Overview

- written as A* Algorithm
- [[Shortest Path Algorithms]] to find a single destination
- based on [[Breadth-First Search]] and [[Dijkstra's Algorithm]]
- informed
  - does not search uniformly
  - uses heuristics
  - prioritizes towards the direction of the goal

Heuristics

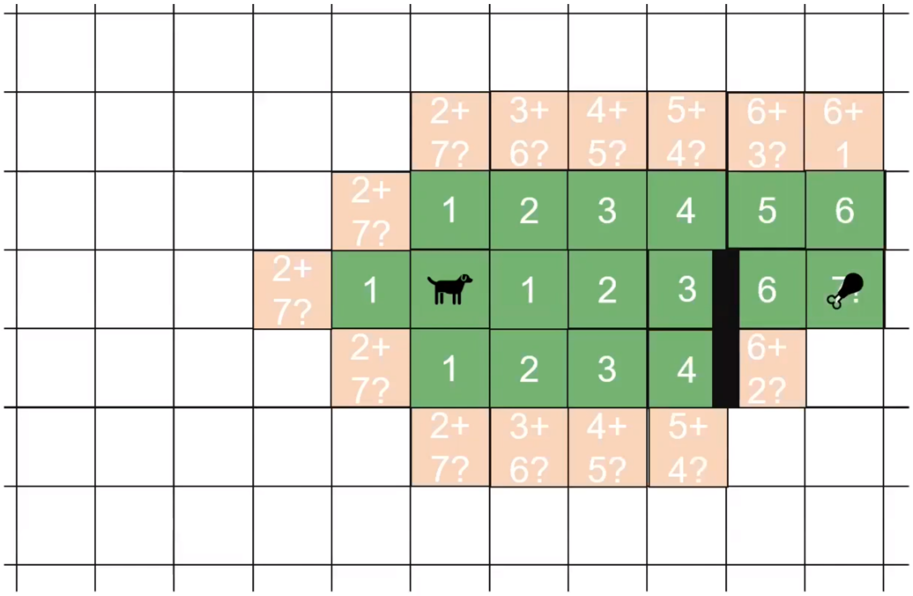
- **Def:** A heuristic is **consistent** if for every edge $\{u, v\} \in E$ we have
$$h(u) \leq w(u, v) + h(v)$$
- perfect heuristic
  - border line impossible
  - requires perfect knowledge lol
- overestimate
  - fast
  - not admissible
  - might not find path even if one exists

A* Heuristics

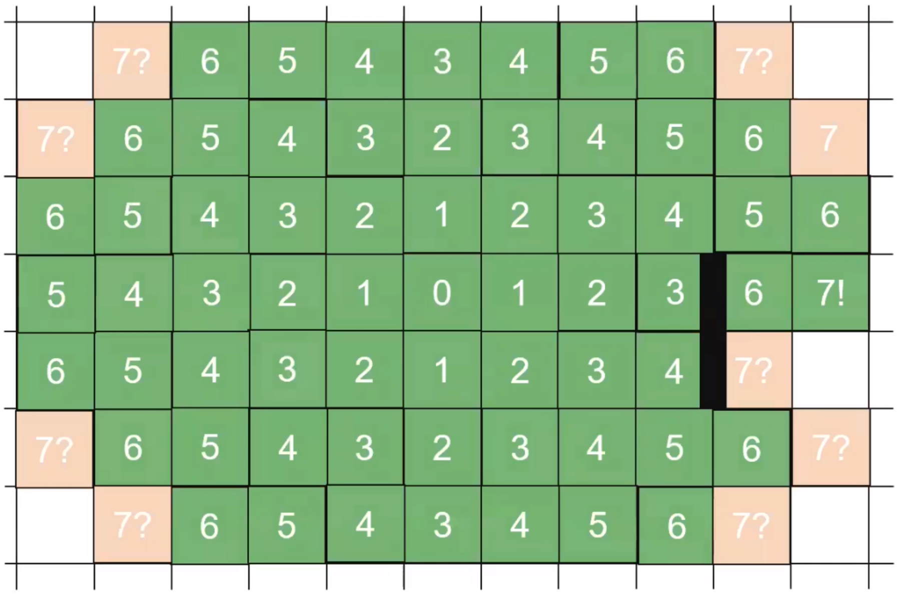
- $g(v)$
  - distance from start to current vertex
- $h(u)$
  - distance from current vertex to end
    - "Luftlinie" - as the crow flies
  - ignores obstacles which may block the path
  - underestimates the future cost
    - good characteristic
- therefore pretty efficient

