Jak działają Google Maps

Jakub Cisło

Uniwersytet Jagielloński

jakub@cislo.net.pl

2 czerwca 2017

Plan

1. Algorytm
2. Grafy
3. Najkrótsza droga
   - Algorytm BFS
   - Algorytm Dijkstry

Algorytm

Jak zrobić jajecznicę?

Przepis na jajecznicę

Przepis na jajecznicę

1. Rozbij 4 jajka do miski.

Przepis na jajecznicę

1. Rozbij 4 jajka do miski.
2. Dobrze wymieszaj żółtka z białkiem.

Przepis na jajecznicę

1. Rozbij 4 jajka do miski.
2. Dobrze wymieszaj żółtka z białkiem.
3. Dopraw solą i pieprzem.

Przepis na jajecznicę

1. Rozbij 4 jajka do miski.
2. Dobrze wymieszaj żółtka z białkiem.
3. Dopraw solą i pieprzem.
4. Na patelni rozpuść kawałek masła.

Przepis na jajecznicę

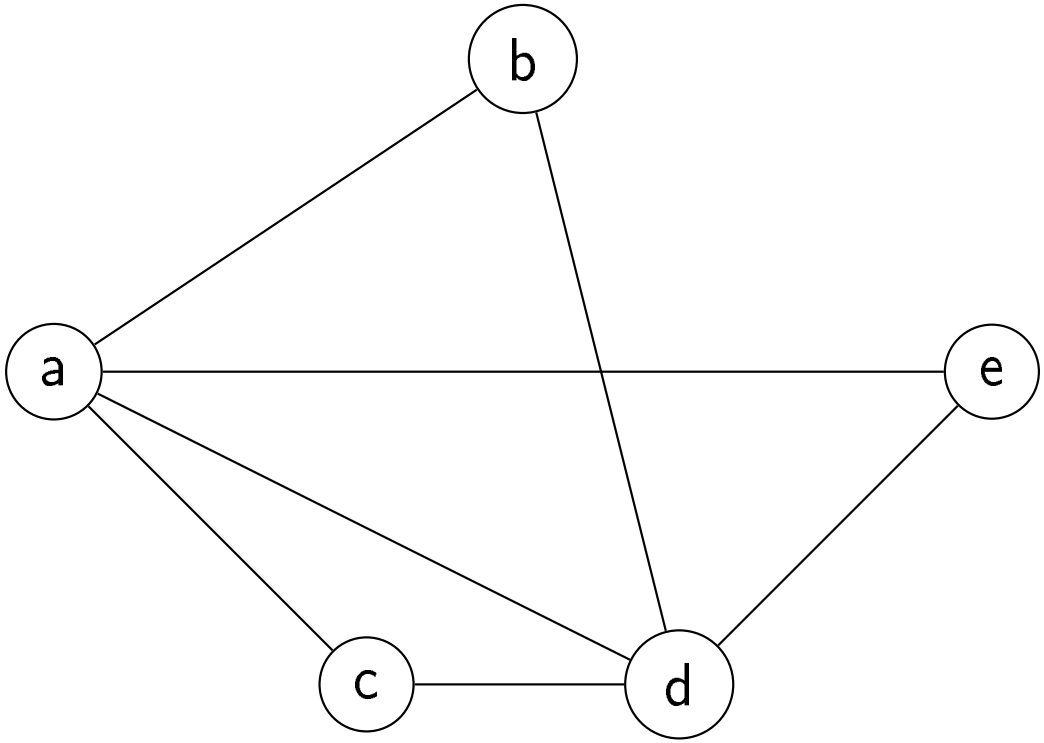
1. Rozbij 4 jajka do miski.
2. Dobrze wymieszaj żółtka z białkiem.
3. Dopraw solą i pieprzem.
4. Na patelni rozpuść kawałek masła.
5. Wlej wymieszane żółtka i białko na patelnię.

Przepis na jajecznicę

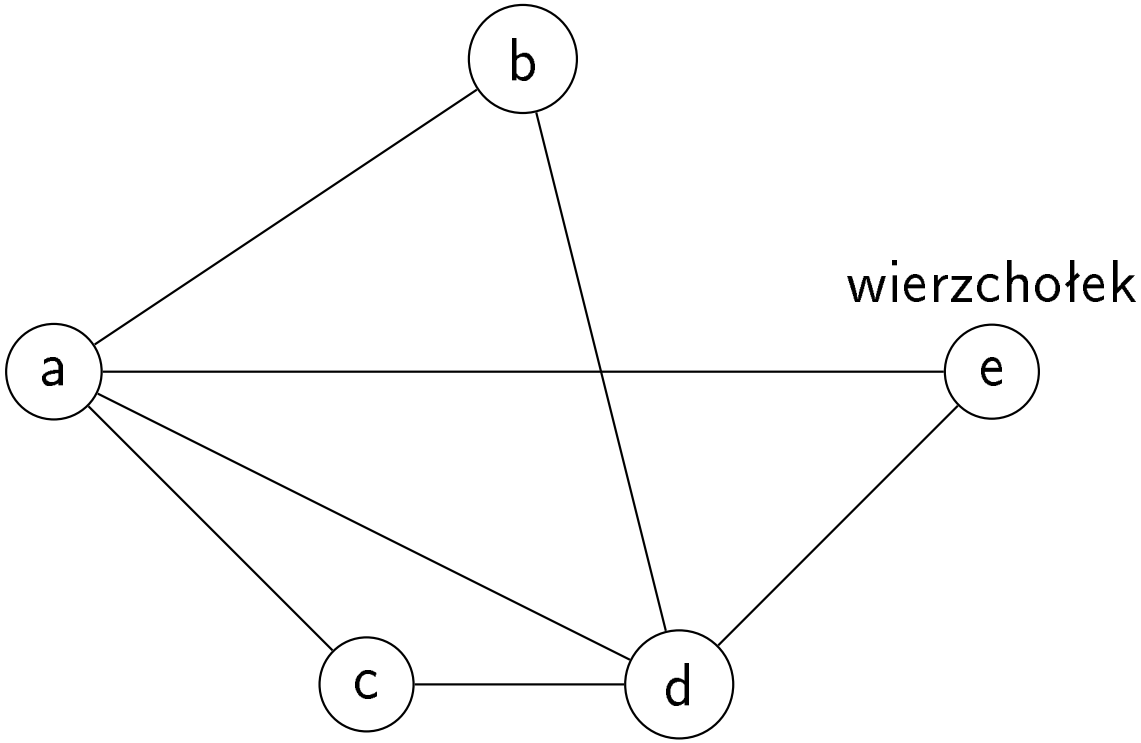
1. Rozbij 4 jajka do miski.
2. Dobrze wymieszaj żółtka z białkiem.
3. Dopraw solą i pieprzem.
4. Na patelni rozpuść kawałek masła.
5. Wlej wymieszane żółtka i białko na patelnię.
6. Mieszaj zawartość patelni aż jajecznica się zetnie.

Grafy

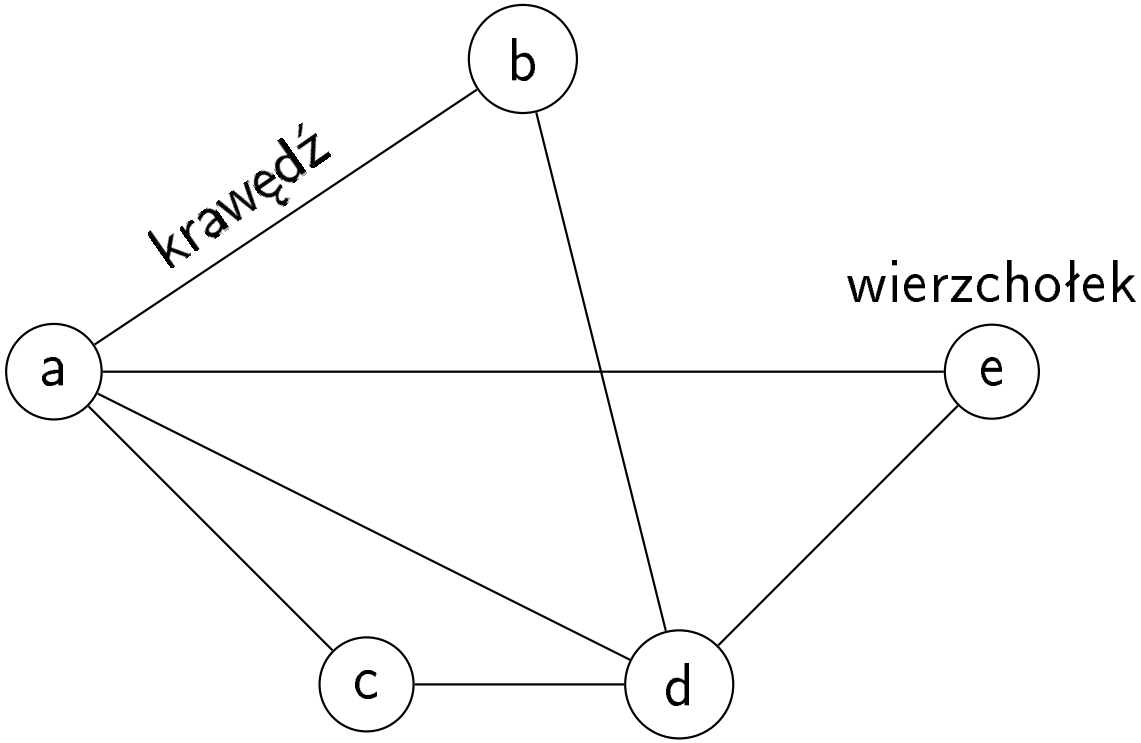
Graf nieskierowany



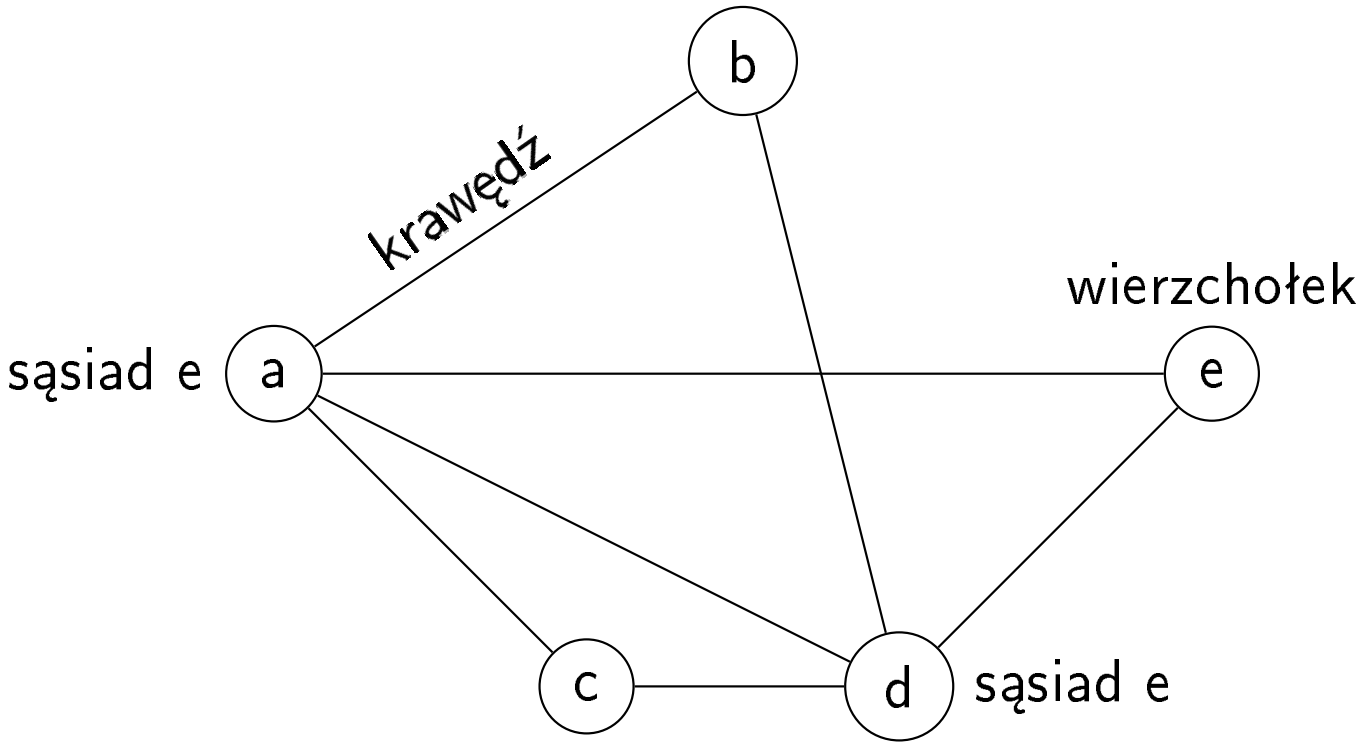
Graf nieskierowany



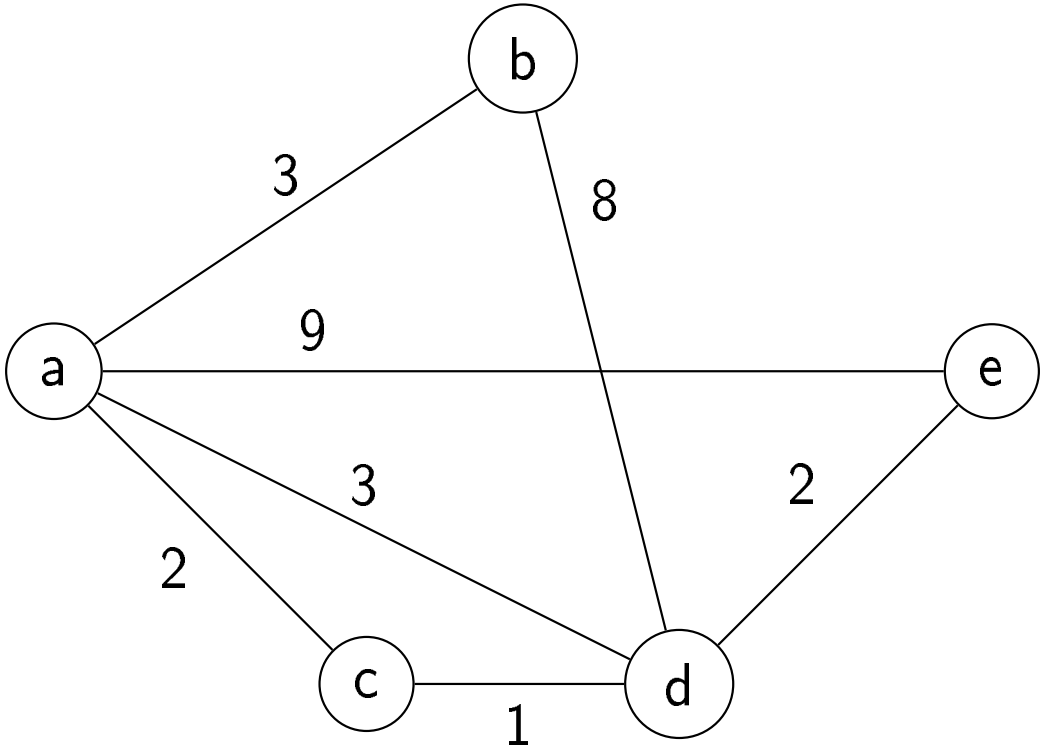
Graf nieskierowany



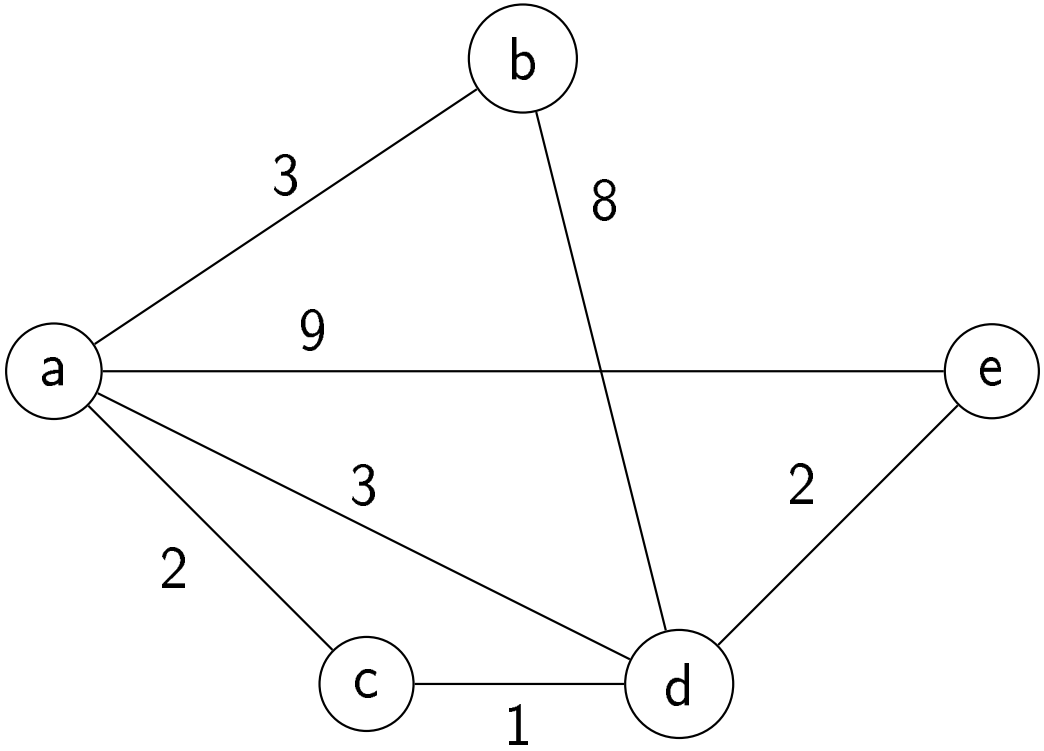
Graf nieskierowany



Graf ważony

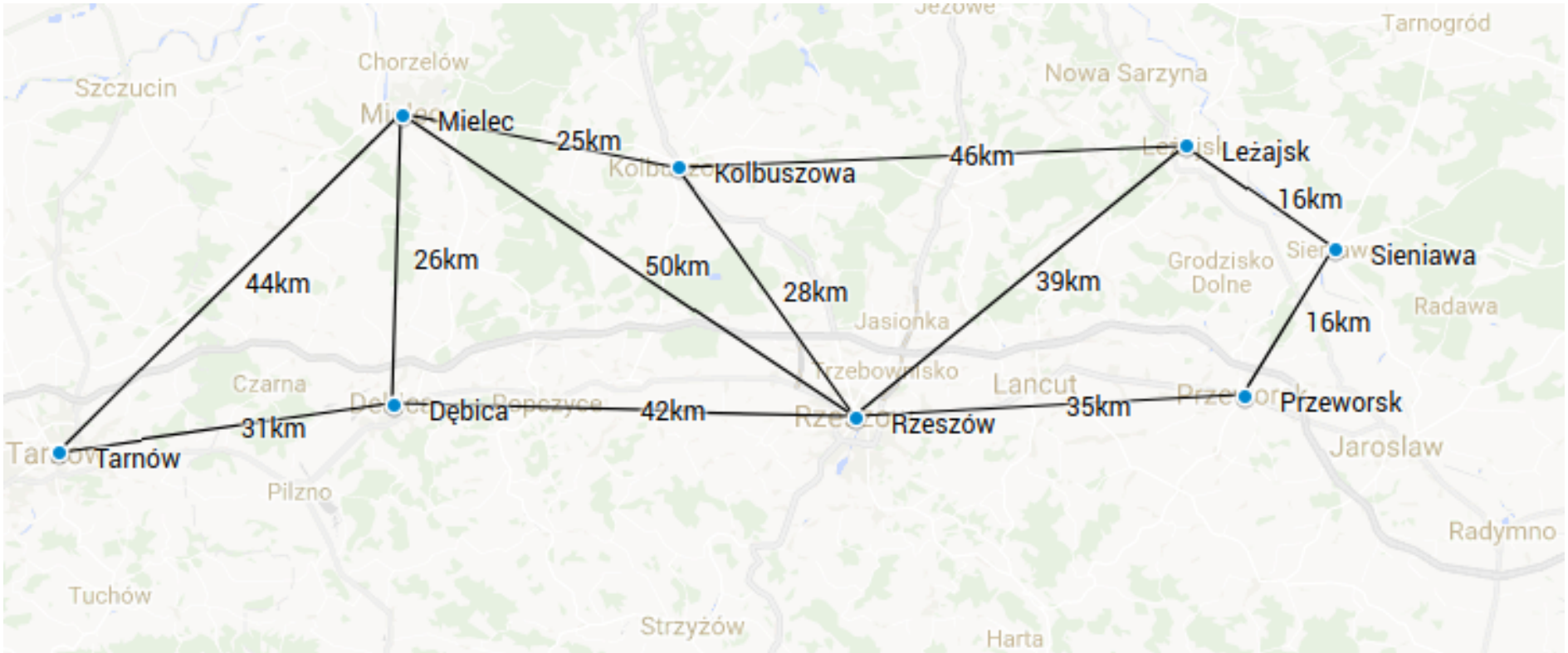


Graf ważony

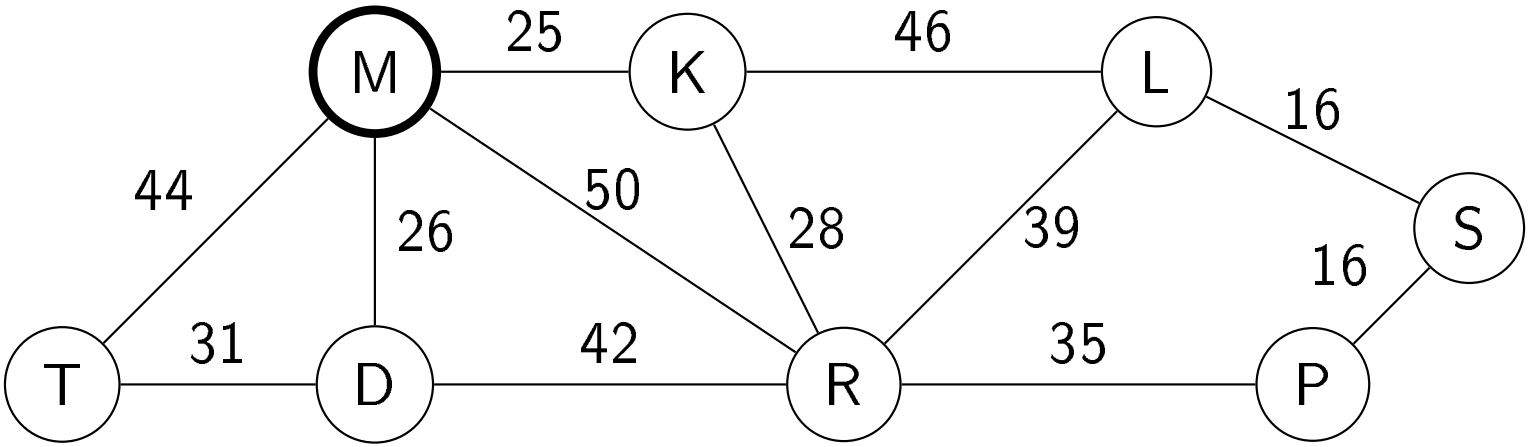


Najkrótsza droga

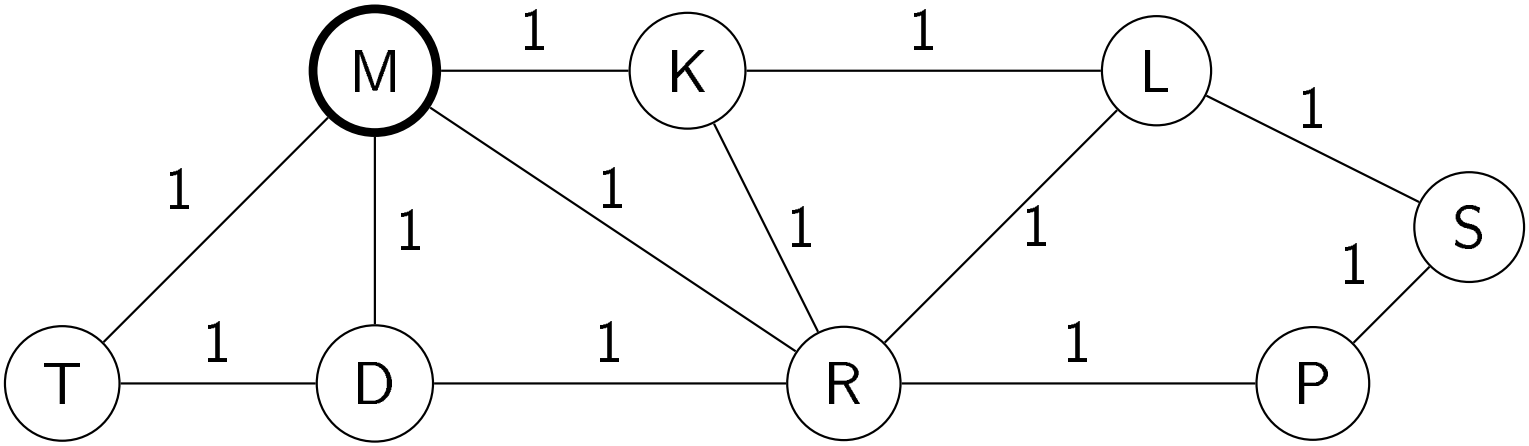
Mapa


Mielec
Kolbuszowa
Leżajsk
Sieniawa
Przeworsk
Rzeszów
Dębica
Tarnów
25km
46km
16km
26km
50km
28km
39km
16km
44km
31km
42km
35km
Chorzelów
Szczucin
Nowa Sarzyna
Tarnogród
Grodzisko Dolne
Radawa
Jasionka
Lancut
Jaroslaw
Pilzno
Radymno
Tuchów
Strzyżów
Harta

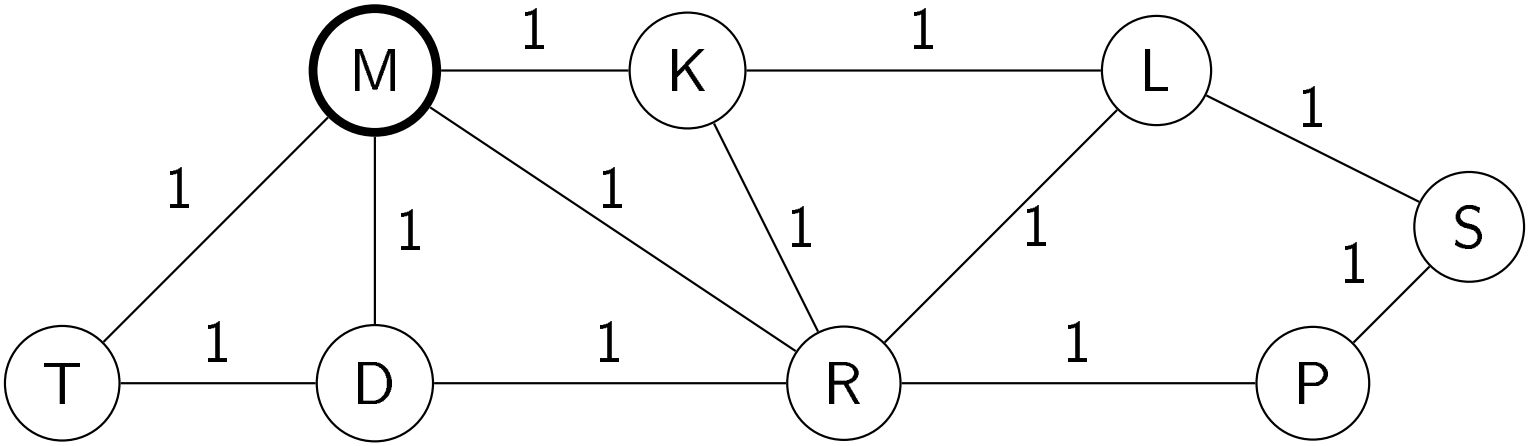
Graf z mapy



Uprośćmy



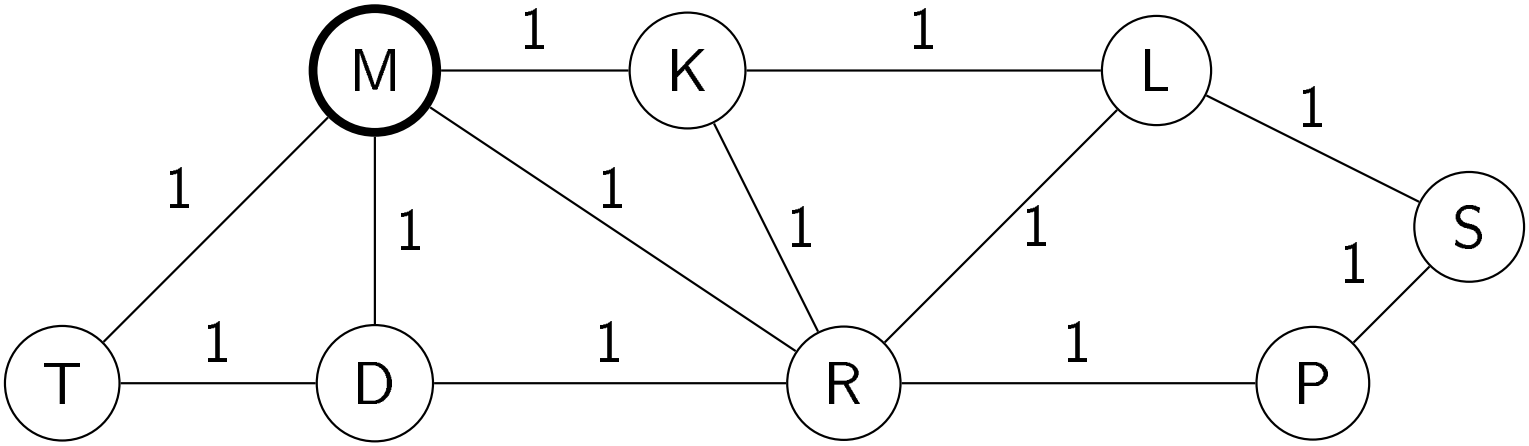
Algorytm BFS



wierzchołek	M	T	D	R	K	L	P	S
odległość								

Kolejka:

Algorytm BFS

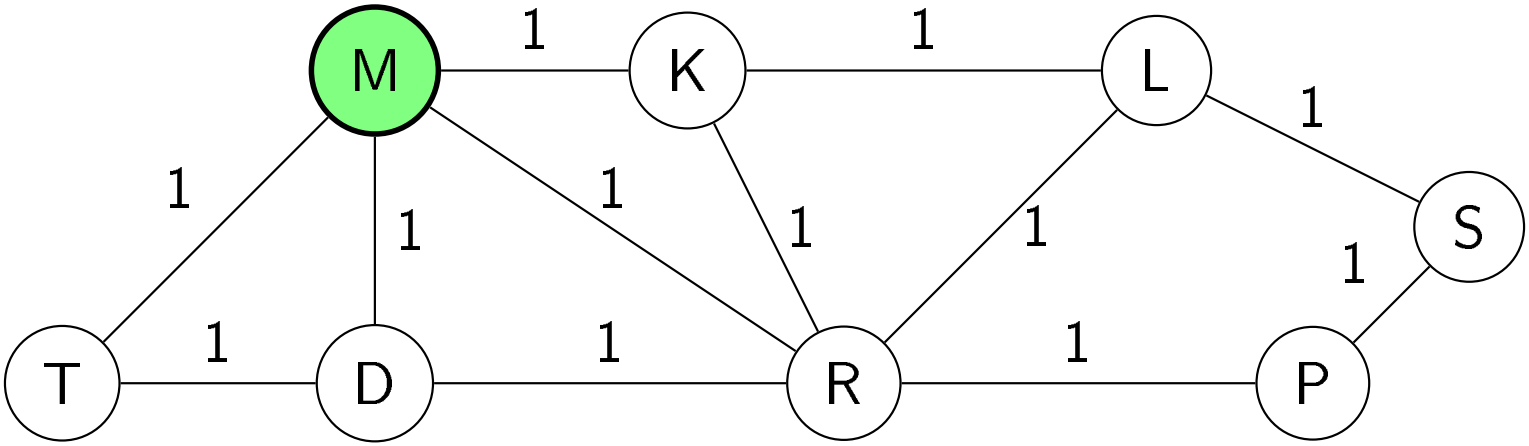


wierzchołek	M	T	D	R	K	L	P	S
odległość	0							

Kolejka:

M

Algorytm BFS

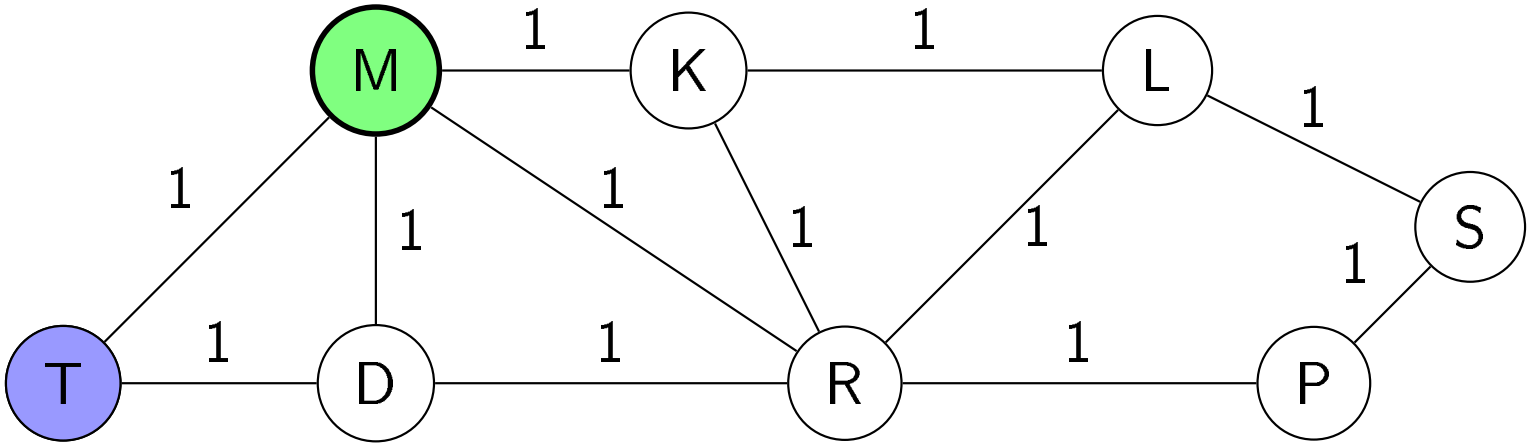


wierzchołek	M	T	D	R	K	L	P	S
odległość	0							

Kolejka:

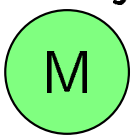
M

Algorytm BFS

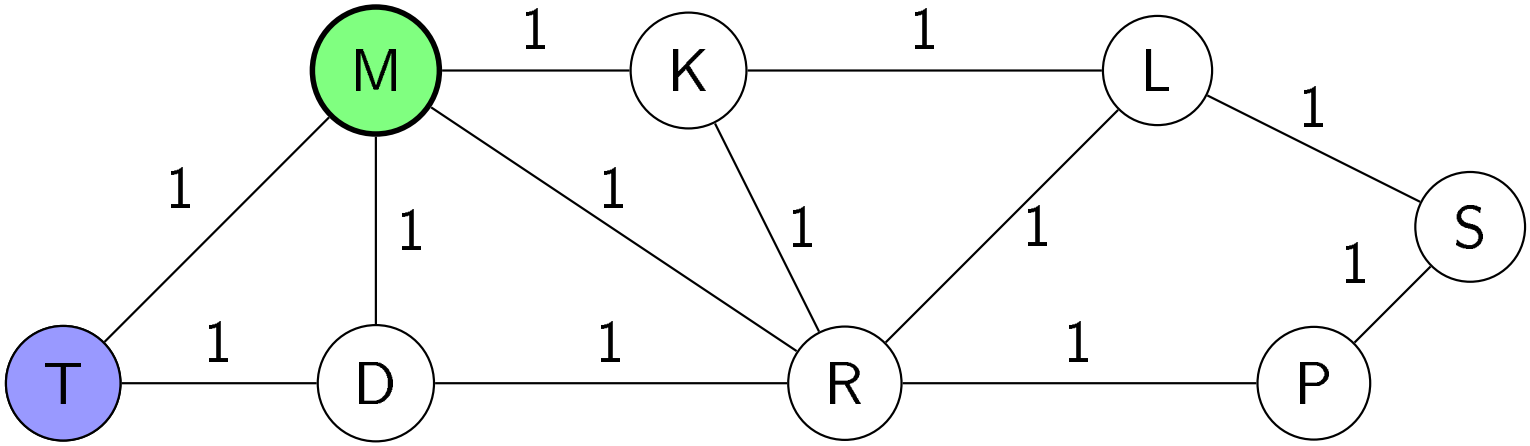


wierzchołek	M	T	D	R	K	L	P	S
odległość	0							

Kolejka:

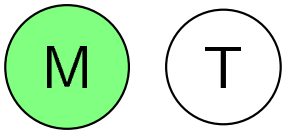


Algorytm BFS

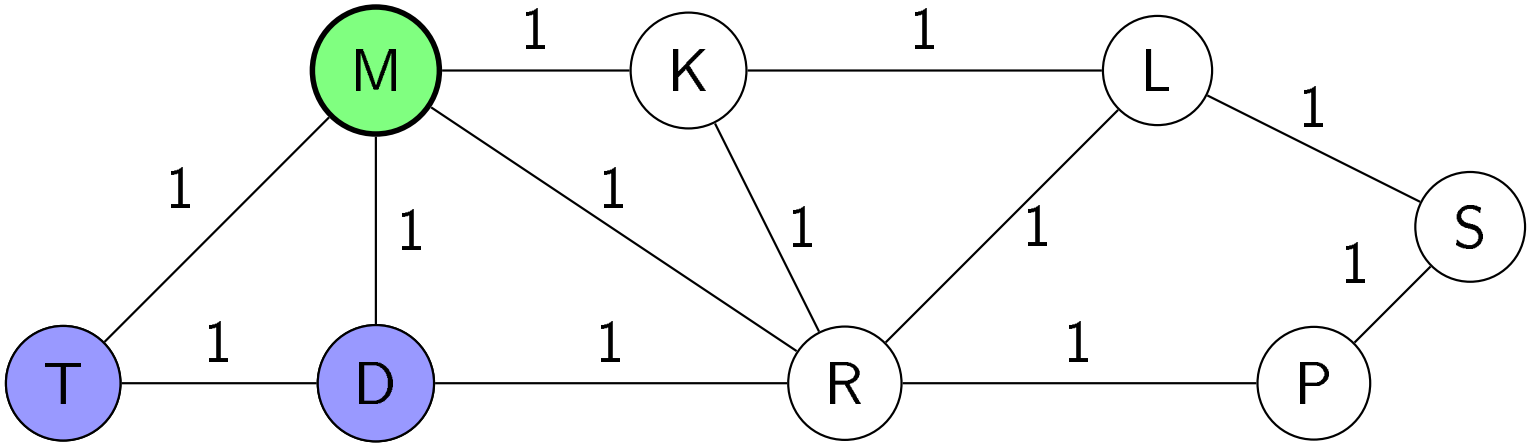


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1						

Kolejka:

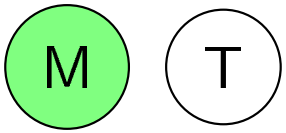


Algorytm BFS

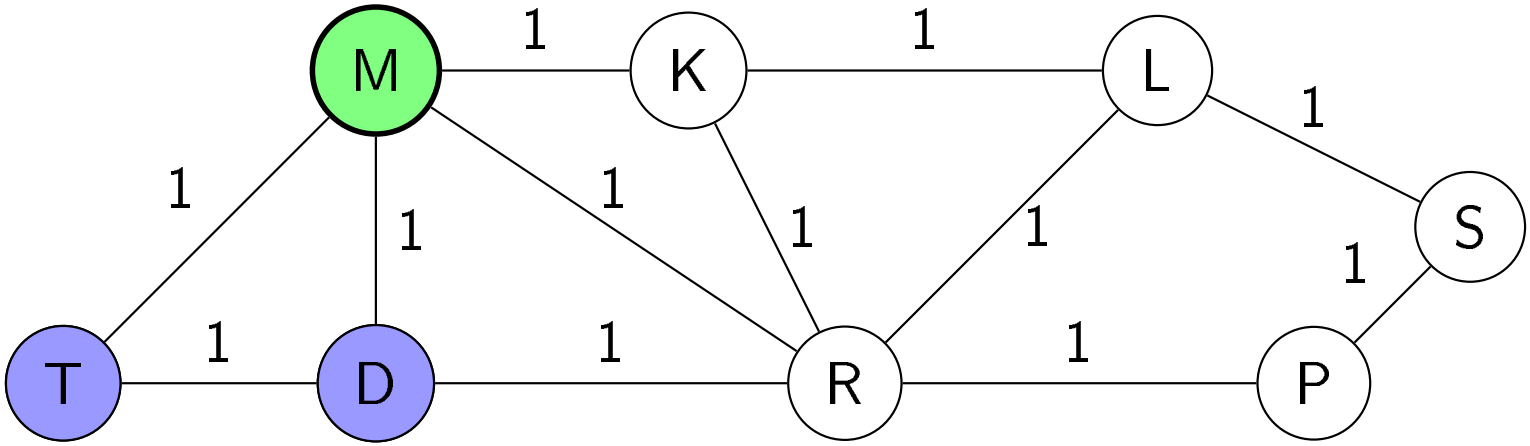


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1						

Kolejka:

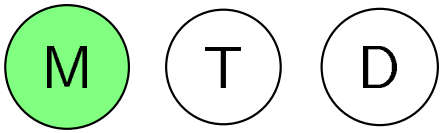


Algorytm BFS

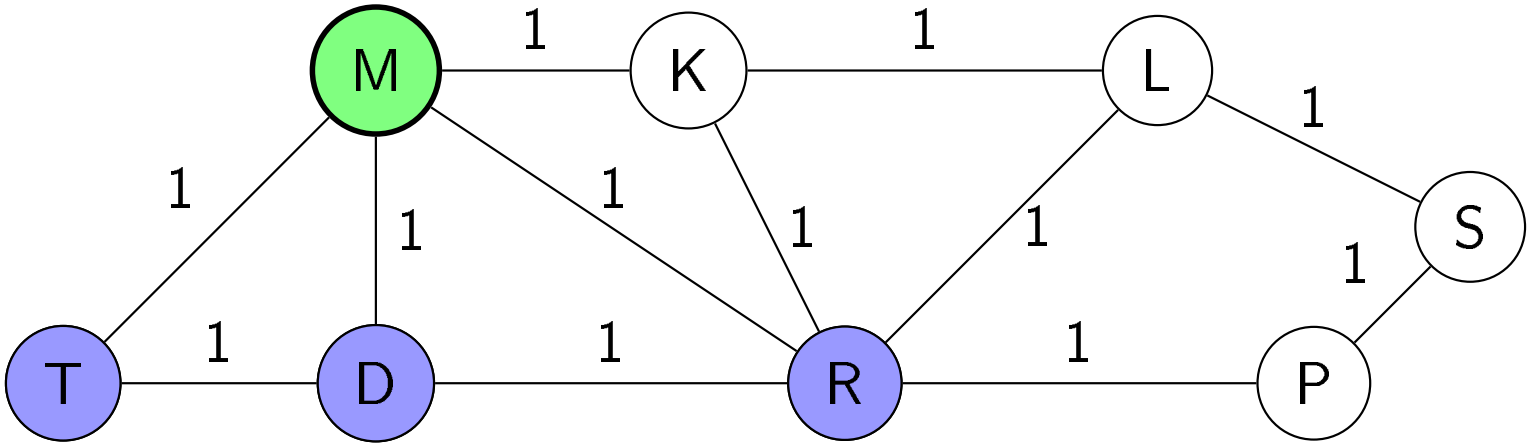


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1					

Kolejka:

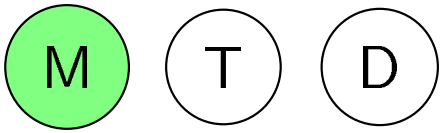


Algorytm BFS

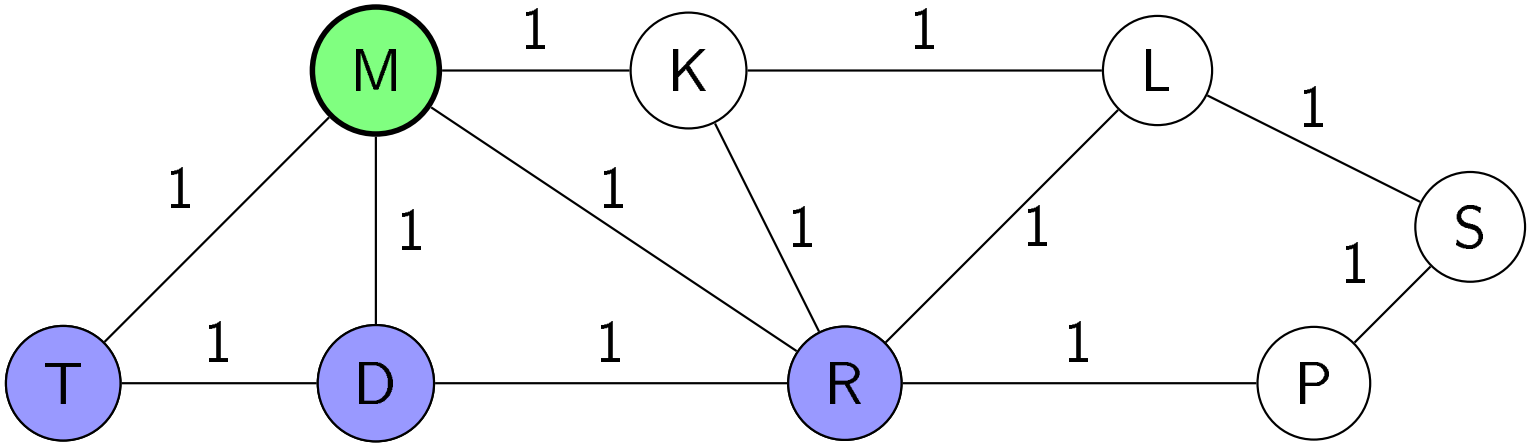


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1					

Kolejka:

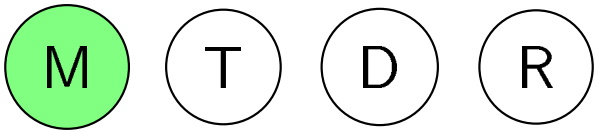


Algorytm BFS

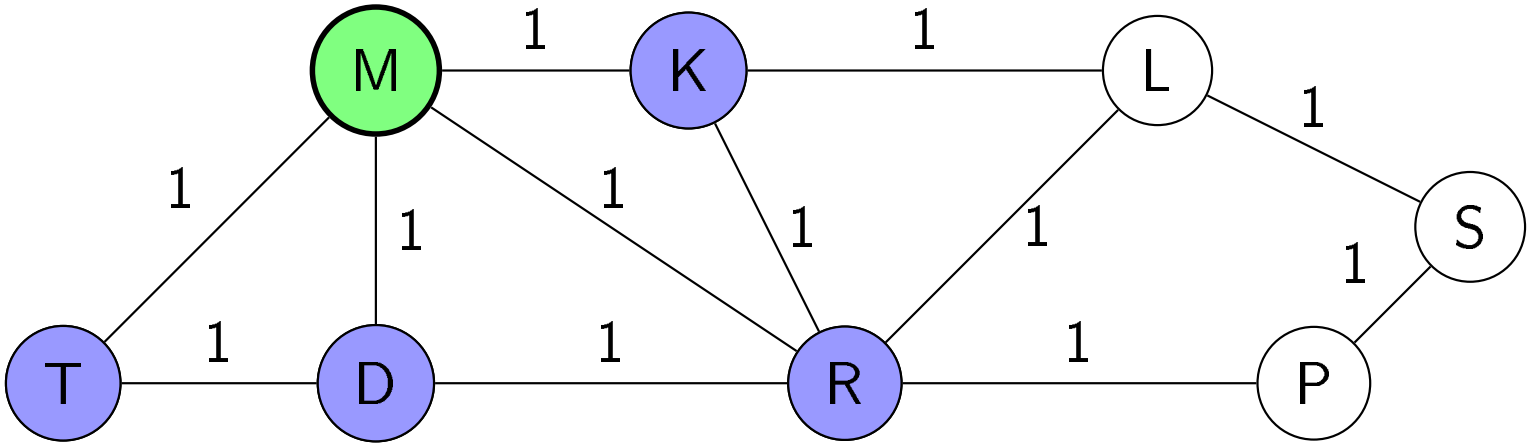


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1				

Kolejka:

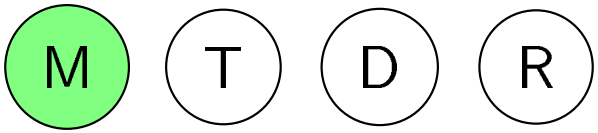


Algorytm BFS

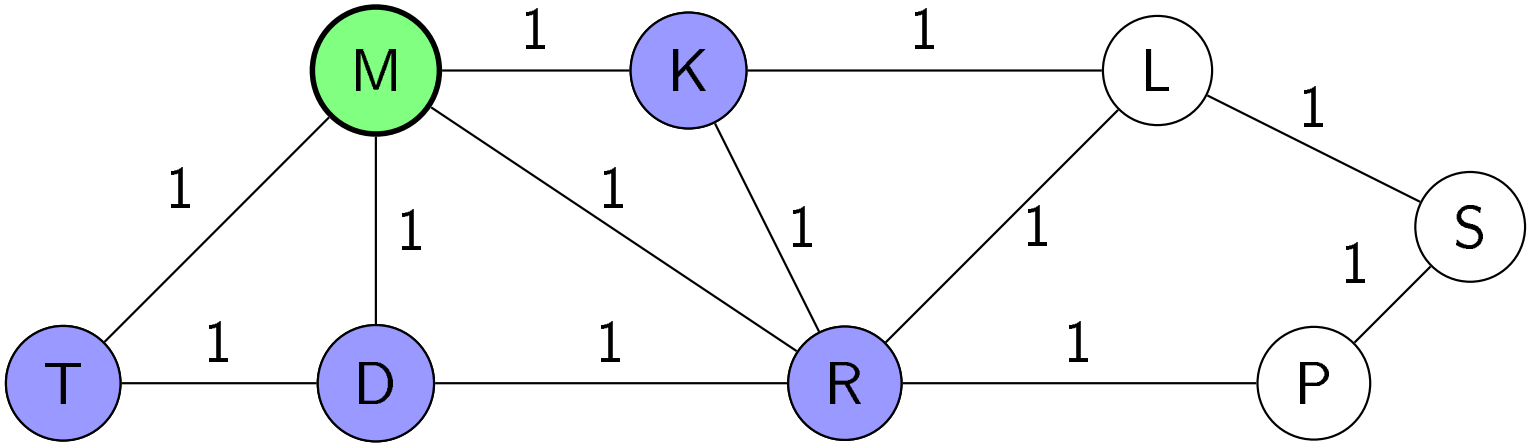


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1				

Kolejka:

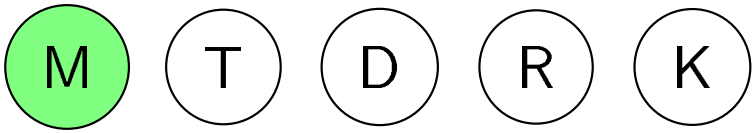


Algorytm BFS

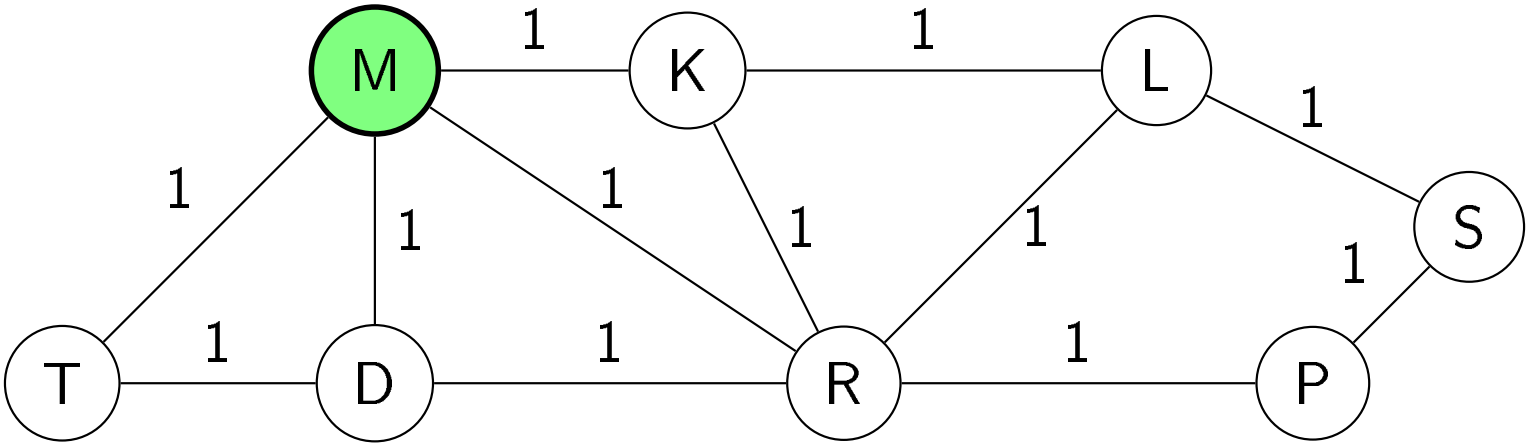


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

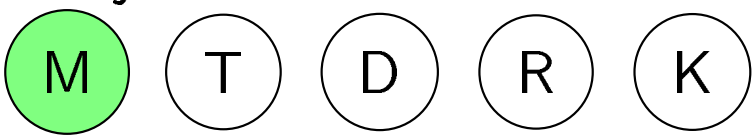


Algorytm BFS

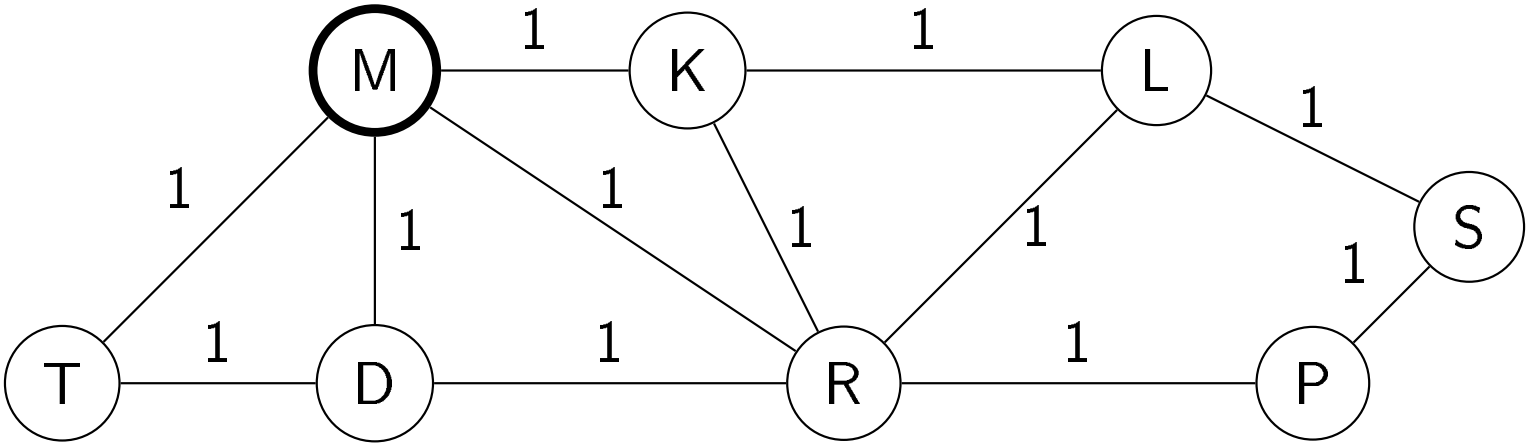


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

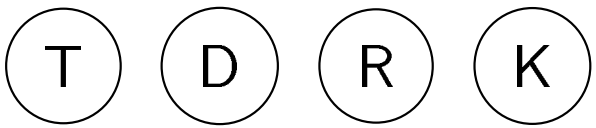


Algorytm BFS

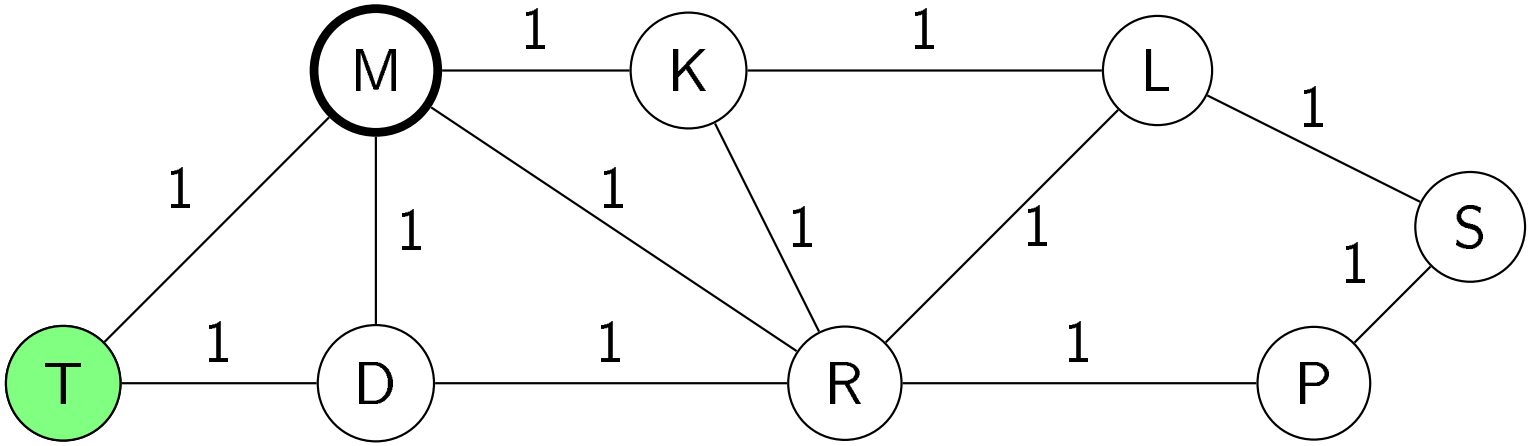


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

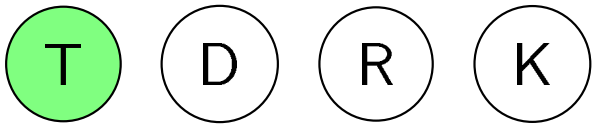


Algorytm BFS

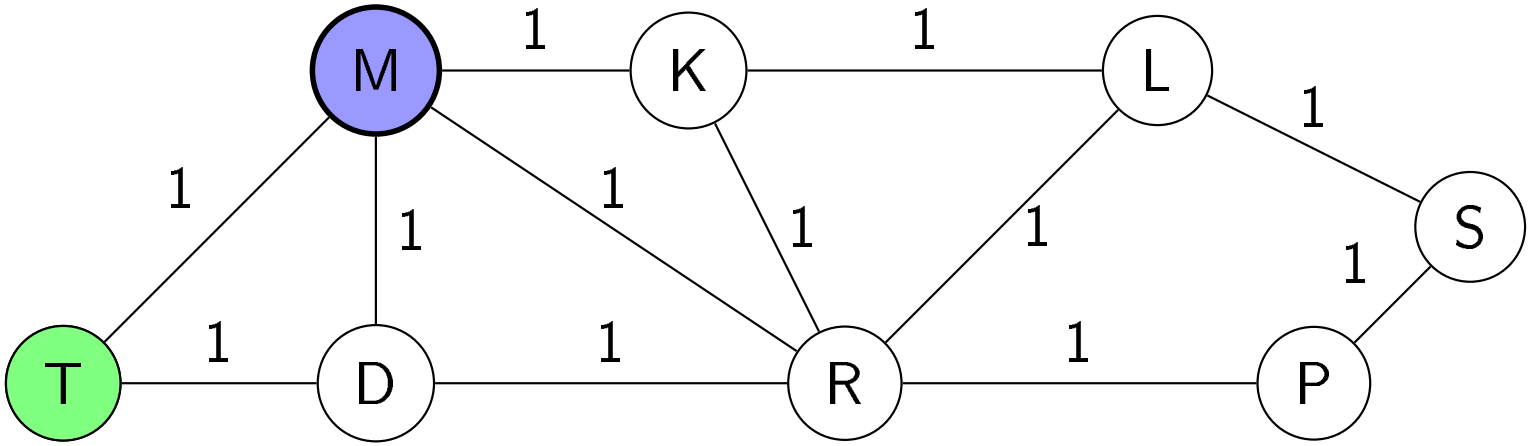


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

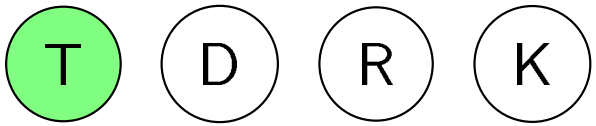


Algorytm BFS

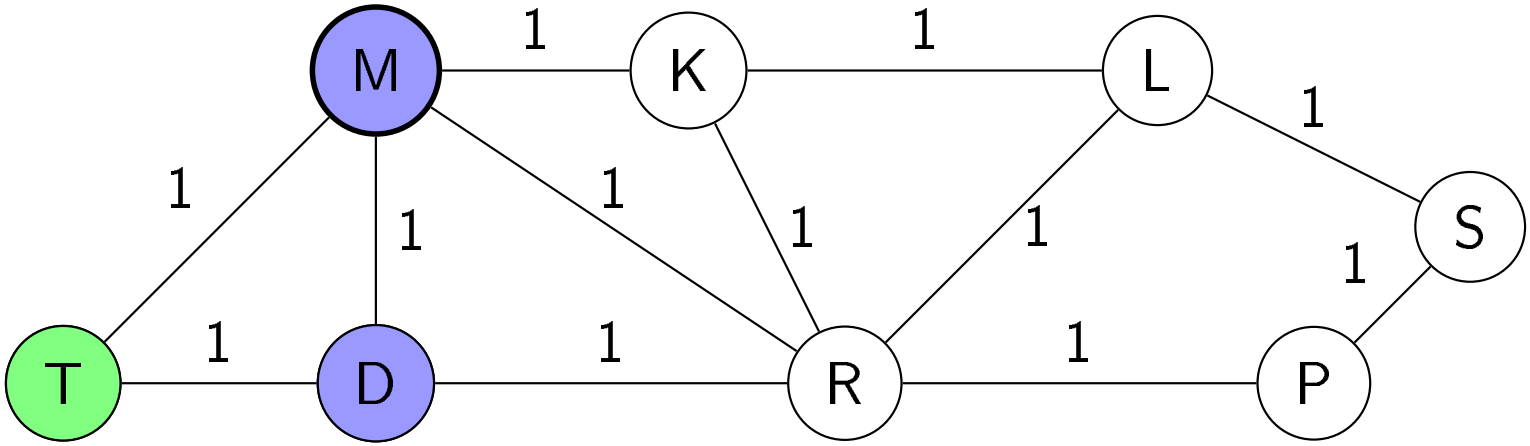


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

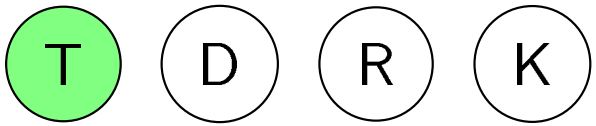


Algorytm BFS

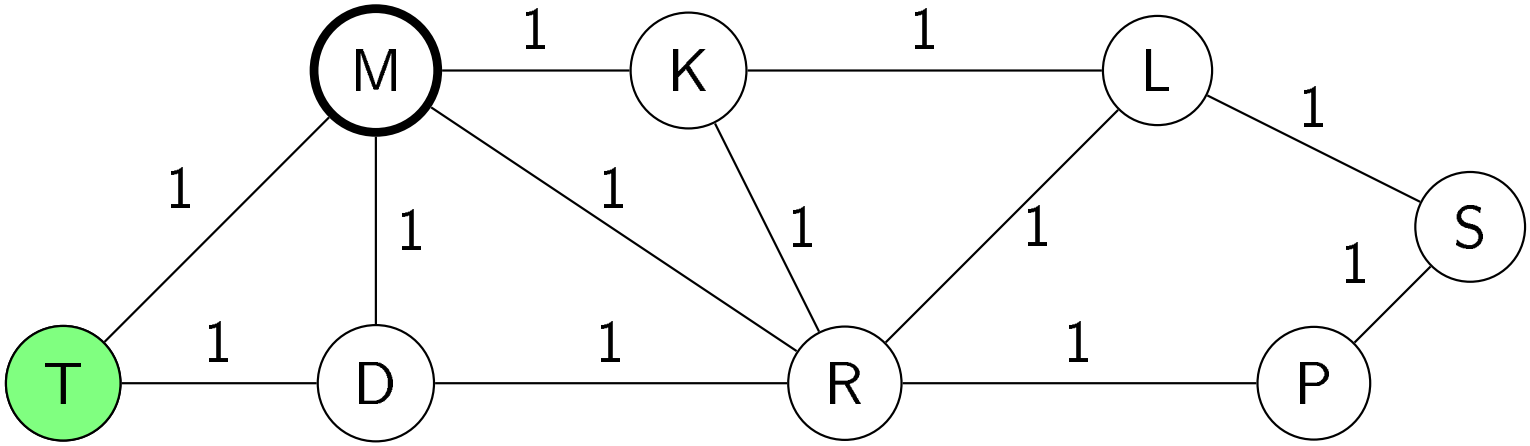


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

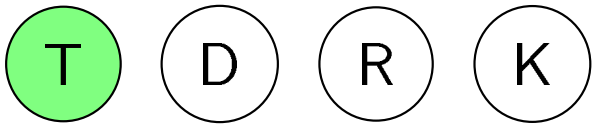


Algorytm BFS

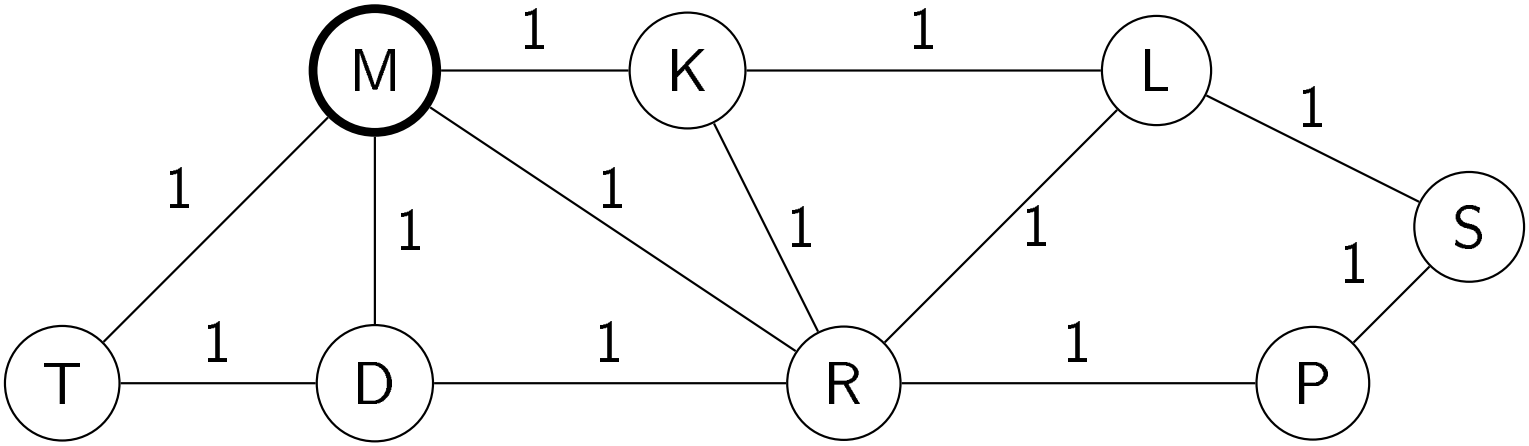


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

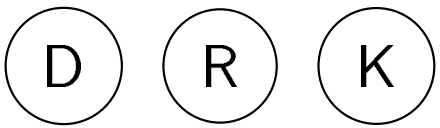


Algorytm BFS

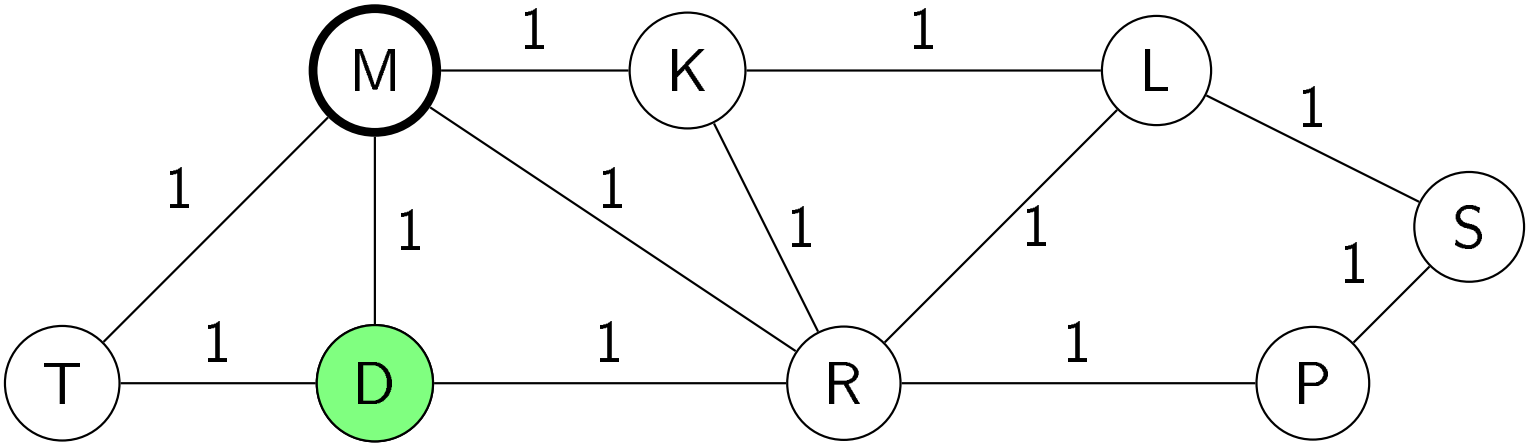


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

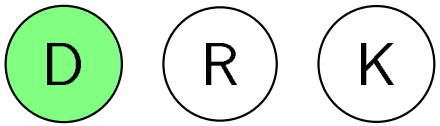


Algorytm BFS

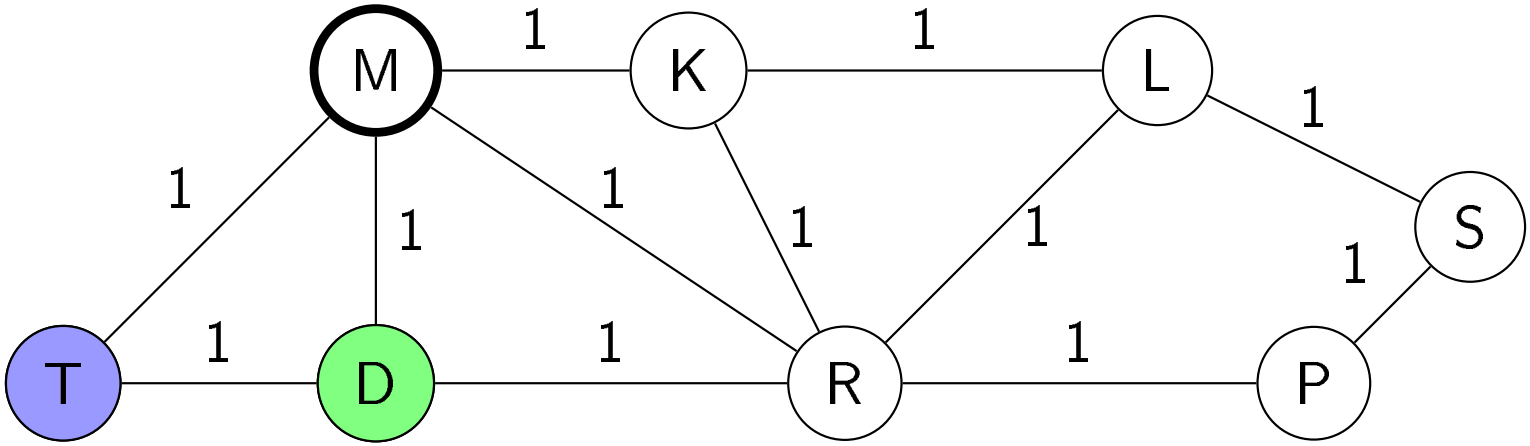


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

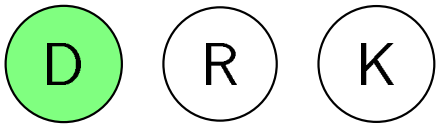


Algorytm BFS

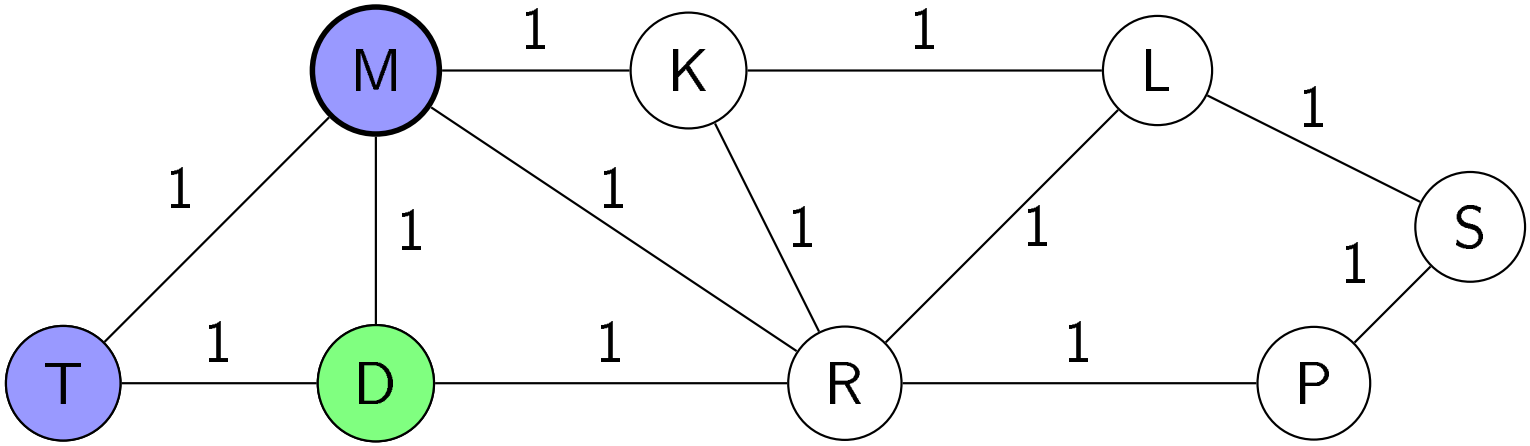


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

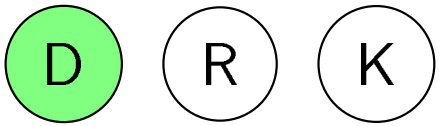


Algorytm BFS

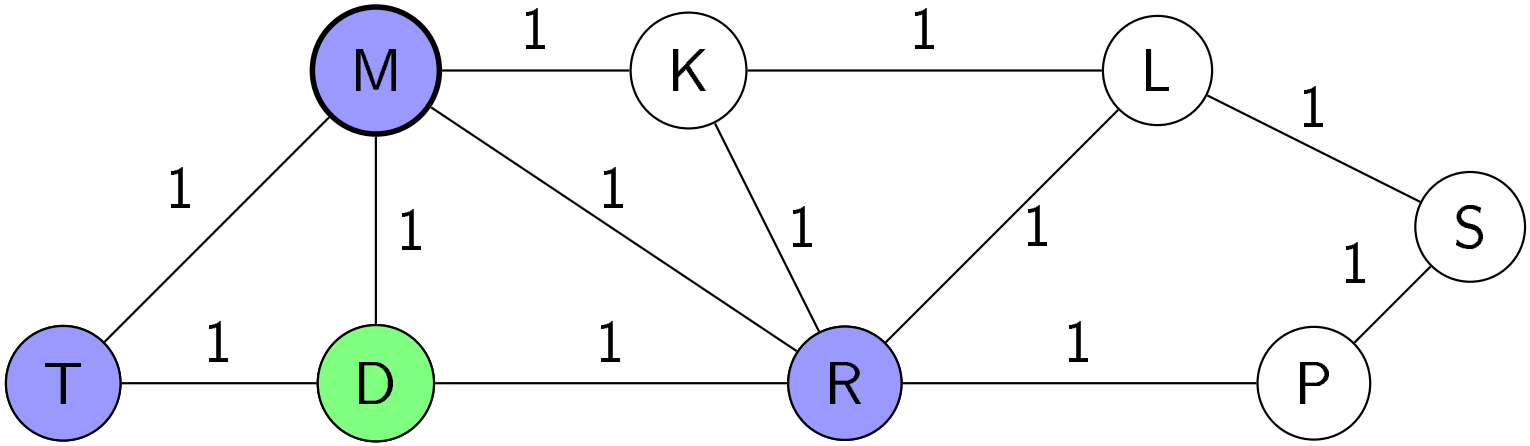


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

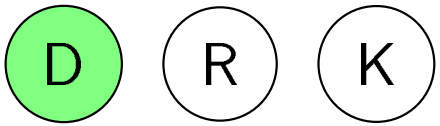


Algorytm BFS

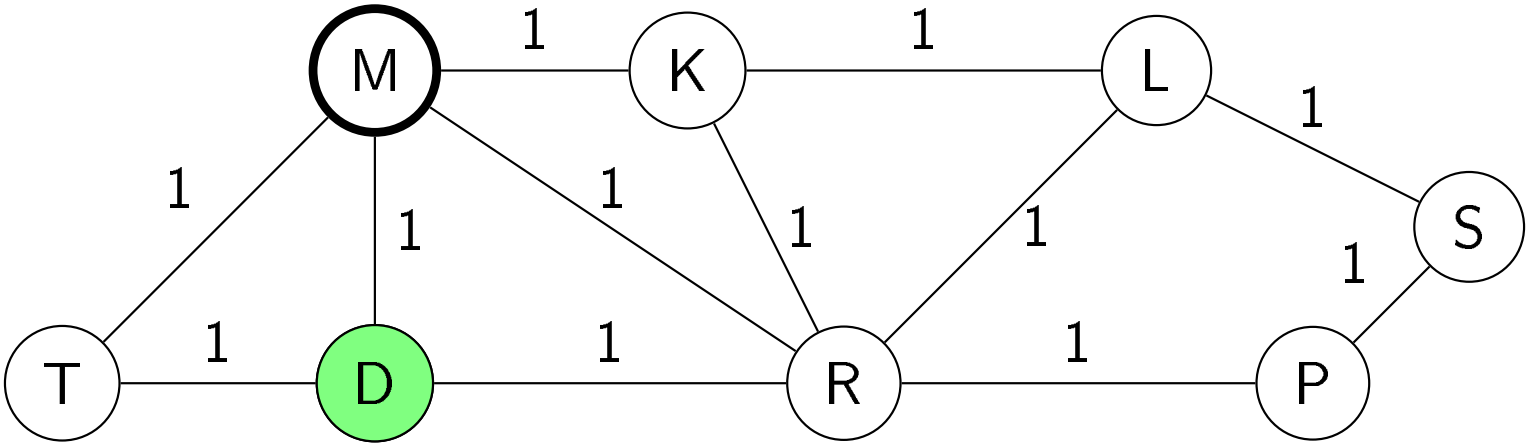


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

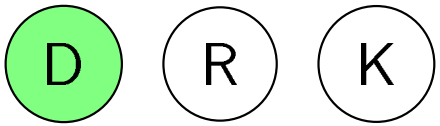


Algorytm BFS

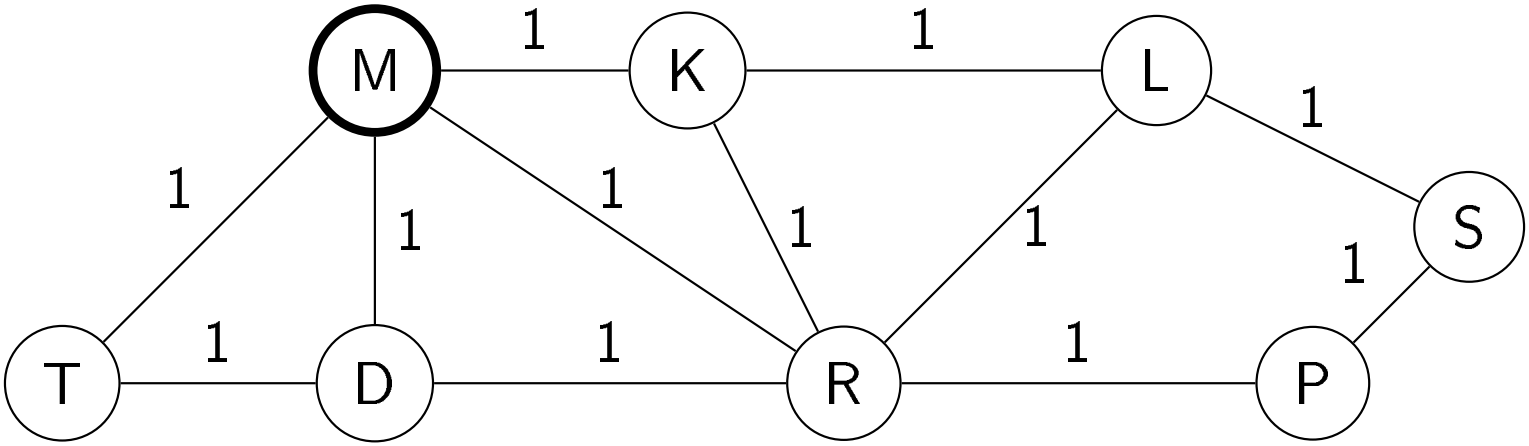


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

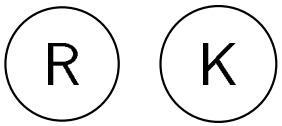


Algorytm BFS

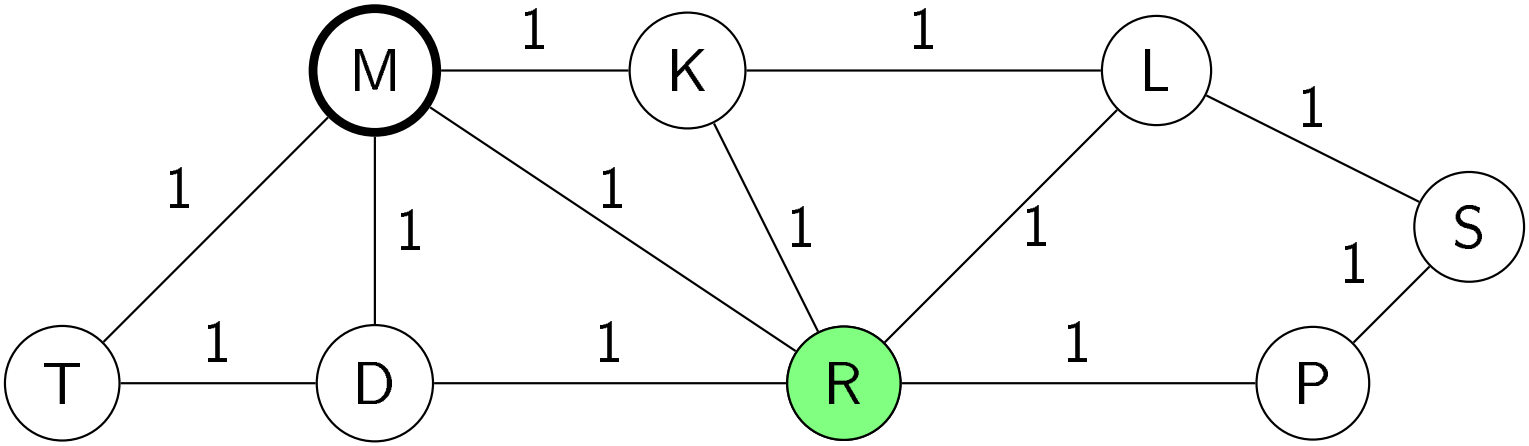


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

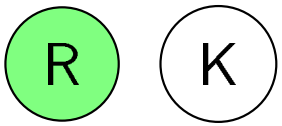


Algorytm BFS

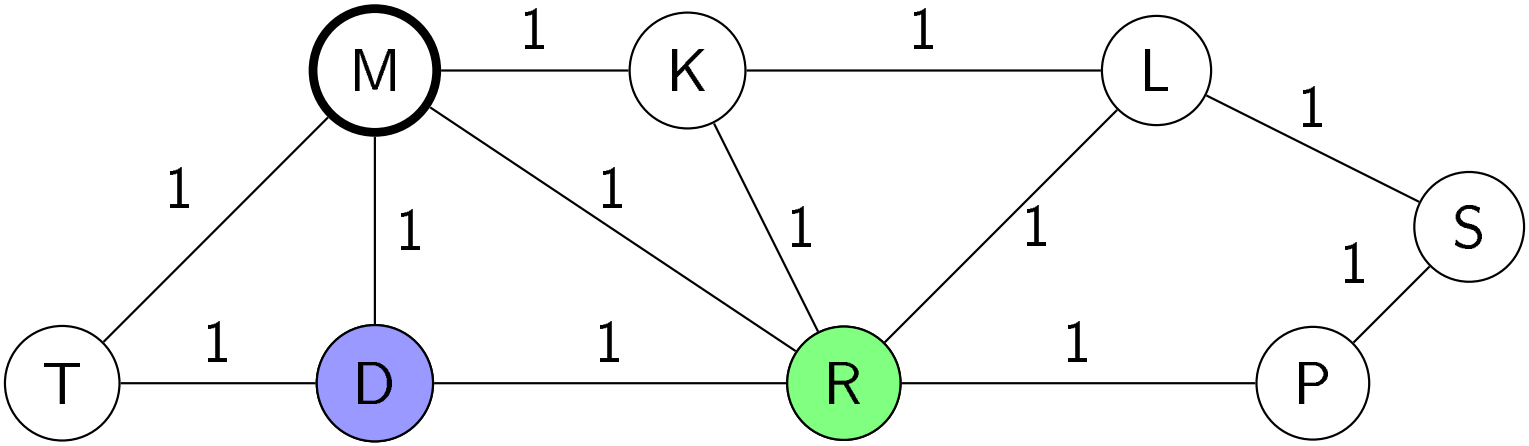


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

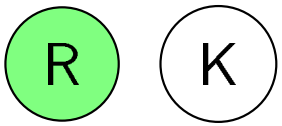


Algorytm BFS

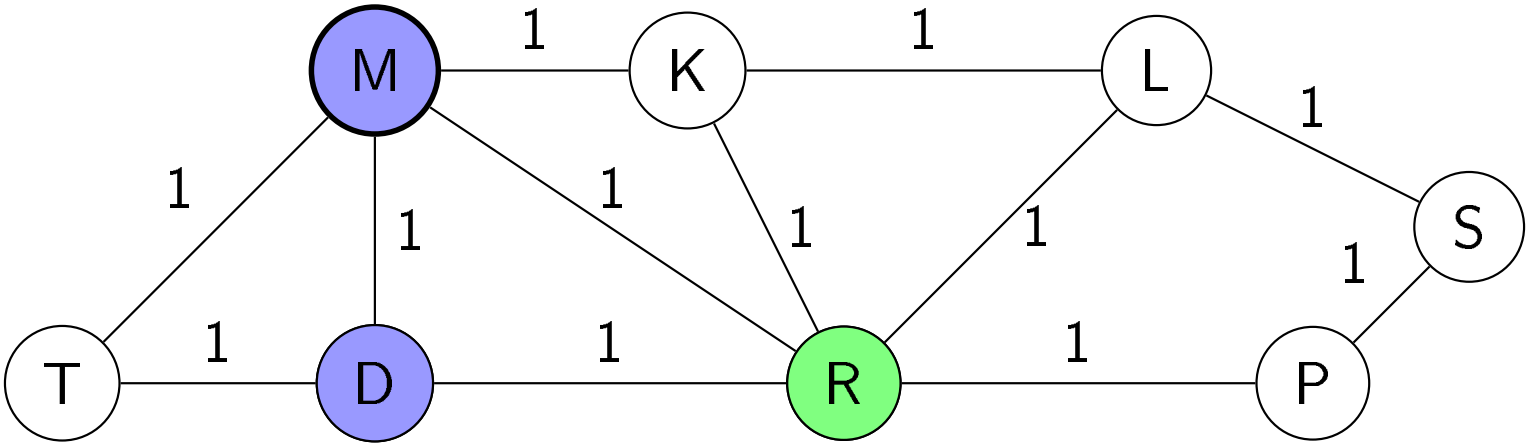


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

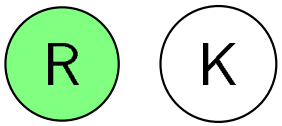


Algorytm BFS

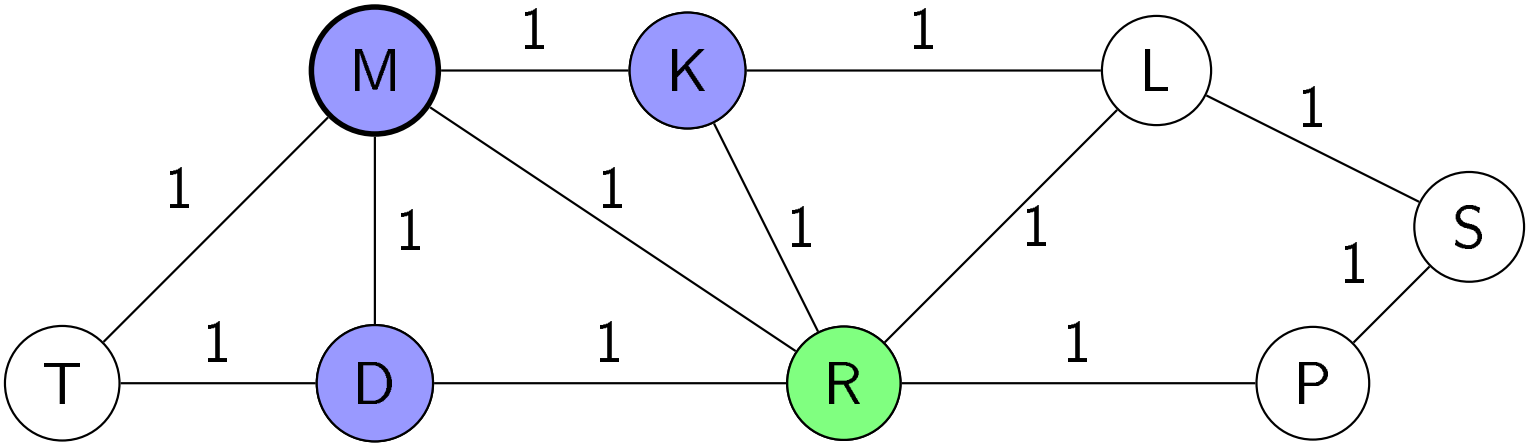


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

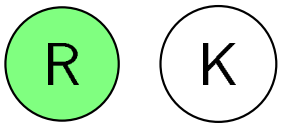


Algorytm BFS

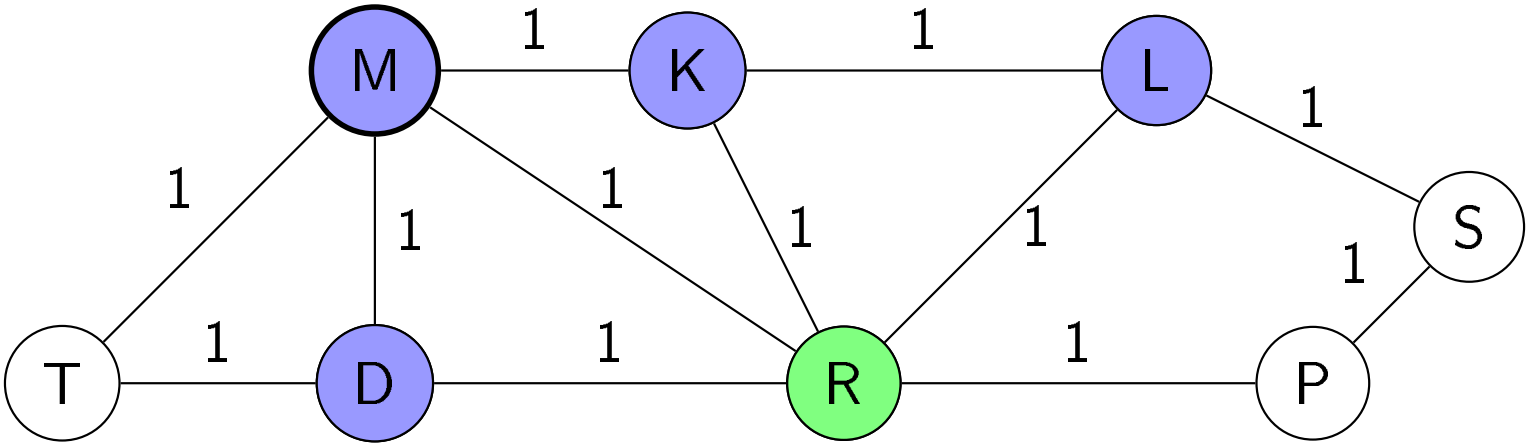


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

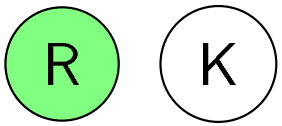


Algorytm BFS

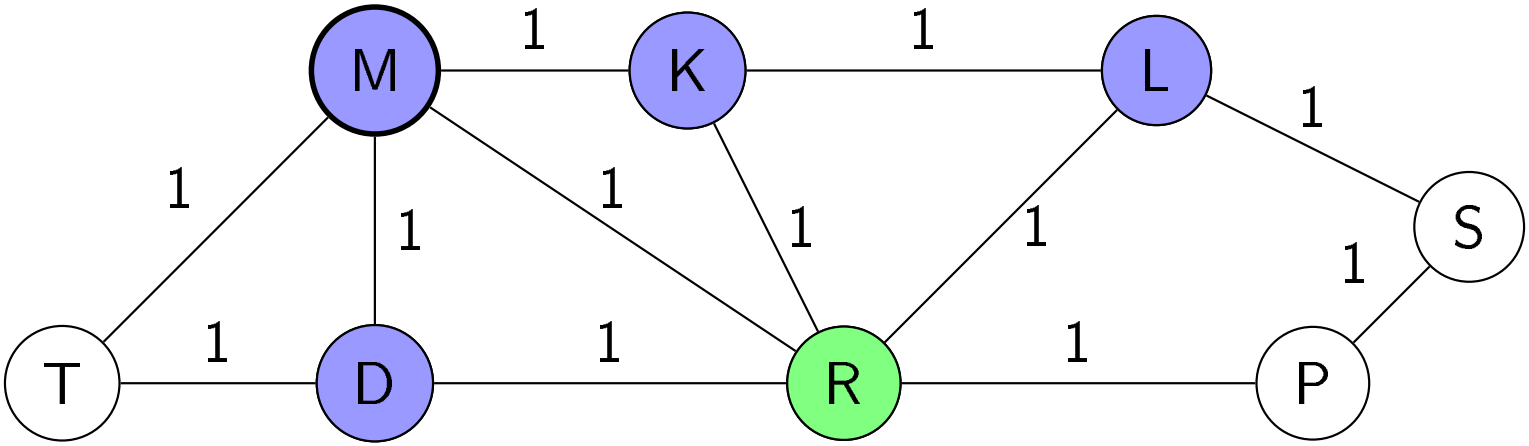


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1			

Kolejka:

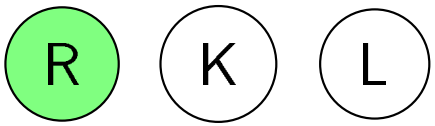


Algorytm BFS

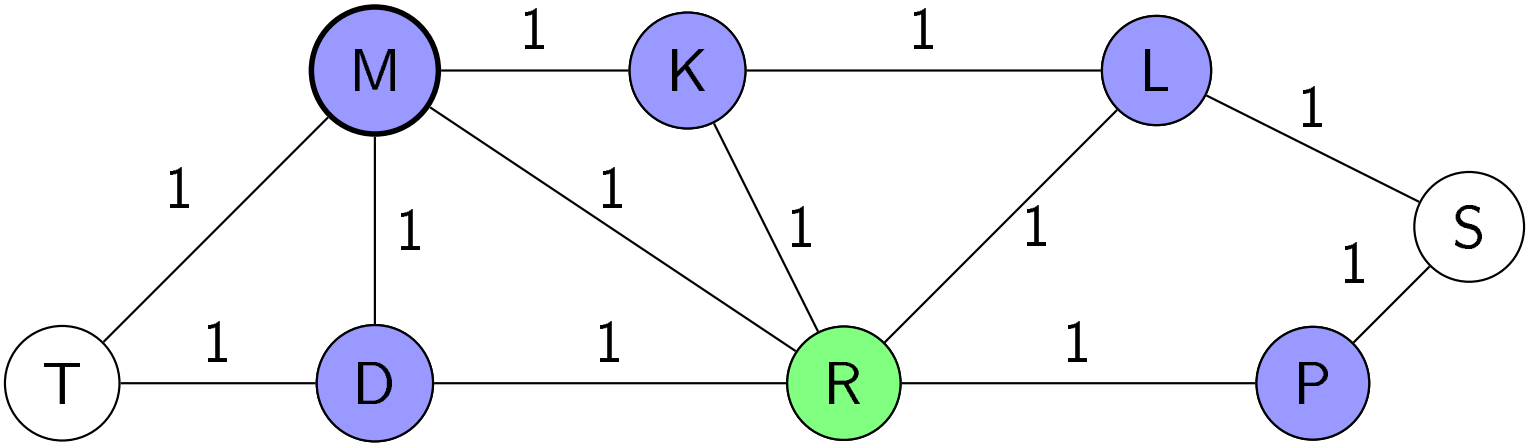


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2		

Kolejka:

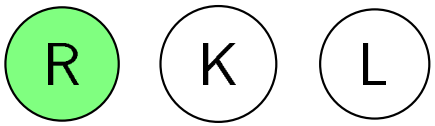


Algorytm BFS

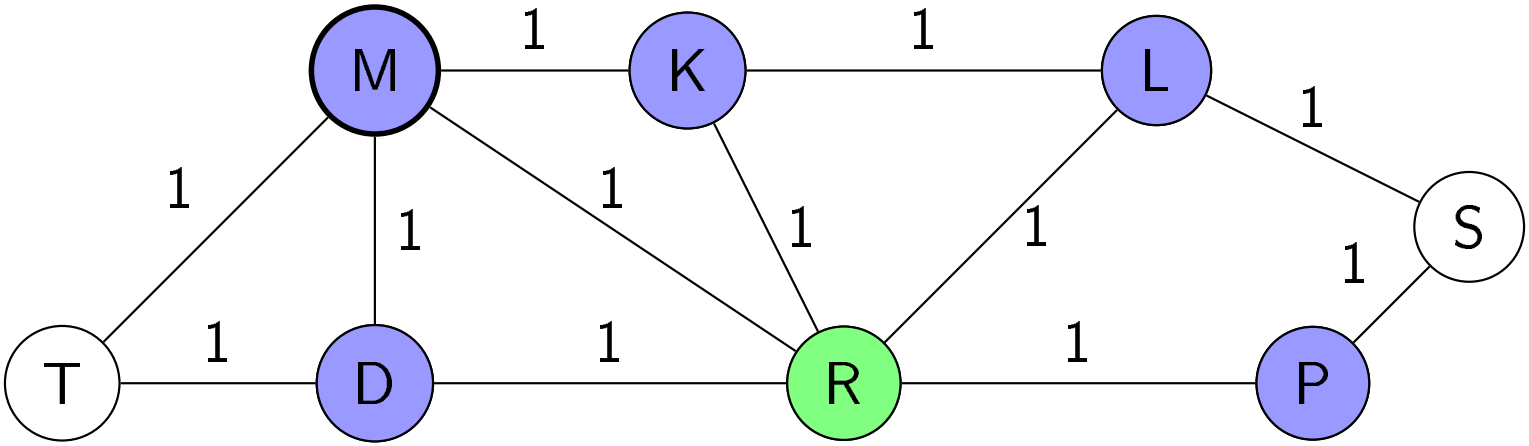


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2		

Kolejka:

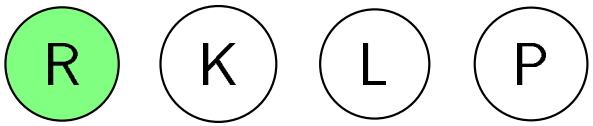


Algorytm BFS

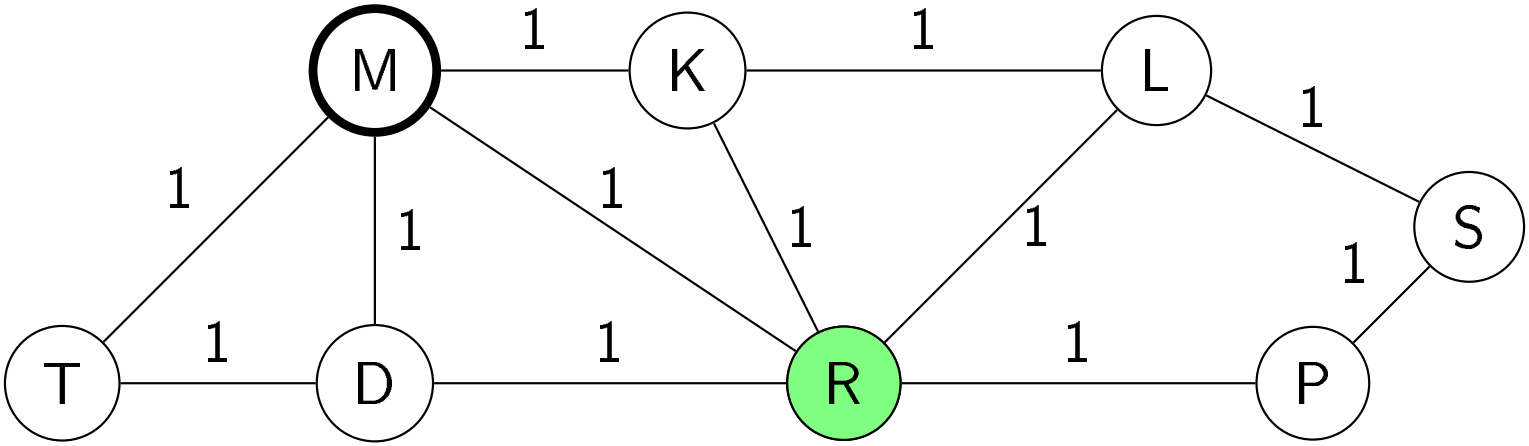


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

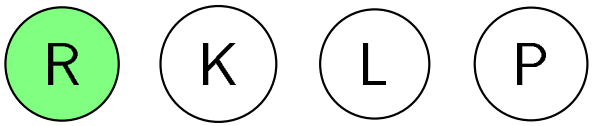


Algorytm BFS

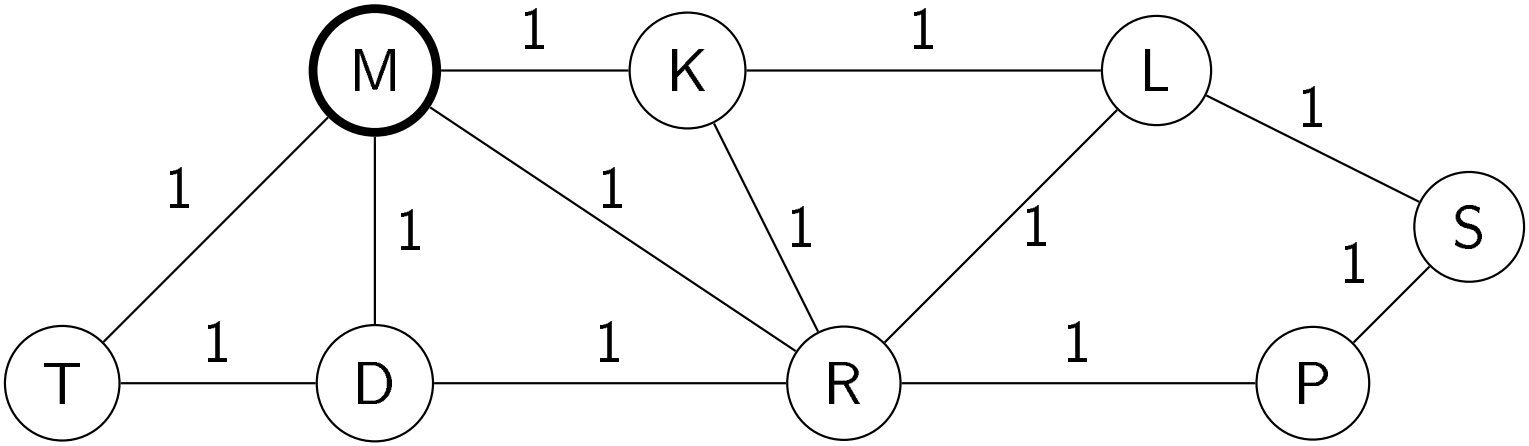


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

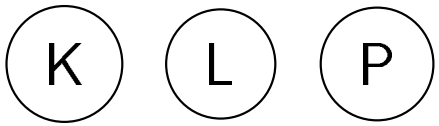


Algorytm BFS

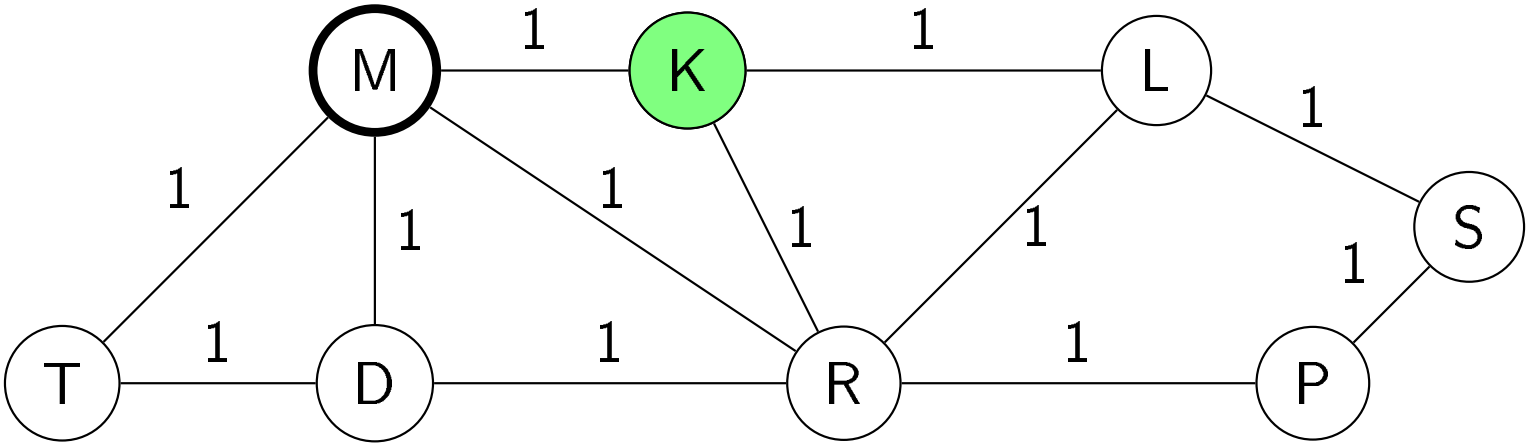


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

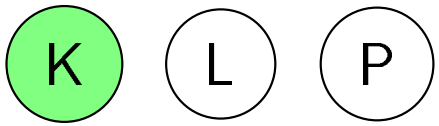


Algorytm BFS

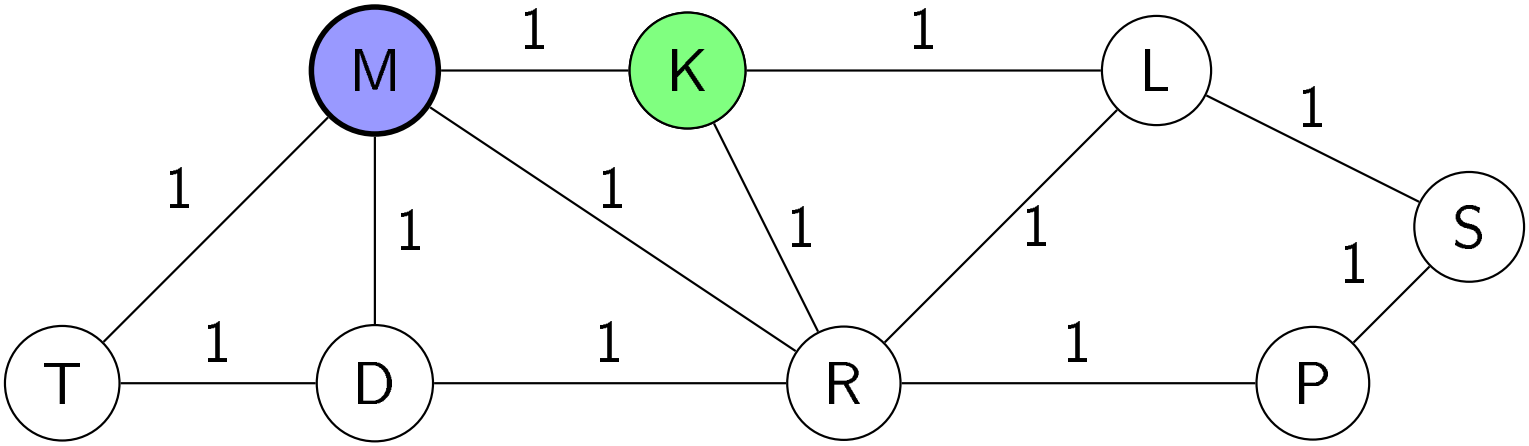


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

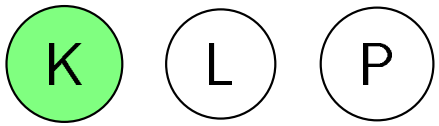


Algorytm BFS

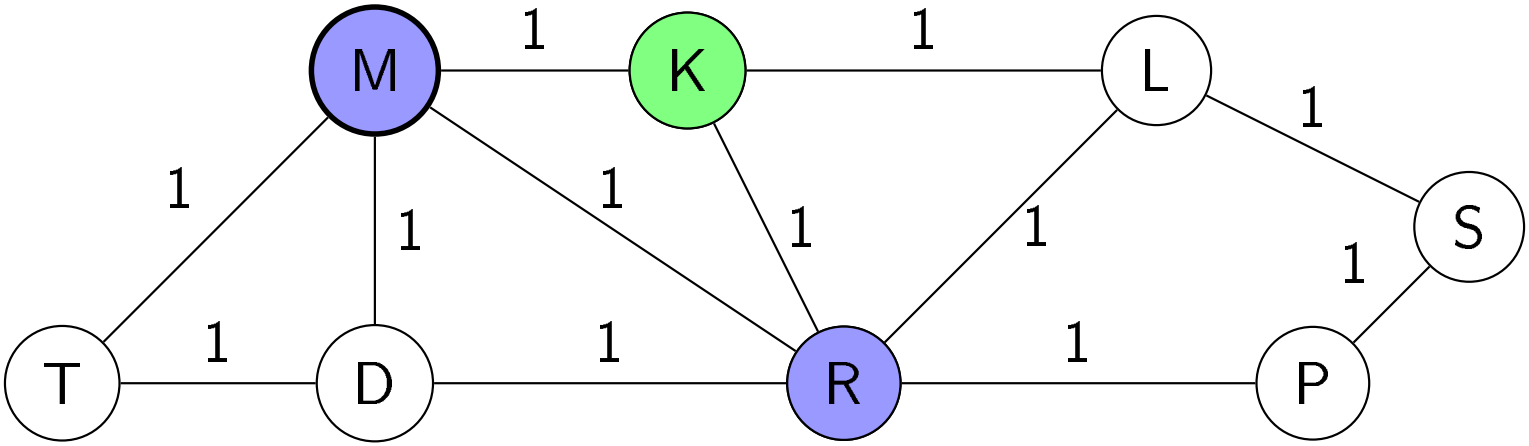


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

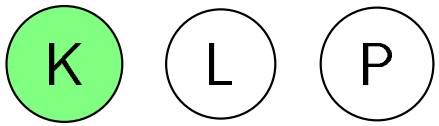


Algorytm BFS

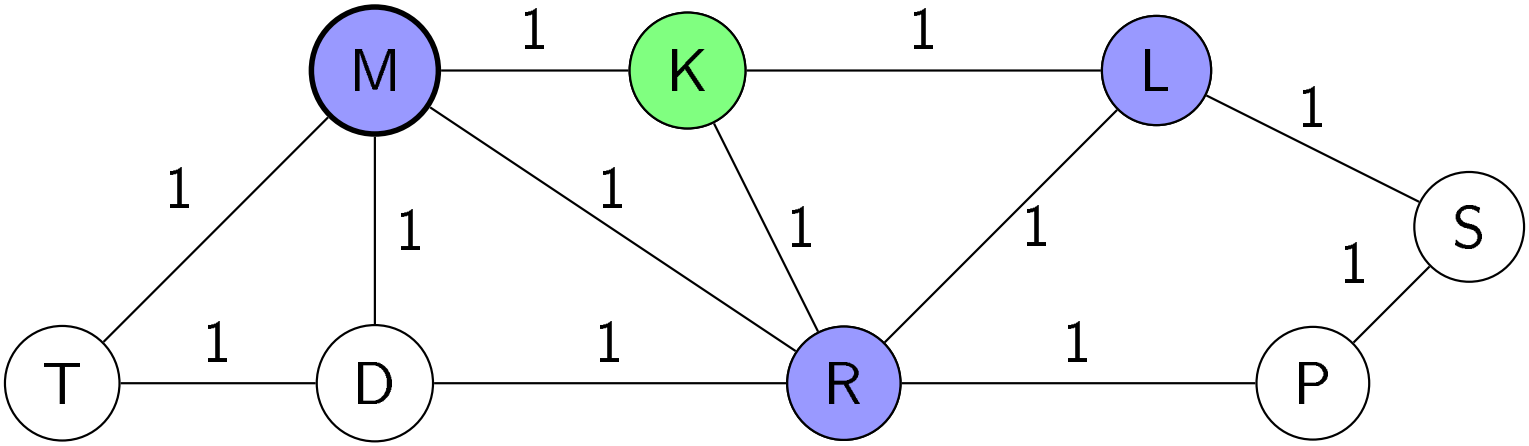


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

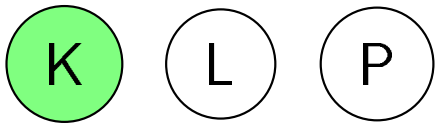


Algorytm BFS

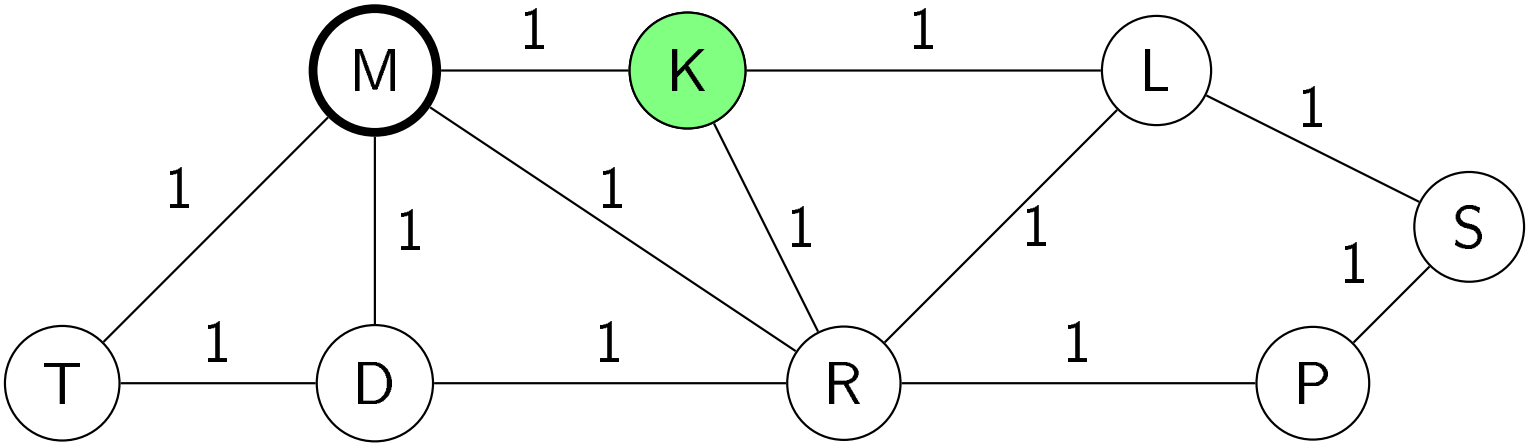


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

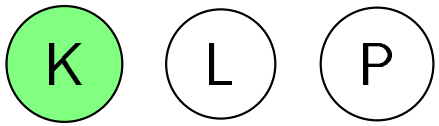


Algorytm BFS

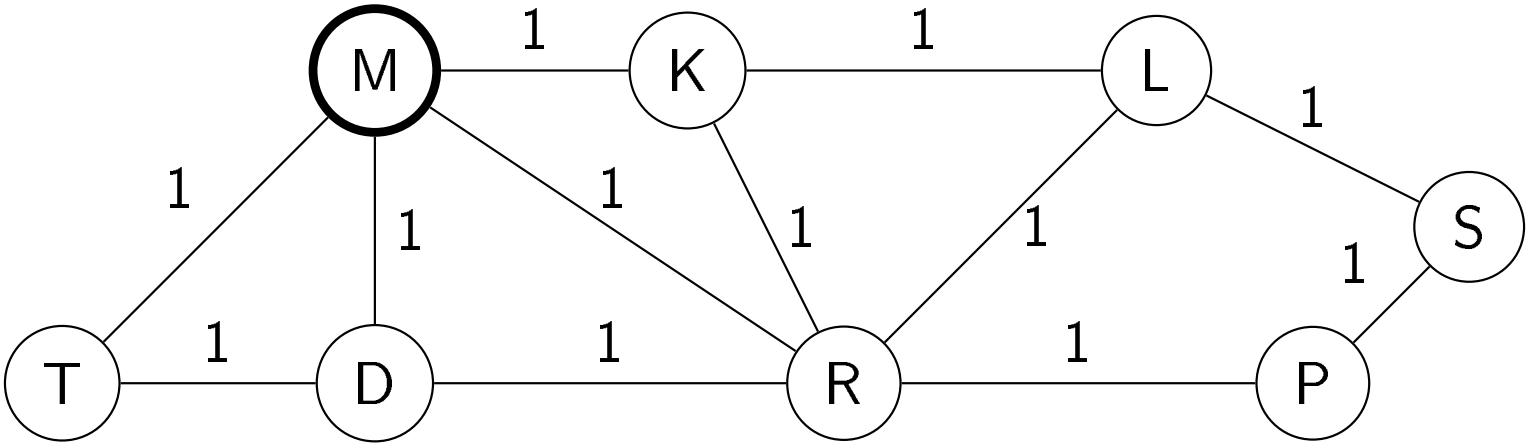


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

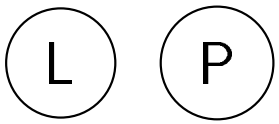


Algorytm BFS

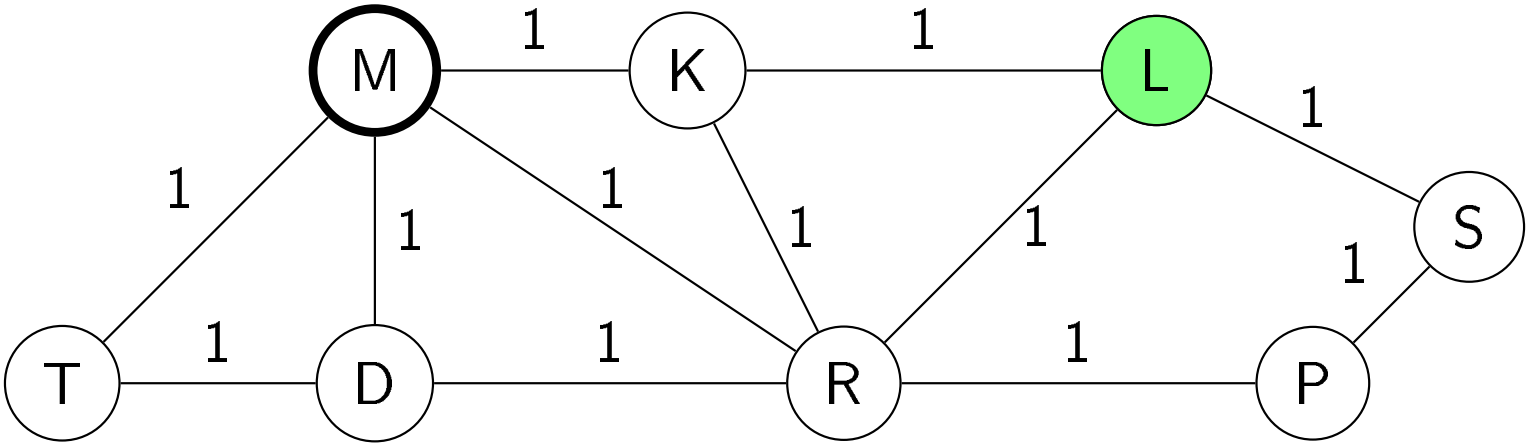


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

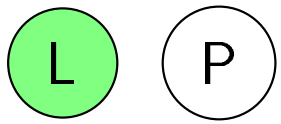


Algorytm BFS

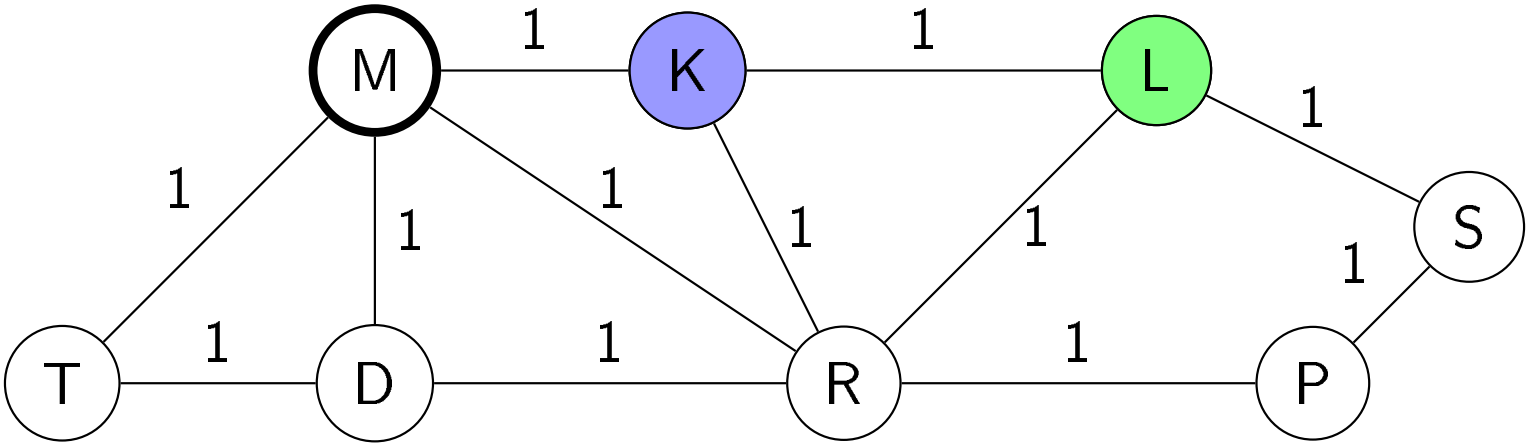


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

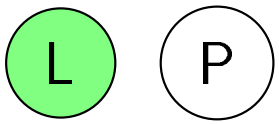


Algorytm BFS

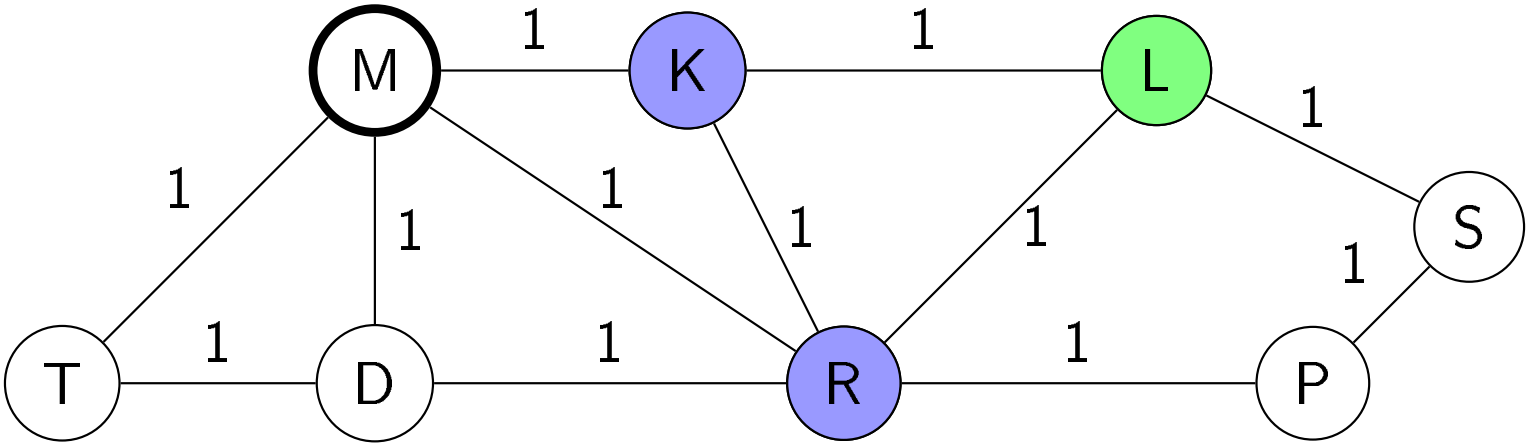


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

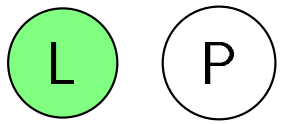


Algorytm BFS

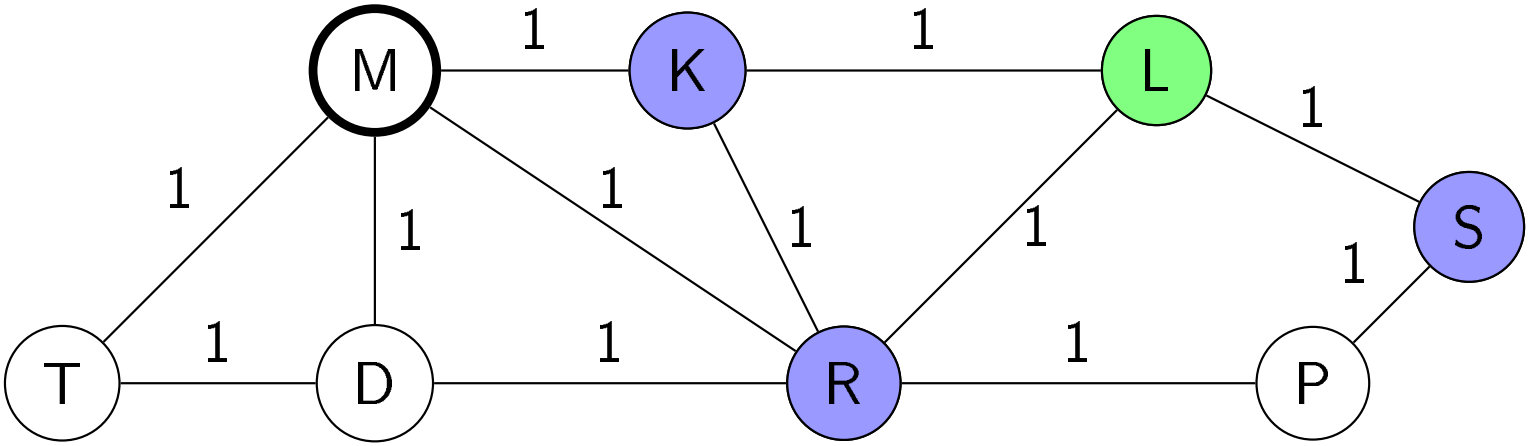


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

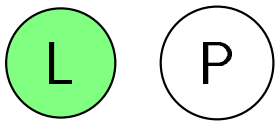


Algorytm BFS

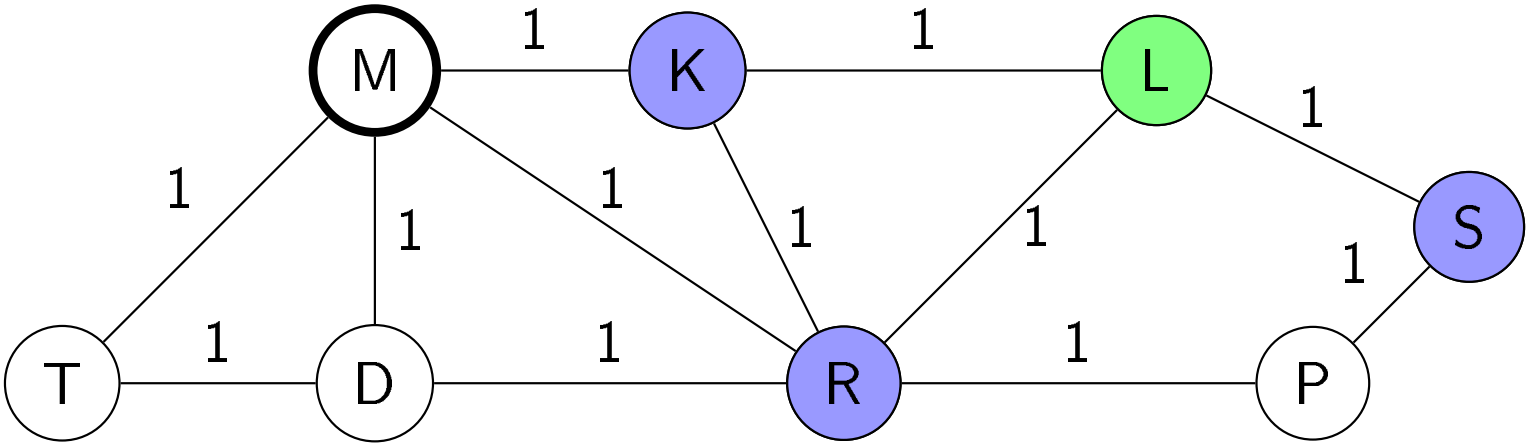


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	

Kolejka:

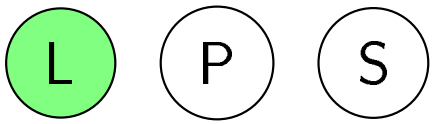


Algorytm BFS

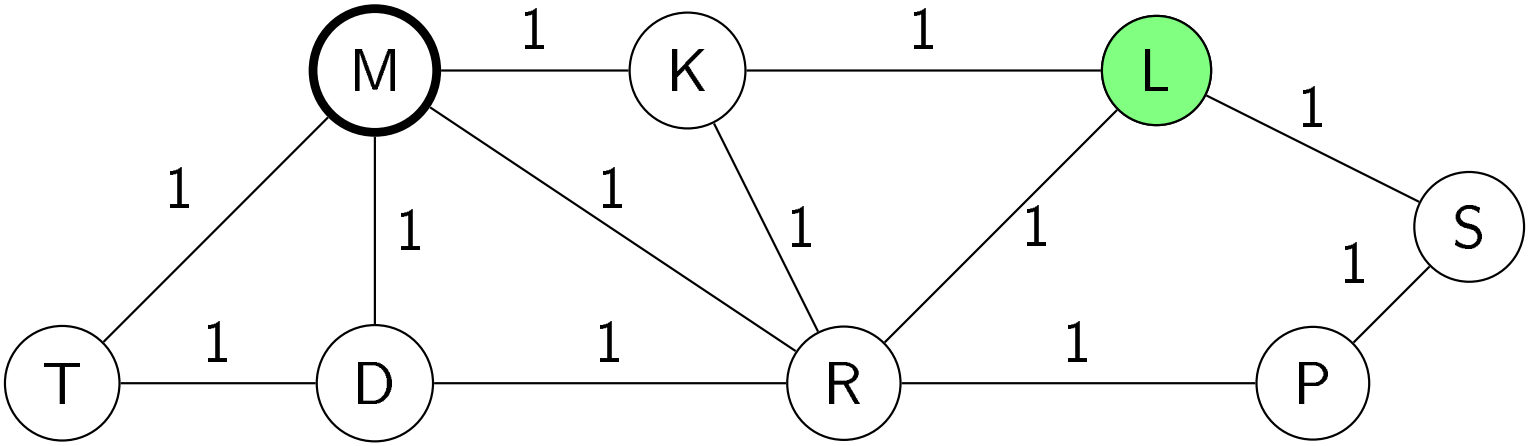


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

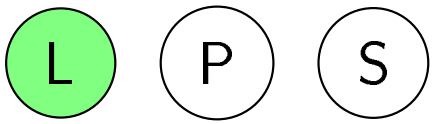


Algorytm BFS

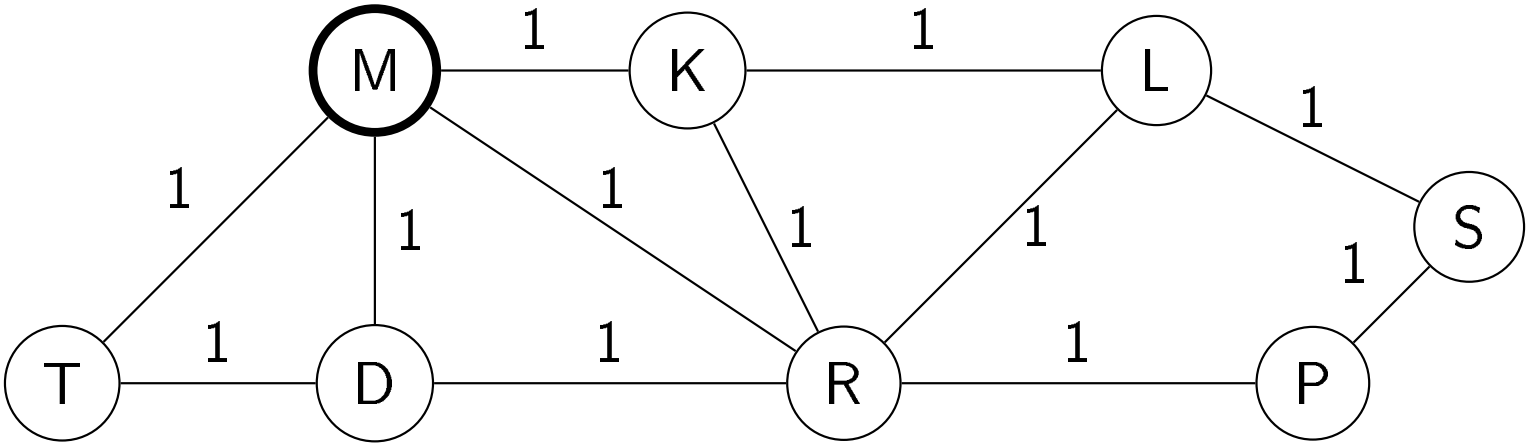


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

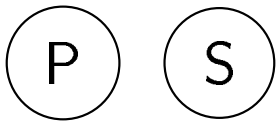


Algorytm BFS

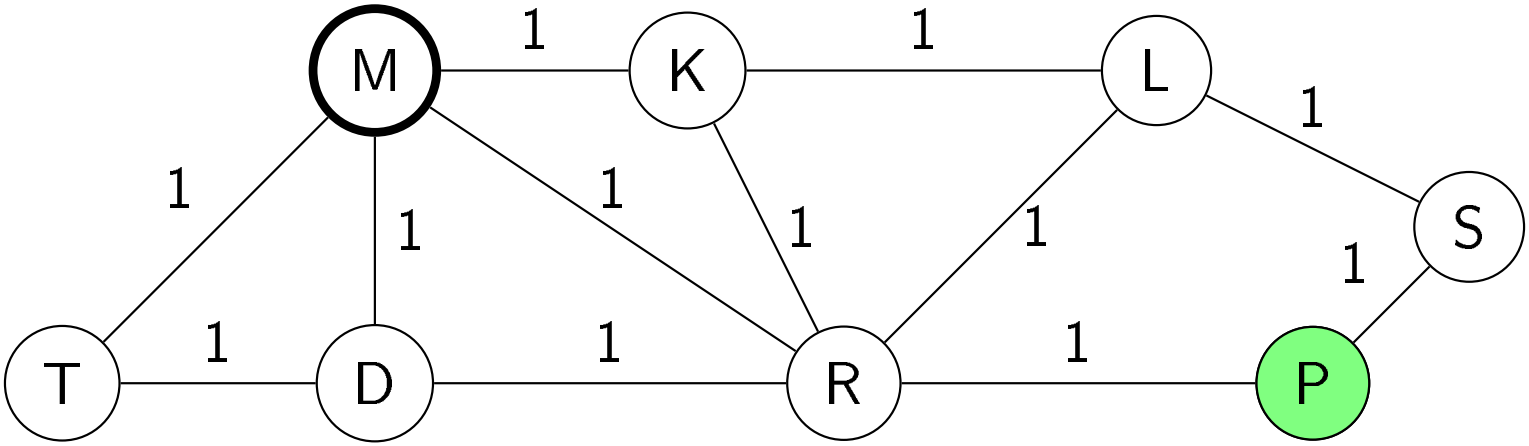


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

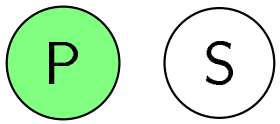


Algorytm BFS

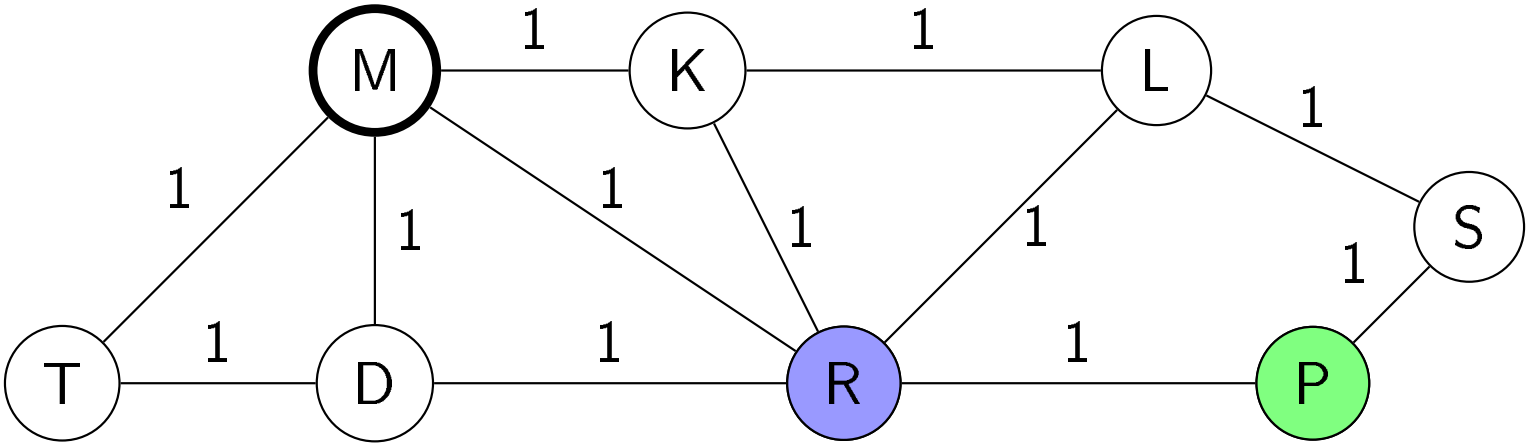


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

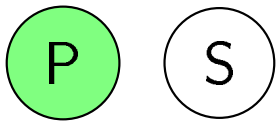


Algorytm BFS

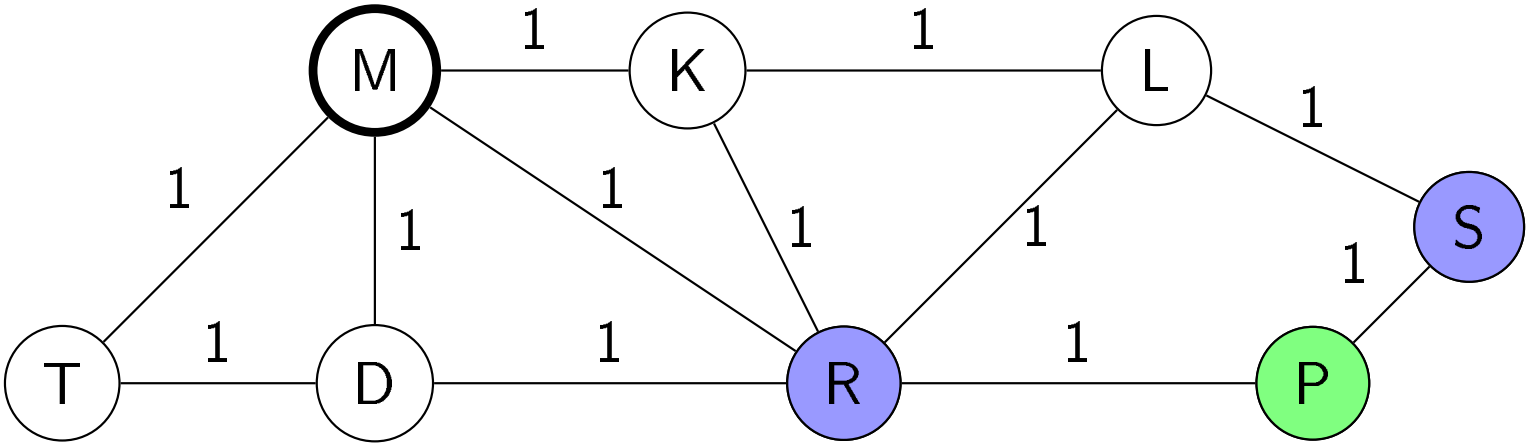


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

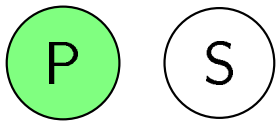


Algorytm BFS

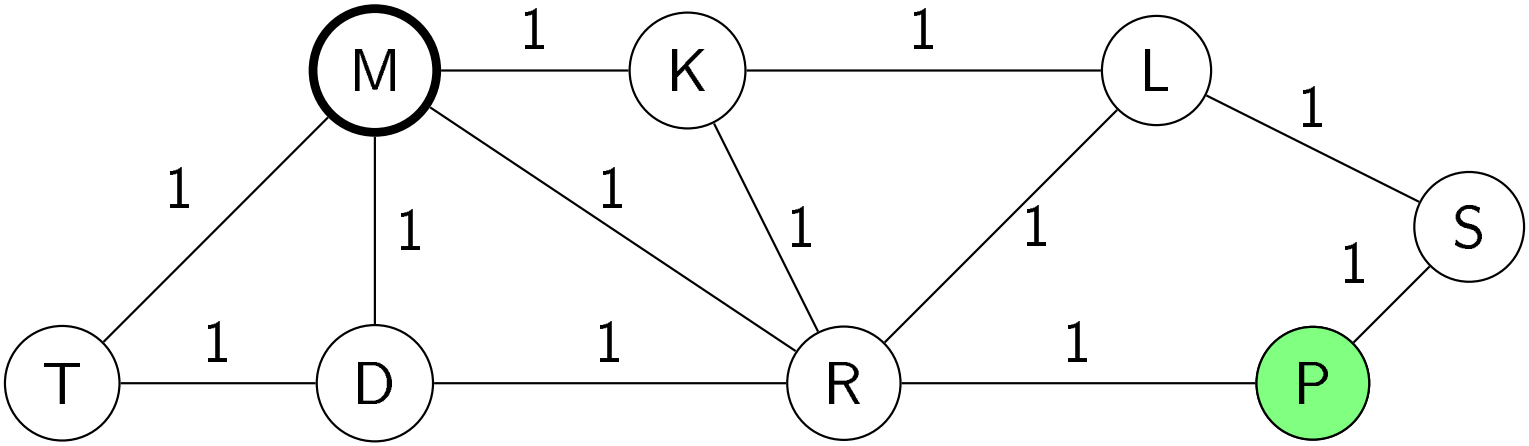


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

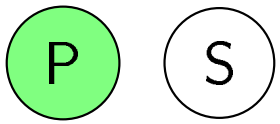


Algorytm BFS

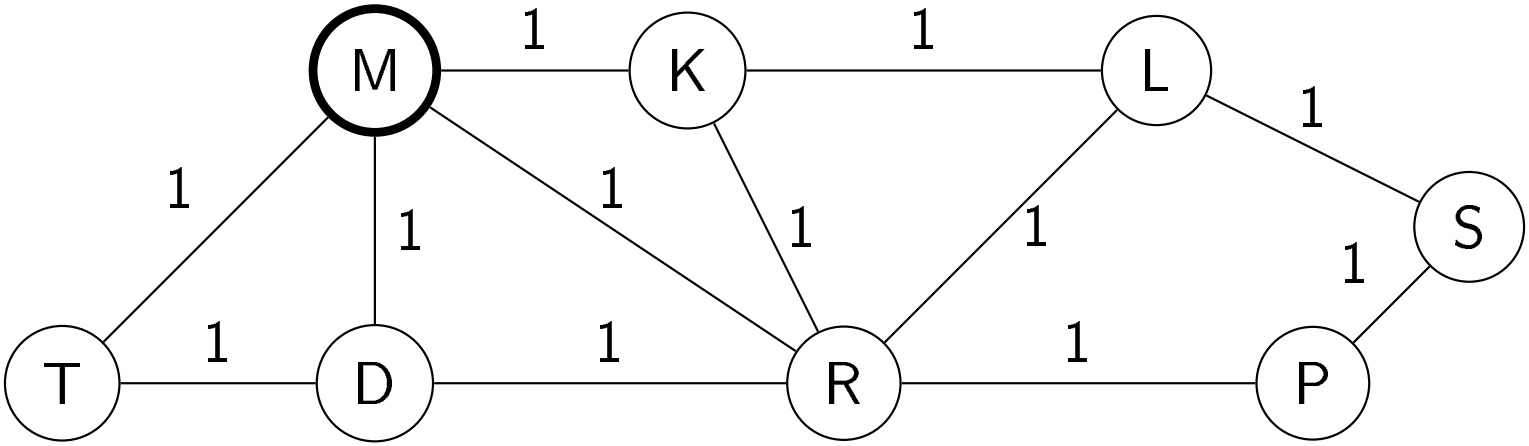


wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:



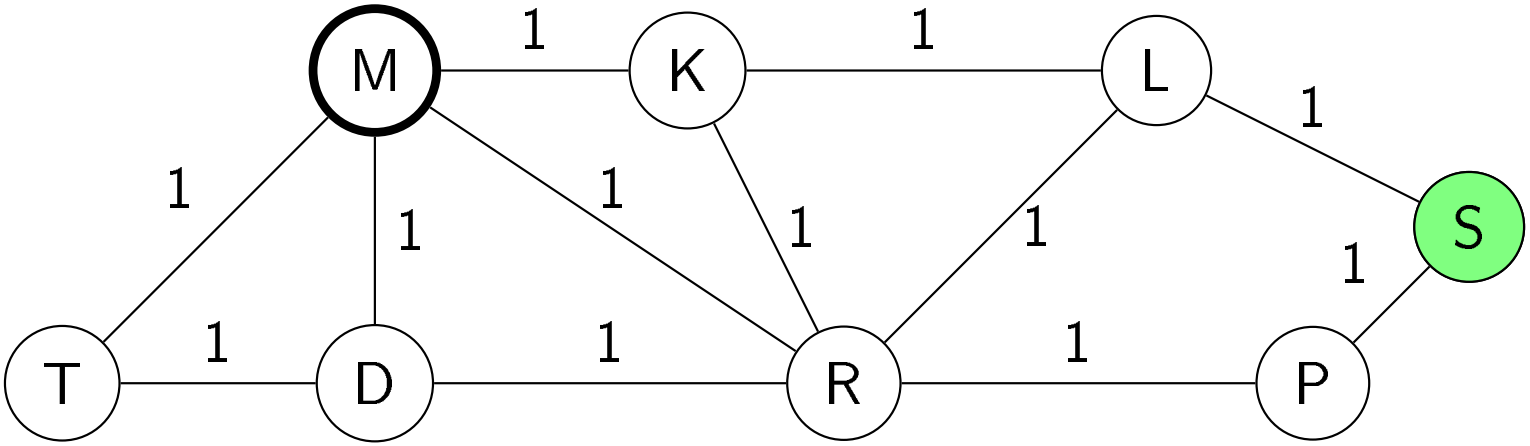
Algorytm BFS



wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

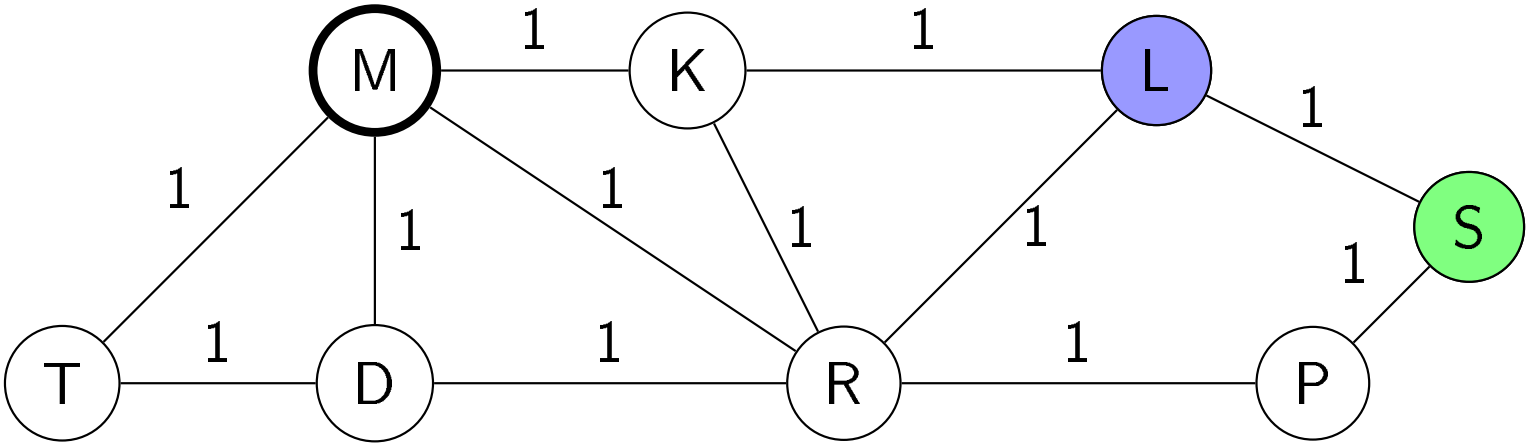
Algorytm BFS



wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

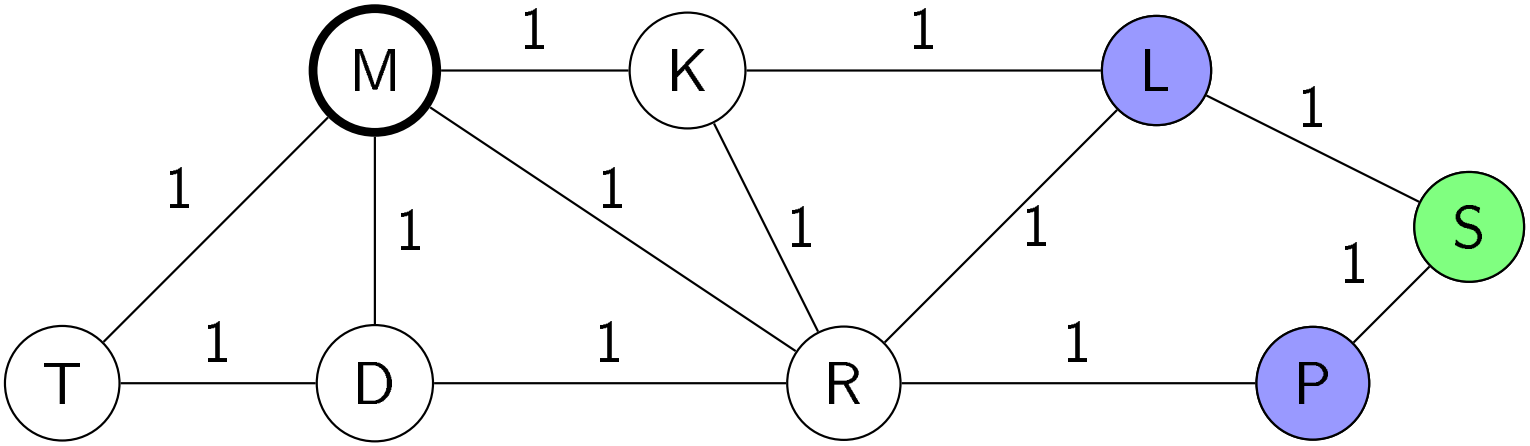
Algorytm BFS



wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

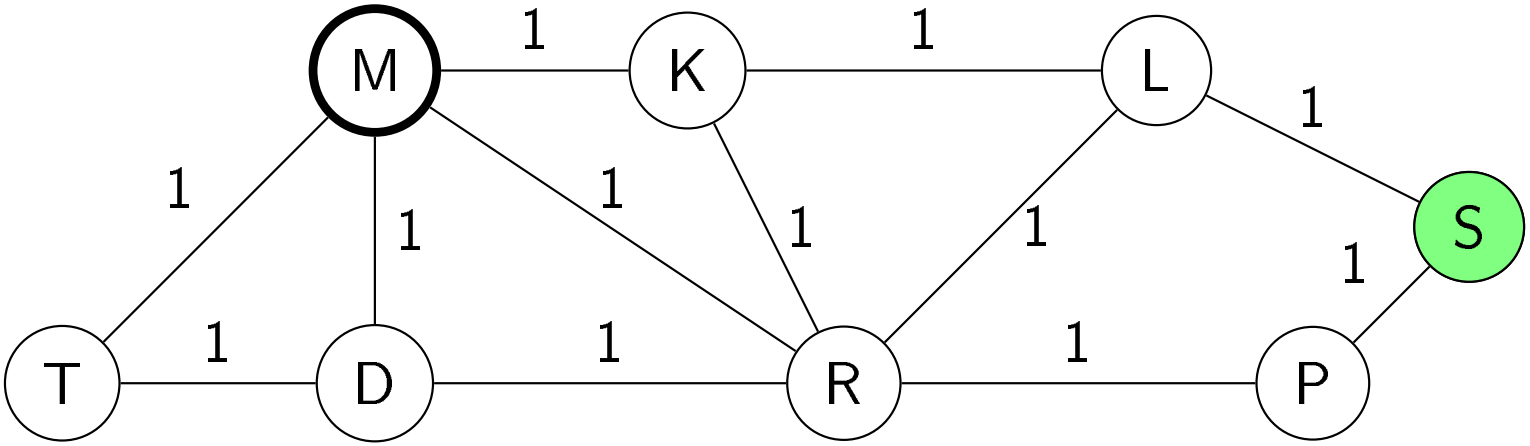
Algorytm BFS



wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

Algorytm BFS



wierzchołek	M	T	D	R	K	L	P	S
odległość	0	1	1	1	1	2	2	3

Kolejka:

Algorytm BFS

Wejście: Graf G, wierzchołek startowy s

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona to:

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona to:
      2. Ustaw ją: `odległość[u] = odległość[w] + krawędź[w, u]`.

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona to:
      2. Ustaw ją: `odległość[u] = odległość[w] + krawędź[w, u]`.
      3. Wrzuć `u` do kolejki.

Algorytm BFS

Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona to:
      2. Ustaw ją: `odległość[u] = odległość[w] + krawędź[w, u]`.
      3. Wrzuć `u` do kolejki.
      4. W przeciwnym wypadku nic nie rób.

Algorytm BFS

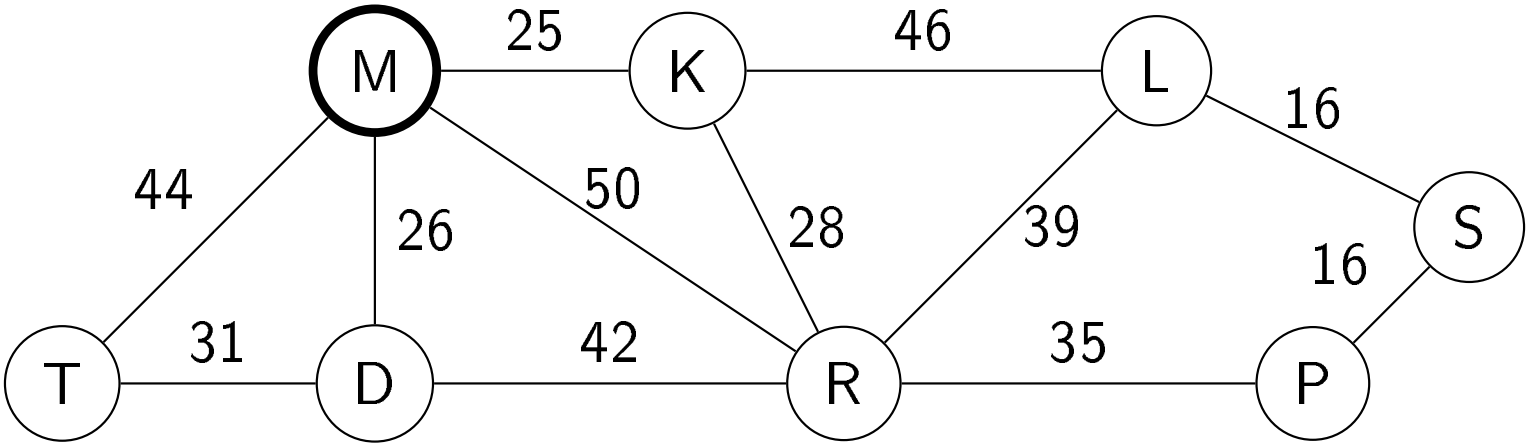
Wejście: Graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona to:
      2. Ustaw ją: `odległość[u] = odległość[w] + krawędź[w, u]`.
      3. Wrzuć `u` do kolejki.
      4. W przeciwnym wypadku nic nie rób.
4. W tablicy `odległość` znajdują się wyniki.

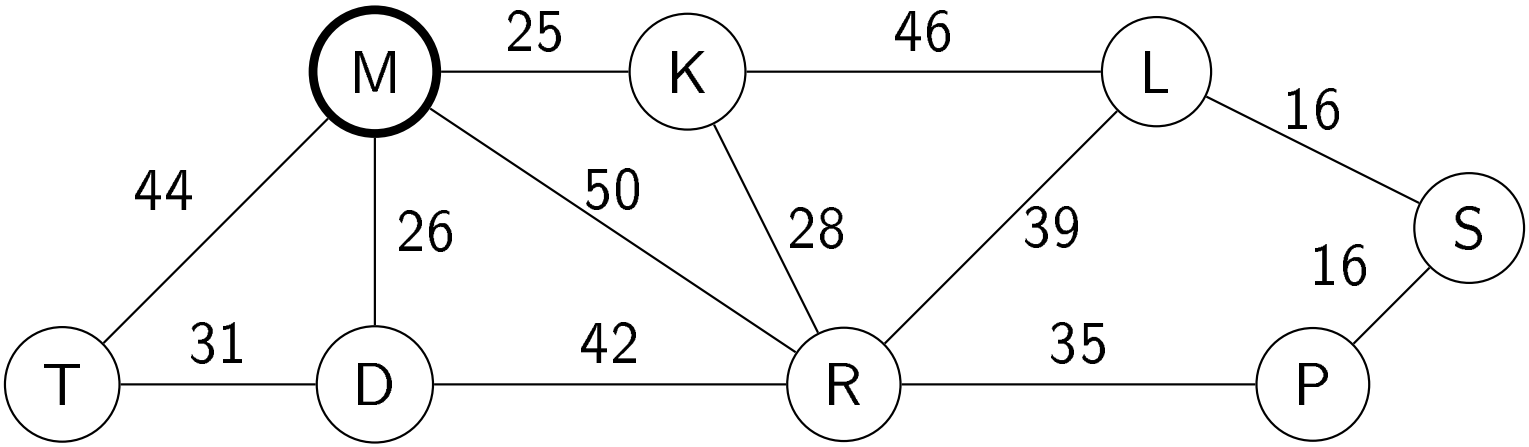
Algorytm BFS

```
def bfs(G, s):
    odleglosc = {}
    Q = Queue()
    Q.put(s)
    while len(Q) > 0:
        w = Q.get()
        for u, koszt_krawedzi in G[w]:
            if u not in odleglosc:
                odleglosc[u] = odleglosc[w] + koszt_krawedzi
                Q.put(u)
    return odleglosc
```

Wracamy



Dijkstra

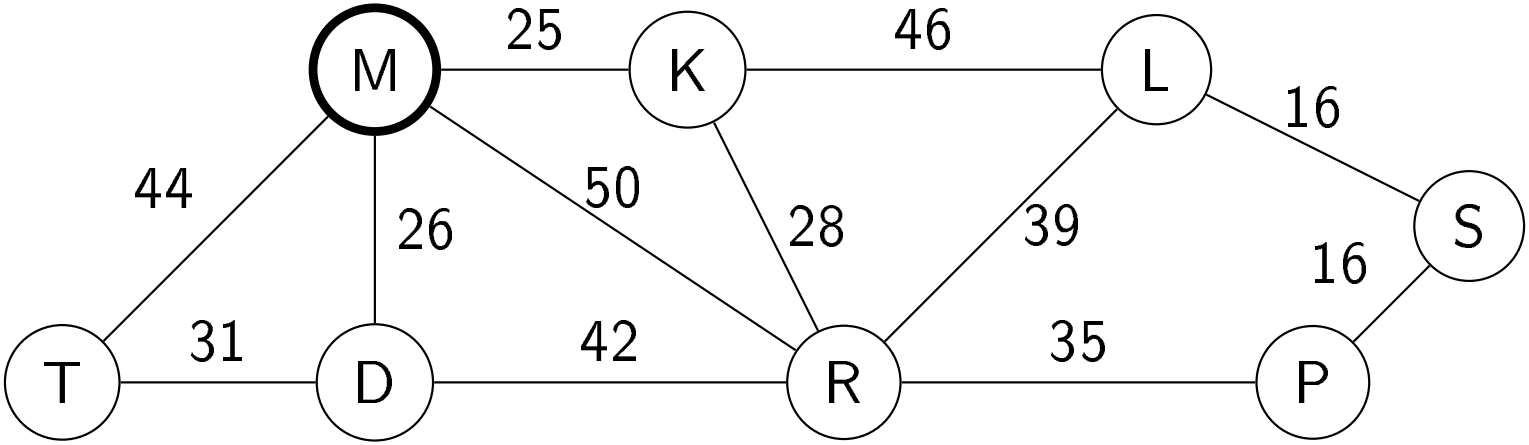


Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość								

Kolejka:

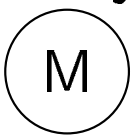
Dijkstra



Długość aktualnej ścieżki:

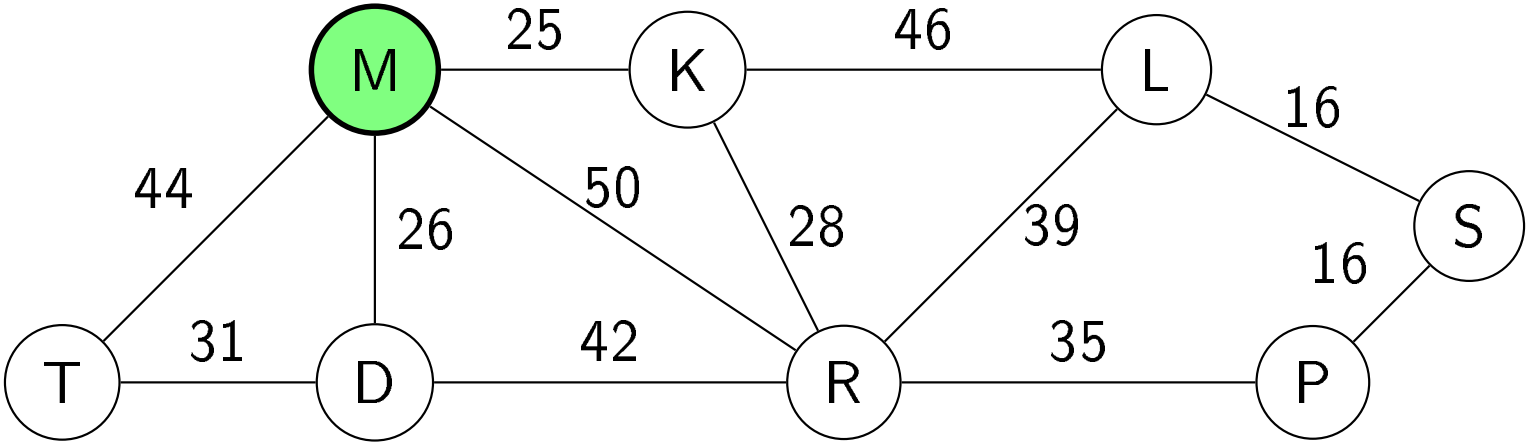
wierzchołek	M	T	D	R	K	L	P	S
odległość	0							

Kolejka:



0

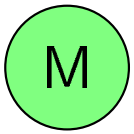
Dijkstra



Długość aktualnej ścieżki:

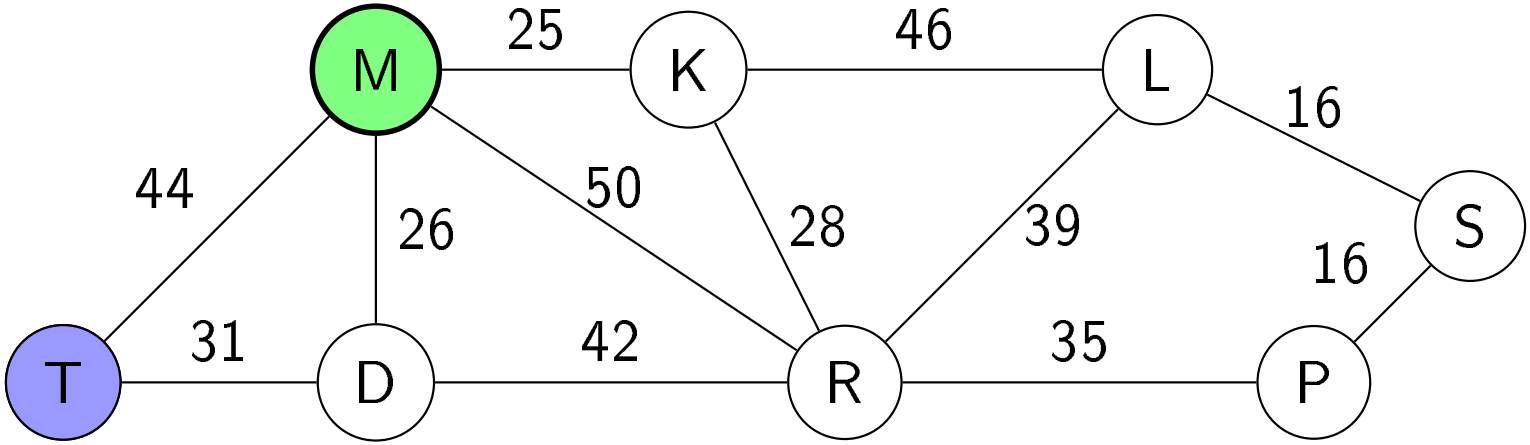
wierzchołek	M	T	D	R	K	L	P	S
odległość	0							

Kolejka:



0

Dijkstra



Długość aktualnej ścieżki: $0 + 44 = 44$

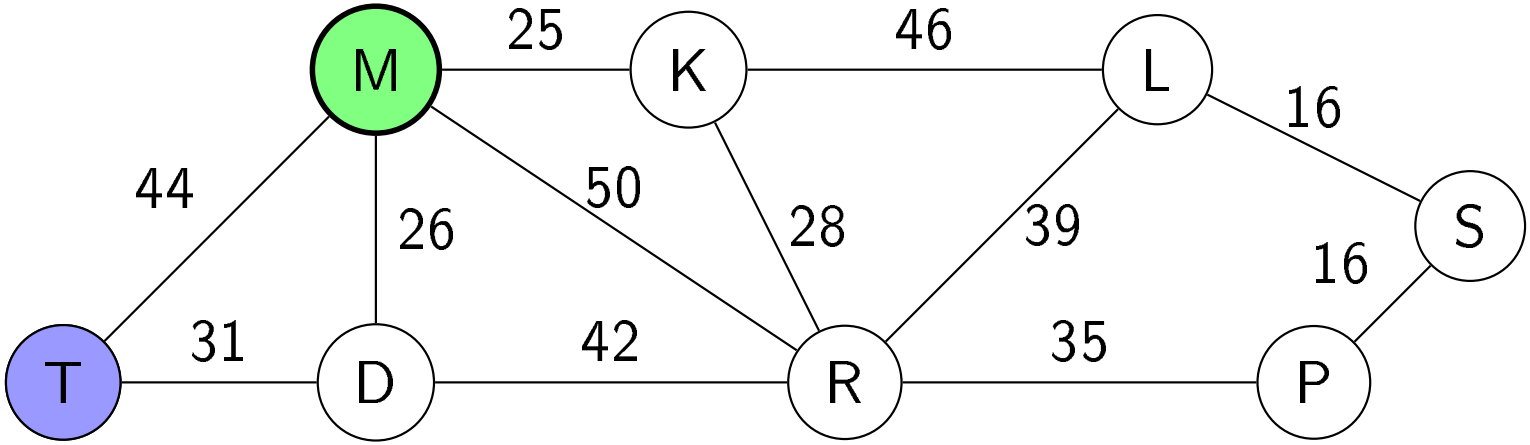
wierzchołek	M	T	D	R	K	L	P	S
odległość	0							

Kolejka:

M

0

Dijkstra

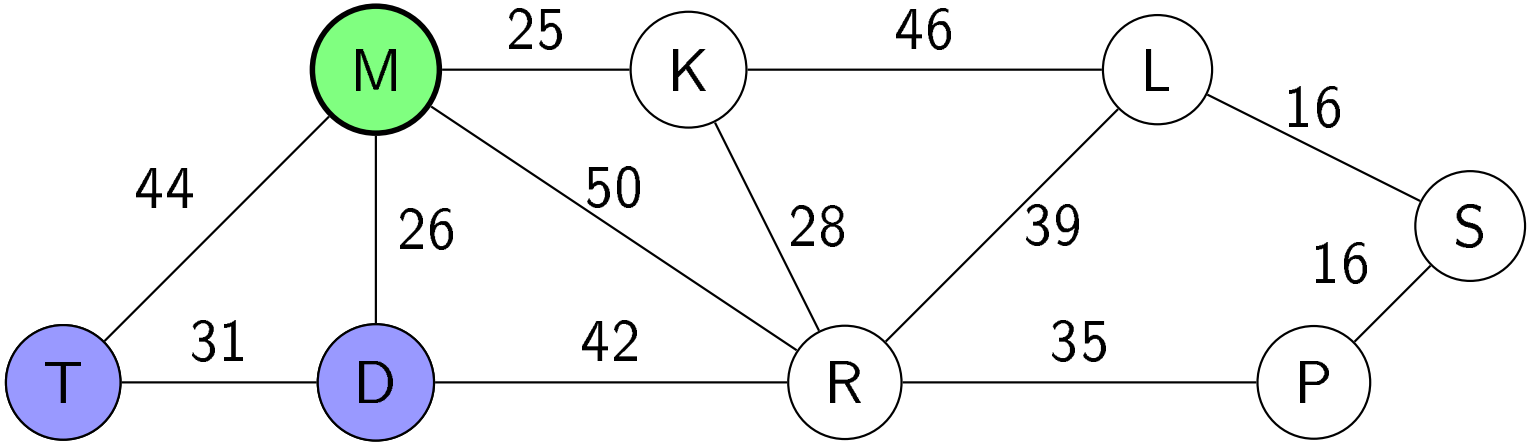


Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44						

Kolejka:

Dijkstra

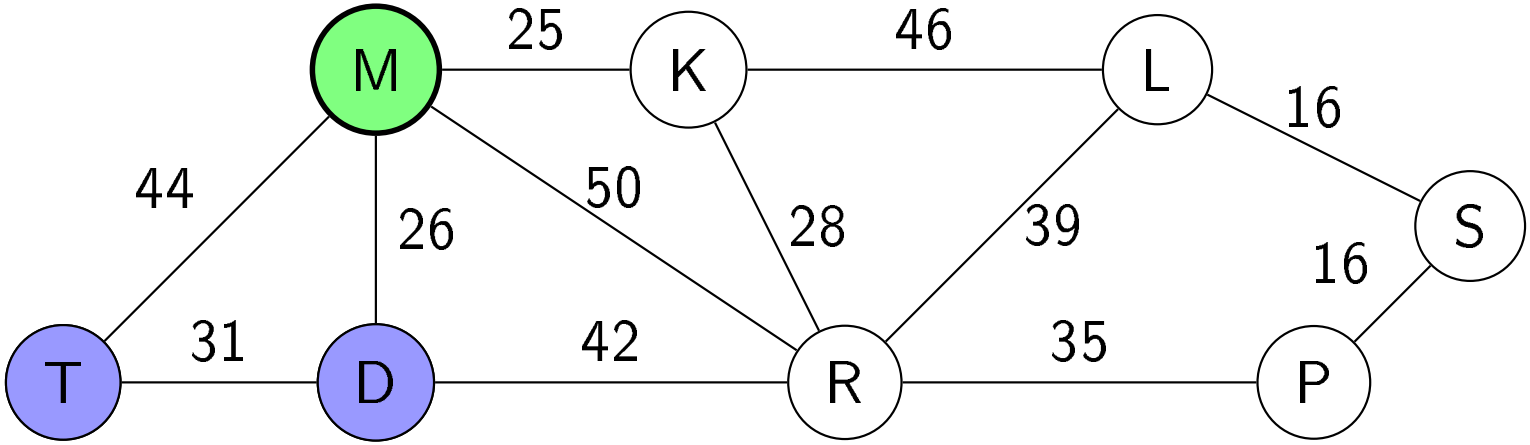


Długość aktualnej ścieżki: $0 + 26 = 26$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44						

Kolejka:

Dijkstra

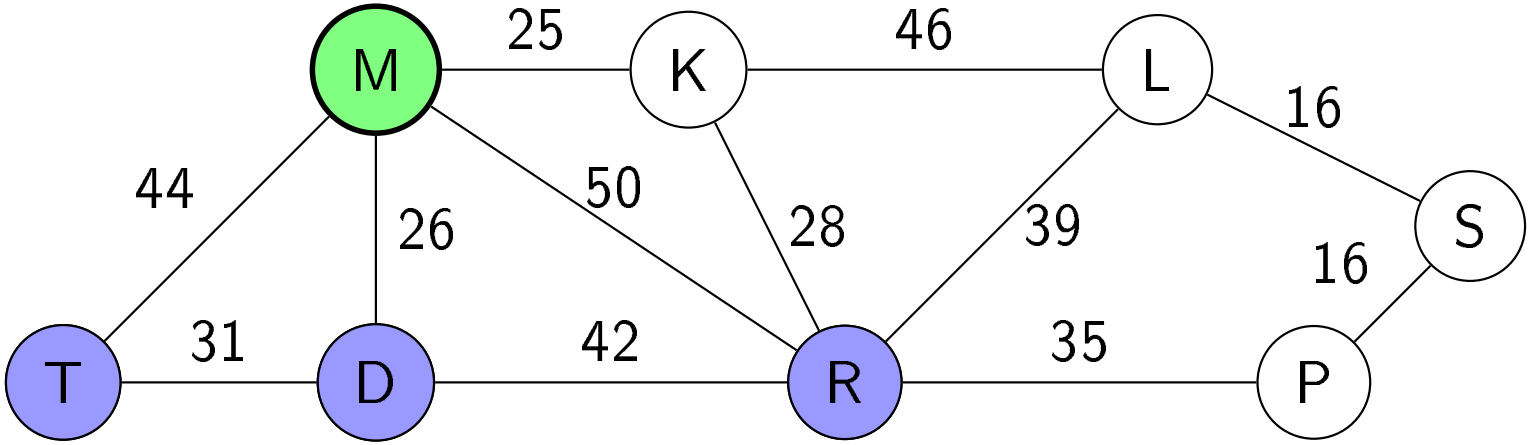


Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26					

Kolejka:

Dijkstra

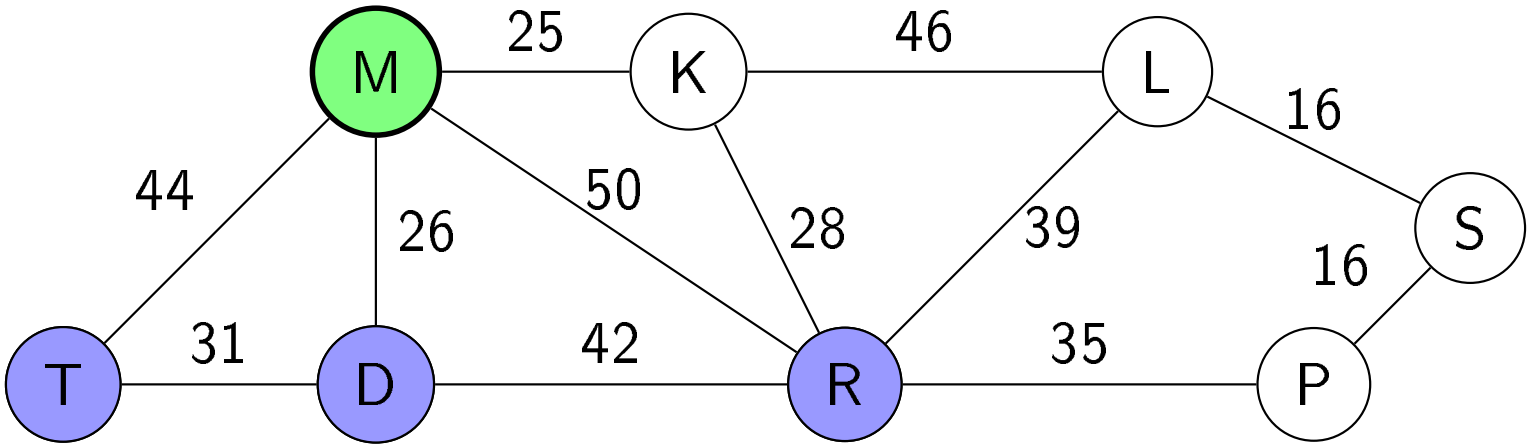


Długość aktualnej ścieżki: $0 + 50 = 50$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26					

Kolejka:

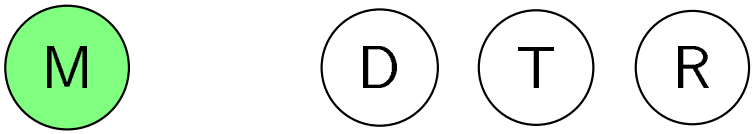
Dijkstra



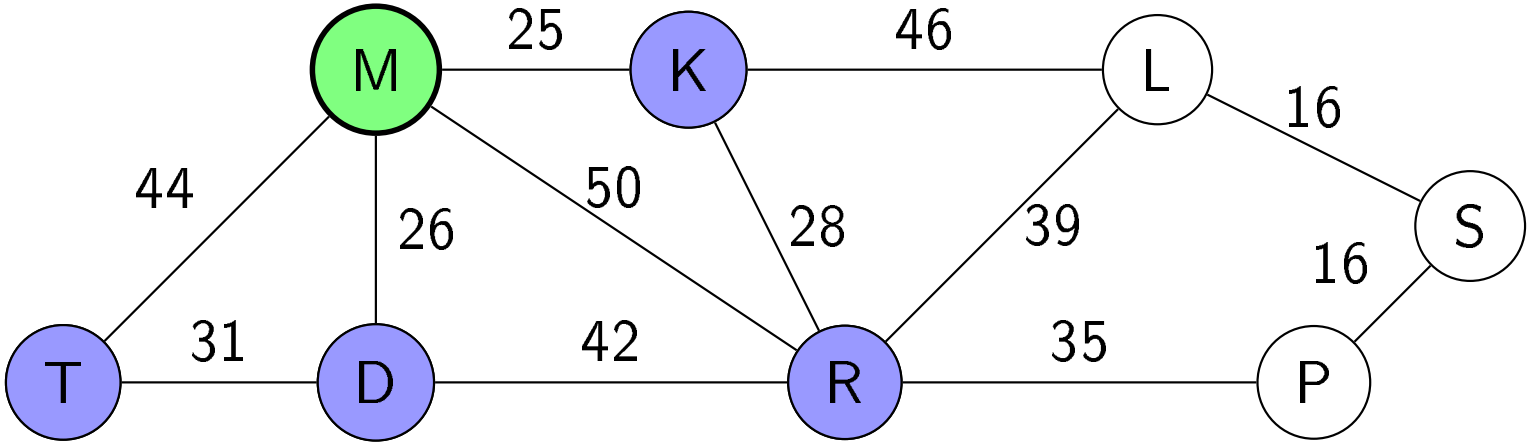
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50				

Kolejka:



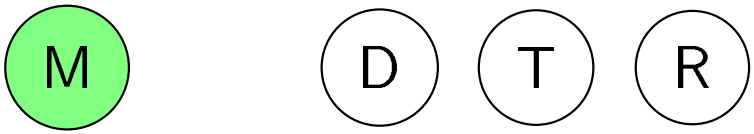
Dijkstra



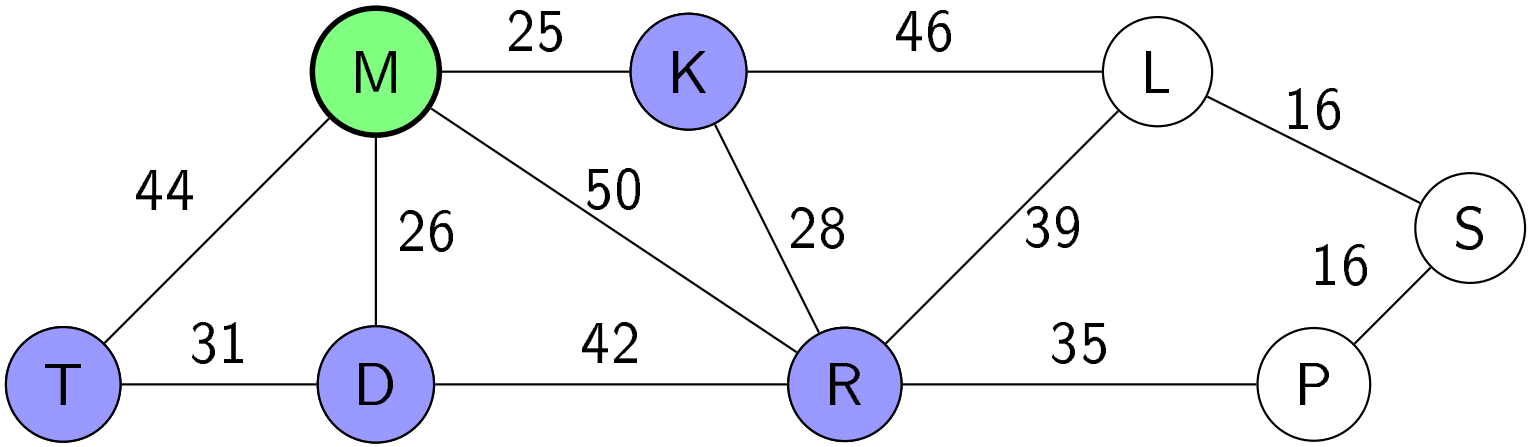
Długość aktualnej ścieżki: $0 + 25 = 25$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50				

Kolejka:



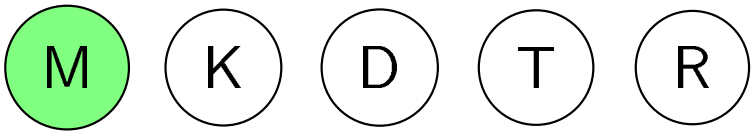
Dijkstra



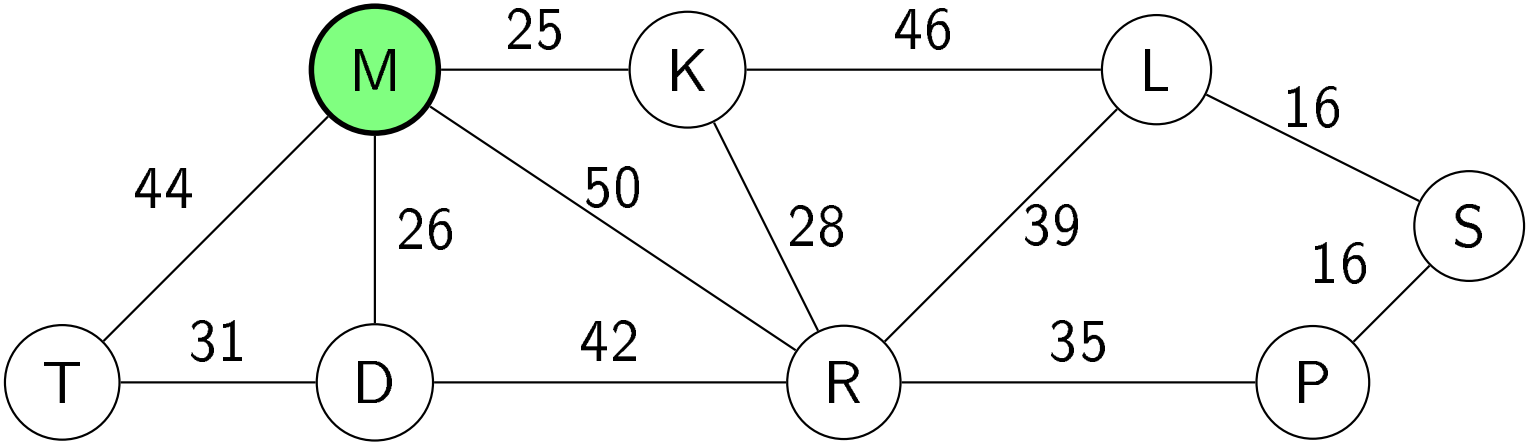
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25			

Kolejka:



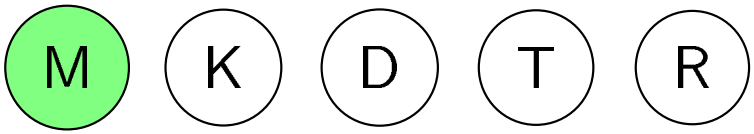
Dijkstra



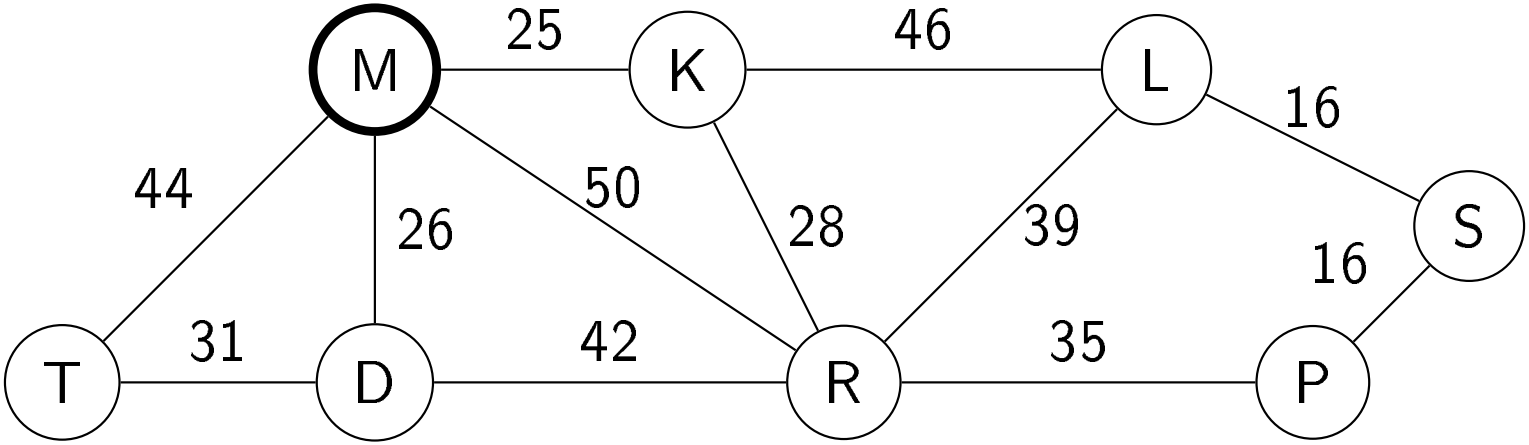
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25			

Kolejka:



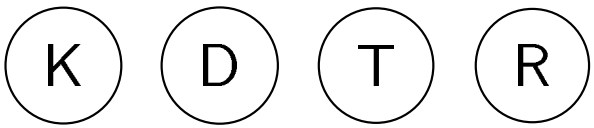
Dijkstra



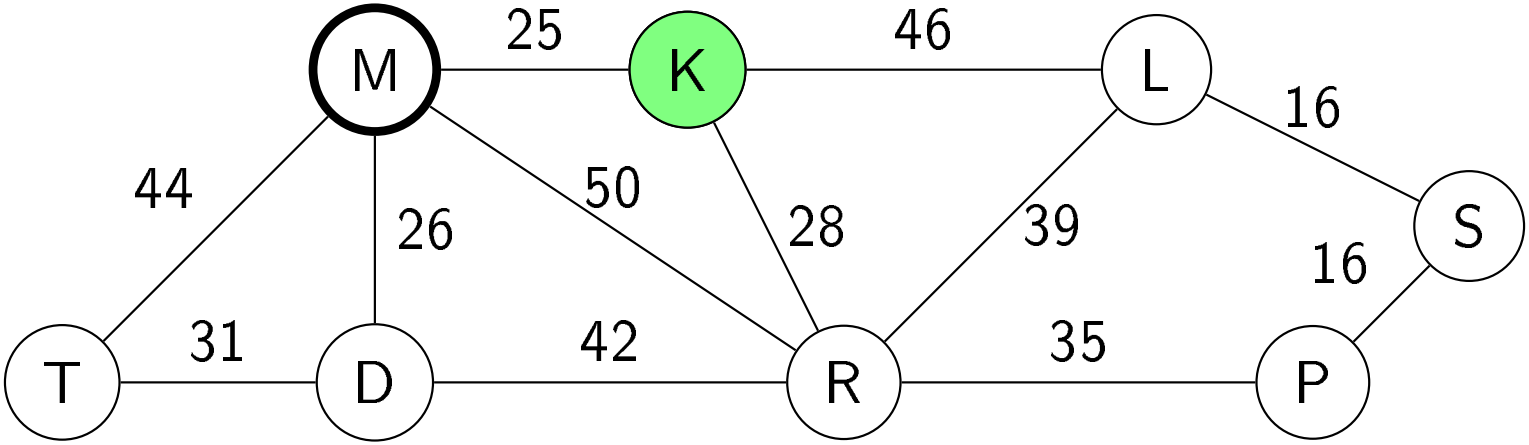
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25			

Kolejka:



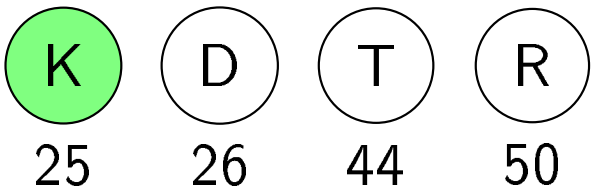
Dijkstra



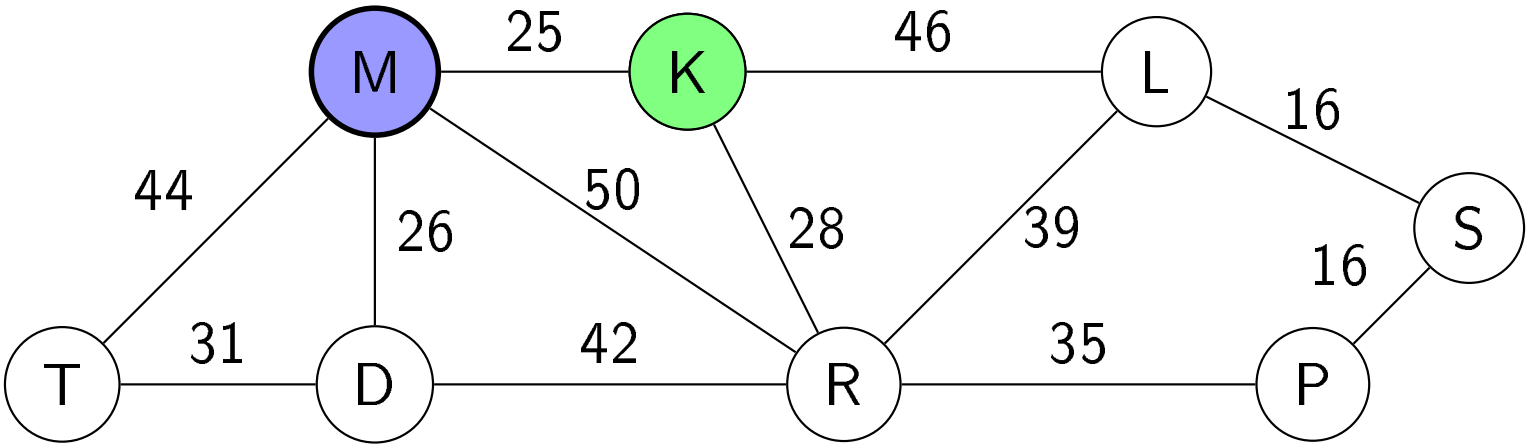
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25			

Kolejka:



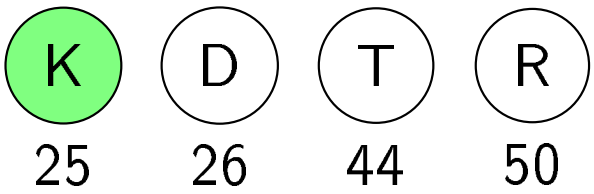
Dijkstra



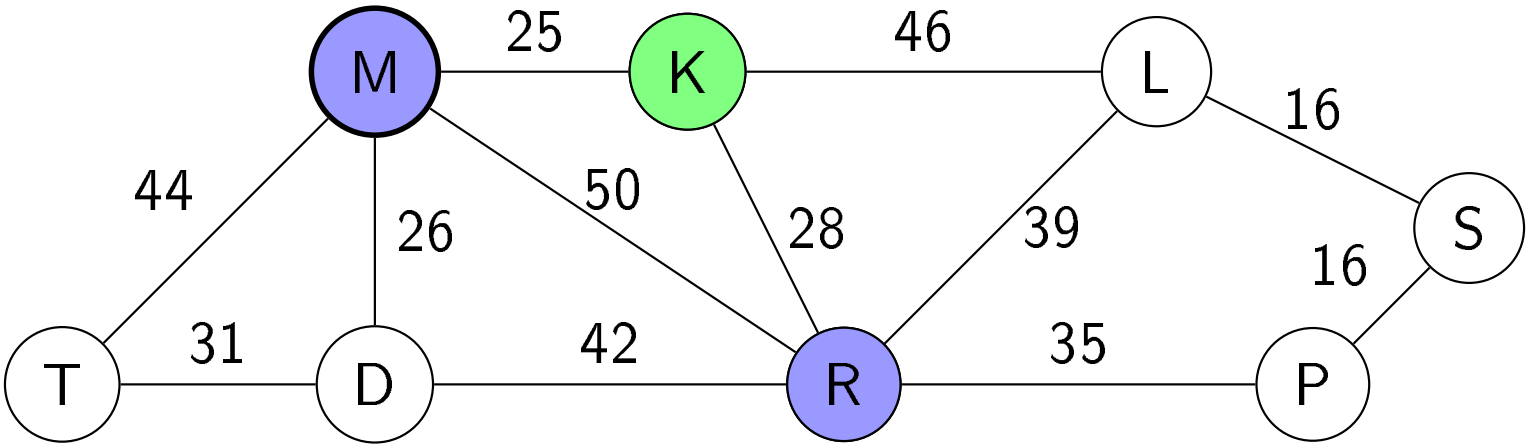
Długość aktualnej ścieżki: $25 + 25 = 50$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25			

Kolejka:



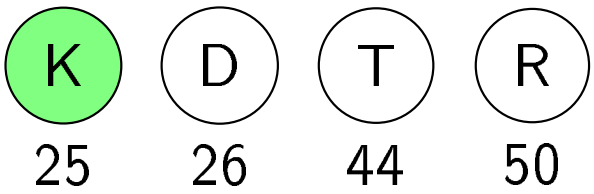
Dijkstra



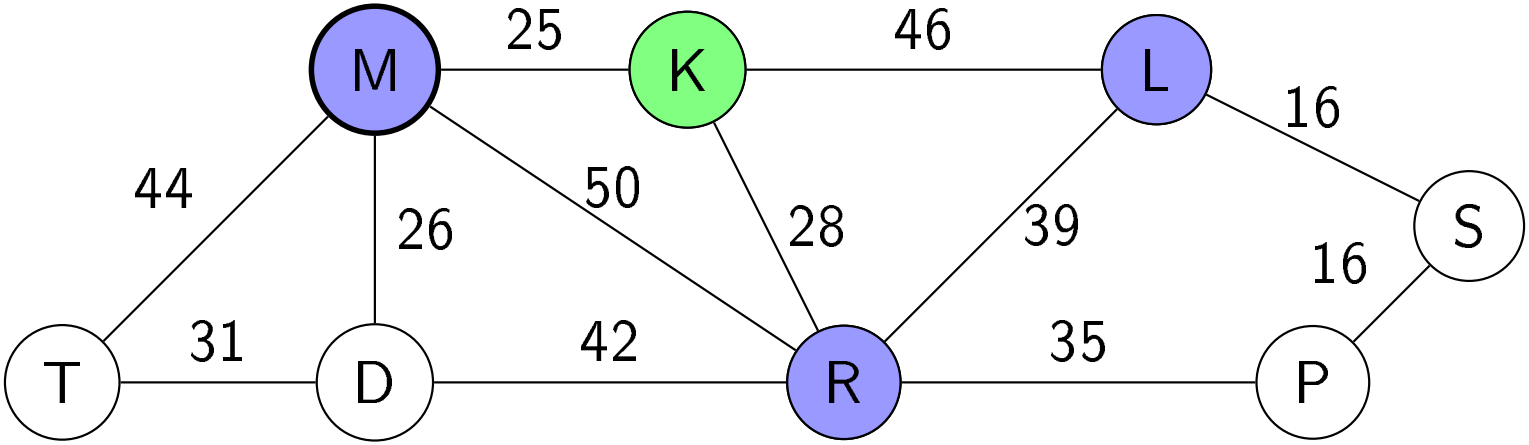
Długość aktualnej ścieżki: $25 + 28 = 53$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25			

Kolejka:



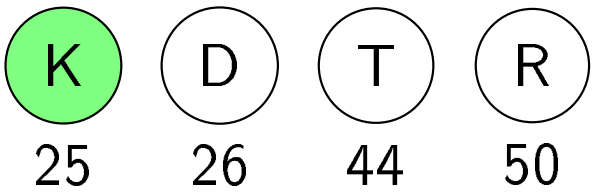
Dijkstra



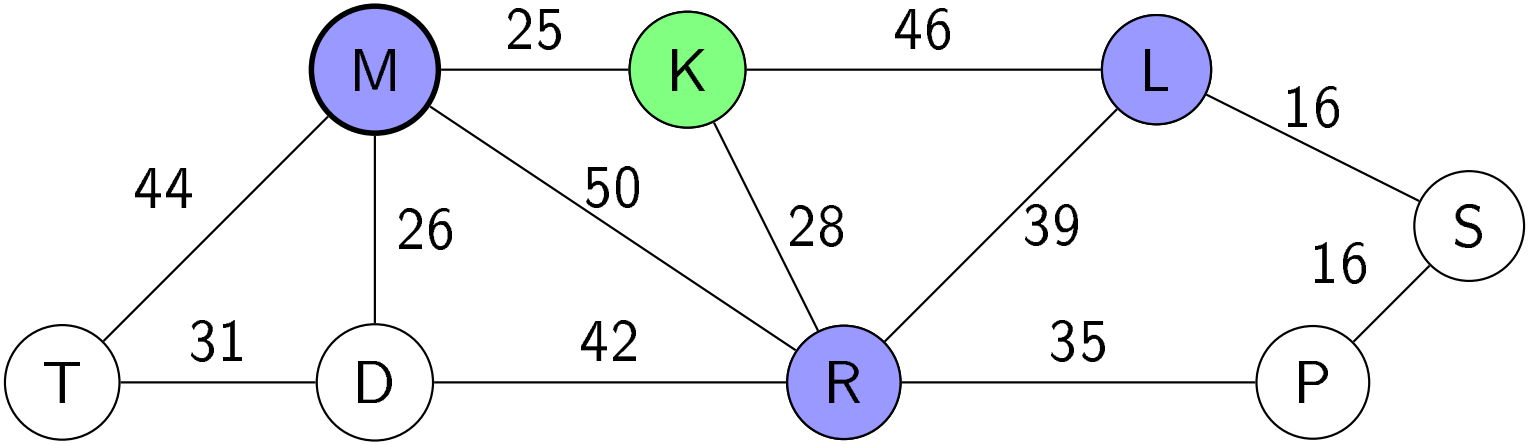
Długość aktualnej ścieżki: $25 + 46 = 71$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25			

Kolejka:



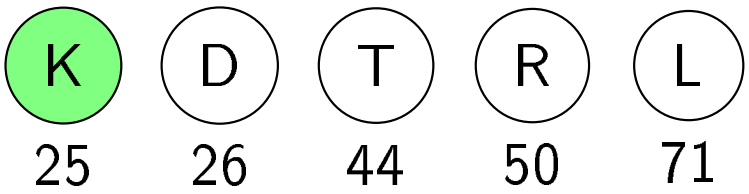
Dijkstra



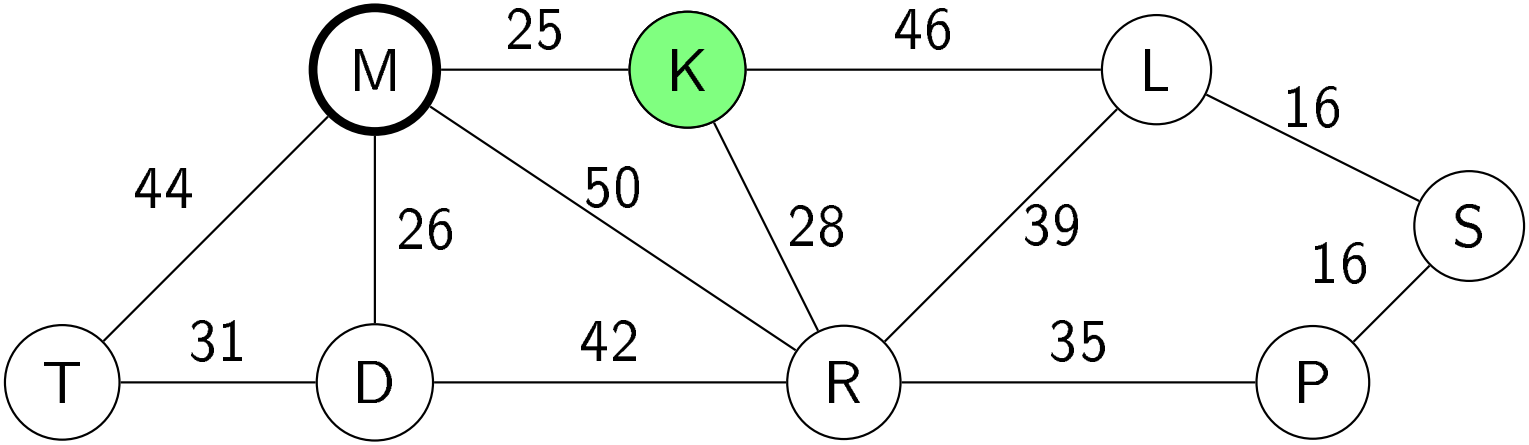
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



