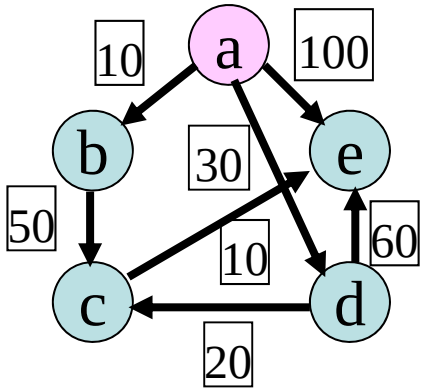
Dijkstra - overview

Graph

- 7 edges



Cost Matrix

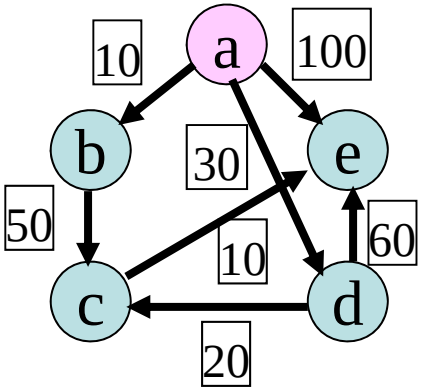
	a	b	c	d	e
a		**10**	§	**30**	**100**
b	§		**50**	§	§
c	§	§		§	**10**
d	§	§	**20**		**60**
e	§	§	§	§	

Dijkstra - overview

Graph

- 7 edges

- **a→b 10**
- **a→d 30**
- **a→e 100**
- **b→c 50**
- **c→e 10**
- **d→c 20**
- **d→e 60**

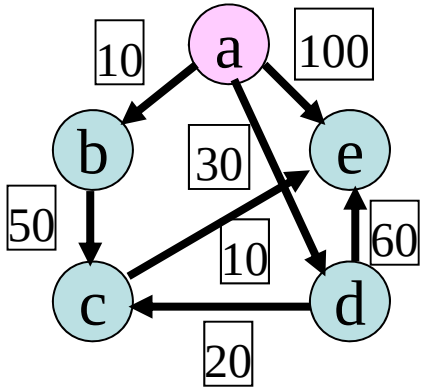


Paths a to {b,c,d,e}

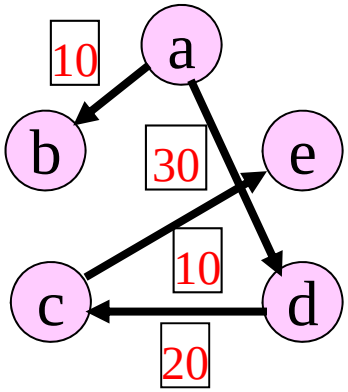
- 8 paths

- **a→b 10**
- **a→b→c 60**
- **a→d→c 50 (shortest path)**
- **a→d 30**
- **a→e 100**
- **a→b→c→e 70**
- **a→d→e 90**
- **a→d→c→e 60 (shortest path)**

Dijkstra - overview



	Look at the paths	
a→b	**a→b** (10)	a→**b→c** (**60**)
a→d	**a→d** (30)	a→**d→c** (**50**) a→**d→e** (**90**)
a→c	§ infinite	a→d→**c→e** (**60**)
a→e	**a→e** (100)	



