Dijkstra's Shortest Path Algorithm

The shortest path between u and v is denoted $\delta(u,v)$, if there is no path, then $\delta(u,v)=\infty$

1 Can a shortest path contain a cycle?

A directed cycle is:

- Positive: if its edge weights sum up to a positive number
- Negative: if its edge weights sum up to a negative number

If there is a positive cycle in the graph, it will not be contained in any shortest path between u and v so we can assume that the shortest paths we find contain no positive cycles.

However if there is a negative cycle between u and v, then $\delta(u,v)=-\infty$ so we shall assume that the graphs we consider do not contain negative cycles.

2 Single-Source Shortest Paths

- Aim: to describe an algorithm that solves the single-source shortest paths problem, i.e. an algorithm that finds the shortest path from a specific source vertex
- This is a generalization of BFS
- So the output of the algorithm should be two arrays d, π where for each vertex v:
  - $d(v)=\delta(s,v)$
  - $\pi(v)$ is the predecessor of v

3 Relaxation

- Assume that the weight on every edge is non-negative
- We do not directly compute the entry $d(v)=\delta(s,v)$
- Instead, at every step, $d(v)$ is an estimate for $\delta(s,v)$
  - Initially, $d(v)=\infty$, and it always remains $d(v)\geqslant\delta(s,v)$
  - $d(v)$ is updated (i.e. it decreases) as shorter paths are found
  - At the end of the algorithm we have $d(v)=\delta(s,v)$

Listing 1 Initialise-Single-Source(G,s)

```
for each vertex v∈V(g) do
    d(v)=∞
    π(v)=NIL
d(s)=0
```

The process of relaxing an edge (u,v):

- Test whether we can improve the shortest path from s to v that we found so far, by going through u
- If yes, then update $d(v)$ and $\pi(v)$
  - Decrease the estimate $d(v)$
  - Update the predecessor $\pi(v)$ to u
- The algorithm first calls initialise-single-source and then it repeatedly relaxes the appropriate edges (according to the weight function w)

Listing 2 Relax(u,v,w)

```
if d(v)>d(u)+w(u,v) then
    d(v)=d(u)+w(u,v)
    π(v)=u
```

4 Dijkstra's Algorithm