Faculty of Mathematics and Physics
Charles University in Prague
31st September 2016

Time to apply stuff…

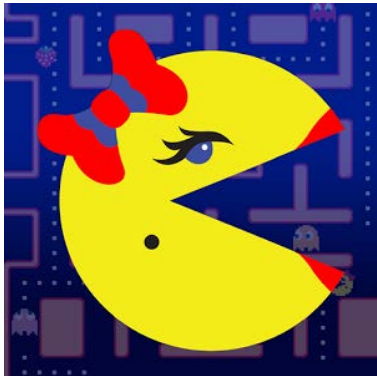
Artificial Intelligence 1

Lab 03 – Path-finding
a.k.a. Informed Graph Search

Today – Path-Finding (again)

INFO

ACTIONS



https://github.com/kefik/MsPacMan-vs-Ghosts-AI

A* Algorithm

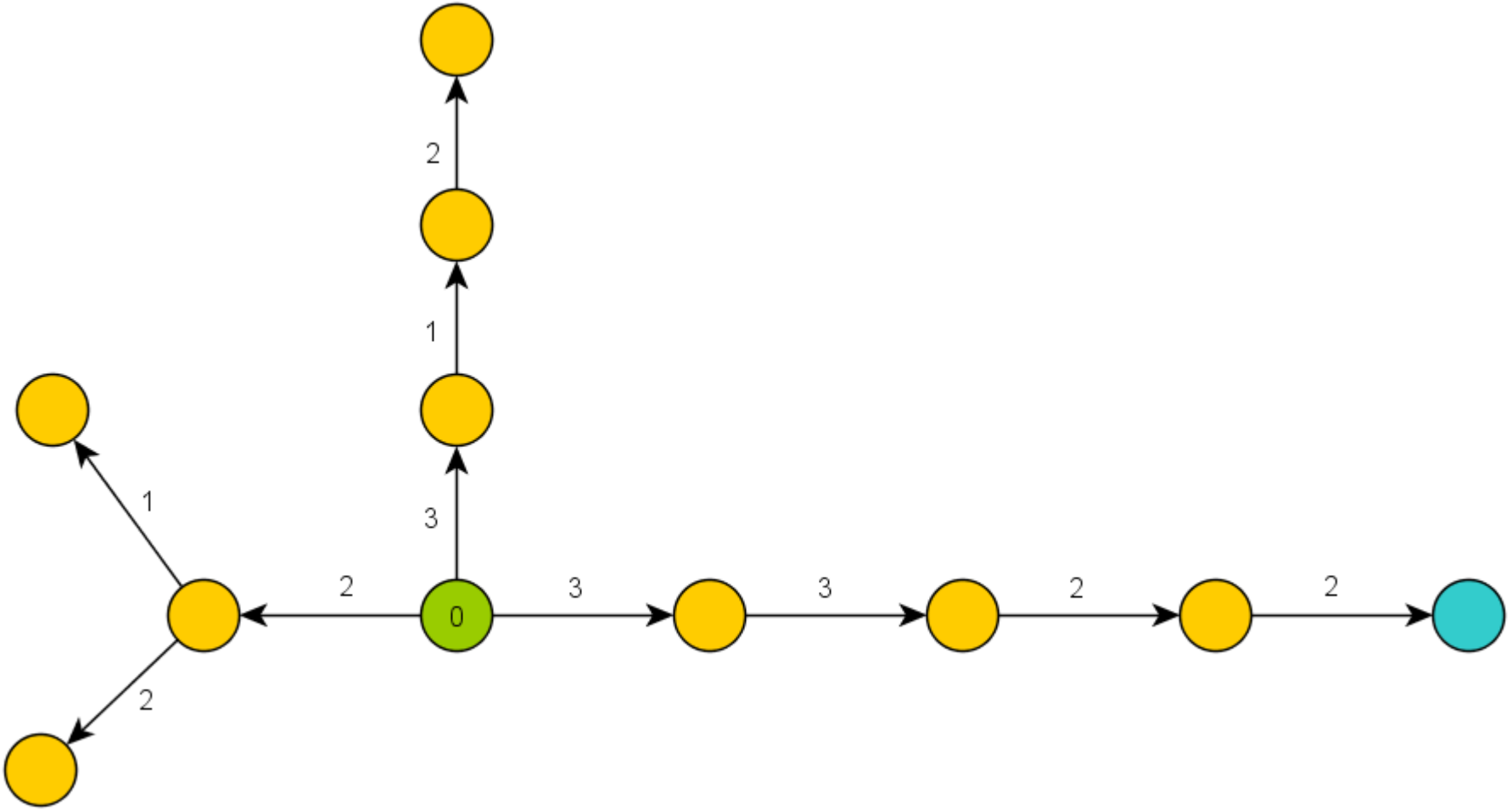
Dijkstra

- Remembering Dijkstra's alg?
- Roughly speaking…

```
Nodes = {start}
while (!nodes.empty) {
  Node = pick_shortest_path(nodes)
  if (Node == Target)
    return reconstruct_path(Node)
  Nodes = Nodes \ Node
  expand(Node, Nodes)
}
```