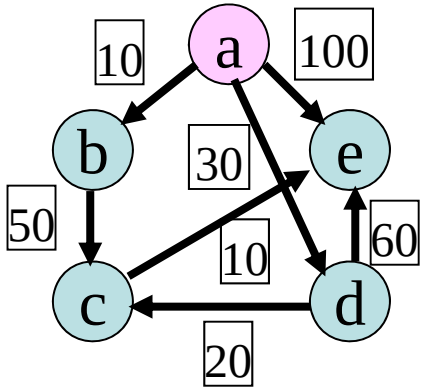
a	b	c	d	e
i	10	§	30	100

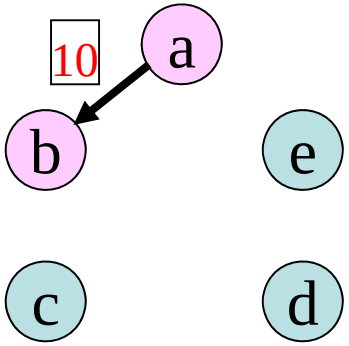
a	b	c	d	e
ii	10	**60**	30	100

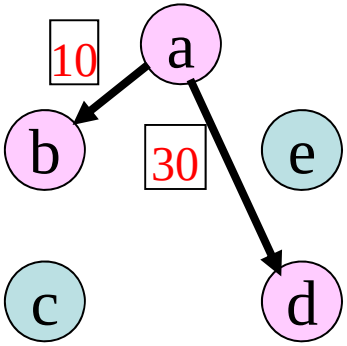
a	b	c	d	e
iii	10	**50**	30	**90**

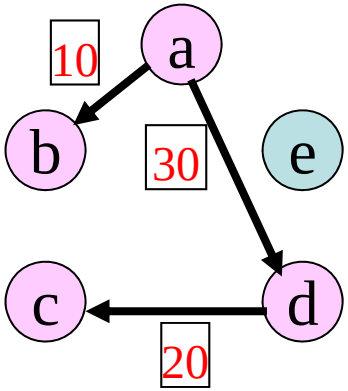
a	b	c	d	e
iv	10	**50**	30	**60**

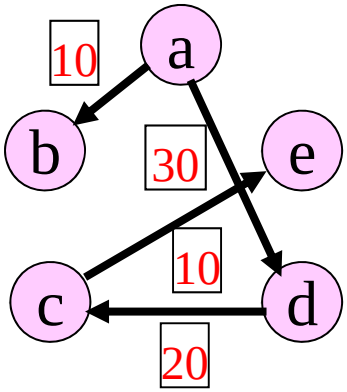
Dijkstra - overview







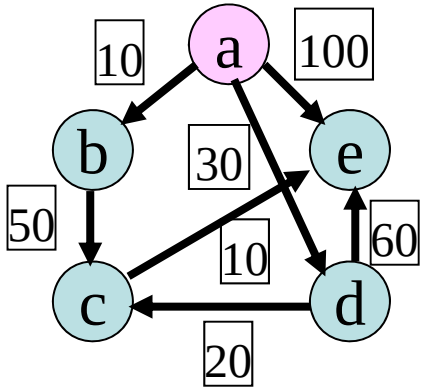




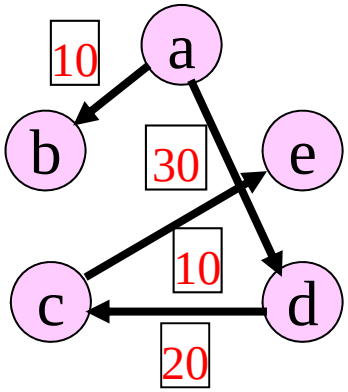
a	b	c	d	e
i	10	§	30	100

a	b	c	d	e
iv	10	**50**	30	**60**

Dijkstra - overview



	Look at the paths	
a→b	**a→b** (10)	a→**b→c** (60)
a→d	**a→d** (30)	a→**d→c** (**50**) a→**d→e** (90)
a→c	§ infinite	a→d→**c→e** (**60**)
a→e	**a→e** (100)	



a	**b**	**c**	**d**	**e**
	10	**50**	30	**60**
	a	**d**	a	**c**
	10	**20**	30	**10**

