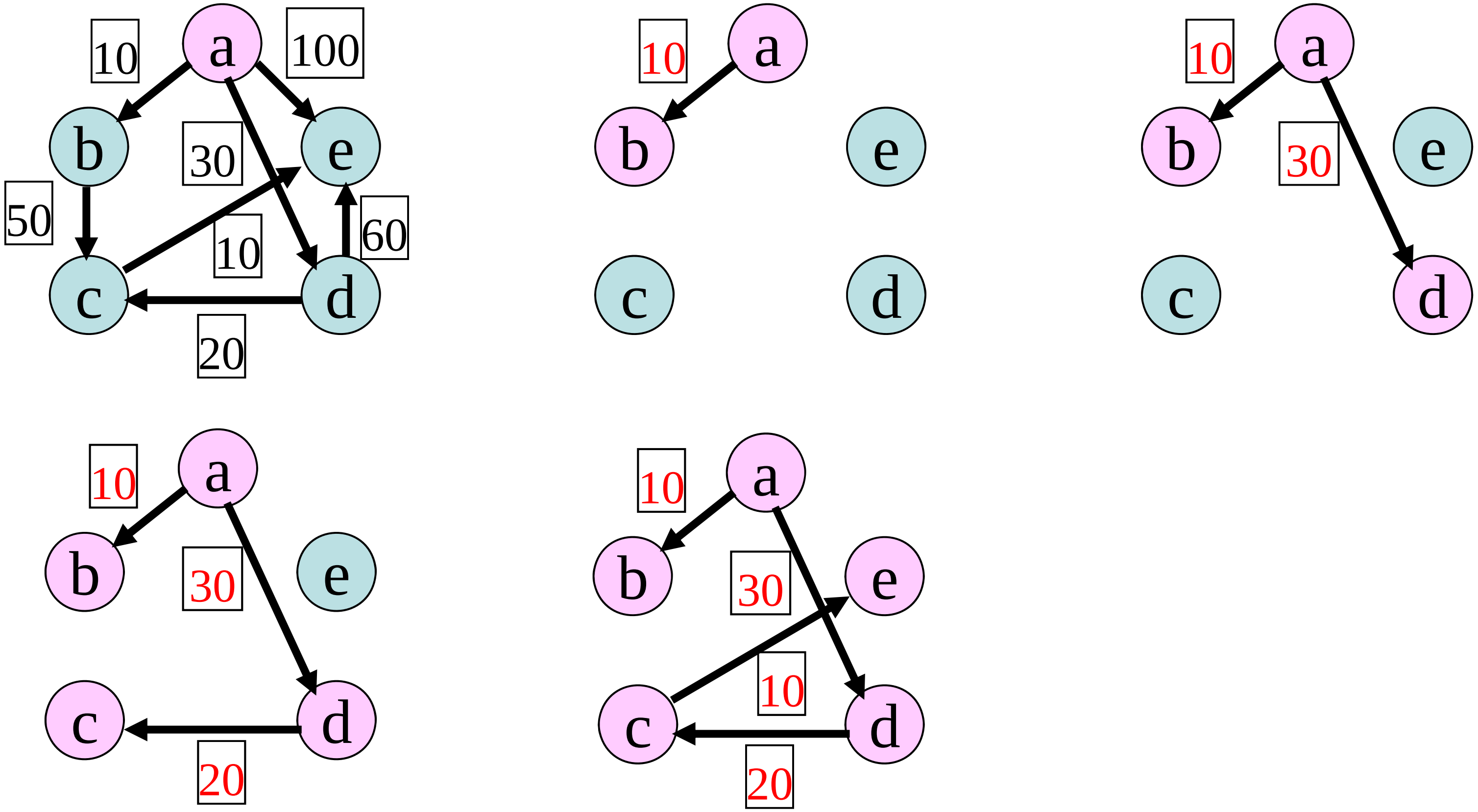
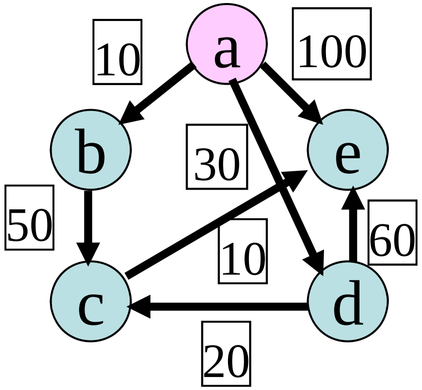
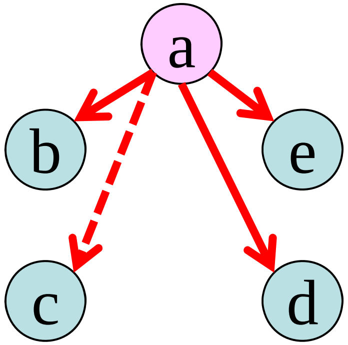
Dijkstra - overview



