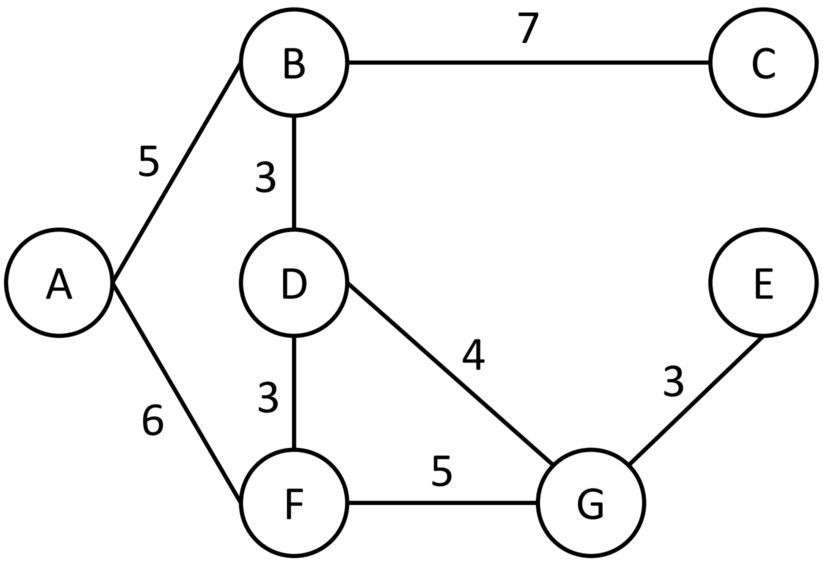
# A*

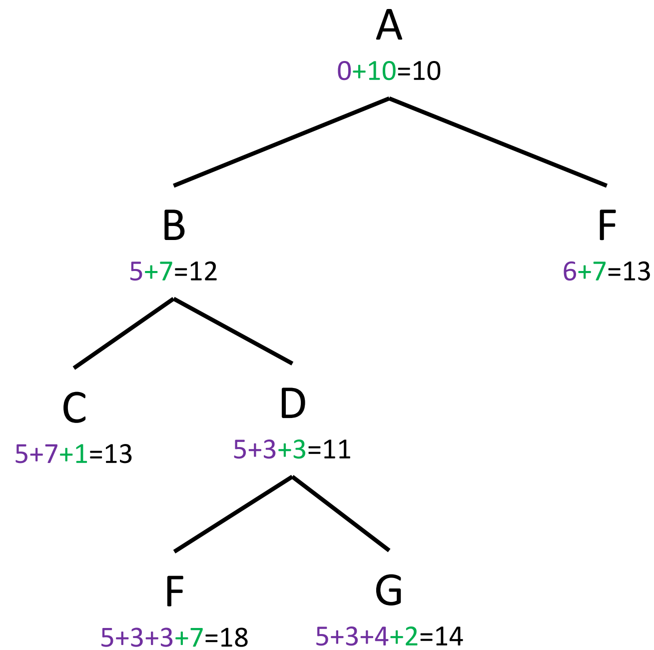



| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |

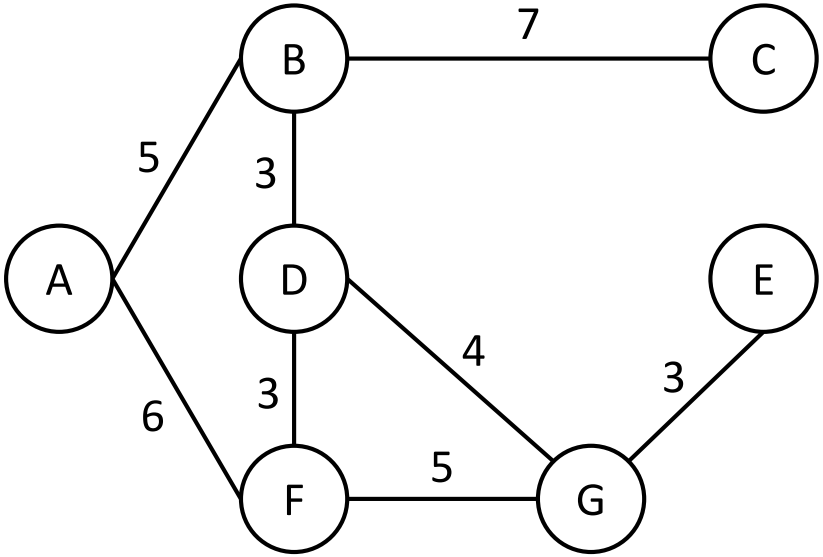
# A*

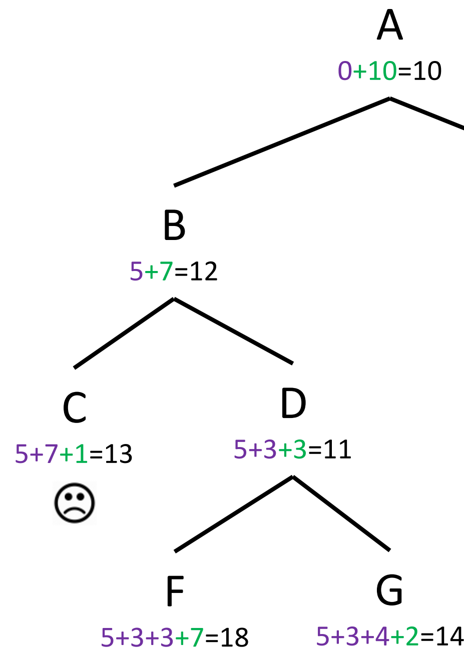



| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |

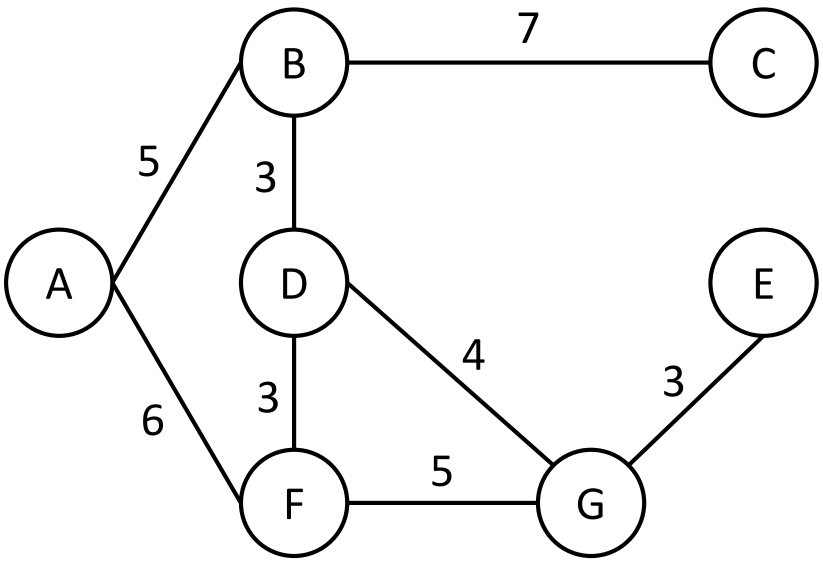
# A*

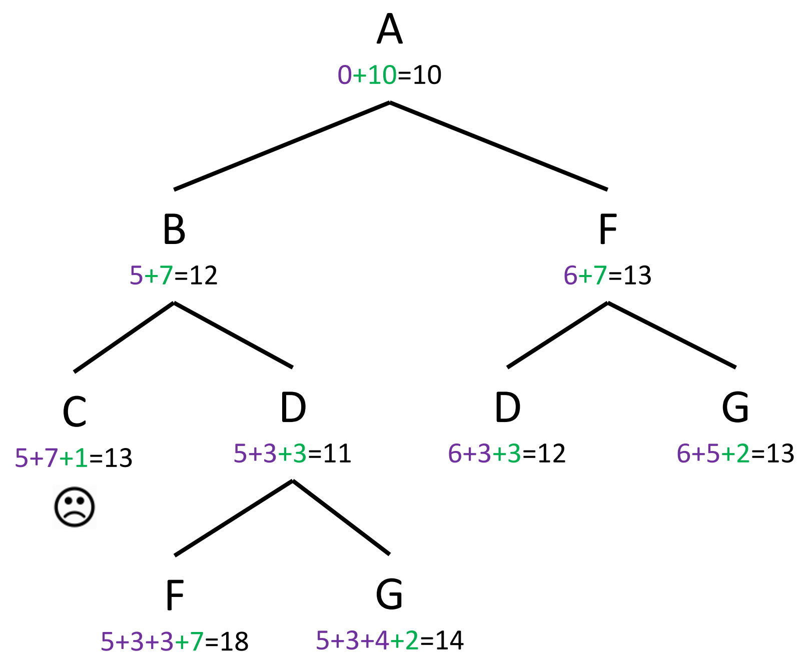



| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |

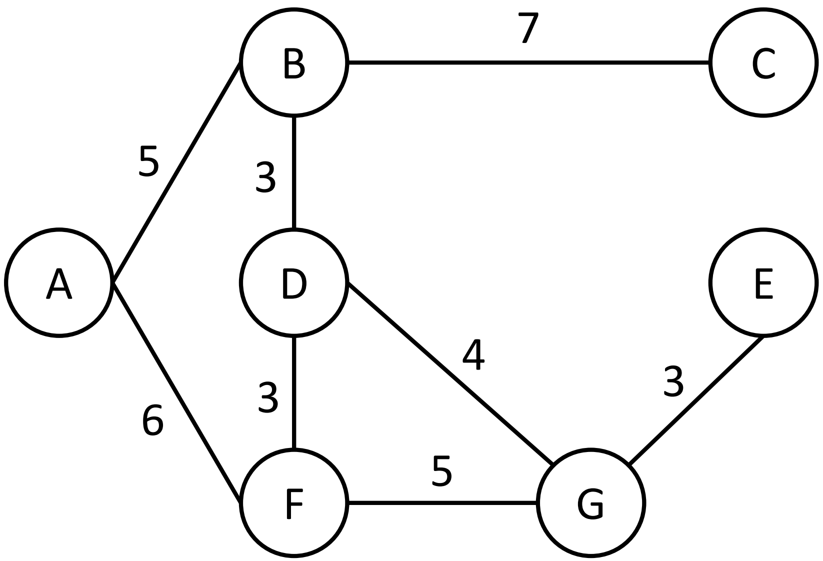
# A*



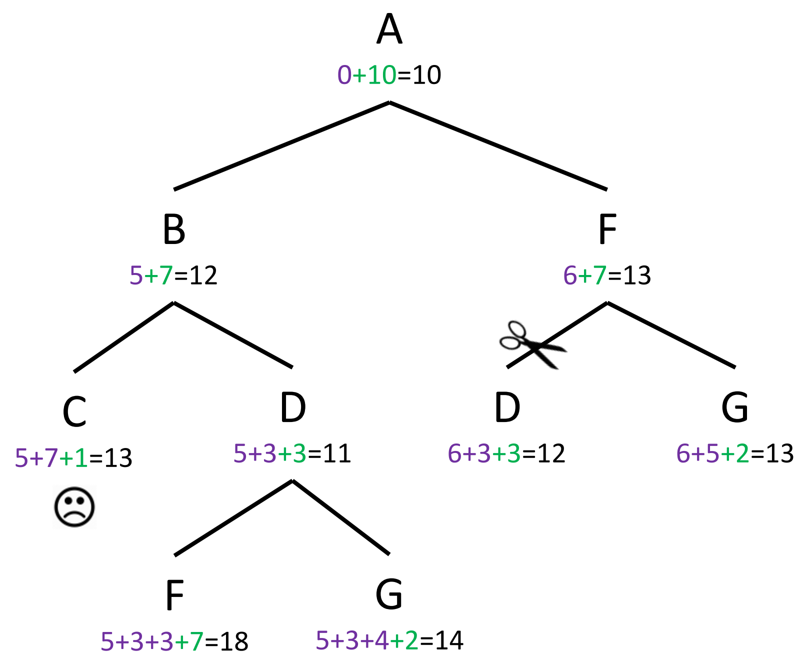




| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |

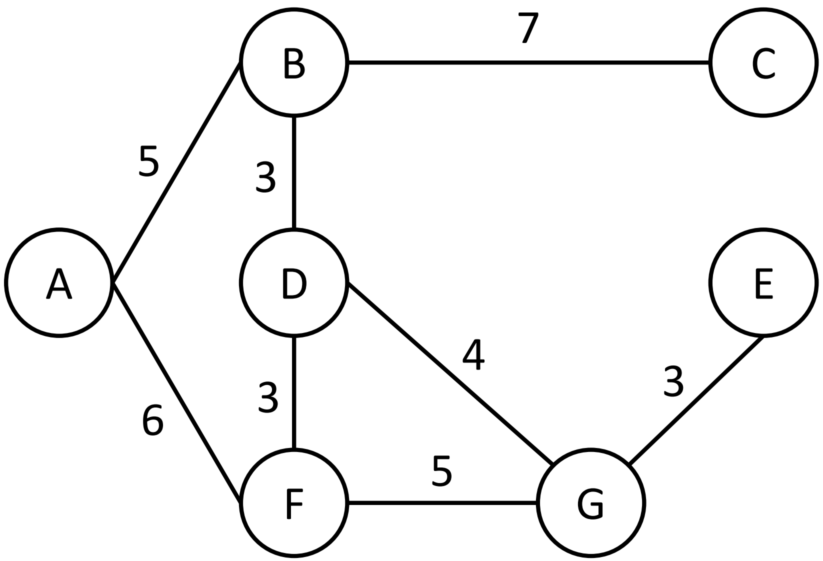
# A*

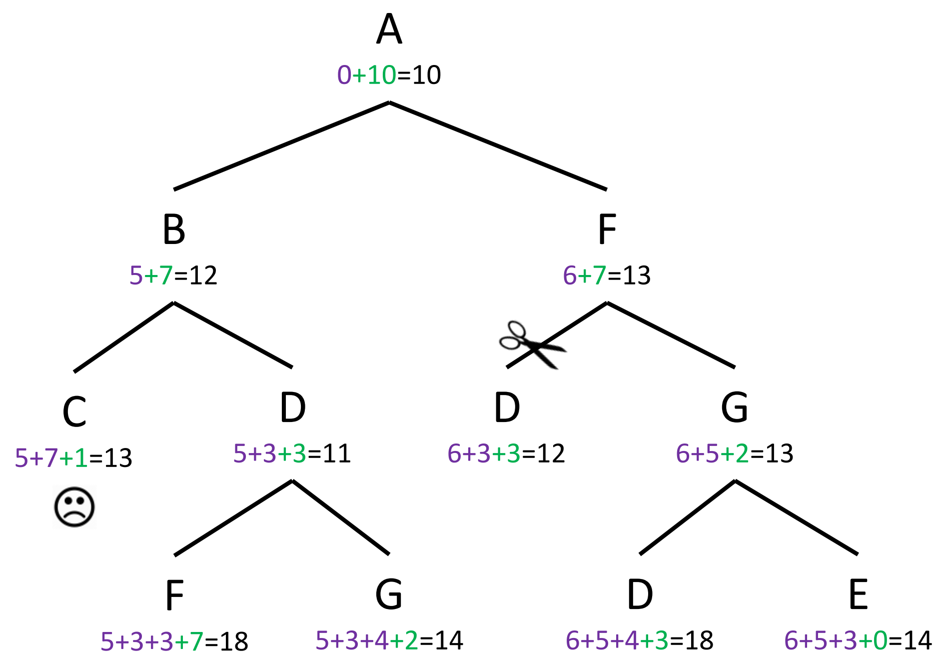



| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |



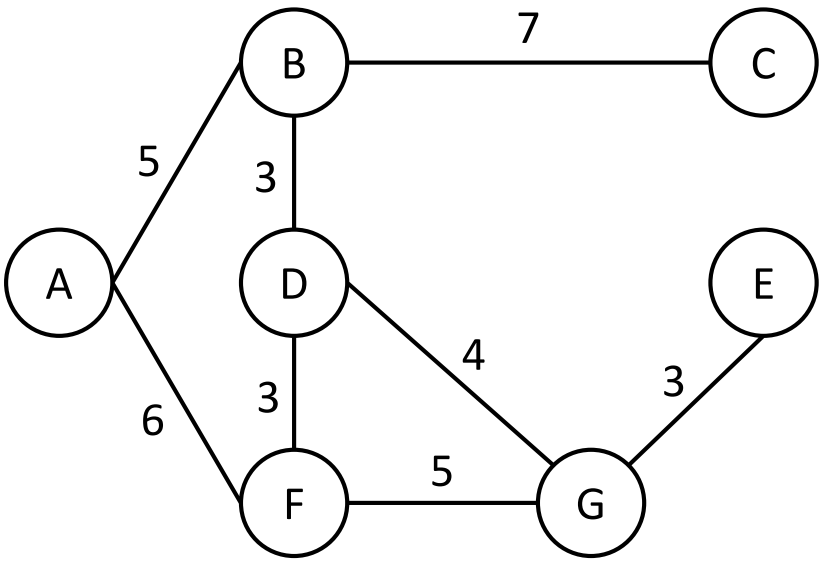
# A*



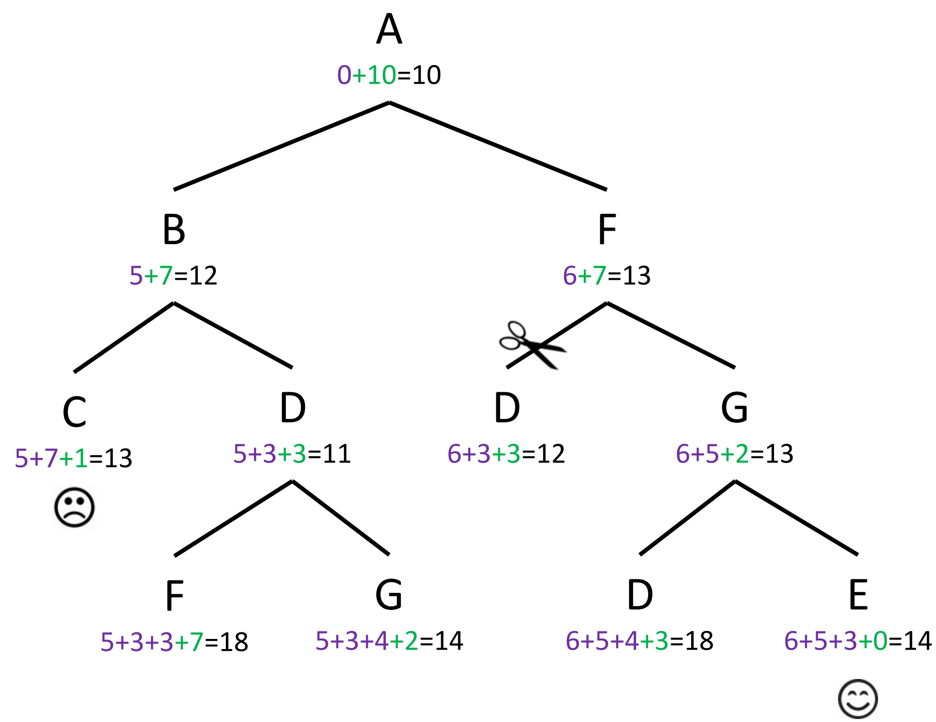




| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |

# A*



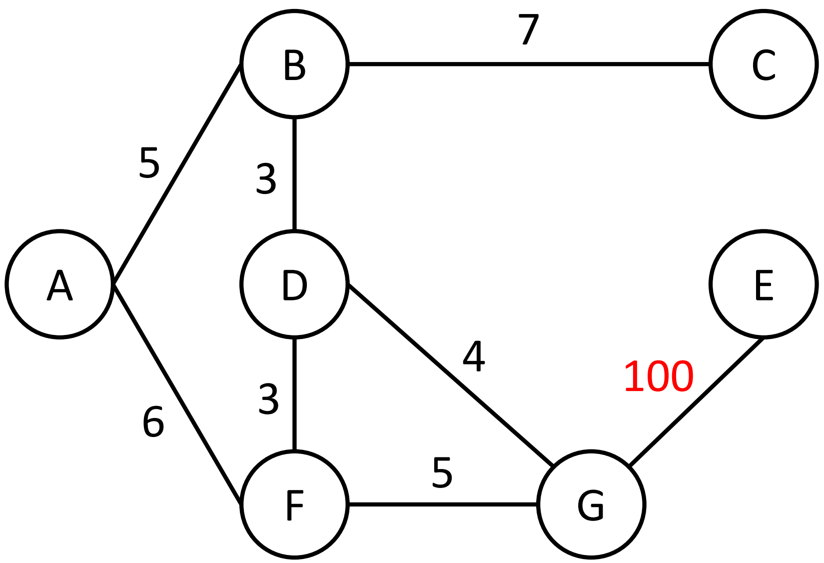

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 7 |
| C | 1 |
| D | 3 |
| E | 0 |
| F | 7 |
| G | 2 |