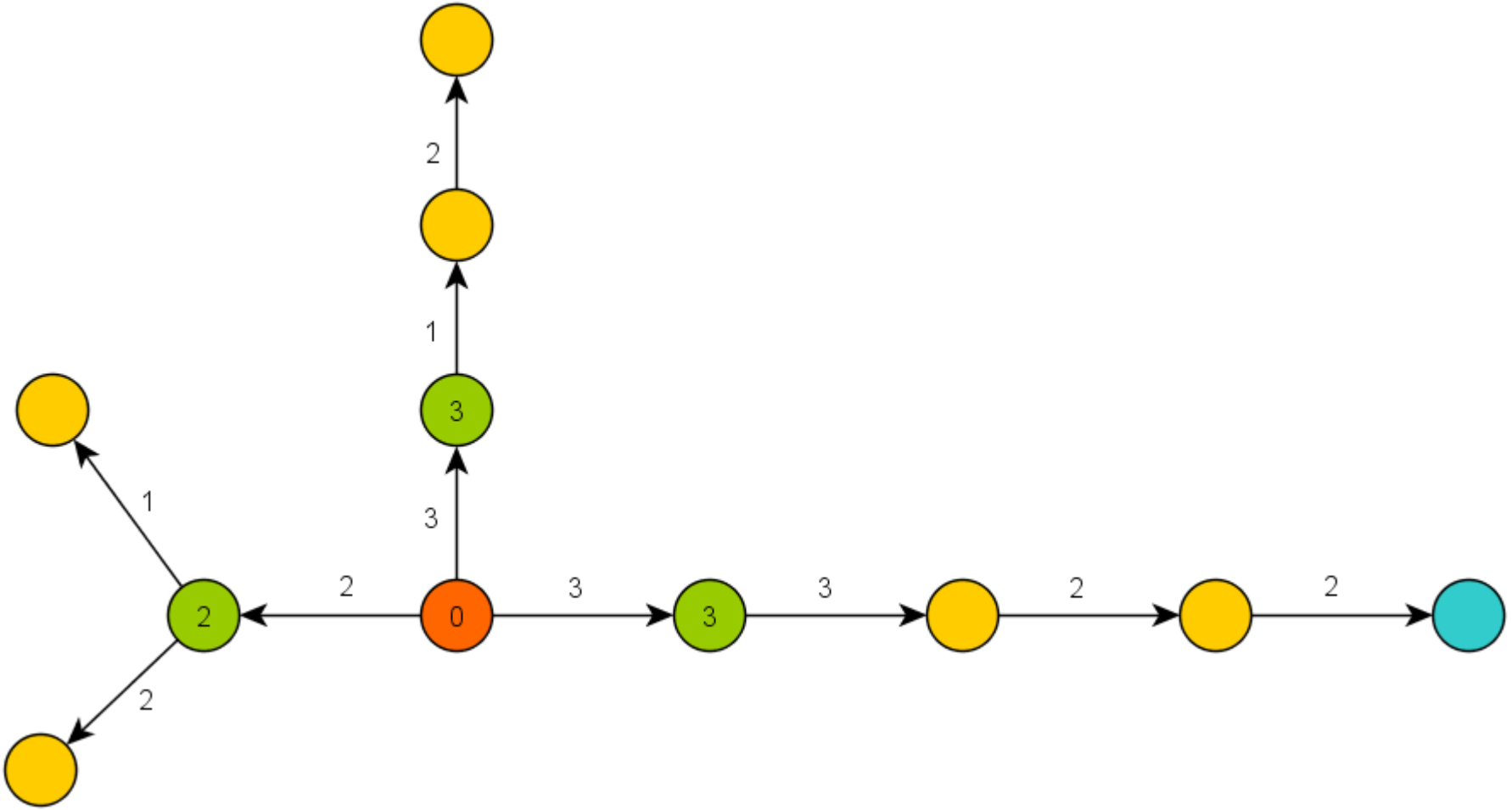
A* Algorithm

Dijkstra Example I



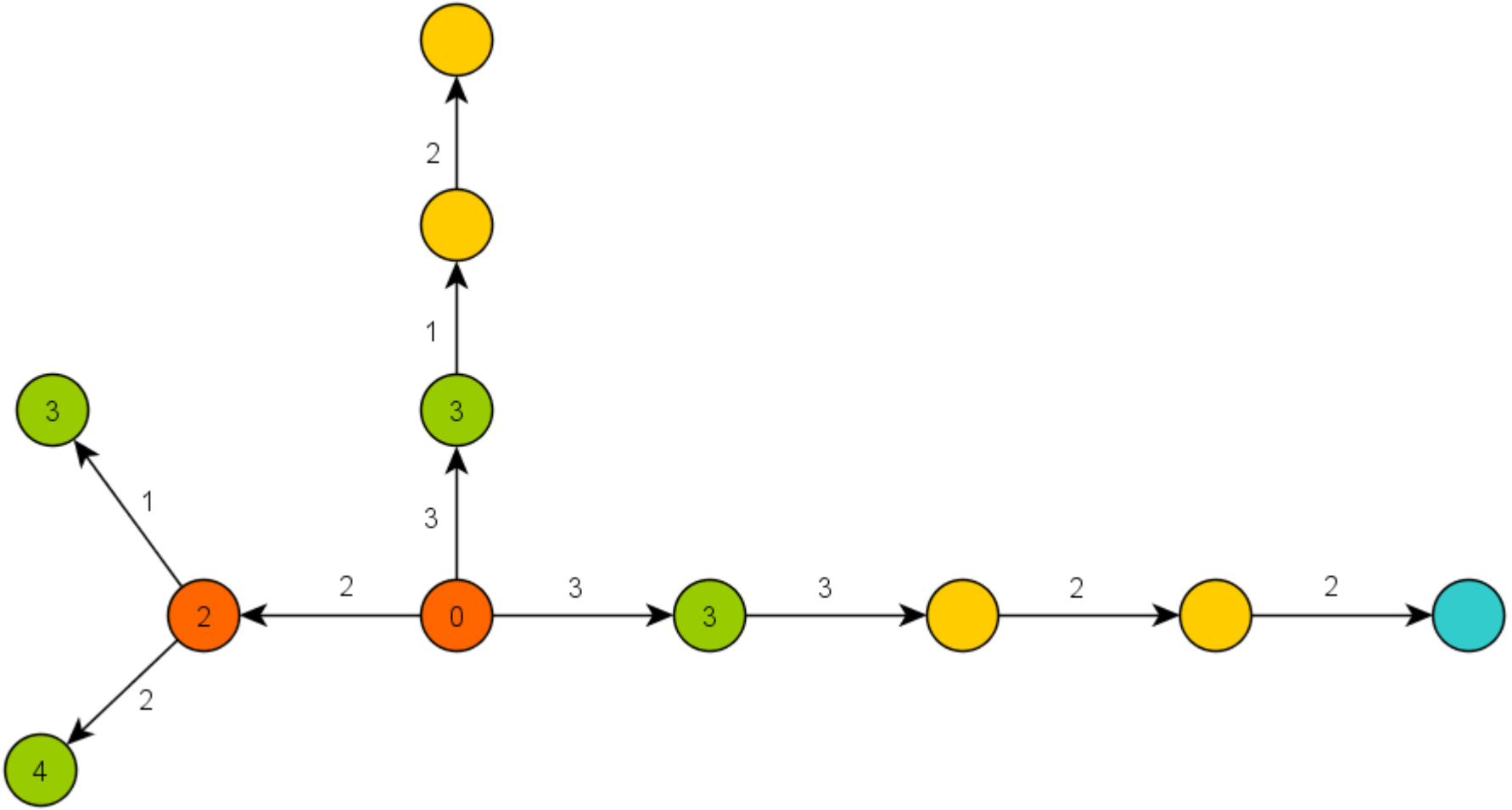
A* Algorithm

Dijkstra Example II



A* Algorithm

Dijkstra Example III



A* Algorithm
Basics

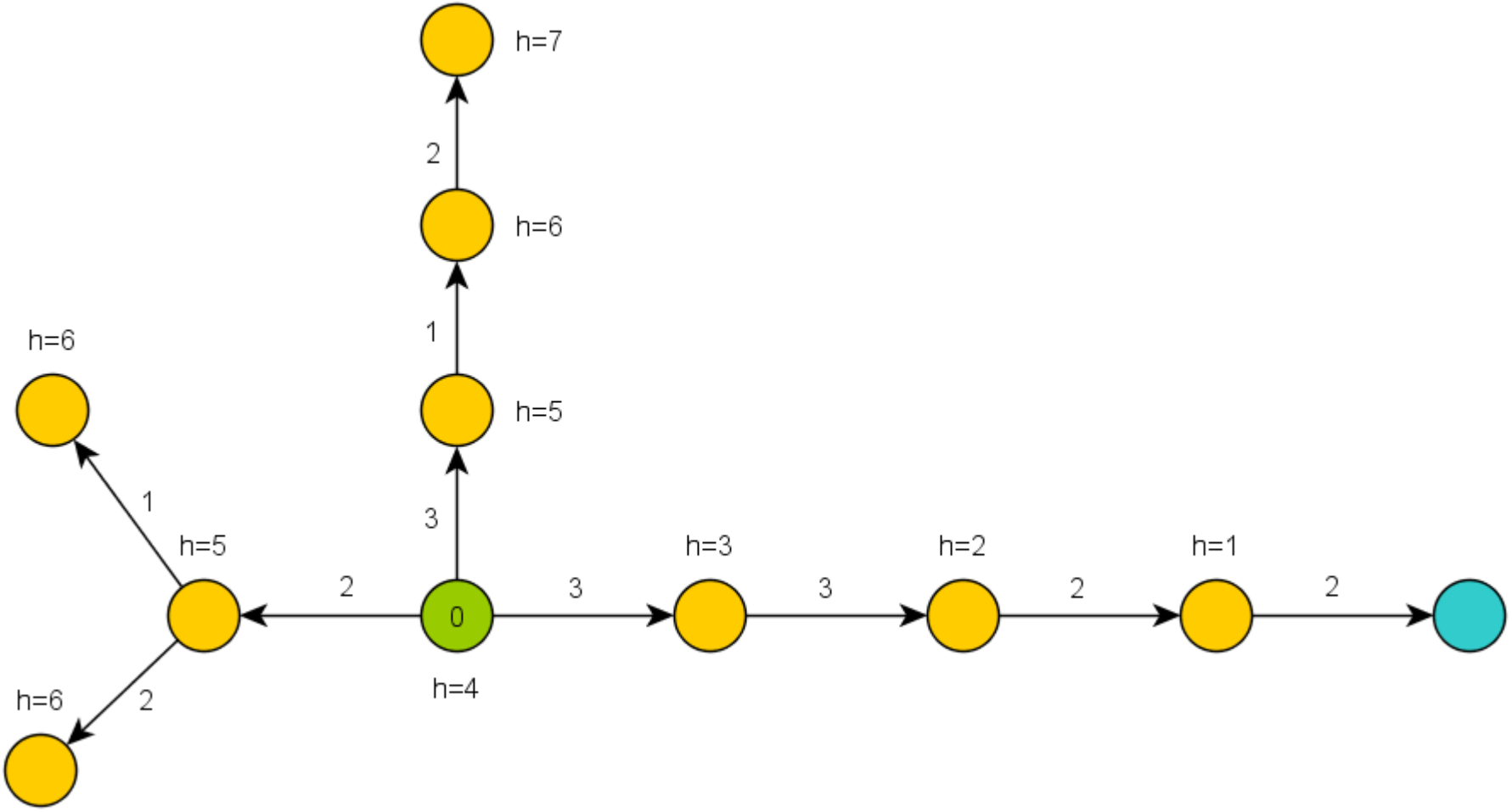
- A* trick

- Roughly speaking…

```
Nodes = {start}
while (!nodes.empty) {
  Node = pick_the_most_promising(nodes)
  if (Node == Target) return
       reconstruct_path(Node)
  Nodes = Nodes \ Node
  expand(Node, Nodes)
}
```

