

## · Idea: Very similar to Dijkstra but with one difference: Dijkstra

- (1) Create distTo and edgeTo lists and priority grew PQ ( which prioritizes nodes closed to soura) · 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()
- (2) Initialize:
  - · for each index i corresponding to vertex i, set dist To (i) = 00; except vertex 0 → distro(0)=0.
  - · For each index i corresponding to vertex i, set elgeTo(i) = null.
- (3) Repeat:
  - · Visit V : remove closest vertex V from Pto · Relax edges

Por each outgang edge e from V: if dist (5, V) + e < dist To (W): distTo(v) = dist(s, v) + e edge To (v) = V

eageTo[

The Find: when PQ is empty

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

- 1) Create distTo and edgeTo lists and priority queue PQ (which prioritizes nodes closed to source)
  - · distTo(v): best known distance from source s to v
  - · edge TO (V): best Known vertex predecessor to V
  - · PQ contains all unvisited vertices in order of listTo()+h (90al)
- (2) InHalize:
  - · for each index i corresponding to vertex i, set dist To (i) = ∞; except vertex 0 → dutro(0)=0.
  - · for each index i corresponding to vertex i, set edge To(i) = null.
- (3) Repeat:
  - · Visit V : remove closest vertex V from Po

· Relax edges for each outgoing edge e from V: if dist (5, V) + e < dist To (v): distTo(v) = dist(s, v) + e edge To (v) = V

(4) End: when PQ is empty