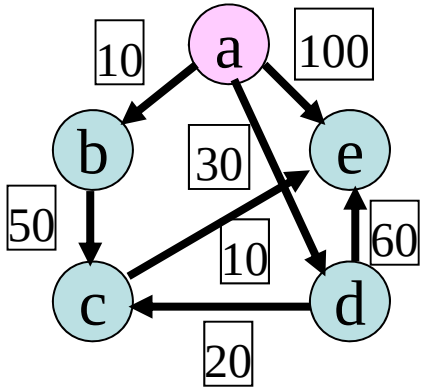
SPT Shortest Path Tree

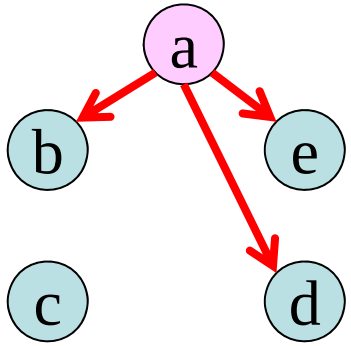
← Distance from a

← Edges

← Edge costs

Initial State





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

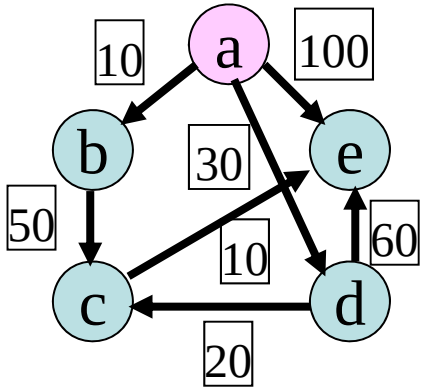
S = {a}

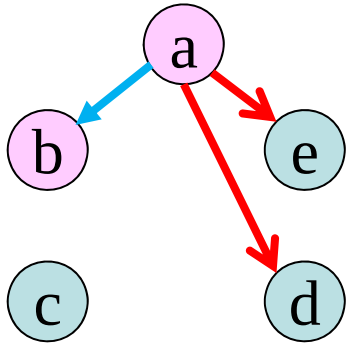
V-S = {b, c, d, e}

	a	b	c	d	e
D (distance)		10	§	30	100

D represents the **path length**
§ = infinity i.e. no edge

Iteration 1: initialisation





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

S = {a, b}

V-S = {c, d, e}

	a	**b**	c	d	e
D (distance)		**10**	**§**	30	100

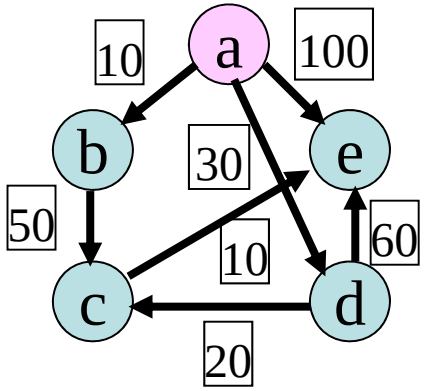
Choose minimum {b,c,d,e} ➔ b

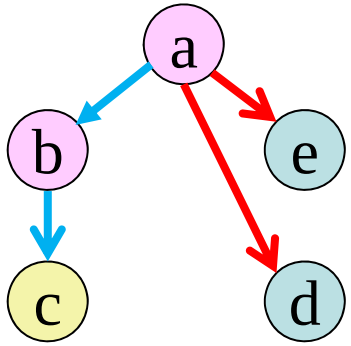
Cheapest path from node a

(here edge a→b)

(a, b, 10)

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





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

S = {a, b}

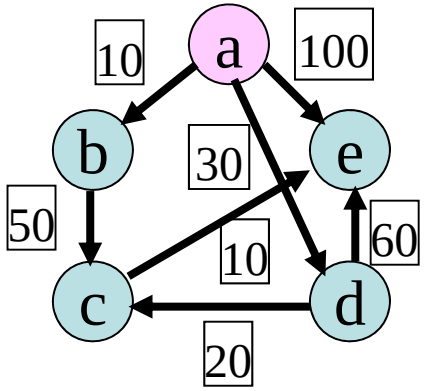
V-S = {c, d, e}

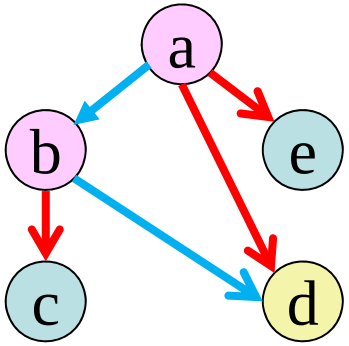
	a	b	c	d	e
D (distance)		10	**60**	30	100