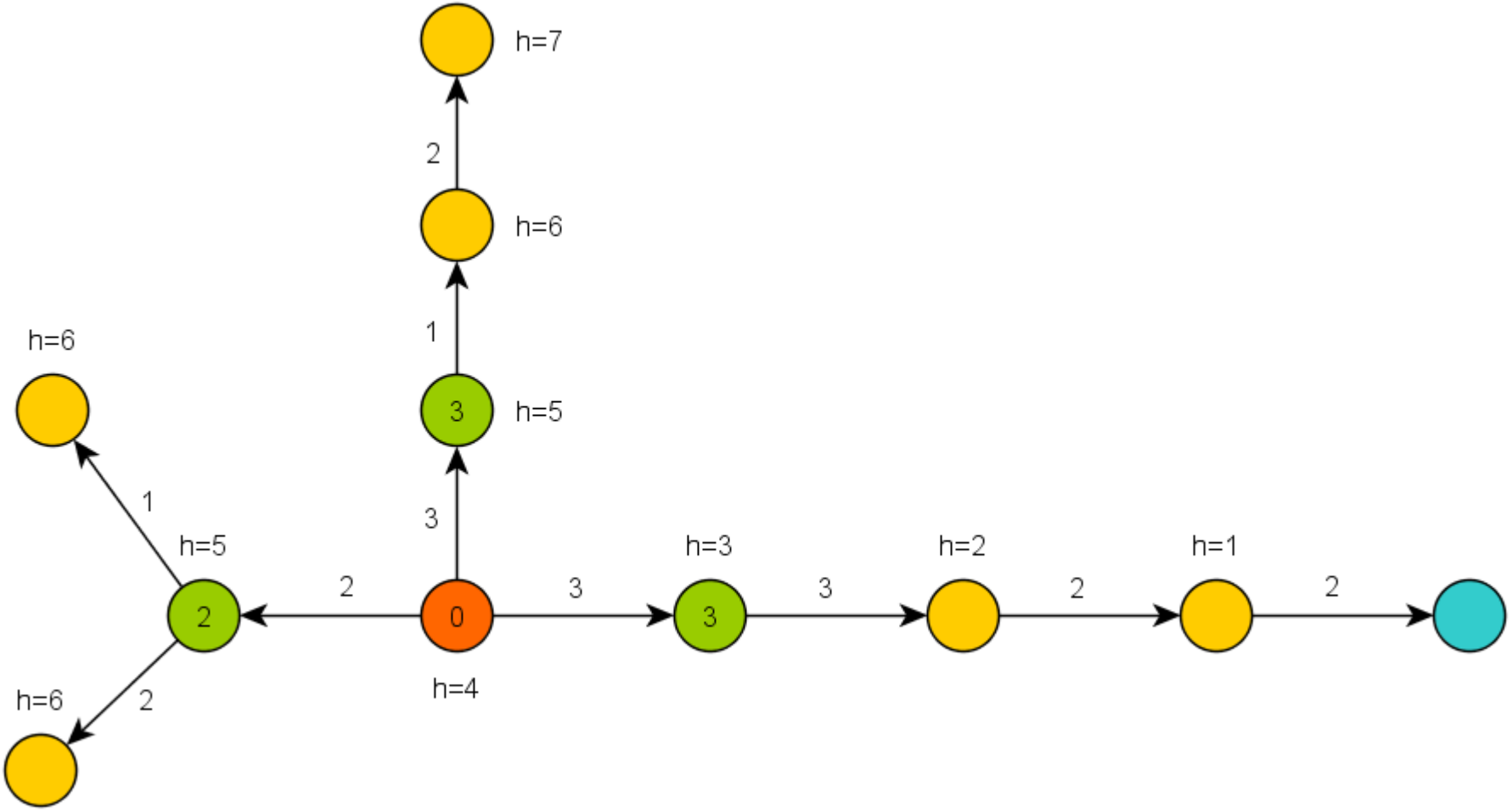
A* Algorithm

A* Example I



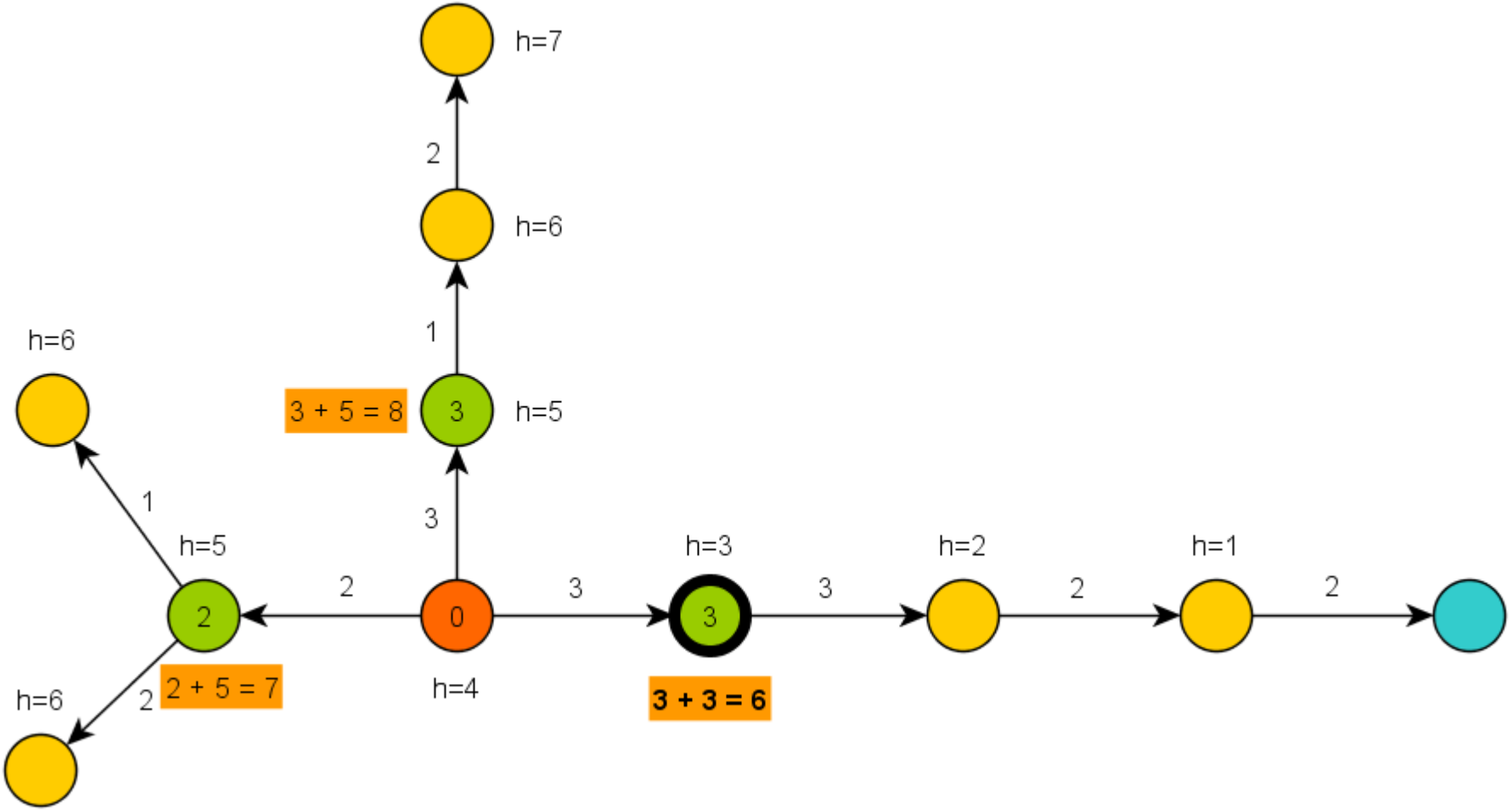
A* Algorithm

A* Example II



A* Algorithm

A* Example III



A* Algorithm
Basics

- A* heuristic function must be… ?
  1. Admissible for correctness
     - Do not over-estimate the path-cost
  2. Consistent == Monotone (for efficiency)
     - “triangle inequation”

- Blah! Let’s hack it!
  - What if we impose additional COST to some nodes or links?