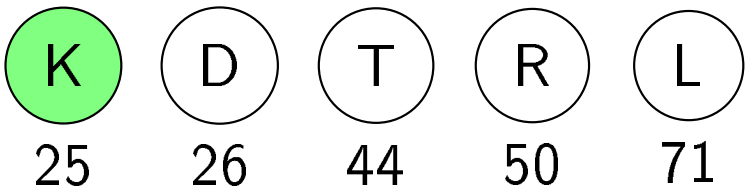
Dijkstra



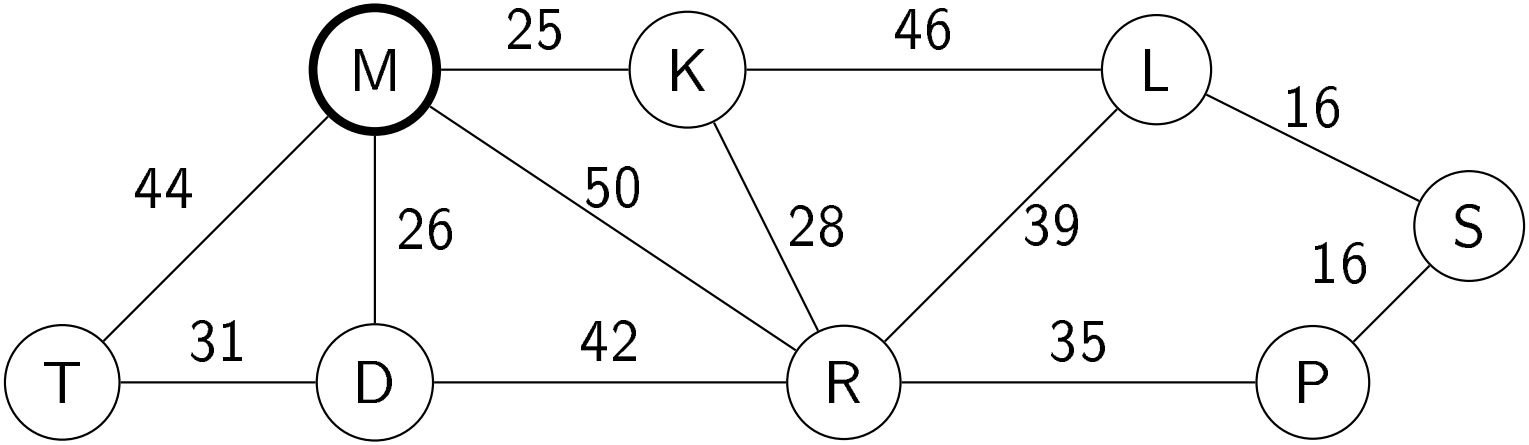
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



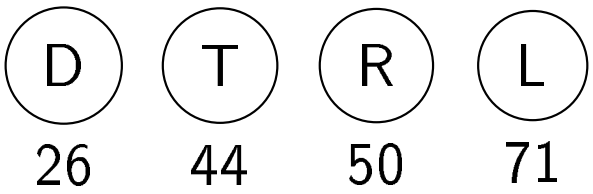
Dijkstra



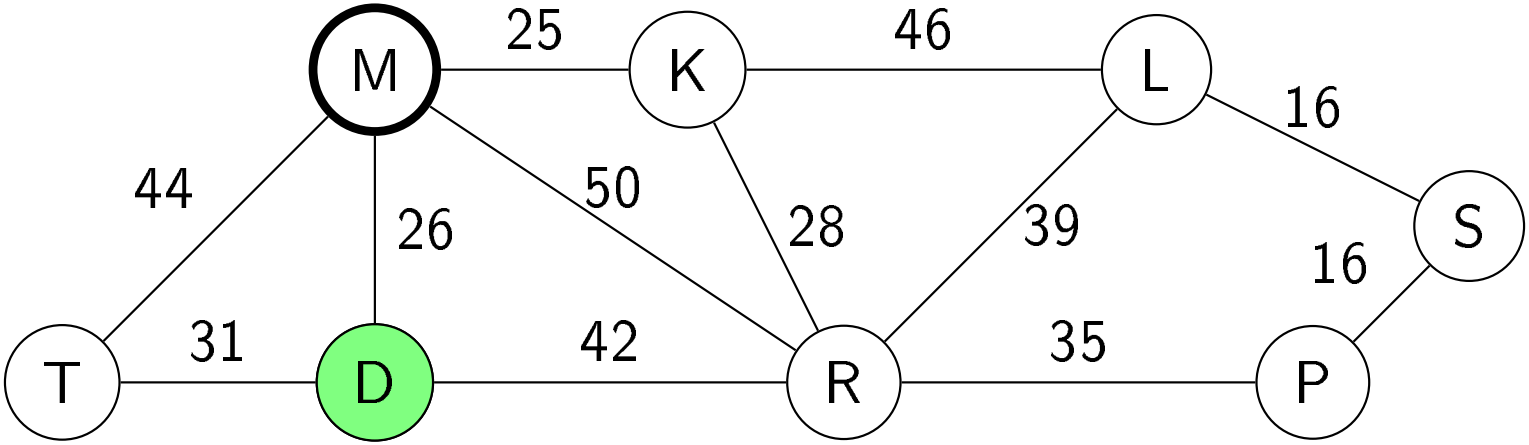
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



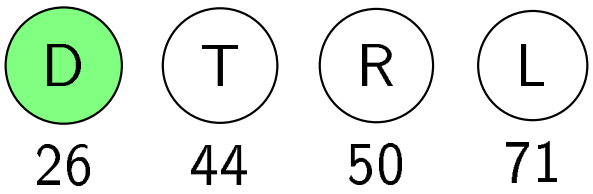
Dijkstra



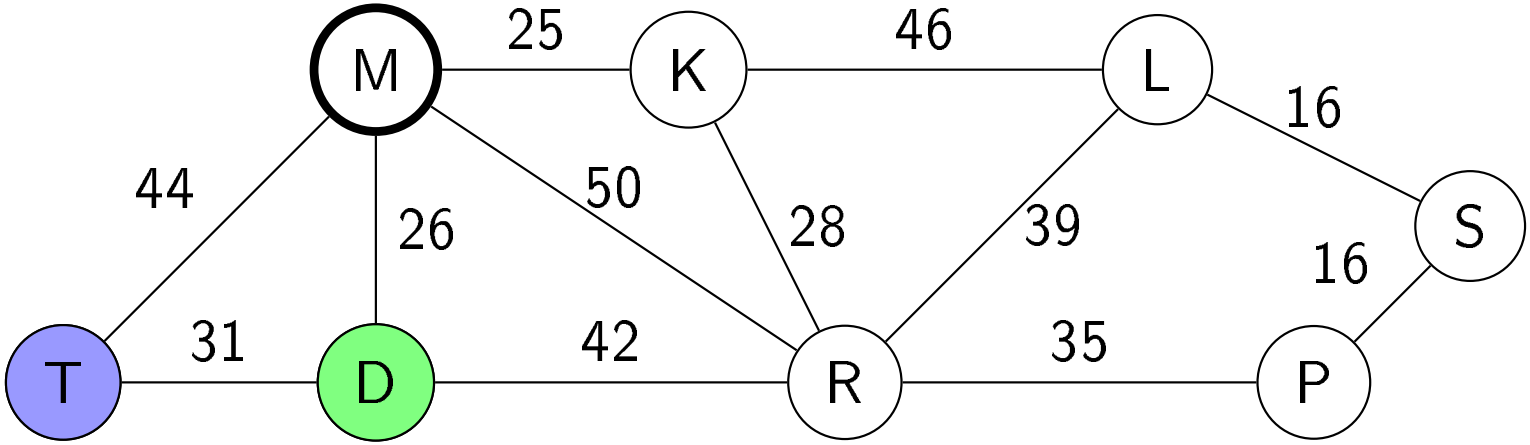
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



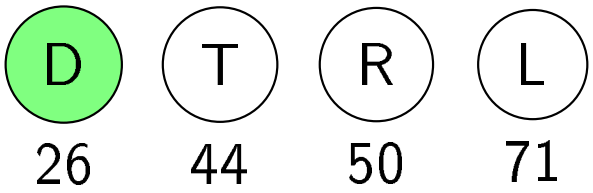
Dijkstra



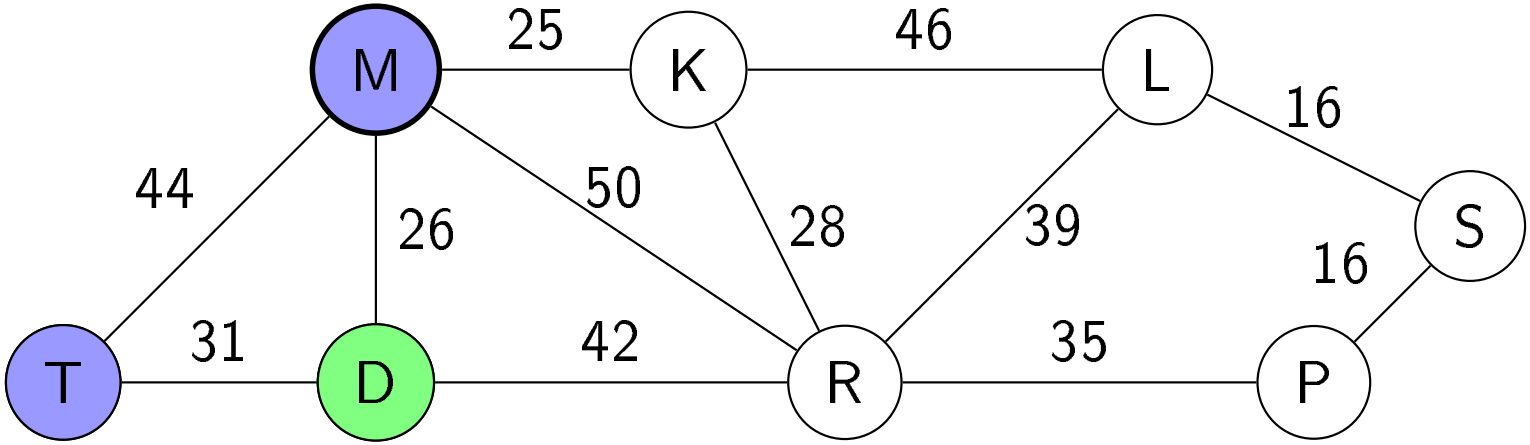
Długość aktualnej ścieżki: $26 + 31 = 57$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



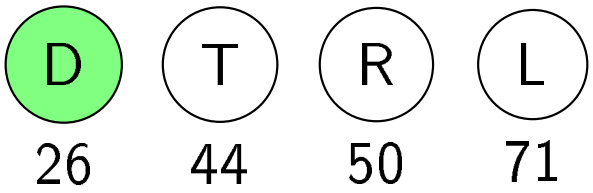
Dijkstra



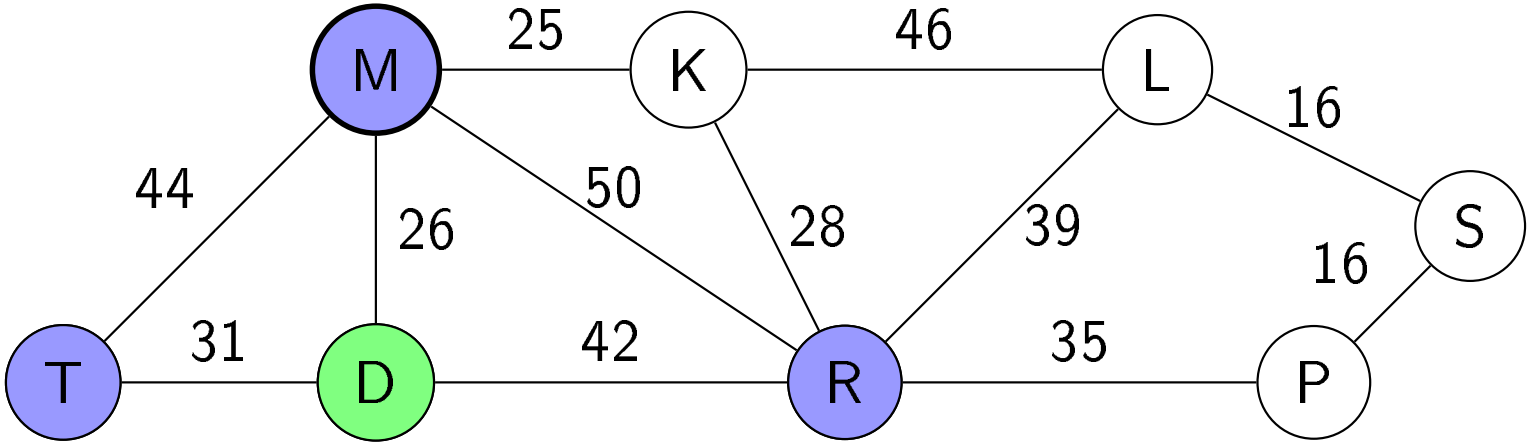
Długość aktualnej ścieżki: $26 + 26 = 52$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



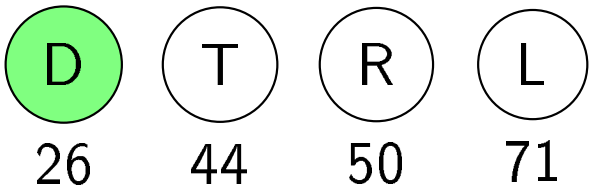
Dijkstra



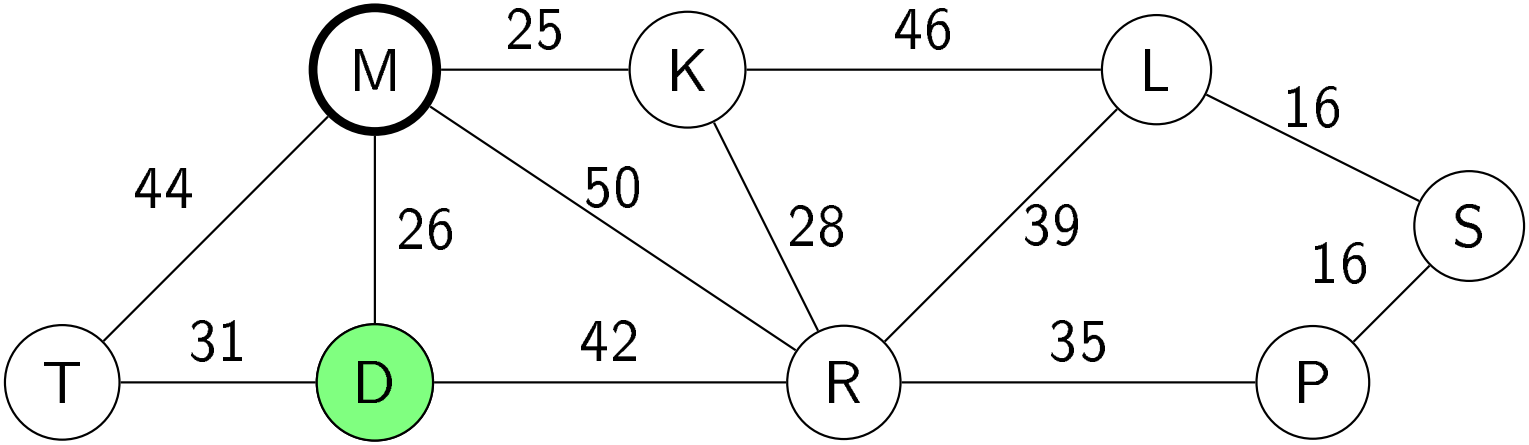
Długość aktualnej ścieżki: $26 + 42 = 68$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



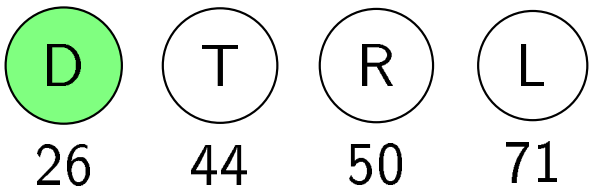
Dijkstra



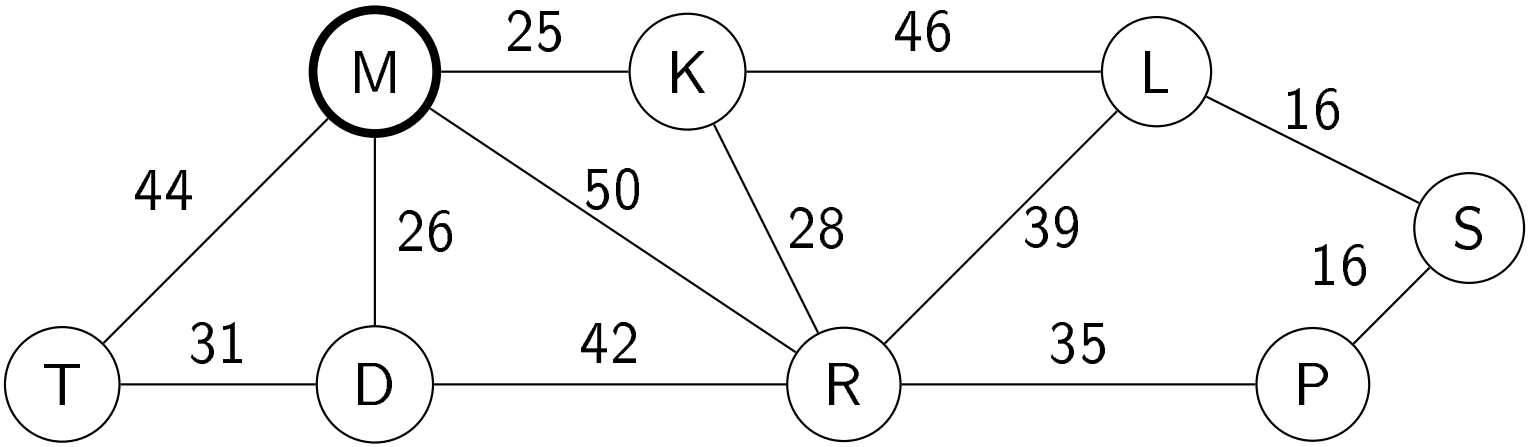
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



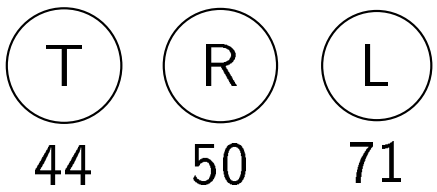
Dijkstra



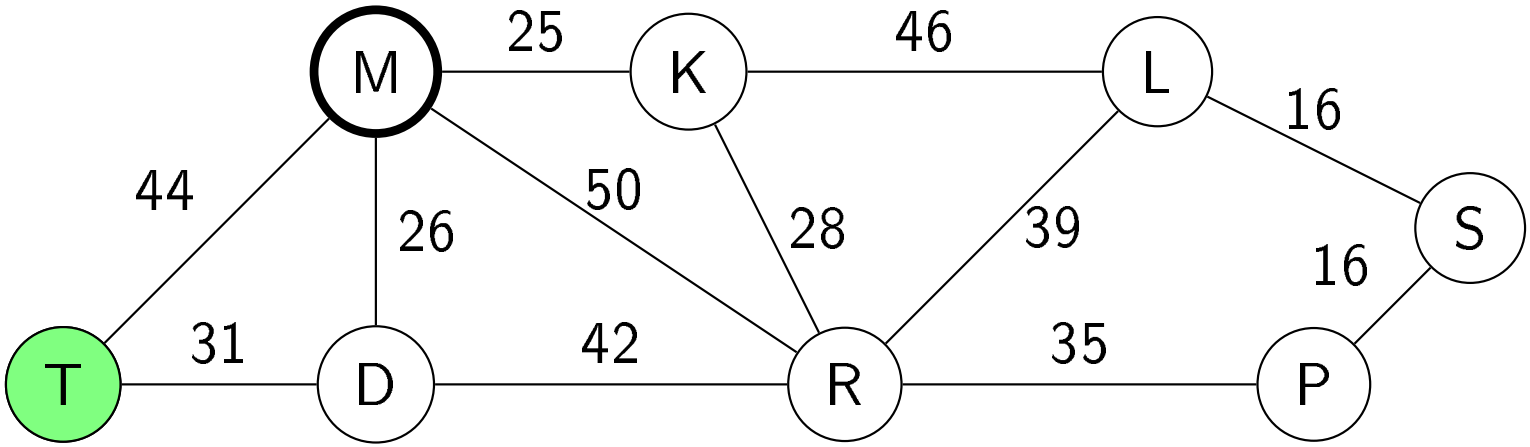
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



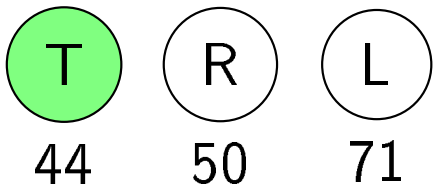
Dijkstra



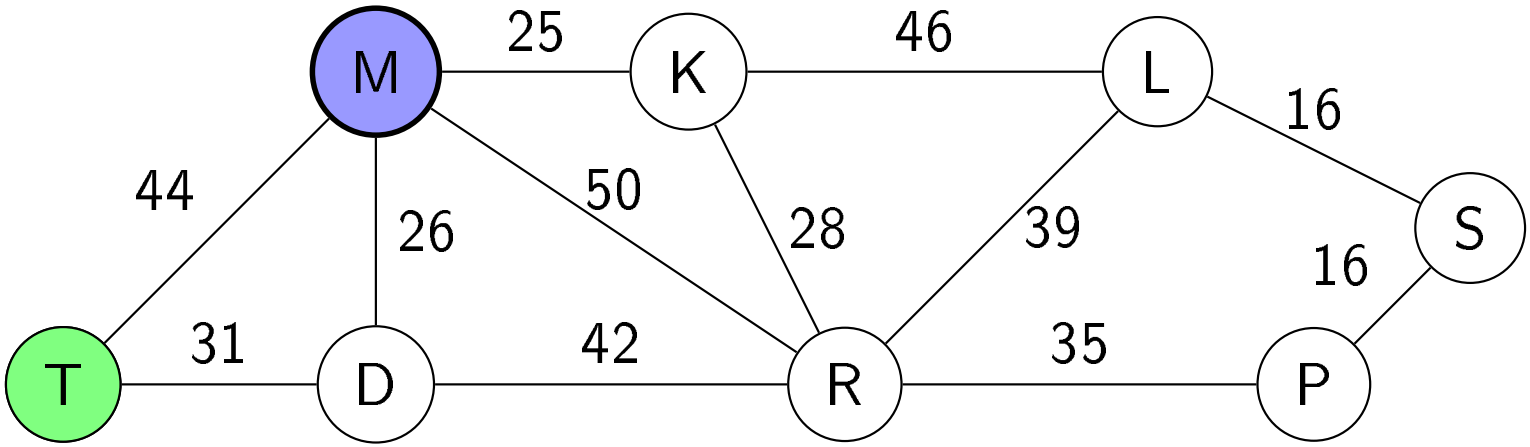
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



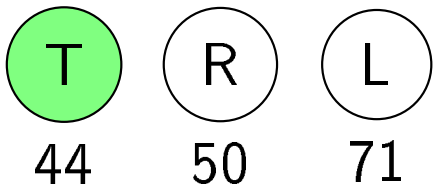
Dijkstra



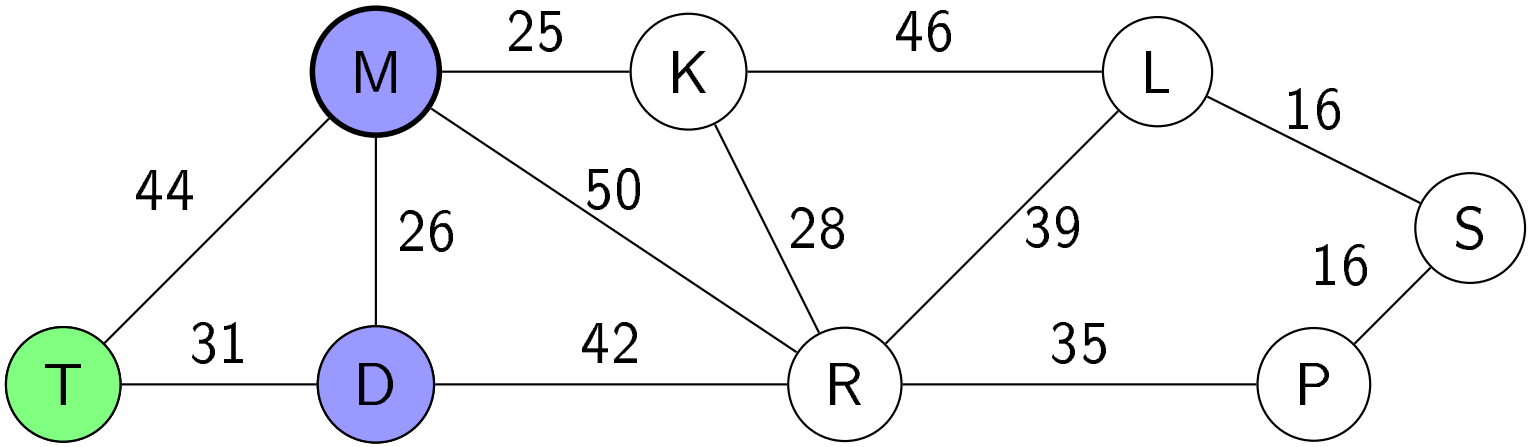
Długość aktualnej ścieżki: $44 + 44 = 88$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



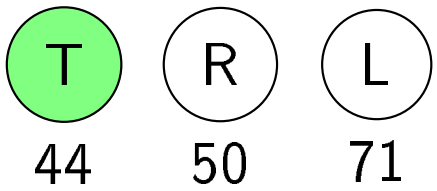
Dijkstra



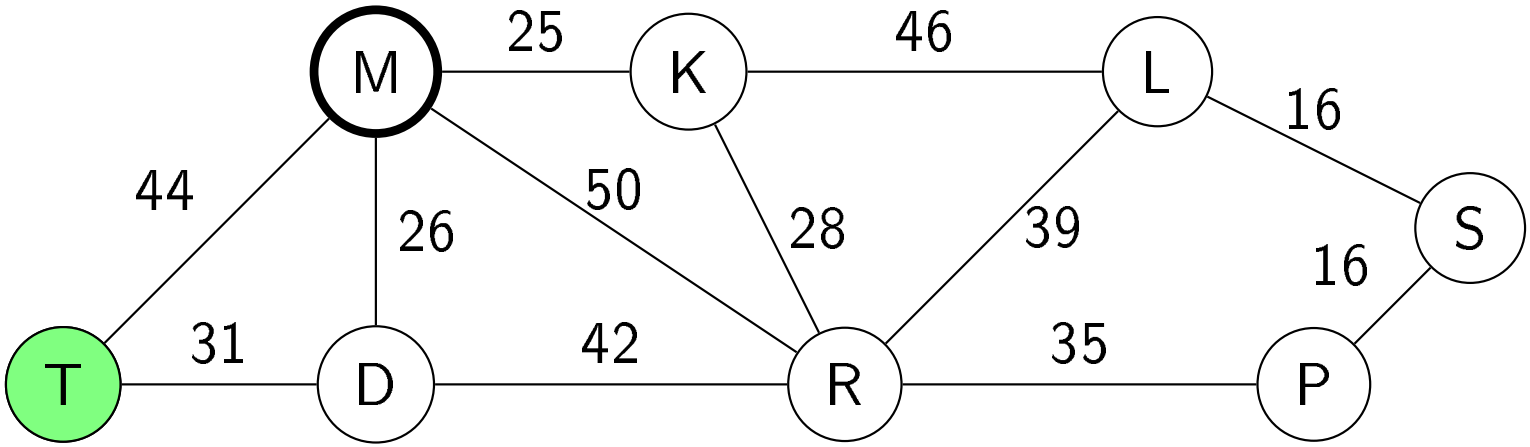
Długość aktualnej ścieżki: $44 + 31 = 75$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



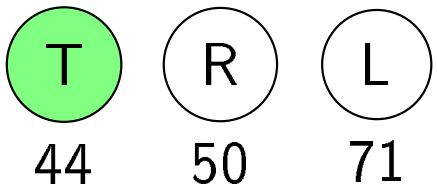
Dijkstra



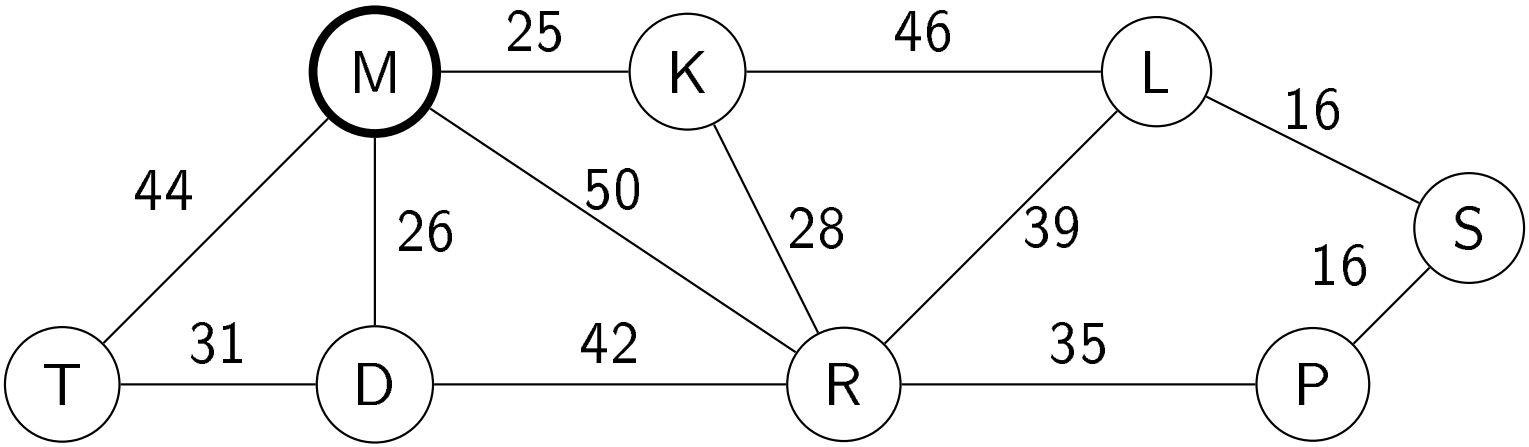
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



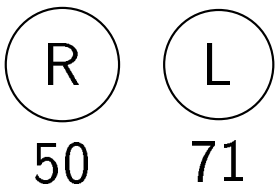
Dijkstra



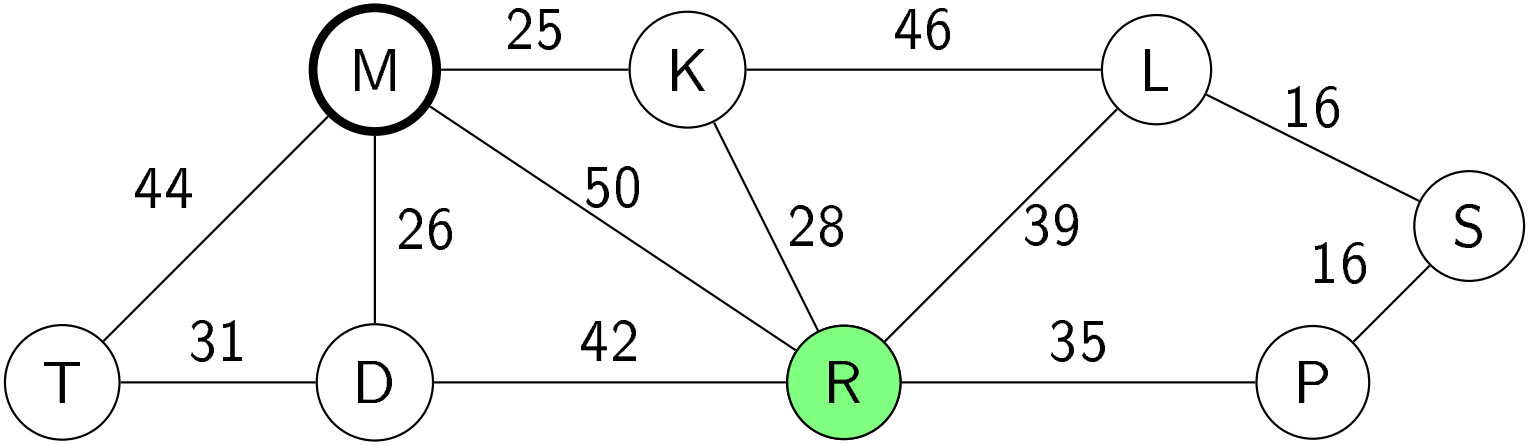
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



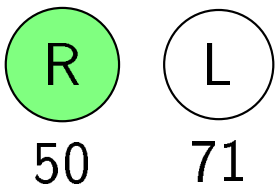
Dijkstra



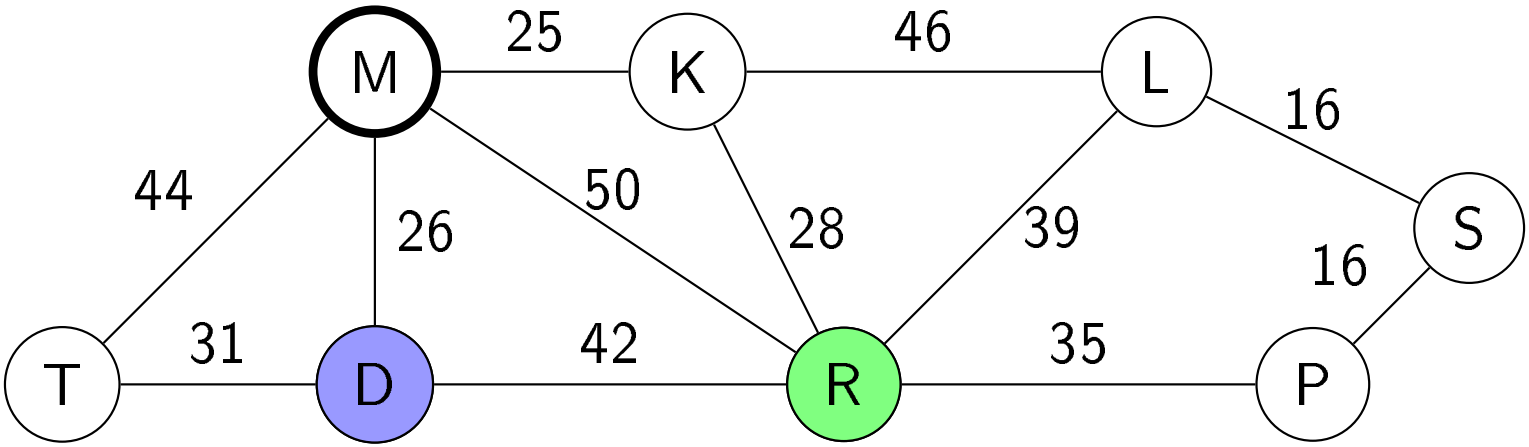
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



