Initialise-Single-Source(G,s)

1 for kaikille solmuille v ∈ V

2 distance[v] = ∞

3 path[v] = NIL

4 distance[s] = 0

Relax(u,v,w)

1 if distance[v] > distance[u] + w(u,v)

2 distance[v] = distance[u]+w(u,v)

3 path[v] = u

Dijkstra(G,w,s)

1 Initialise-Single-Source(G,s)

2 S = ∅

3 while ( kaikki solmut eivät vielä ole joukossa S )

4 valitse solmu u ∈ V \ S, jonka etäisyysarvio lähtösolmuun s on lyhin

5 S = S ∪ {u}

6 for jokaiselle solmulle v ∈ vierus[u] // kaikille u:n vierussolmuille v

7 Relax(u,v,w)

14 // tyhjä rivi

Dijkstra-with-heap(G,w,s)

1 Intialise-Single-Source(G,s)

2 S = ∅

3 for kaikille solmuille v ∈ V

4 heap-insert(H,v,distance[v])

5 while not empty(H)

6 u = heap-del-min(H)

7 S = S ∪ {u}

8 for jokaiselle solmulle v ∈ vierus[u] // kaikille u:n vierussolmuille v

9 Relax(u,v,w)

10 heap-decrease-key(H,v,distance[v]) // ei tee mitään, jos distance[v] ei ole muuttunut

shortest-path(G,v)

1 u = path[v]

2 while u ≠ s

3 push(pino,u)

4 u = path[u]

5 print("lyhin polku solmusta s solmuun v kulkee seuraavien solmujen kautta:")

6 while not empty(pino)

7 u = pop(P)

8 print(u)

Astar(G,w,a,b)

// G tutkittava verkko, a lähtösolmu, b kohdesolmu ja w kaaripainot kertova funktio

1 for kaikille solmuille v ∈ V

2 alkuun[v] = ∞

3 loppuun[v] = arvioi suora etäisyys v → b

4 polku[v] = NIL

5 alkuun[a] = 0

6 S = ∅

7 while ( solmu b ei ole vielä joukossa S )

8 valitse solmu u ∈ V \ S, jolle alkuun[v]+loppuun[v] on pienin

9 S = S ∪ {u}

10 for jokaiselle solmulle v ∈ Adj[u] // kaikille u:n vierussolmuille v

11 if alkuun[v] > alkuun[u] + w(u,v)

12 alkuun[v] = alkuun[u]+w(u,v)

13 polku[v] = u

14 // tyhjä rivi

AstarWithHeap(G,w,a,b)

// G tutkittava verkko, a lähtösolmu, b kohdesolmu ja w kaaripainot kertova funktio

1 for kaikille solmuille v ∈ V

2 alkuun[v] = ∞

3 loppuun[v] = arvioi suora etäisyys v → b

4 polku[v] = NIL

5 alkuun[a] = 0

6 S = ∅

3 for kaikille solmuille v ∈ V

4 heap-insert(H,v, alkuun[v]+loppuun[v])

7 while ( solmu b ei ole vielä joukossa S )

6 u = heap-del-min(H)

9 S = S ∪ {u}

10 for jokaiselle solmulle v ∈ Adj[u] // kaikille u:n vierussolmuille v

11 if alkuun[v] > alkuun[u] + w(u,v)

12 alkuun[v] = alkuun[u]+w(u,v)

13 polku[v] = u

10 heap-decrease-key(H,v, alkuun[v]+loppuun[v])

// ei tee mitään, jos alkuun[v]+loppuun[v] ei ole muuttunut