A* Algorithm

Juggling with node/link costs

- Let's choose some "nodes" or "links" that we want to avoid
  - `B` … BADDIES … nodes or links with extra cost
  - `EC(B)` … EXTRA COST … sum of extra cost over the B set
- We then have two types of metrics for the path
  - `Len(p)` … PATH LENGTH … real environment path length
  - `Cost(p)` … PATH COST … `Len(p) + EC(p)`
- Thus we can run A* using those two metrics
  - `A*-Len(N,M)` … outputs the shortest path between nodes N and M
  - `A*-Cost(N,M)` … outputs the least costly path between nodes N and M
- What do `A*-Len(N,M)` and `A*-Cost(N,M)` look like?

A* Algorithm

Juggling with node/link costs

- Let's choose some "nodes" or "links" that we want to avoid
  - `B` … BADDIES … nodes or links with extra cost
  - `EC(B)` … EXTRA COST … sum of extra cost over the B set
- We then have two types of metrics for the path
  - `Len(p)` … PATH LENGTH … real environment path length
  - `Cost(p)` … PATH COST … `Len(p) + EC(p)`
- Thus we can run A* using those two metrics
  - `A*-Len(N,M)` … outputs the shortest path between nodes N and M
  - `A*-Cost(N,M)` … outputs the least costly path between nodes N and M
- What do `A*-Len(N,M)` and `A*-Cost(N,M)` look like?

1. `A*-Len(N,M) != A*-Cost(N,M)`
   - `A*-Len(N,M)` path contains some `B'` that are not on the path of `A*-Cost(N,M)`

⇒ We have found a detour that is shorter than `EC(B')`!

- `Cost(A*-Cost(N,M)) < Len(A*-Len(N,M)) + EC(A*-Len(N,M))`

A* Algorithm

Juggling with node/link costs

- Let's choose some "nodes" or "links" that we want to avoid
  - `B` … BADDIES … nodes or links with extra cost
  - `EC(B)` … EXTRA COST … sum of extra cost over the B set
- We then have two types of metrics for the path
  - `Len(p)` … PATH LENGTH … real environment path length
  - `Cost(p)` … PATH COST … `Len(p) + EC(p)`
- Thus we can run A* using those two metrics
  - `A*-Len(N,M)` … outputs the shortest path between nodes N and M
  - `A*-Cost(N,M)` … outputs the least costly path between nodes N and M
- What do `A*-Len(N,M)` and `A*-Cost(N,M)` look like?

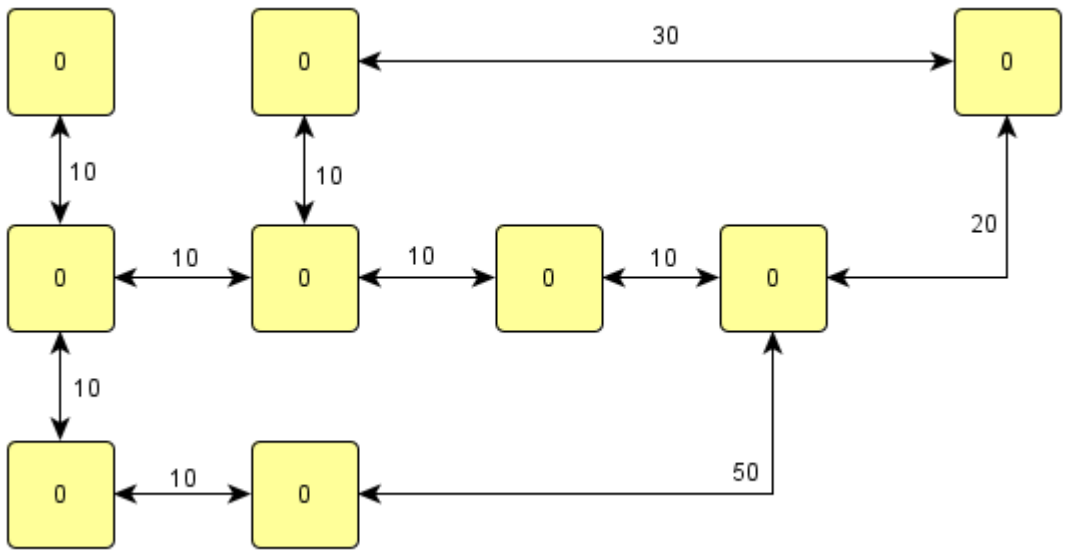
2. `A*-Len(N,M) == A*-Cost(N,M)`
   - Both paths contains `B'` subset of B

⇒ There is no other `PATH(N,M)`, for which following would hold:
- `Cost(PATH(N,M)) < Len(A*-Len(N,M)) + EC(A*-Len(N,M))`
- `Len(PATH(N,M)) + EC(PATH(N,M)) < Len(A*-Len(N,M)) + EC(B')`

⇒ All other paths that would go around `B'` are longer than `EC(B')`!