Initial State





V = {a, b, c, d, e}

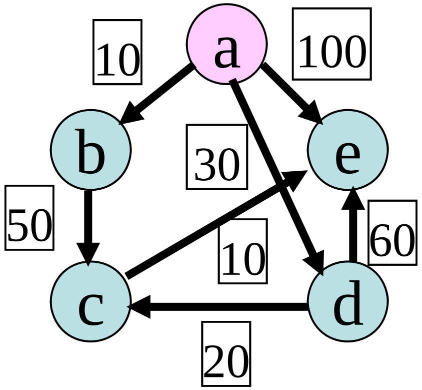
S = {a}

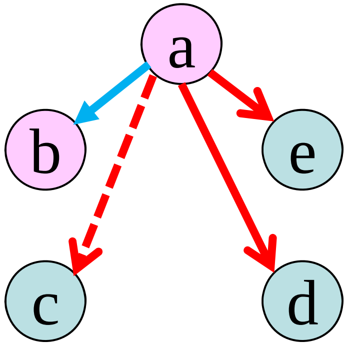
V-S = {b, c, d, e}

	a	b	c	d	e
D (distance)		10	§	30	100
E (edge)		a	**a**	a	a
L (edge cost)		10	§	30	100

D represents the **path length**
§ = infinity i.e. no edge
SPT =
E (edge) + L (edge cost)

Iteration 1: initialisation





	a	**b**	c	d	e
D (distance)		**10**	**§**	30	100
E (edge)		**a**	**a**	a	a
L (edge cost)		**10**	**§**	30	100

Choose minimum {b,c,d,e} ➔ b

V = {a, b, c, d, e}

S = {a, b}

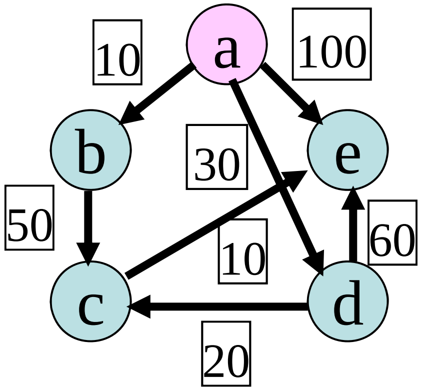
V-S = {c, d, e}

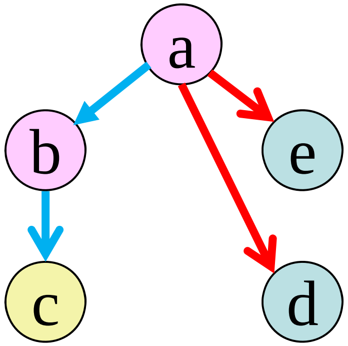
Cheapest path from node a

(here edge a→b)

(a, b, 10)

Iteration 1: recalculate path costs **via b**





	a	b	c	d	e
D (distance)		10	**60**	30	100
E (edge)		a	**b**	a	a
L (edge cost)		10	**50**	30	100