# A*

- Problem: if we work with an extended list, admissibility is not enough!

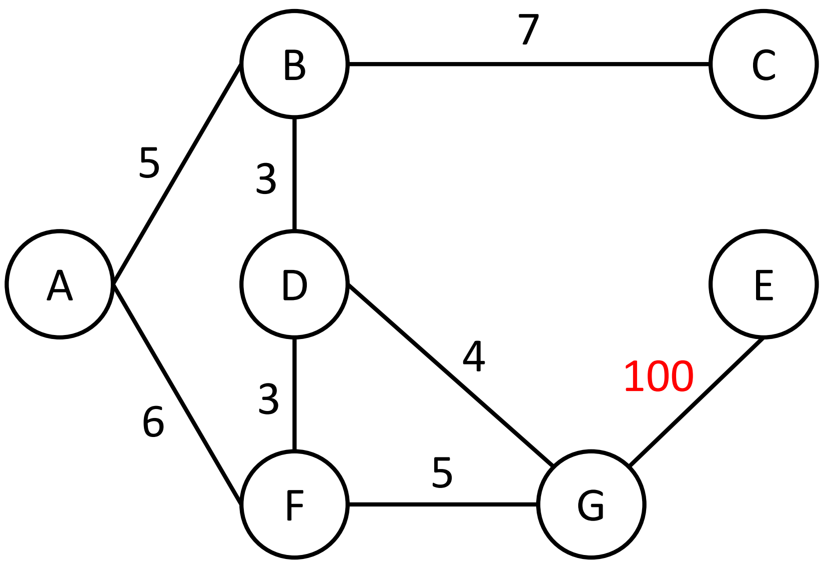
- Let's consider this "pathological" instance:




| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let's consider this "pathological" instance:



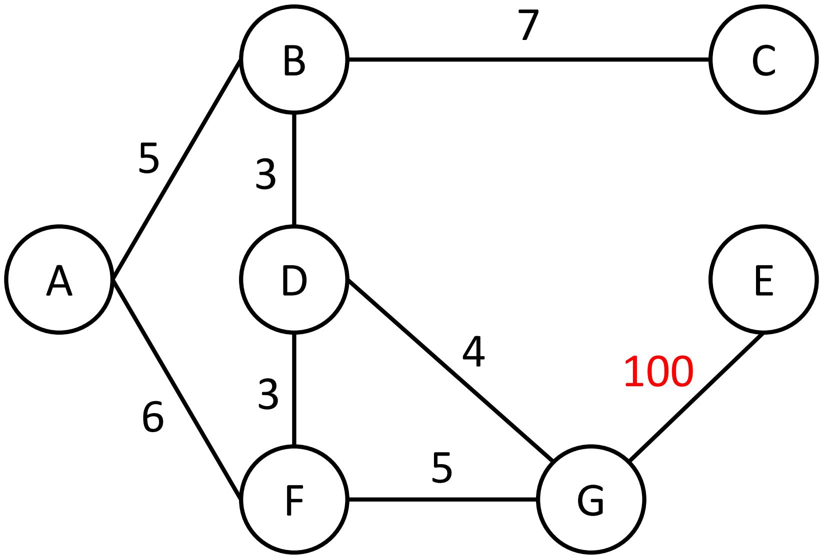

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

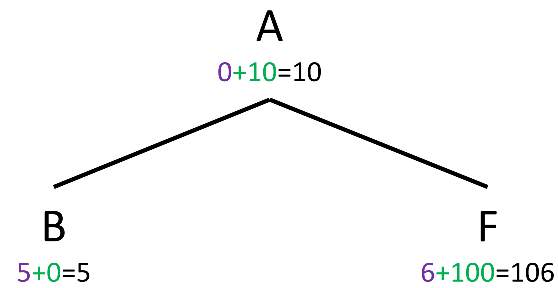
A

0+10=10

# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let's consider this "pathological" instance:

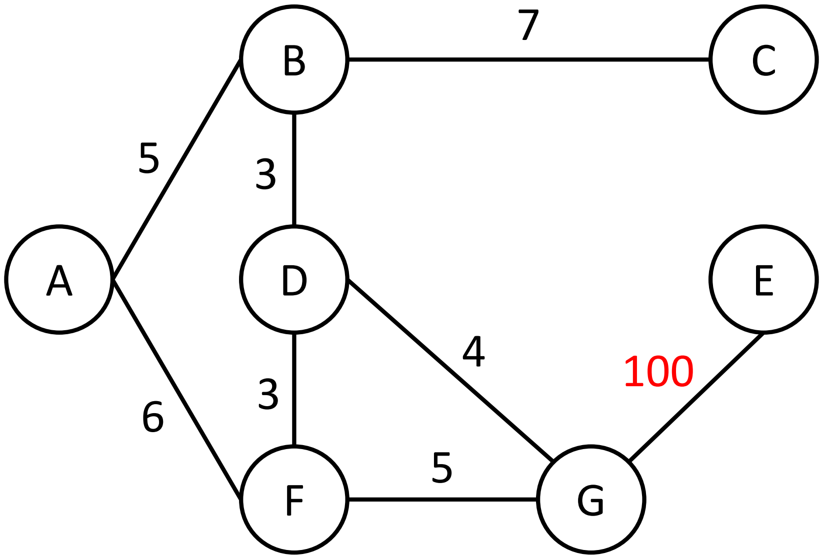


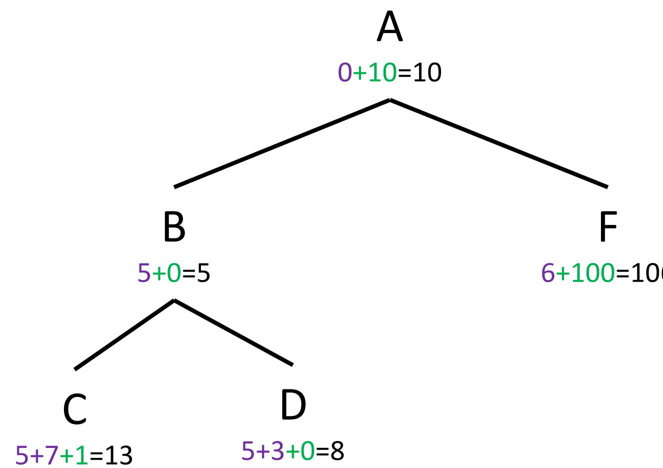




| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let's consider this "pathological" instance:



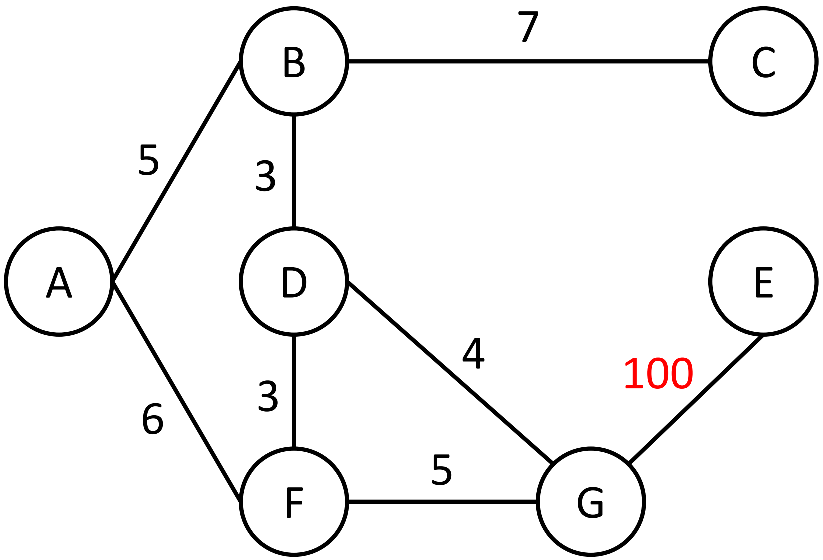



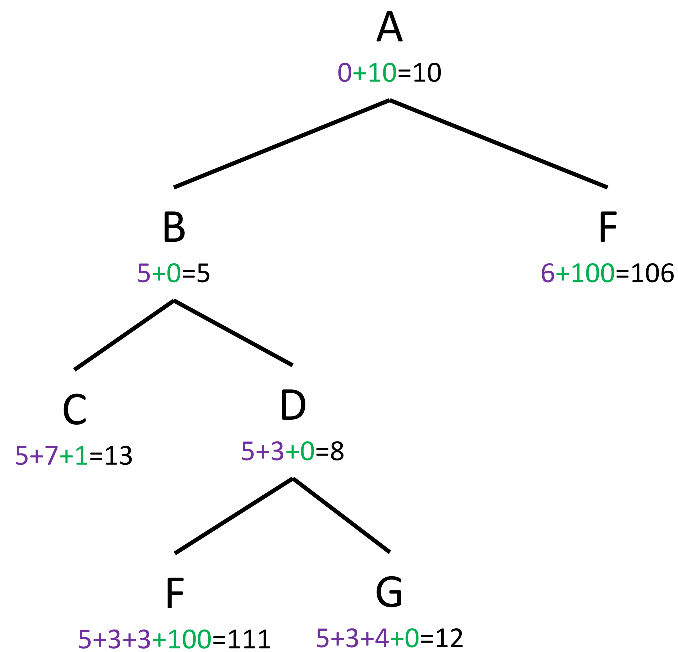

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

# A*

- Problem: if we work with an extended list, admissibility is not enough!

- Let's consider this "pathological" instance:

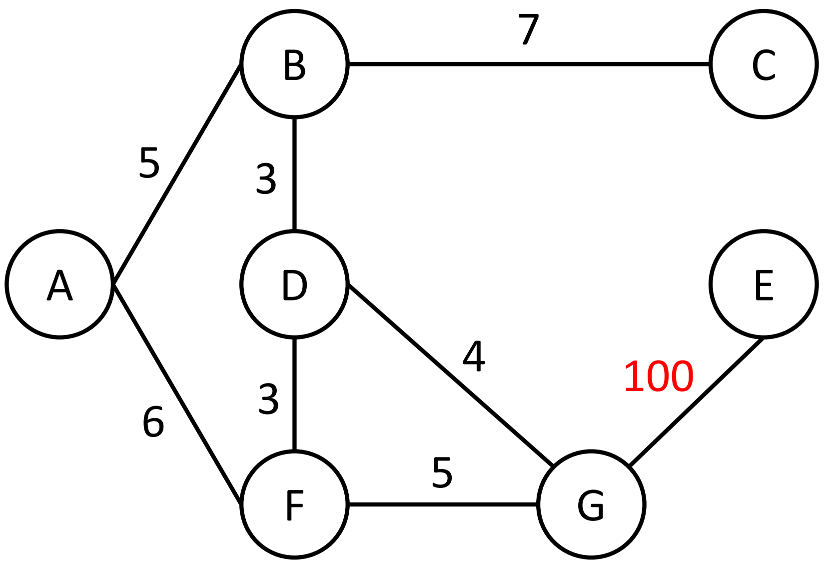


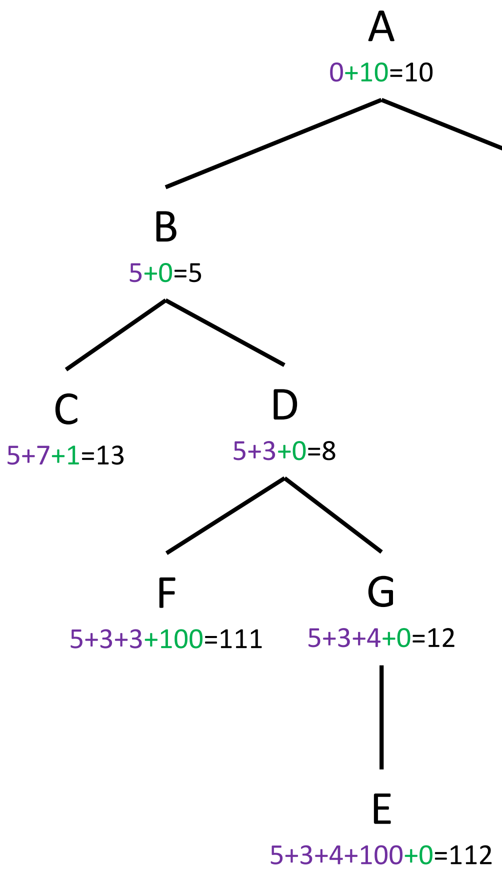

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let's consider this "pathological" instance:



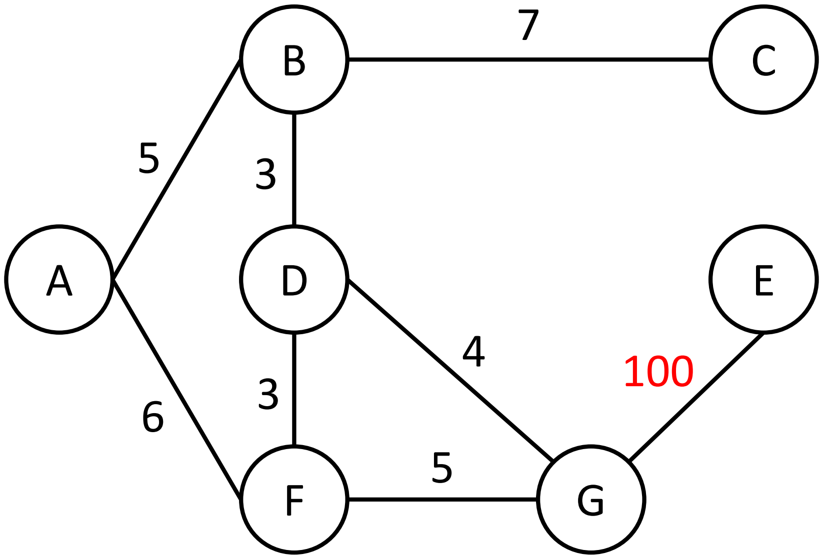

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |



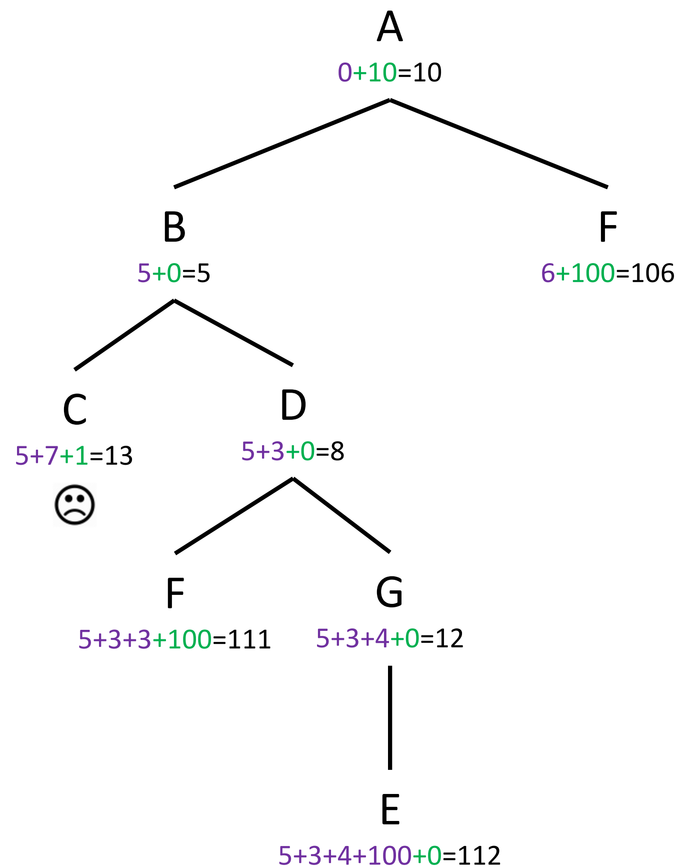
# A*

- Problem: if we work with an extended list, admissibility is not enough!

- Let's consider this “pathological" instance:

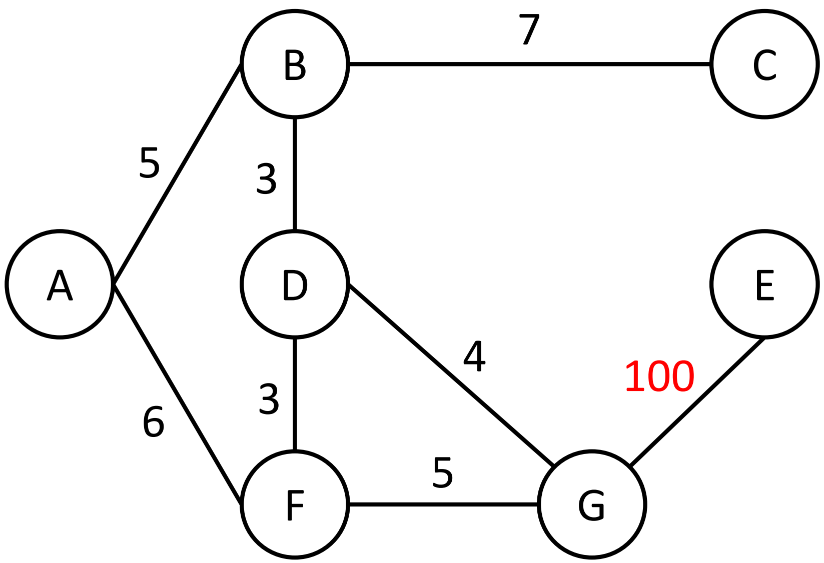



| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

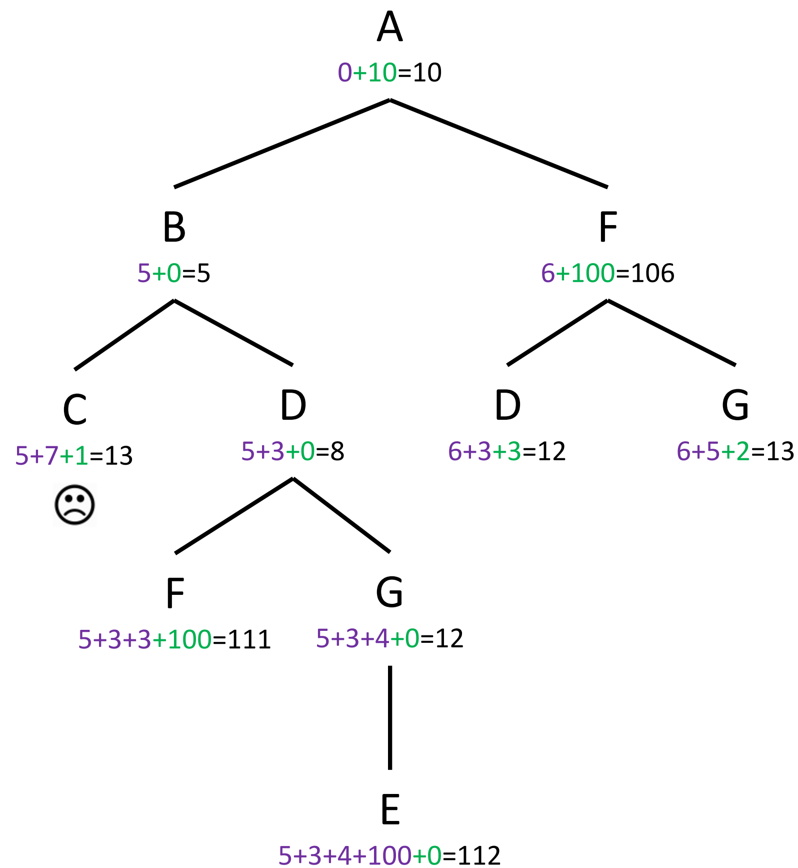


# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let's consider this "pathological" instance:

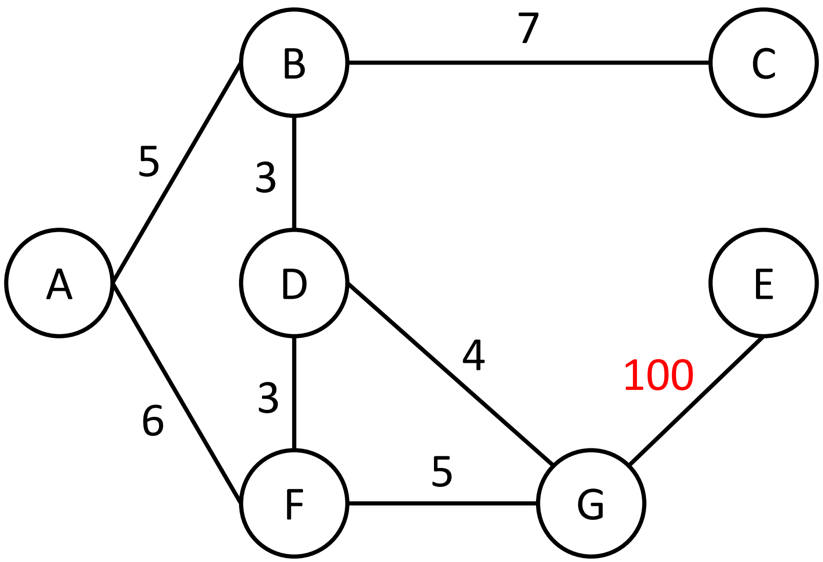



| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

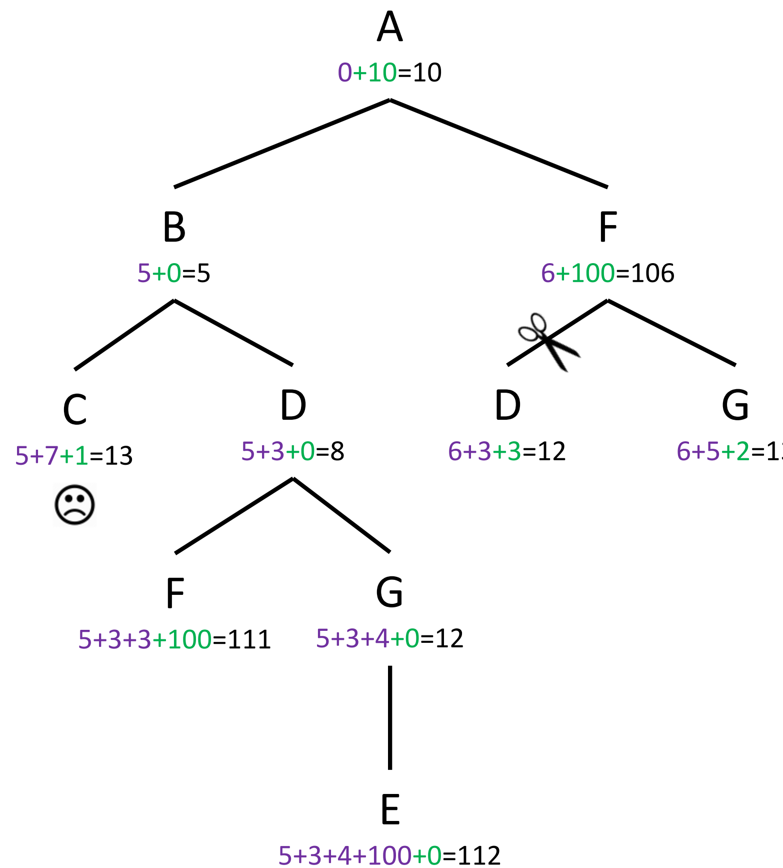


# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let's consider this "pathological" instance:

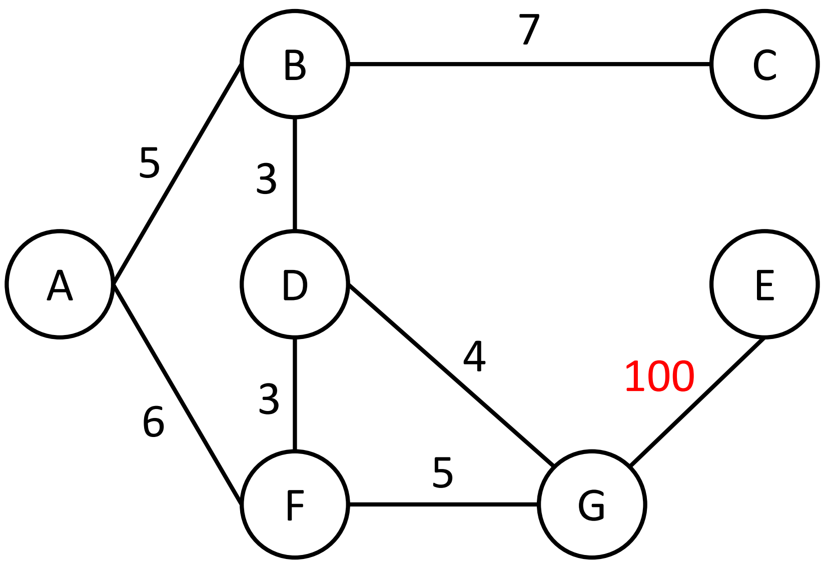


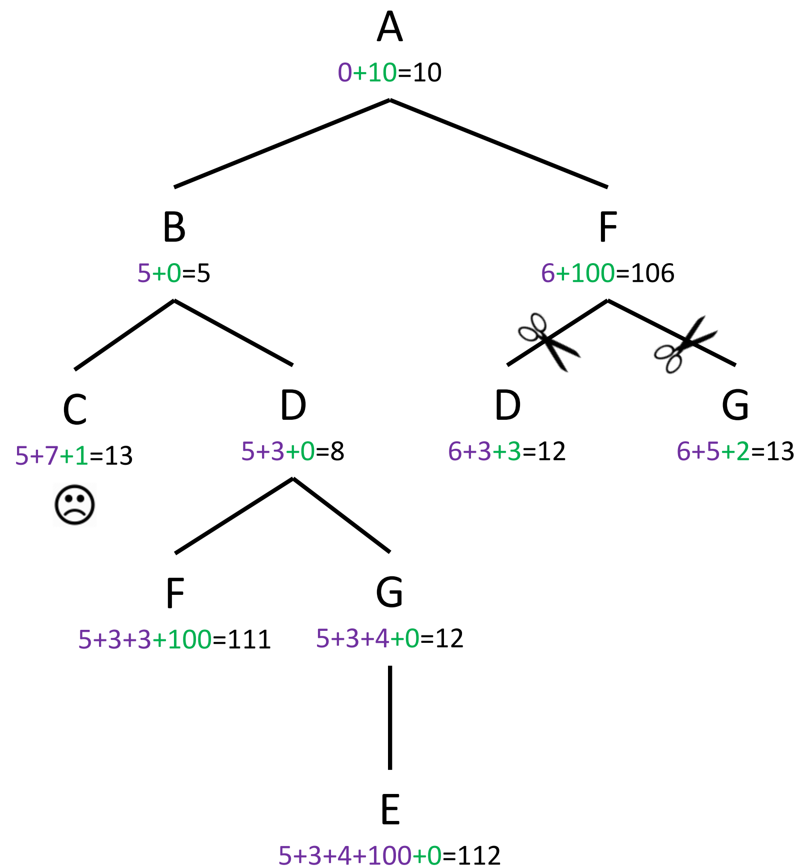

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |



# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let’s consider this “pathological" instance:

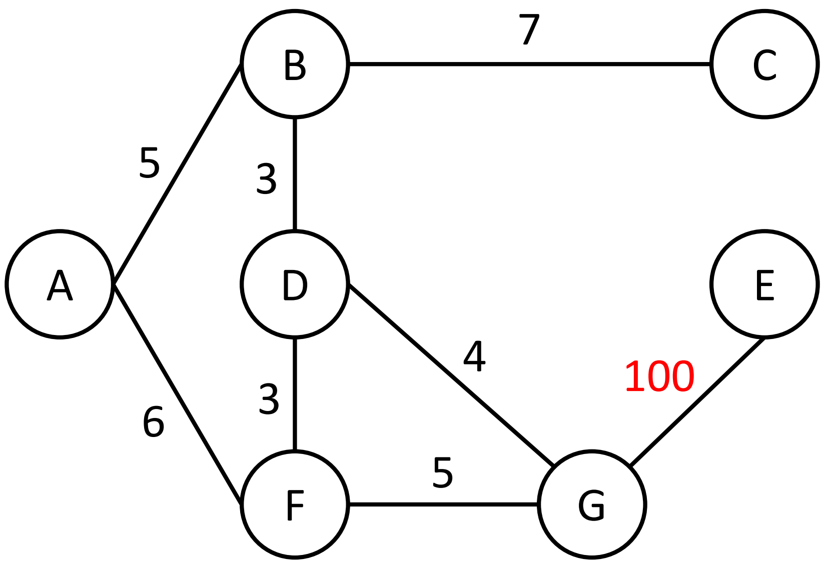


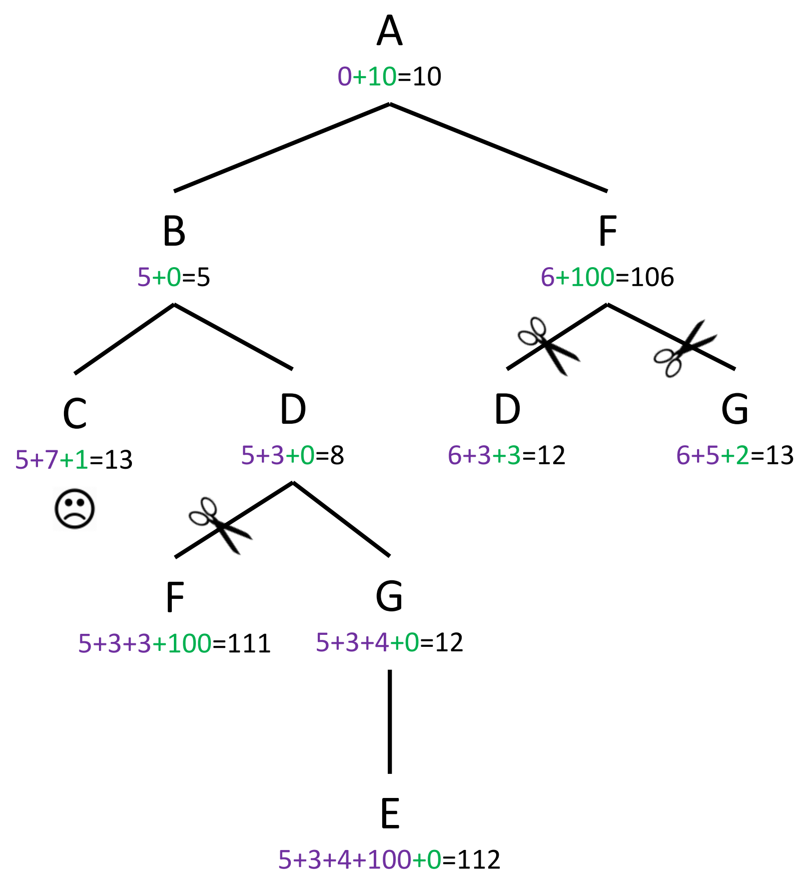

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let's consider this "pathological" instance:

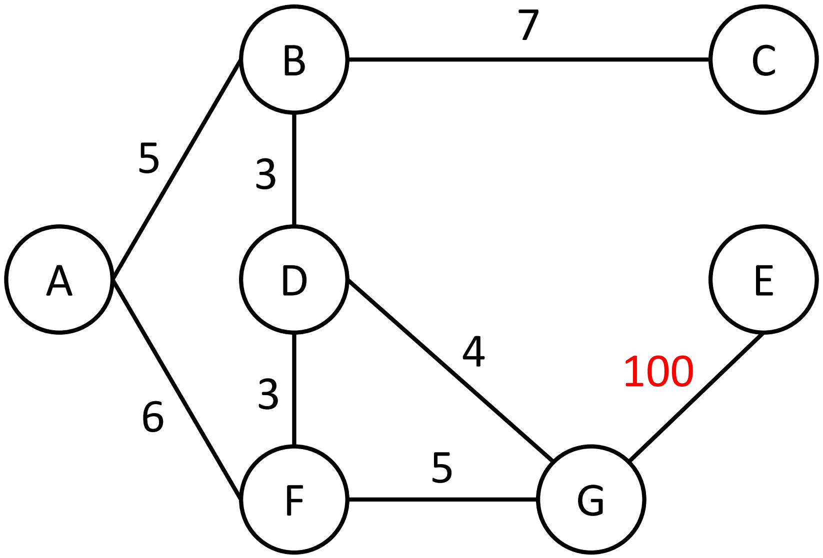


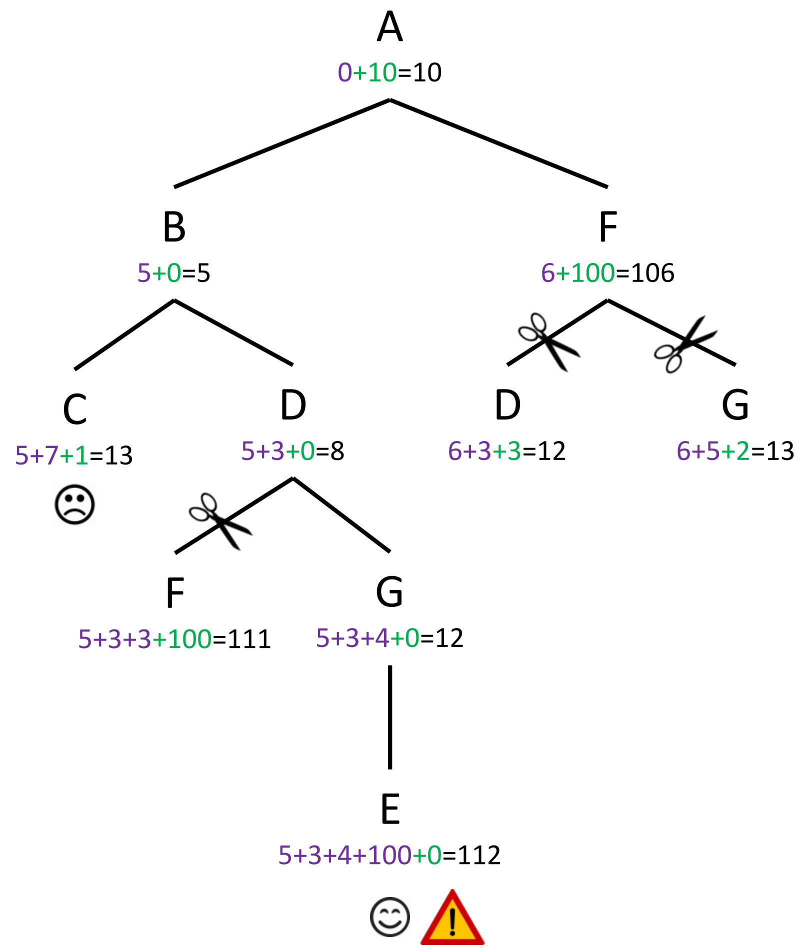

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |



# A*

- Problem: if we work with an extended list, admissibility is not enough!
- Let's consider this "pathological" instance:

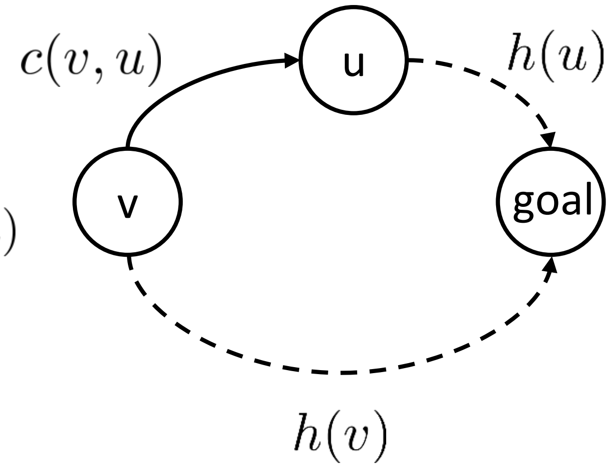



| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |

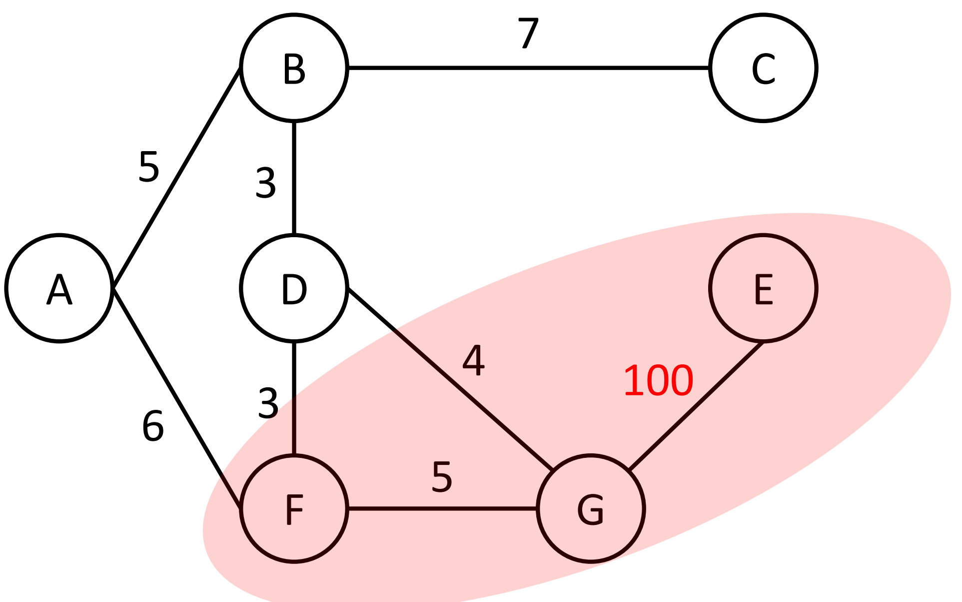
# A*

- We need to require a stronger property: **consistency**
- For any connected nodes u and v: $h(v) \leq c(v,u) + h(u)$