Comparison between A* and Dijkstra's

- A*



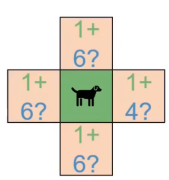
- Dijkstra's

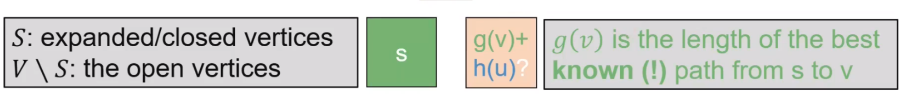


Algorithm

`Input: G(V,E,W) start point: s, end point: t`

`Initialize S={s},` $g(s)=0$, $g(V\backslash N(s))=\infty$, $g(v)=w(s,v)$, $v\in N(s)$



- 

```
While t ∉ S do
    u = argmin_{u∈V\S}{g(u) + h(u,t)}
    For v s.t. {u,v}∈ E do
        temp=min{g(v),g(u)+w(u,v)}
        If temp<g(v) then:
            g(v)=temp
            S=S\{v}          //does nothing if v ∉ S
    S= S ∪{u}
```

g(v)+ h(u)?

$g(v)$ is the length of the best **known (!)** path from s to v

$h(u,t)$ is a heuristic guess for the path from u to t.

S might decrease!

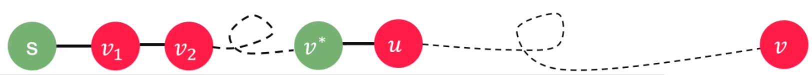
- red parts differ from [[Dijkstra's Algorithm]]

Properties

- nodes may expand more than once
  - $g(v)$ heuristic value can change
  - always terminates if a path exists
- optimal if a path exists

Lemma 1: Always, for every *open* node v and every optimal path P from s to v, there exists an ***open** node* u on P with g(u) **=** distance(s, u)

Proof: Let $P = (s = v_0, v_1, v_2, \dots, v = v_k)$.



$C = \{v_i \in P \mid v_i \textbf{ closed}, g(v_i) = d(s,u)\} \neq \emptyset$,

- Let v^*, the vertex in C with highest index. $v^* \neq v$.
- Let u be the successor of v^* in P (possibly $u = v$).

P is optimal path

$$g(u) \le g(v^*) + w(v^*, u) = d(s, v^*) + w(v^*, u) = d(s,u) \le g(u)$$

v^* expanded

definition of v^*

always

u has to be open by definition of C

Corollary: Suppose h is admissible and A* has not terminated. Then, for any optimal path P from s to t, there is an open node u with

g(u) + heuristic(u,t) $\le$ distance(s,t)

Proof: By Lemma 1, we have open node $u \in P$ *with* g(u) **=** distance(s, u)

g(u) + heuristic(u,t) **=** distance(s, u) + heuristic(u,t)

h admissible $\le$ distance(s, u) + futureCost(u,t)

Proof (Optimality):

Suppose A* terminates at t with a suboptimal path, i.e., in the last step we expanded t with

g(t) + $\overbrace{\text{heuristic(t,t)}}^{0}$ > distance(s, t)

But, by the corollary, there existed just before "expanding t", there is an open node u on an optimal path with

g(u) + heuristic(u,t) $\le$ distance(s, t) Contradiction!

- optimally efficient
  - with regards to the number of vertices expanded
- space as bottle neck
  - $g(v) + h(v)$ is stored for each visited v
  - **15-Puzzle:** Search space has a node for each configuration: 16!=20.922.789.888.000 vertices!
  - motivation for memory bounded heuristic search
    - Iterative deepening A*