V = {a, b, c, d, e}

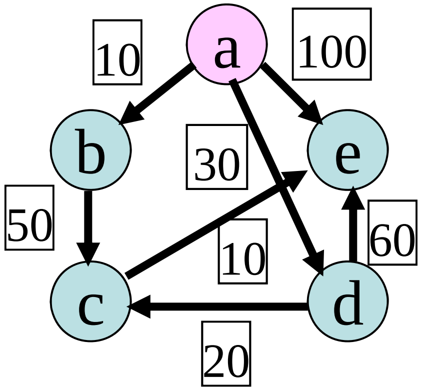
S = {a, b}

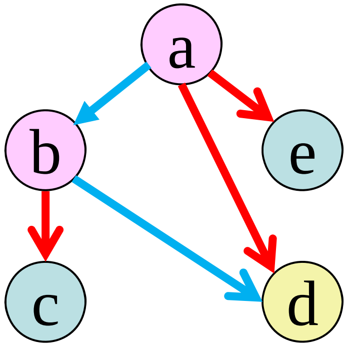
V-S = {c, d, e}

Path from [a→b→c]

D[c] = min(D[c], D[b] + C[b,c]) ➔ min(§, 10+50) ➔ 60 **--- change**

Iteration 1: recalculate path costs **via b**





	a	b	c	d	e
D (distance)		10	**60**	30	100
E (edge)		a	**b**	a	a
L (edge cost)		10	**50**	30	100

V = {a, b, c, d, e}

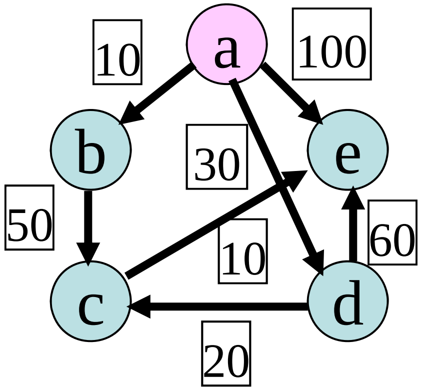
S = {a, b}

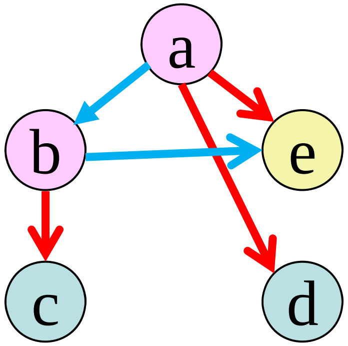
V-S = {c, d, e}

Path from
[a→b→d]

D[d] = min(D[d], D[b] + C[b,d]) ➔ min(30, 10+§) ➔ 30 --- no change

Iteration 1: recalculate path costs **via b**





	a	b	c	d	e
D (distance)		10	60	30	100
E (edge)		a	**b**	a	a
L (edge cost)		10	**50**	30	100

V = {a, b, c, d, e}

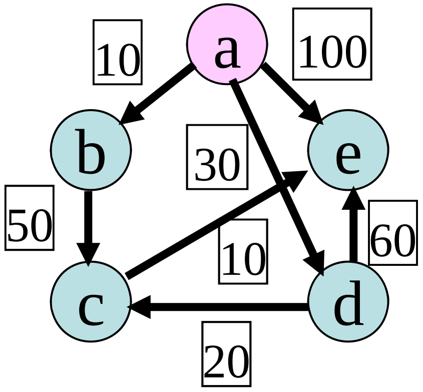
S = {a, b}

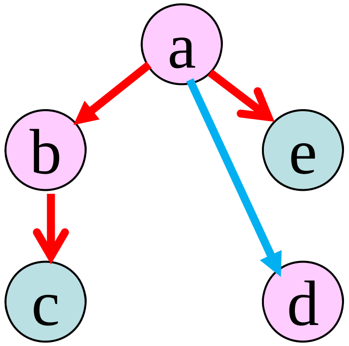
V-S = {c, d, e}

Path from
[a→b→e]

