A*

- Idea: very similar to Dijkstra but with one difference:

Dijkstra:

(1) Create distTo and edgeTo lists and priority queue PQ (which prioritizes nodes closest to source)

- distTo(v): best known distance from source s to v
- edgeTo(v): best known vertex predecessor to v
- PQ contains all unvisited vertices in order of distTo(V)

(2) Initialize:

- for each index i corresponding to vertex i, set distTo(i) = ∞; except vertex 0 → distTo(0) = 0.
- for each index i corresponding to vertex i, set edgeTo(i) = null.

(3) Repeat:

```
· Visit v: remove closest vertex V from PQ
· Relax edges
        for each outgoing edge e from v:
          if dist(S,V) + e < distTo(v):
             distTo(v) = dist(S,V) + e
             edgeTo(v) = V
```

(4) End: when PQ is empty

A*

(0) Define a goal node that we want to find

(1) Create distTo and edgeTo lists and priority queue PQ (which prioritizes nodes closest to source)

- distTo(v): best known distance from source s to v
- edgeTo(v): best known vertex predecessor to v
- PQ contains all unvisited vertices in order of distTo(V)+h(goal, V)

(2) Initialize:

- for each index i corresponding to vertex i, set distTo(i) = ∞; except vertex 0 → distTo(0) = 0.
- for each index i corresponding to vertex i, set edgeTo(i) = null.

(3) Repeat:

```
· Visit v: remove closest vertex V from PQ
· Relax edges
        for each outgoing edge e from v:
          if dist(S,V) + e < distTo(v):
             distTo(v) = dist(S,V) + e
             edgeTo(v) = V
```

(4) End: when PQ is empty