- It's a sort of triangle inequality, let's reconsider our pathological instance:

| node $v$ | $h(v)$ |
|---|---|
| A | 10 |
| B | 0 |
| C | 1 |
| D | 0 |
| E | 0 |
| F | 100 |
| G | 0 |



# Optimality of A*

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

$$f(u) = \overline{g(u)} + h(u) = \overline{g(v) + c(v,u)} + h(u) \geq g(v) + h(v)$$

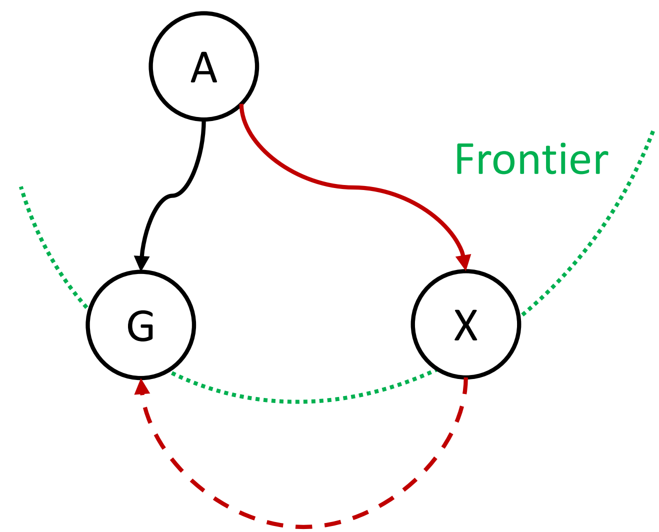
$$f(u) = g(u) + h(u) = g(v) + \underline{c(v,u) + h(u) \geq} g(v) + \underline{h(v)}$$

consistency

$f(u) \geq f(v)$ ⟶ f is non-decreasing along any search trajectory

Hypotheses:

1. A* selects from the frontier a node G that has been generated through a path p
2. p is not the optimal path to G




Given 2 and the frontier separation property, we know that there must exist a node X on the frontier that is on a better path to G

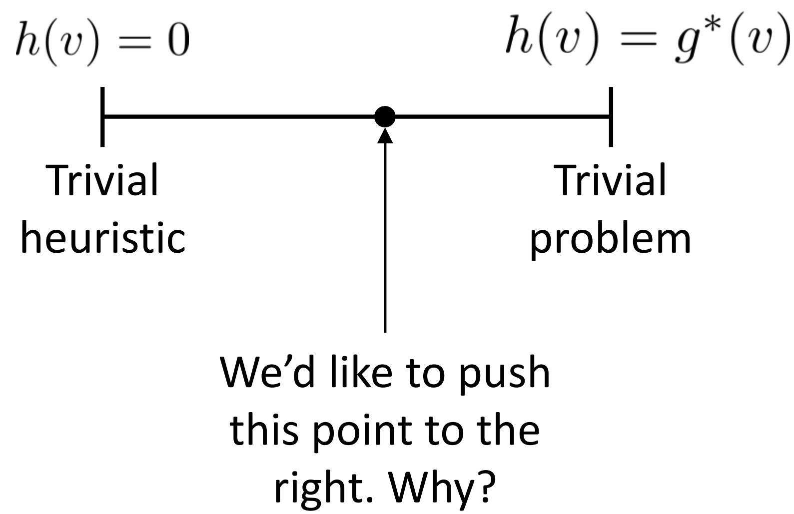
f is non-decreasing: $f(G) \geq f(X)$

A* selected G: $f(G) < f(X)$

*When A* selects a node for expansion, it discovers the optimal path to that node*

# Evaluating heuristics

- How to evaluate if an heuristic is good?




- A* will expand all nodes v such that: $f(v) < g^*(goal) \longrightarrow h(v) < g^*(goal) - g(v)$
- If, for any node v $h_1(v) \leq h_2(v)$

  then A* with $h_2$ will not expand more nodes than A* with $h_1$, in general $h_2$ is better (provided that is consistent and can be computed by an efficient algorithm)

- If we have two consistent heuristics $h_1$ and $h_2$ we can define

  $h_3(v) = \max\{h_2(v), h_1(v)\}$

# Building good heuristics

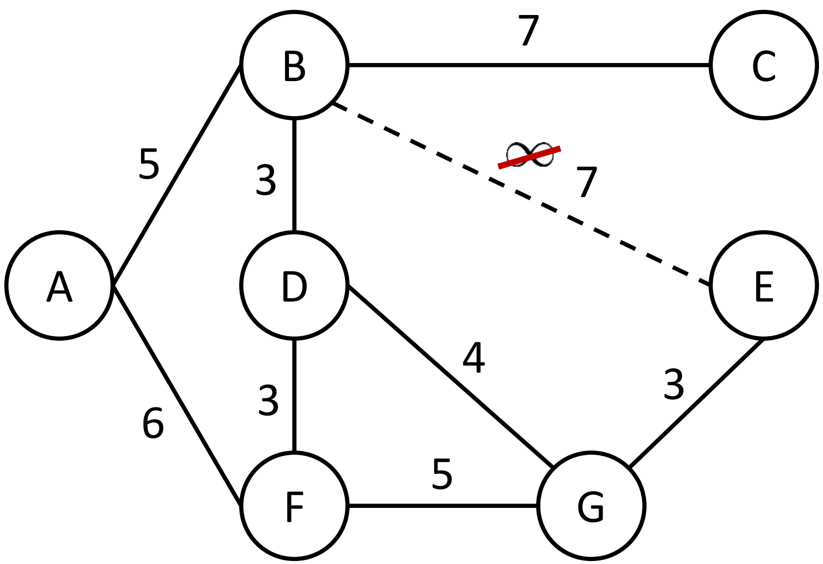
- The “larger heuristics are better” principle is not a methodology to define a good heuristic
- Such a task, seems to be rather complex: heuristics deeply leverage the inner structure of a problem and have to satisfy a number of constraints (admissibility, consistency, efficiency) whose guarantee is not straightforward
- When we adopted the straight-line distance in our route finding examples, we were sure it was a good heuristic
- Would it be possible to generalize what we did with the straight-line distance to define a method to *compute* heuristics for a problem?
- Good news: the answer is yes

# Relaxed problems

- Given a problem P, a relaxation of P is an easier version of P where some constraints have been dropped

$$P \xrightarrow{\text{Removing constraints}} \hat{P}$$

Original problem → Relaxed problem




$$\hat{g}(v,u) \leq g(v,u)$$

Costs in the relaxation

Costs in the original problem

- In our route finding problems removing the constraint that movements should be over roads (links) means that some costs pass from an infinite value to a finite one (the straight-line distance)

# Relaxed problems

- Idea:

Define a relaxation of P: $\hat{P}$ → Apply A* to every node and get $\hat{h}^*(v)$ → Set $h(v) = \hat{h}^*(v)$ in the original problem and run A*

- We can easily define a problem relaxation, it's just matter of removing constraints/rewriting costs

- But what happens to soundness and completeness of A*?

$\hat{h}^*(v) \leq \hat{g}(v,u) + \hat{h}^*(u)$ Path costs are optimal

$h(v) \leq \hat{g}(v,u) + h(u)$ From our idea

$\hat{g}(v,u) \leq g(v,u)$ From the definition of relaxation

$h(v) \leq g(v,u) + h(u)$ **h is consistent**

*Sistemi Intelligenti*
*Corso di Laurea in Informatica, A.A. 2019-2020*
*Università degli Studi di Milano*

**Nicola Basilico**
Dipartimento di Informatica
Via Celoria 18- 20133 Milano (MI)
Ufficio 4008
nicola.basilico@unimi.it
+39 02.503.16289