D[e] = min(D[e], D[b] + C[b,e]) ➔ min(100, 10+**§**) ➔ 100 --- no change

Iteration 2: initialisation





V = {a, b, c, d, e}

S = {a, b, d}

V-S = {c, e}

	a	b	c	d	e
D (distance)		10	60	**30**	100
E (edge)		a	b	**a**	a
L (edge cost)		10	50	**30**	100

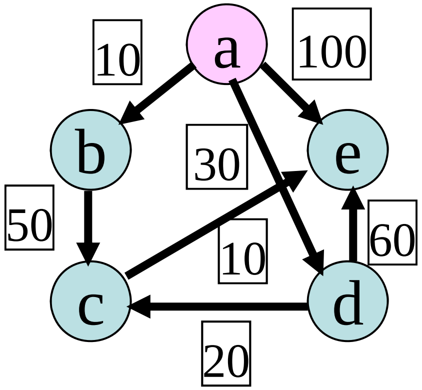
Choose minimum {c,d,e} ➔ d

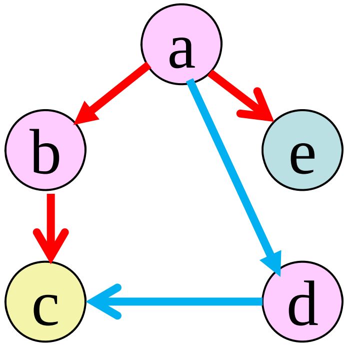
Cheapest path from node [a]

(here edge a→d)

(a, d, 30)

Iteration 2: recalculate path costs **via d**





V = {a, b, c, d, e}

S = {a, b, d}

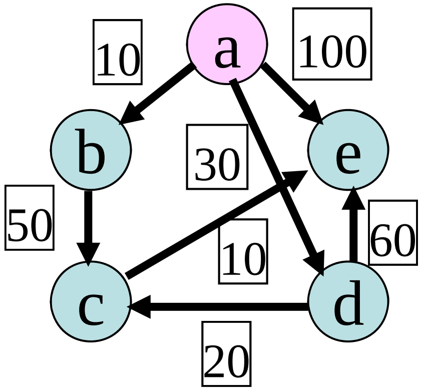
V-S = {c, e}

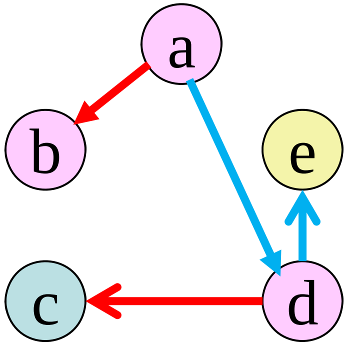
	a	b	c	d	e
D (distance)		10	**50**	30	100
E (edge)		a	**d**	a	a
L (edge cost)		10	**20**	30	100

Path from [a→d→c]

D[c] = min(D[c], D[d] + C[d,c]) ➔ min(60, 30+20) ➔ 50 **--- change**

Iteration 2: recalculate path costs **via d**





	a	b	c	d	e
D (distance)		10	50	30	**90**
E (edge)		a	**d**	a	**d**
L (edge cost)		10	**20**	30	**60**

V = {a, b, c, d, e}

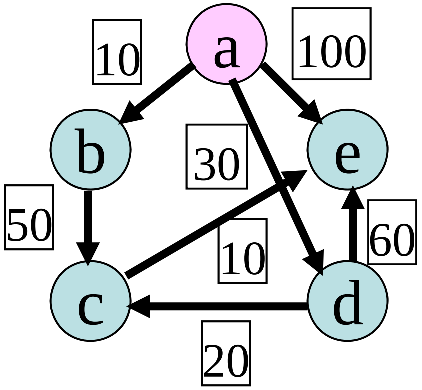
S = {a, b, d}

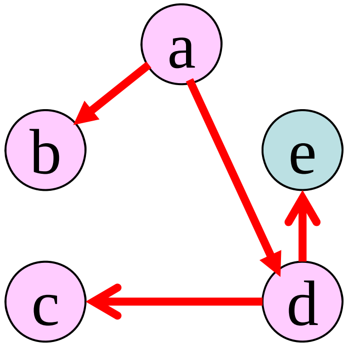
V-S = {c, e}

Path from [a→d→e]