A* Algorithm

Juggling with node/link costs

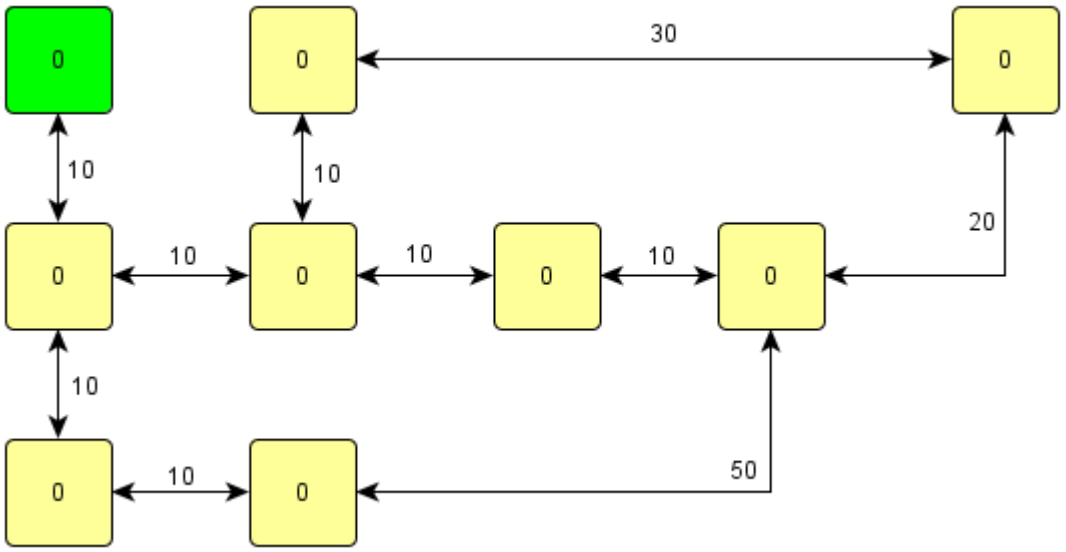


- Example map

A* Algorithm

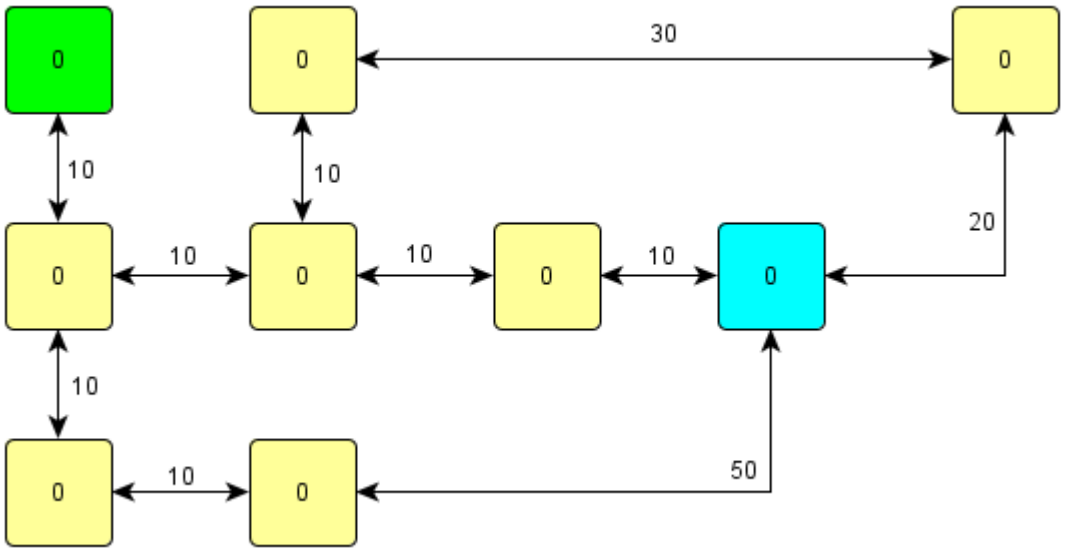
Juggling with node/link costs

- Start-node



A* Algorithm

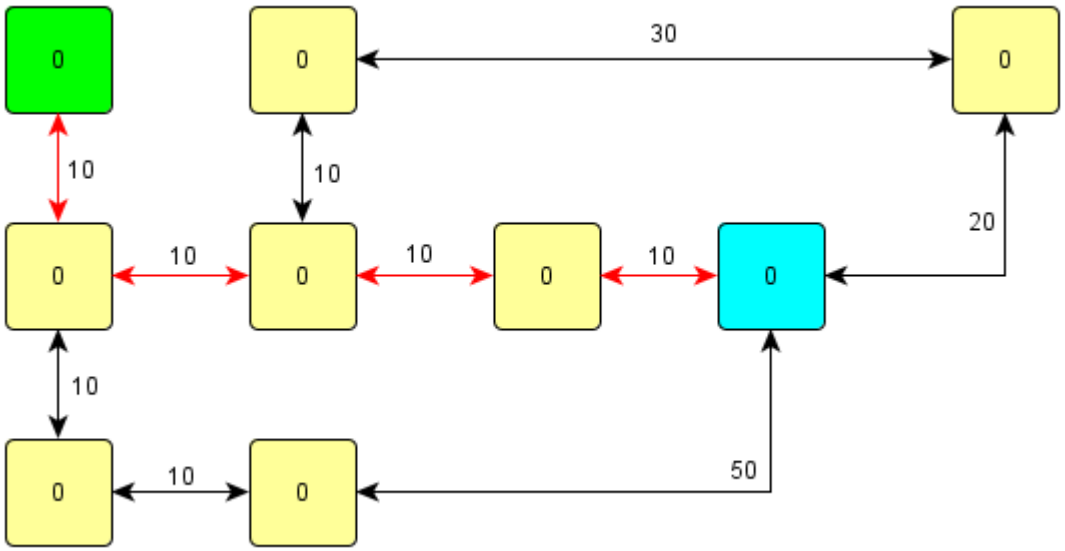
Juggling with node/link costs



- Target-node

A* Algorithm

Juggling with node/link costs