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**





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

S = {a, b}

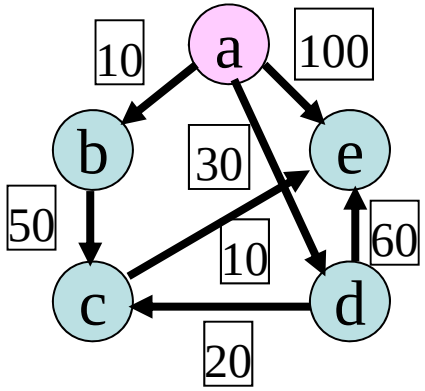
V-S = {c, d, e}

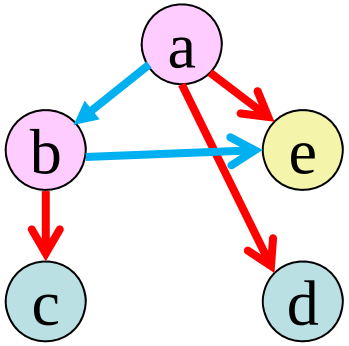
	a	b	c	d	e
D (distance)		10	**60**	30	100

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**





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

S = {a, b}

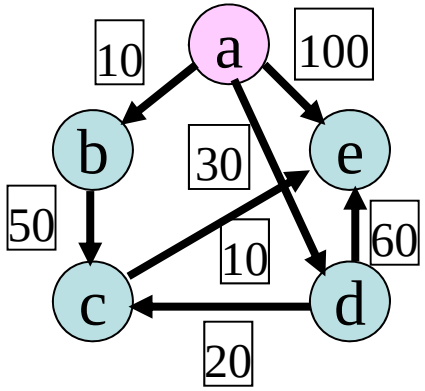
V-S = {c, d, e}

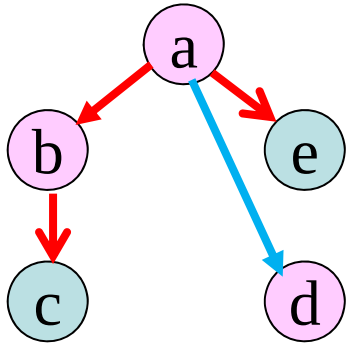
	a	b	c	d	e
D (distance)		10	60	30	100

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

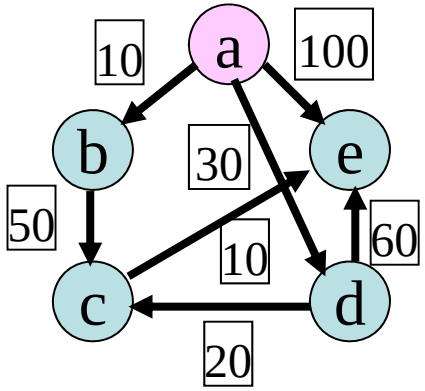
Choose minimum {c,d,e} ➔ d

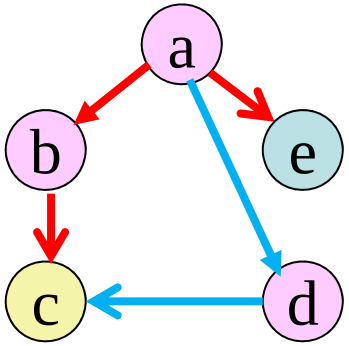
Cheapest path from node [a]

(here edge a→d)

(a, d, 30)