D[e] = min(D[e], D[d] + C[d,e]) ➔ min(100, 30+60) ➔ 90 **--- change**

Iteration 3: initialisation





V = {a, b, c, d, e}

S = {a, b, d, c}

V-S = {e}

	a	b	c	d	e
D (distance)		10	**50**	30	90
E (edge)		a	**d**	a	d
L (edge cost)		10	**20**	30	60

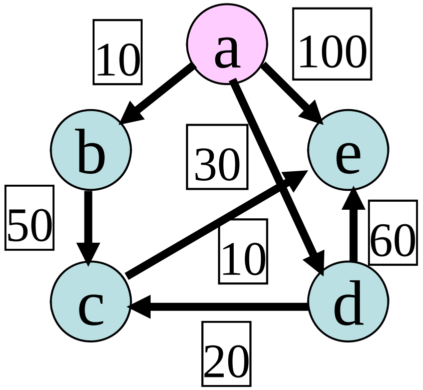
Choose minimum {c,e} ➔ c

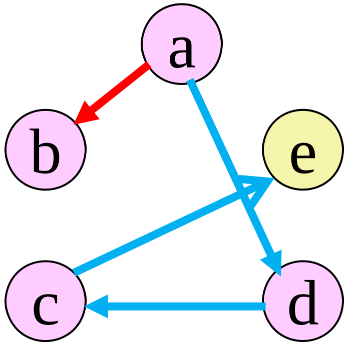
Cheapest path from node [a]

(here path a→d→c)

(a, d, 30) + (d, c, 20)

Iteration 3: recalculate path costs **via c**





	a	b	c	d	e
D (distance)		10	50	30	**60**
E (edge)		a	d	a	**c**
L (edge cost)		10	20	30	**10**

V = {a, b, c, d, e}

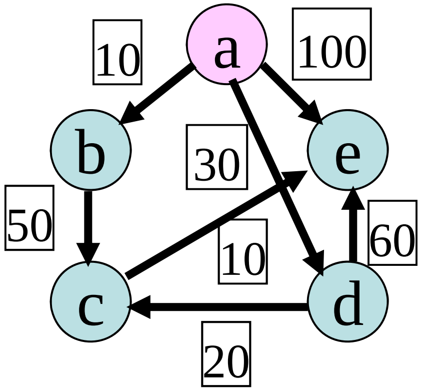
S = {a, b, d, c}

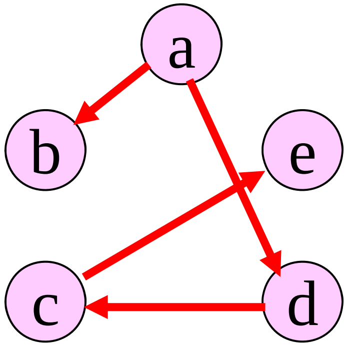
V-S = {e}

Cheapest path from a→d→c to e

D[e] = min(D[e], D[c] + C[c,e]) ➔ min(90, 50+10) ➔ 60 **--- change**

Iteration 4: initialisation/final result





	a	b	c	d	e
D (distance)		10	50	30	60
E (edge)		a	d	a	c
L (edge cost)		10	20	30	10

V = {a, b, c, d, e}

S = {a, b, d, c, e}

V-S = { }

Cheapest path from node [a]

a→d→c→e

V-S is empty

Choose minimum {e} ➔ e