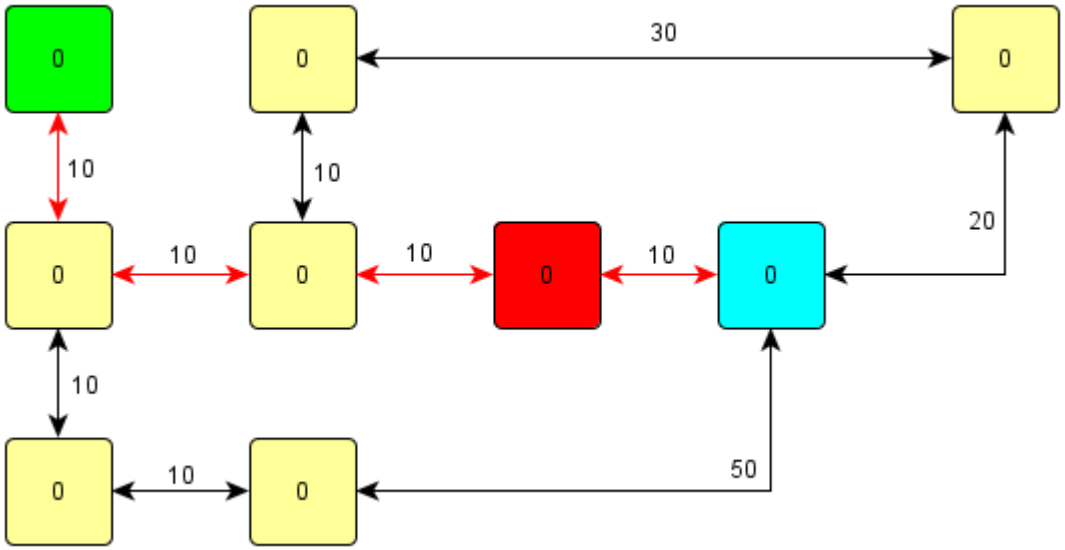


- Shortest path

A* Algorithm

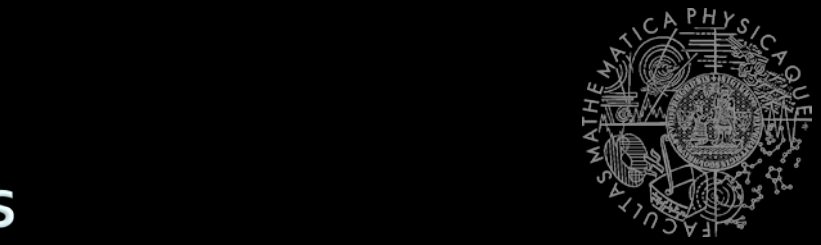
Juggling with node/link costs

- Adversary we want to avoid

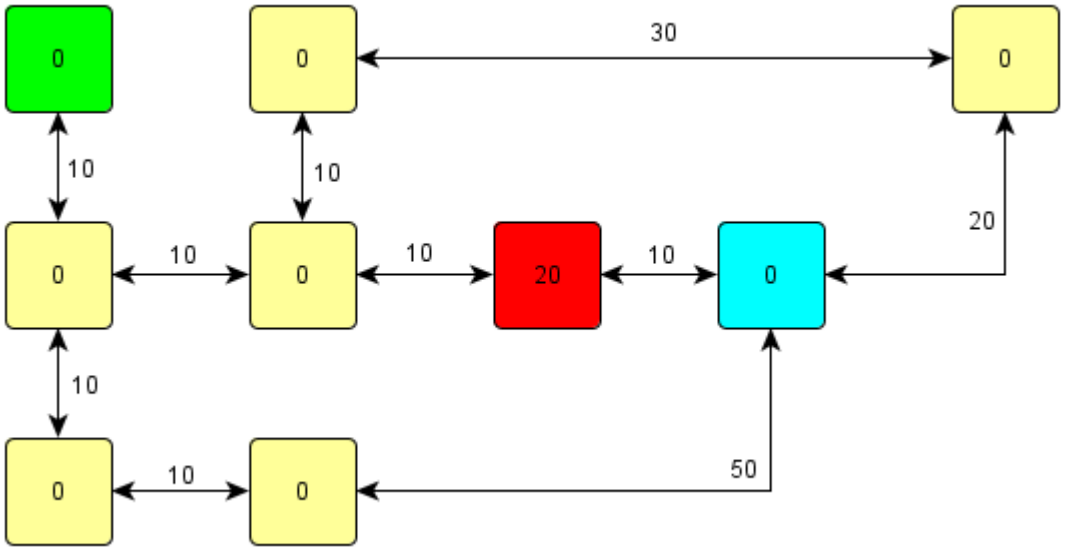


A* Algorithm

Juggling with node/link costs

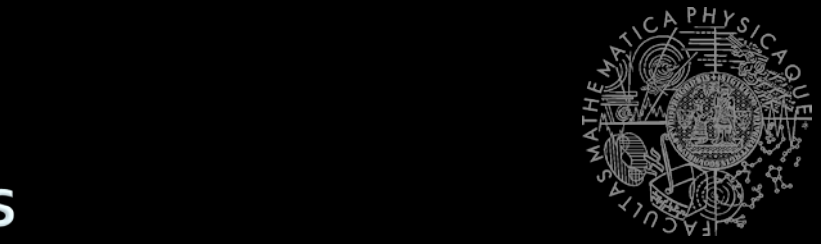


- Let's rise the NODE cost … is it enough?