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





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

S = {a, b, d}

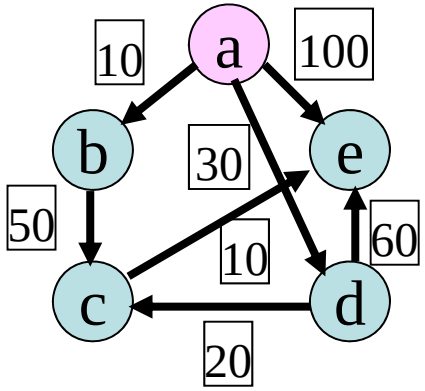
V-S = {c, e}

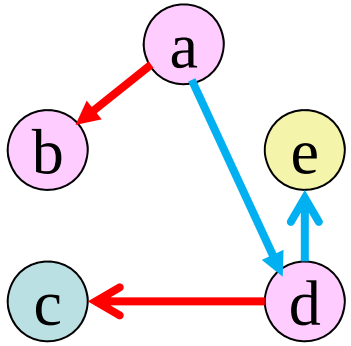
	a	b	c	d	e
D (distance)		10	50	30	100

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**





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

S = {a, b, d}

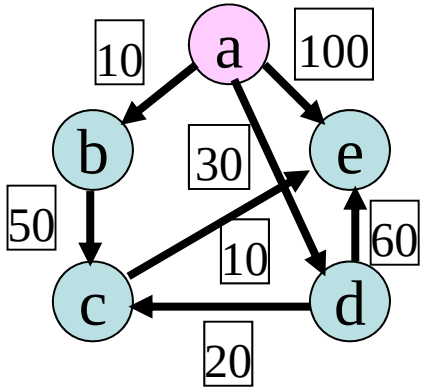
V-S = {c, e}

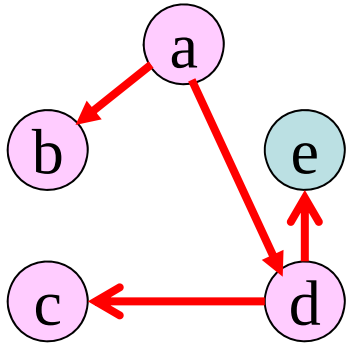
	a	b	c	d	e
D (distance)		10	50	30	90

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

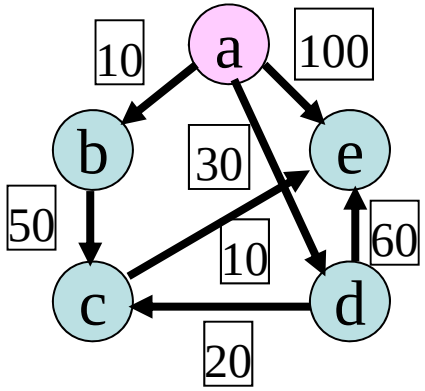
Choose minimum {c,e} ➔ c

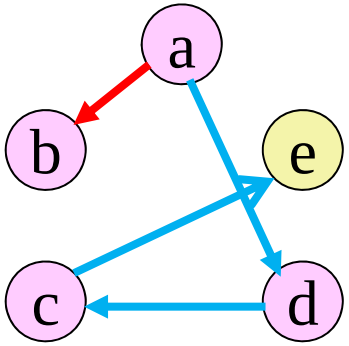
Cheapest path from node [a]

(here path a→d→c)

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

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





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

S = {a, b, d, c}

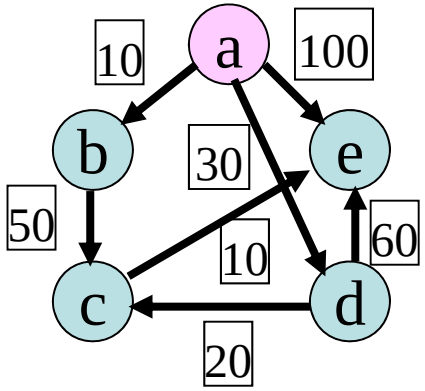
V-S = {e}

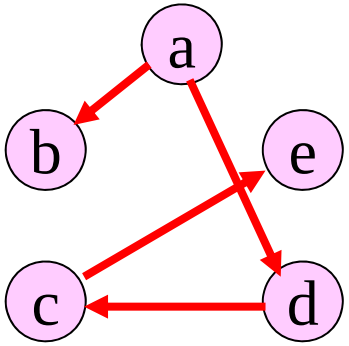
	a	b	c	d	e
D (distance)		10	50	30	60

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





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

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

V-S = { }

	a	b	c	d	e
D (distance)		10	50	30	60

Choose minimum {e} ➔ e

Cheapest path from node [a]

a→d→c→e

V-S is empty