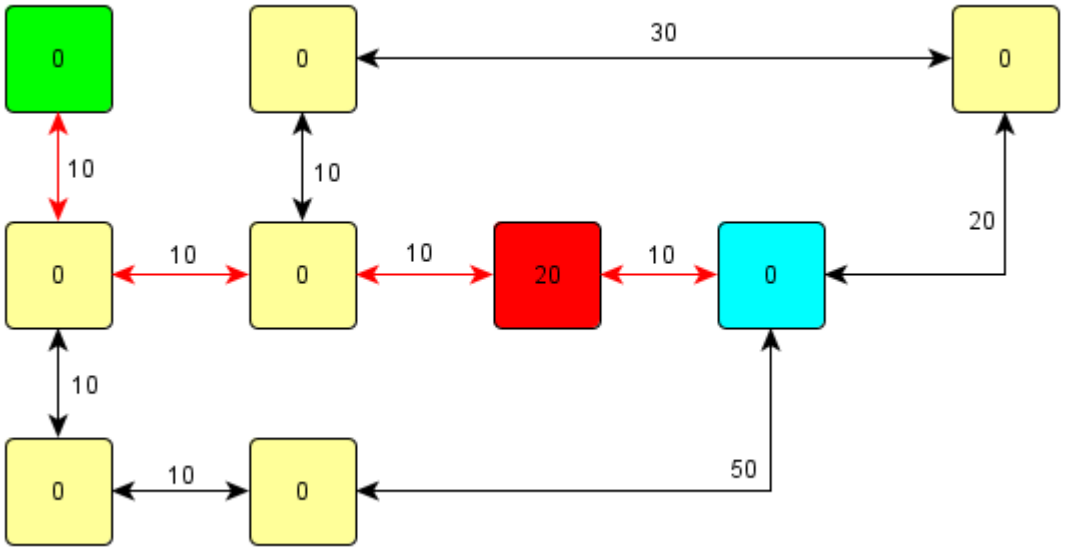


A* Algorithm

Juggling with node/link costs

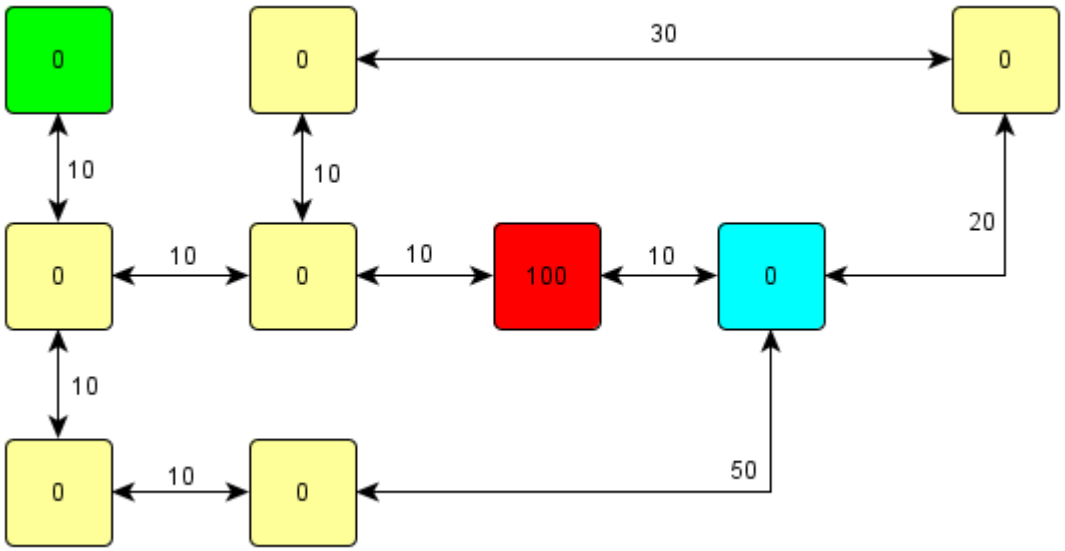


- No…



A* Algorithm

Juggling with node/link costs

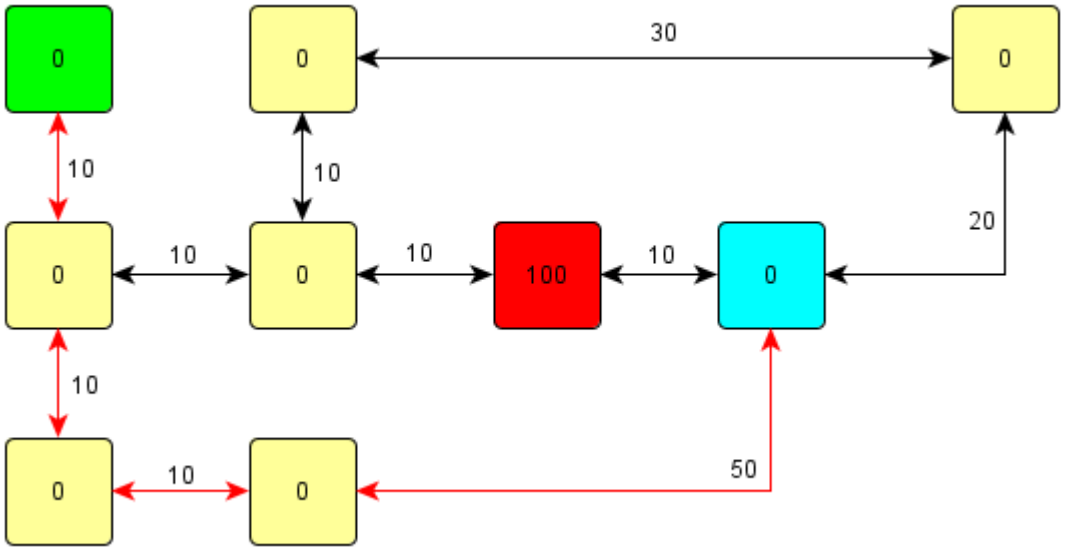


- Rise the NODE cost again… enough now?

A* Algorithm

Juggling with node/link costs

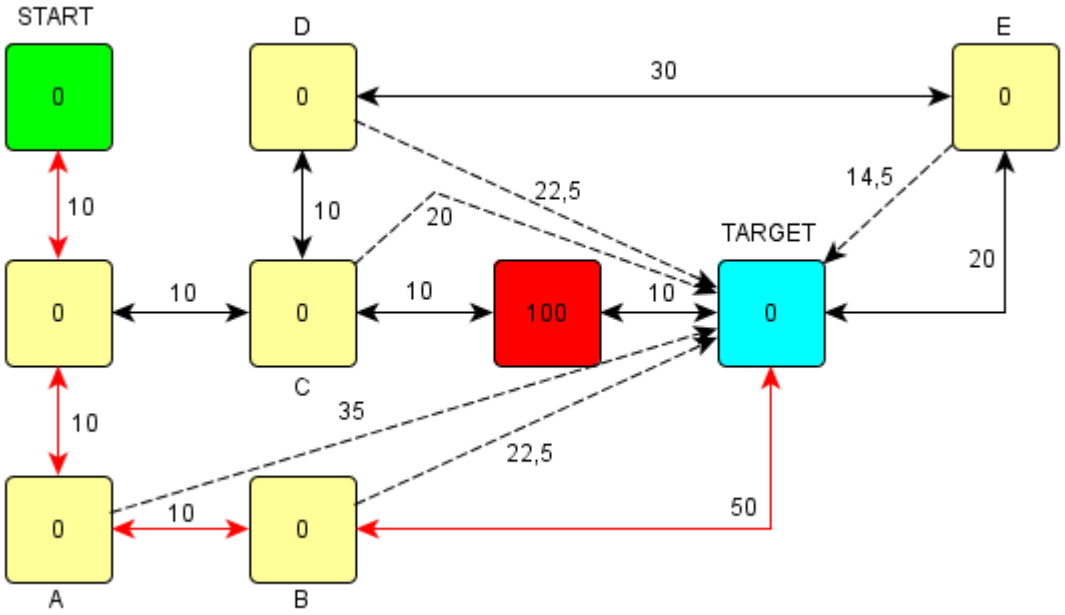
- Here you go!
  - Why was this path found?



A* Algorithm

Juggling with node/link costs

- Adding important heuristic costs
  - So, are we cheating or not?



A* Algorithm
Generic design

- Separating three concerns
  - Goal definition
    - What do we try to achieve?
    - `ISearchGoal`
  - Search strategy
    - How do we define search space and direct the search?
    - `ISearchStrategy`
  - Graph view
    - How do our agent perceive the underlying graph?
    - `IGraphView`

Homework 03

10 Points

1. **See Pac-Man exercise E5**
   - https://github.com/kefik/MsPacMan-vs-Ghosts-AI
2. **Implement informed A-Star search**
   - See `InformedSearch.step()`
3. **Come up with a way to eat everything up in the shortest time …**
   - Hint: greedy way, add new links until it becomes Eulerian, than trigger the search for Eulerian path

- No deadline in here…

Submit your homework

- Completely zip-up your project(s) folder
  - **WITHOUT** the **bin** folder!
- Send it to:
  - Jakub Gemrot
    - gemrot@gamedev.cuni.cz
- Use subject:
  - AI1 – 2016 – H03 – Path-Finding
- Every reported & confirmed bug (within the framework) is for 1 credit!