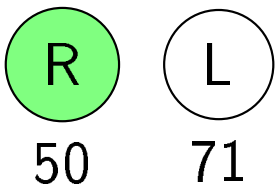
Dijkstra



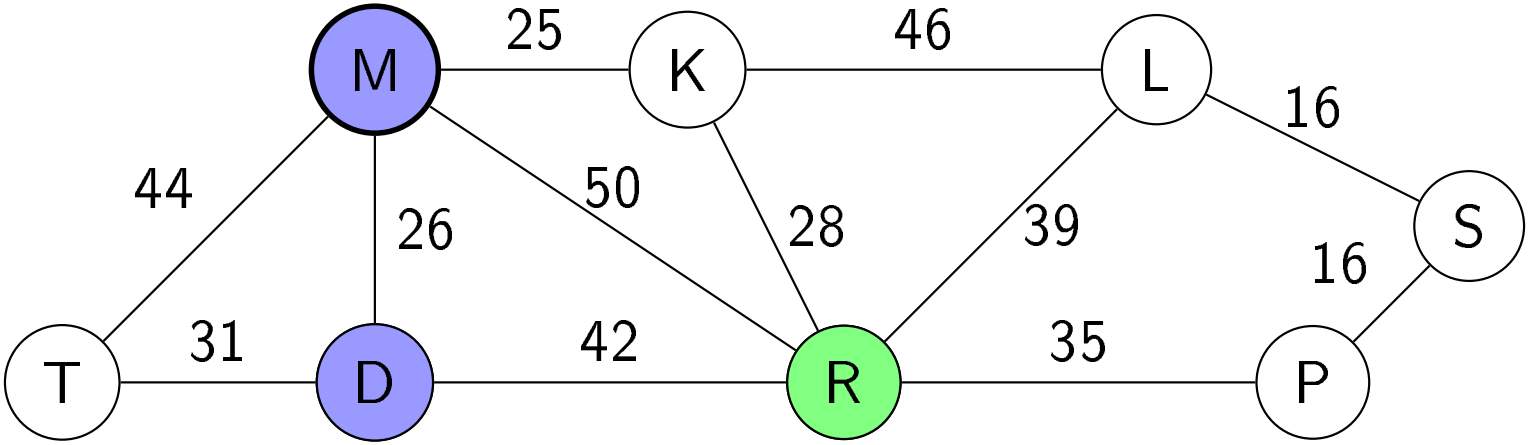
Długość aktualnej ścieżki: $50 + 42 = 92$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



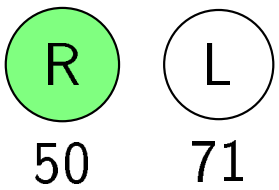
Dijkstra



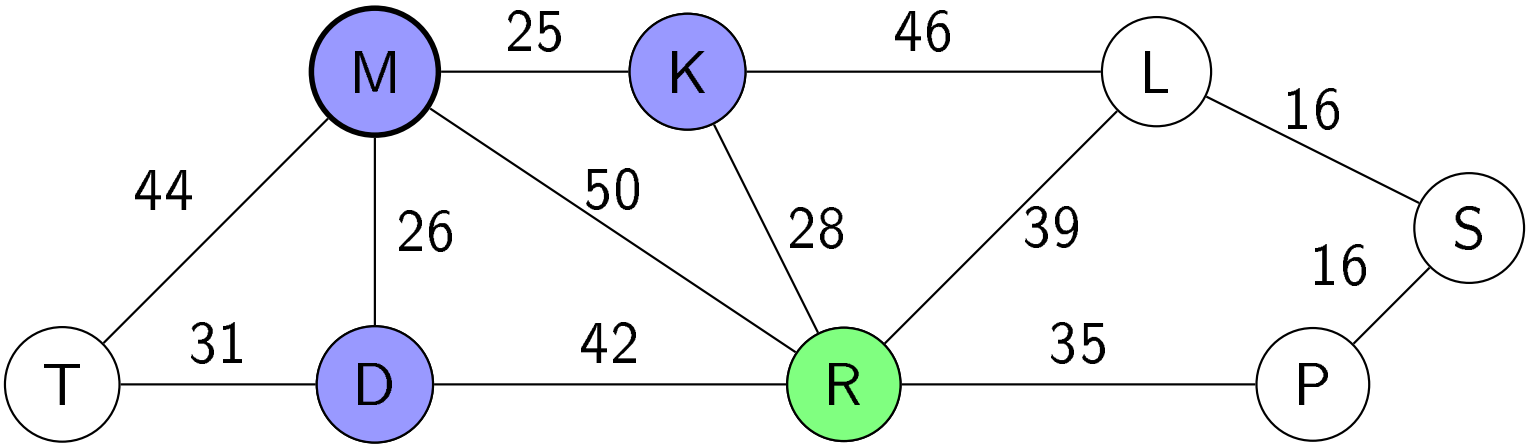
Długość aktualnej ścieżki: $50 + 50 = 100$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



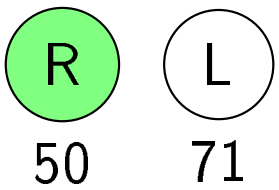
Dijkstra



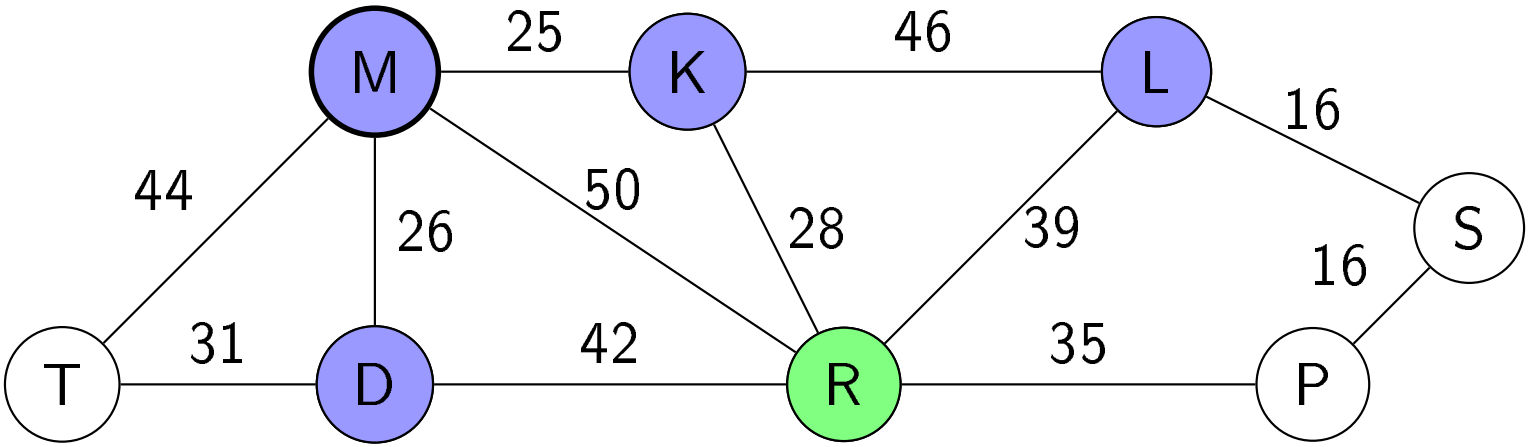
Długość aktualnej ścieżki: $50 + 28 = 78$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



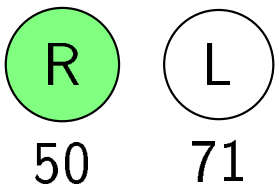
Dijkstra



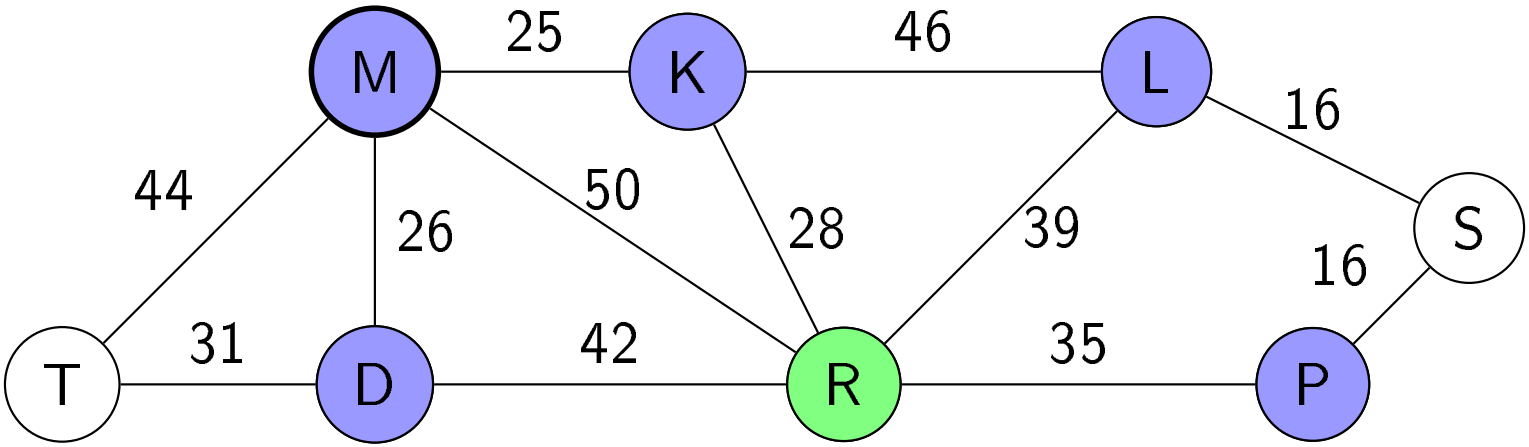
Długość aktualnej ścieżki: $50 + 39 = 89$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



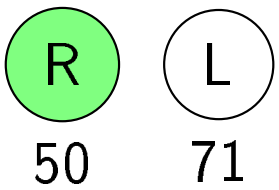
Dijkstra



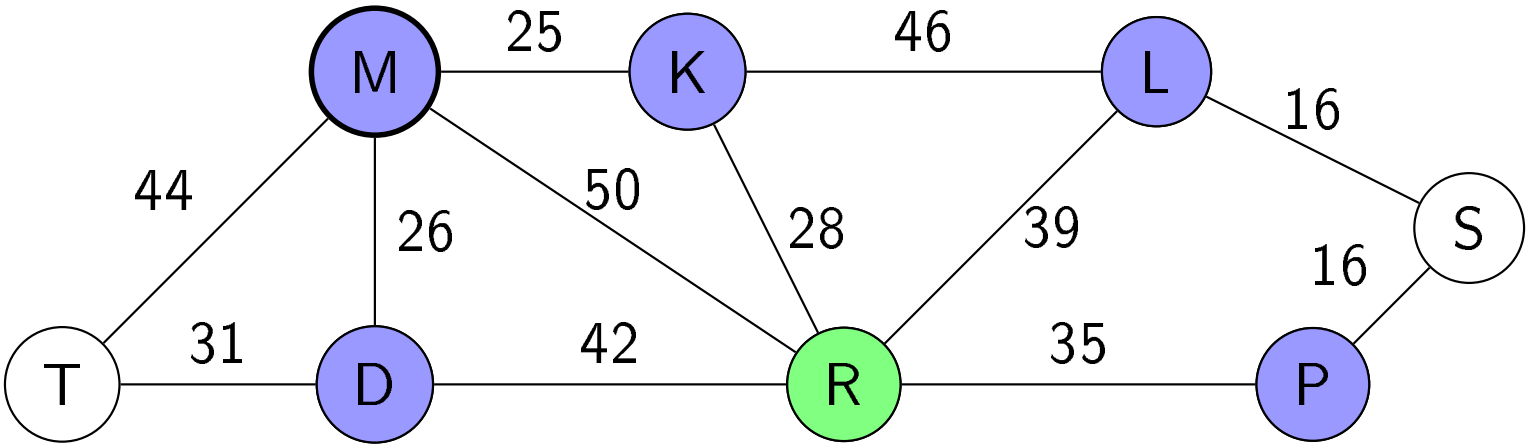
Długość aktualnej ścieżki: $50 + 35 = 85$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71		

Kolejka:



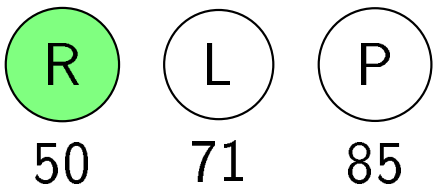
Dijkstra



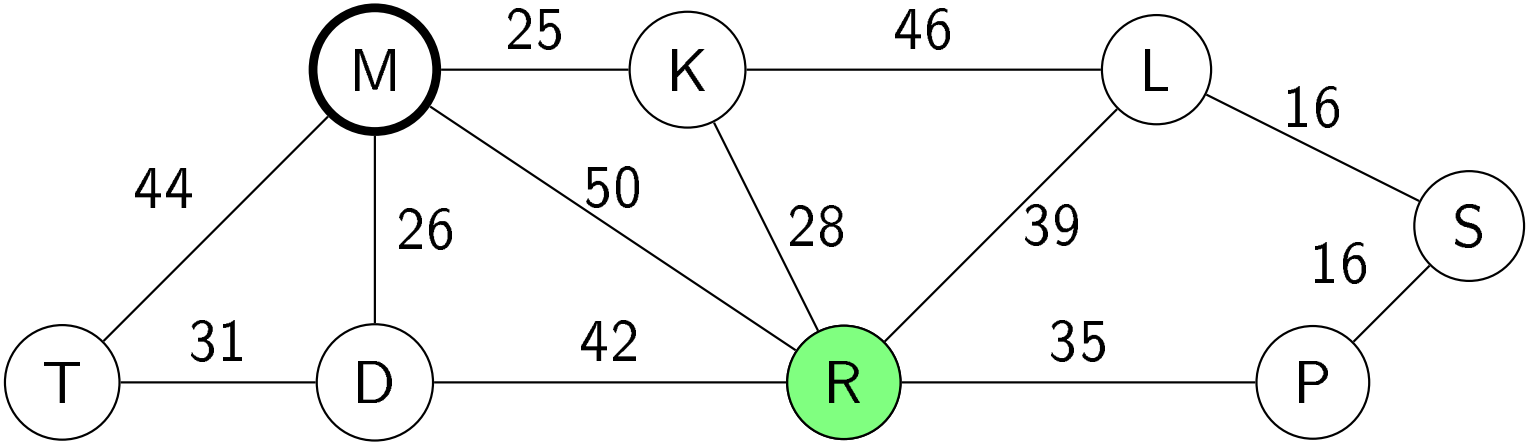
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	

Kolejka:



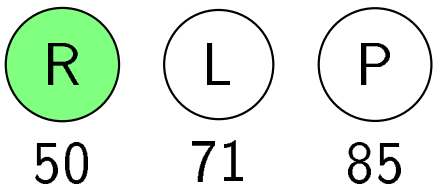
Dijkstra



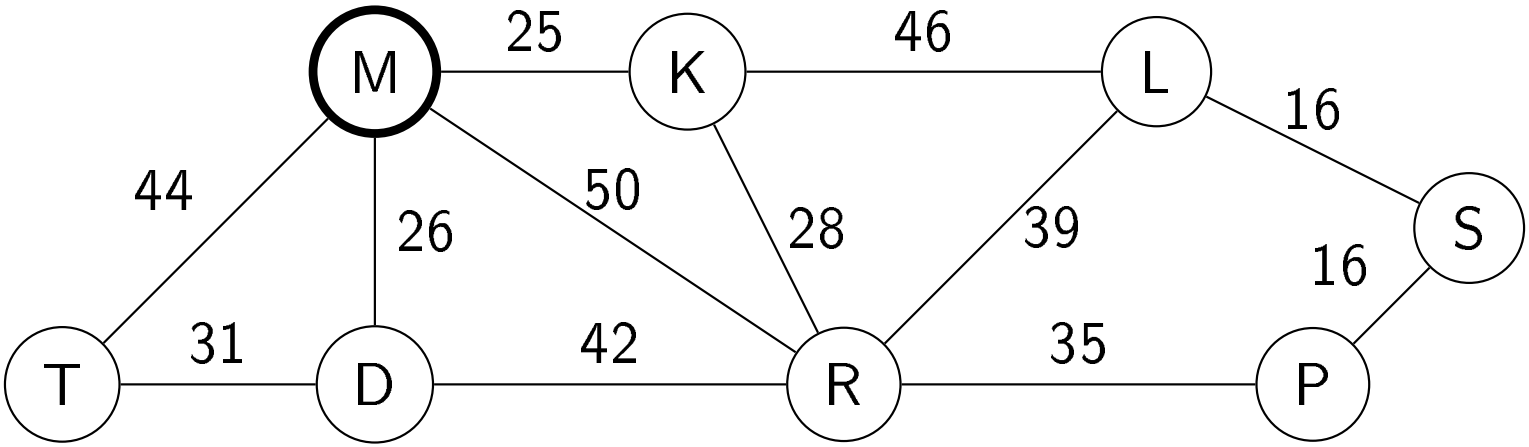
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	

Kolejka:



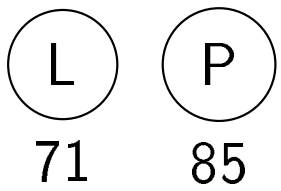
Dijkstra



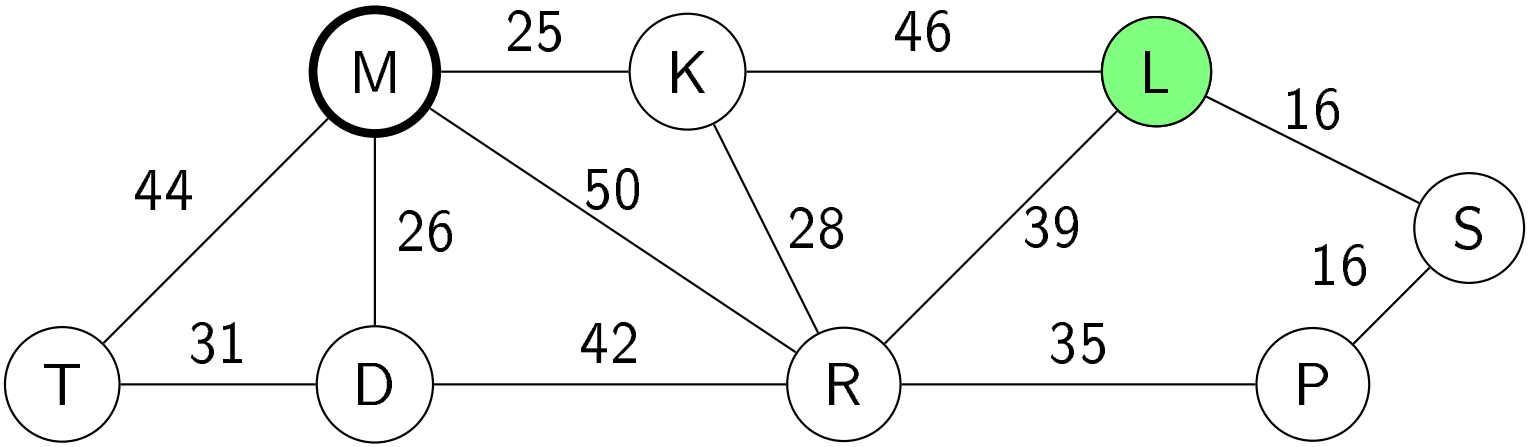
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	

Kolejka:



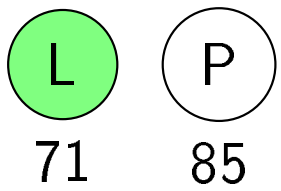
Dijkstra



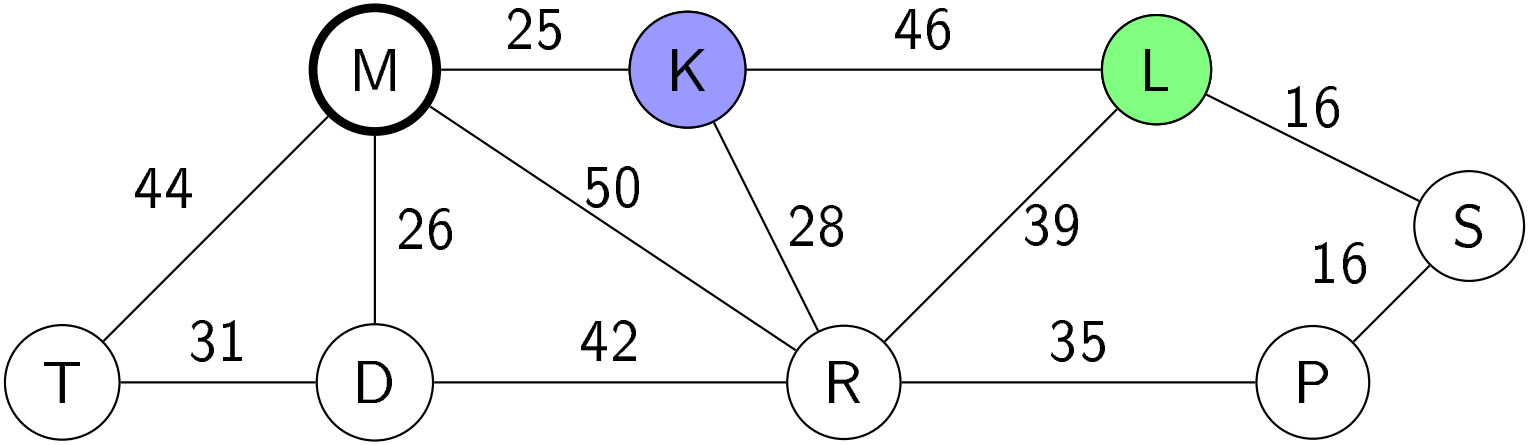
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	

Kolejka:



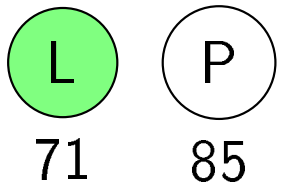
Dijkstra



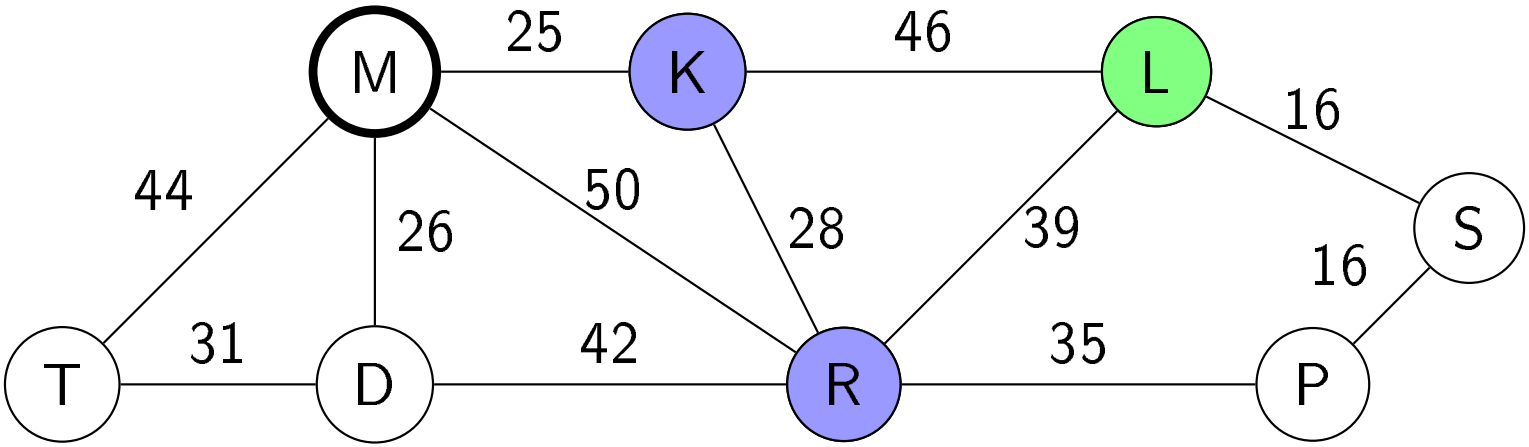
Długość aktualnej ścieżki: $71 + 46 = 117$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	

Kolejka:



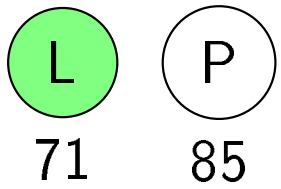
Dijkstra



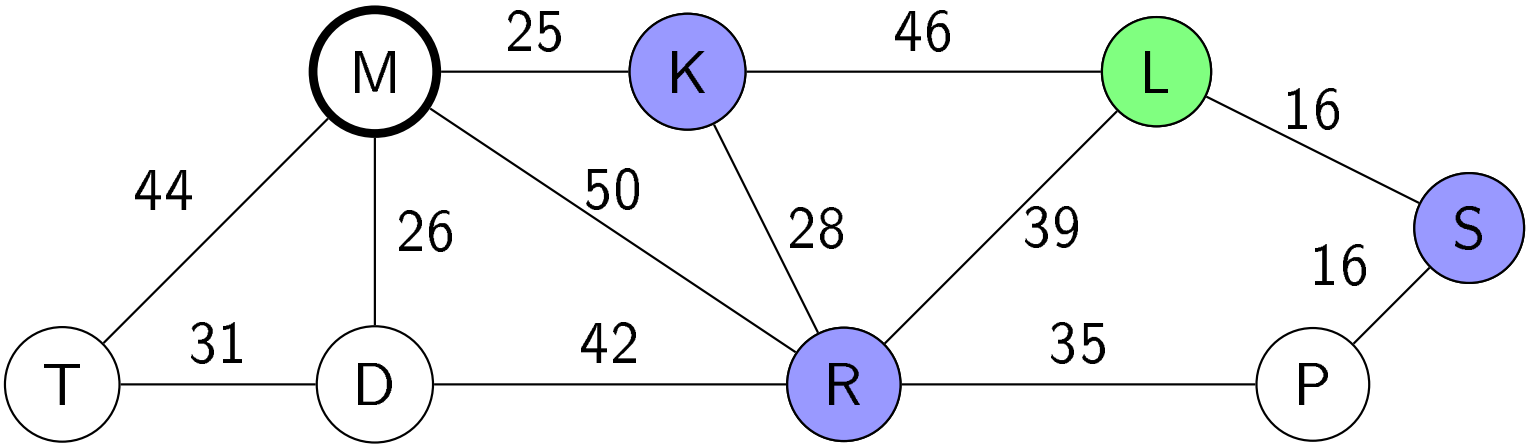
Długość aktualnej ścieżki: $71 + 39 = 110$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	

Kolejka:



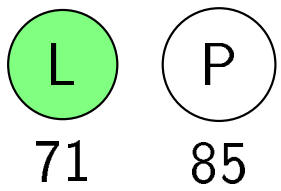
Dijkstra



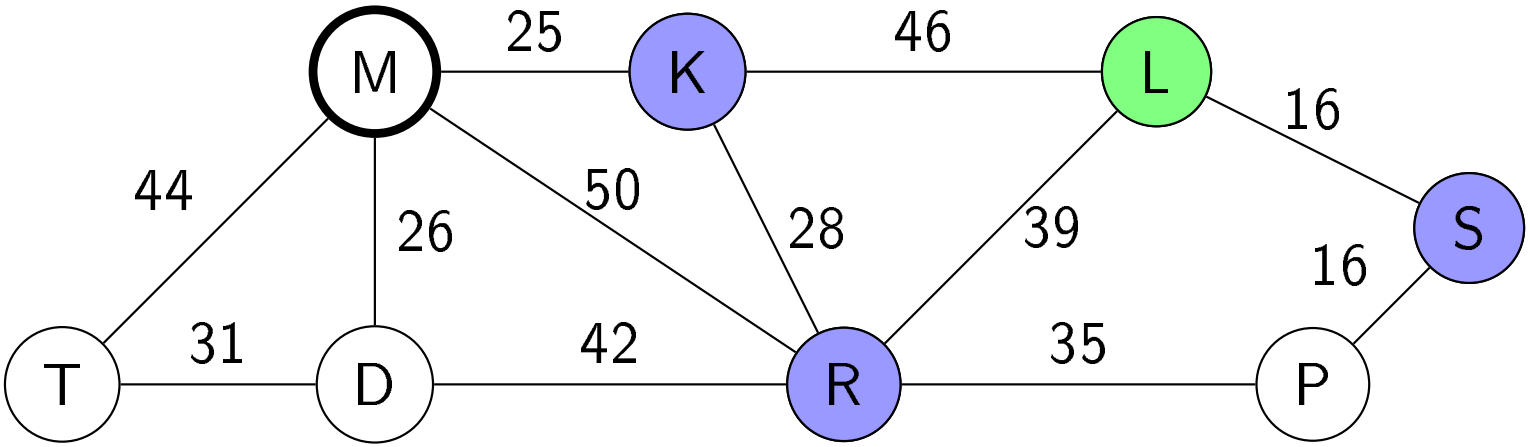
Długość aktualnej ścieżki: $71 + 16 = 87$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	

Kolejka:



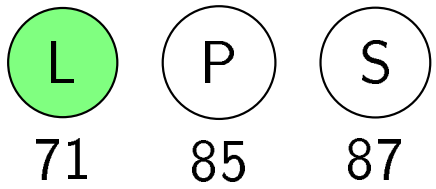
Dijkstra



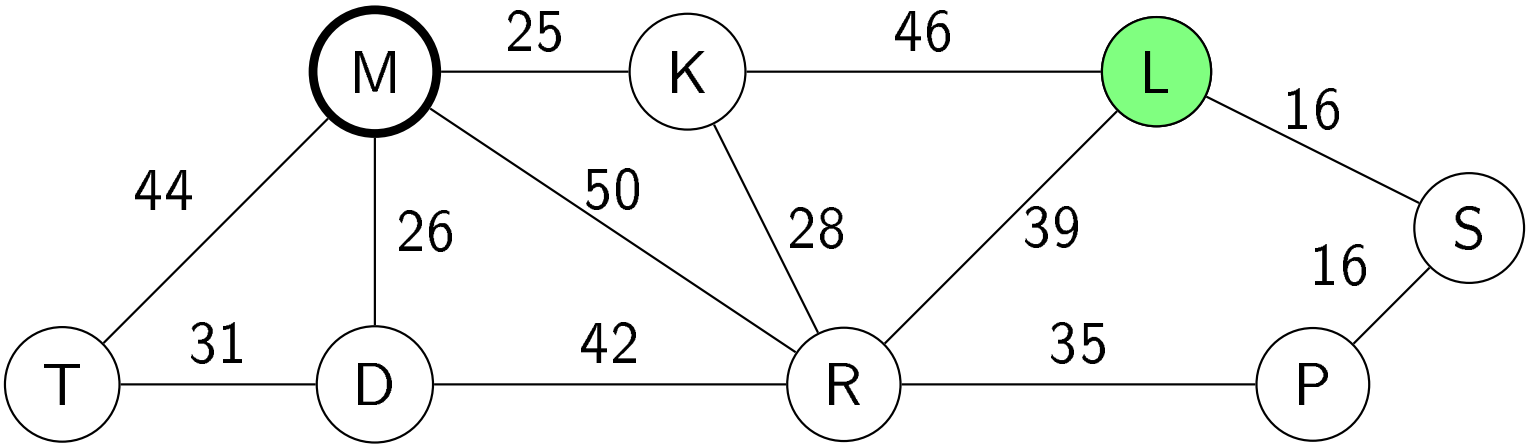
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



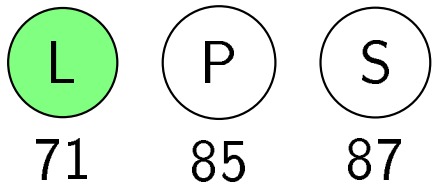
Dijkstra



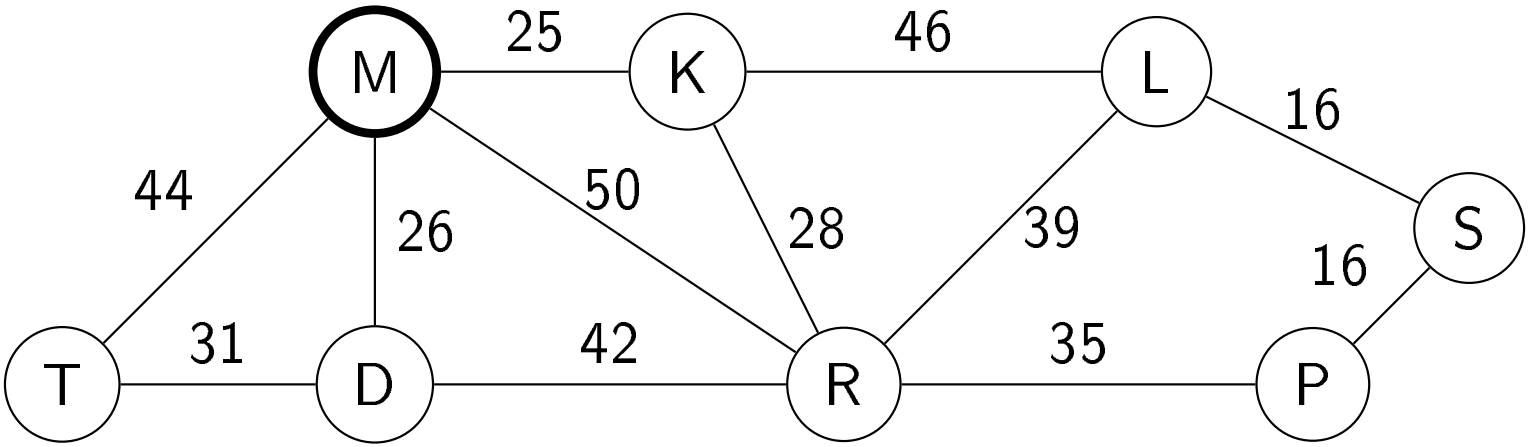
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



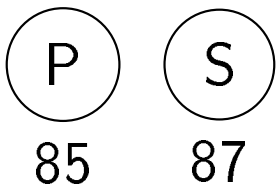
Dijkstra



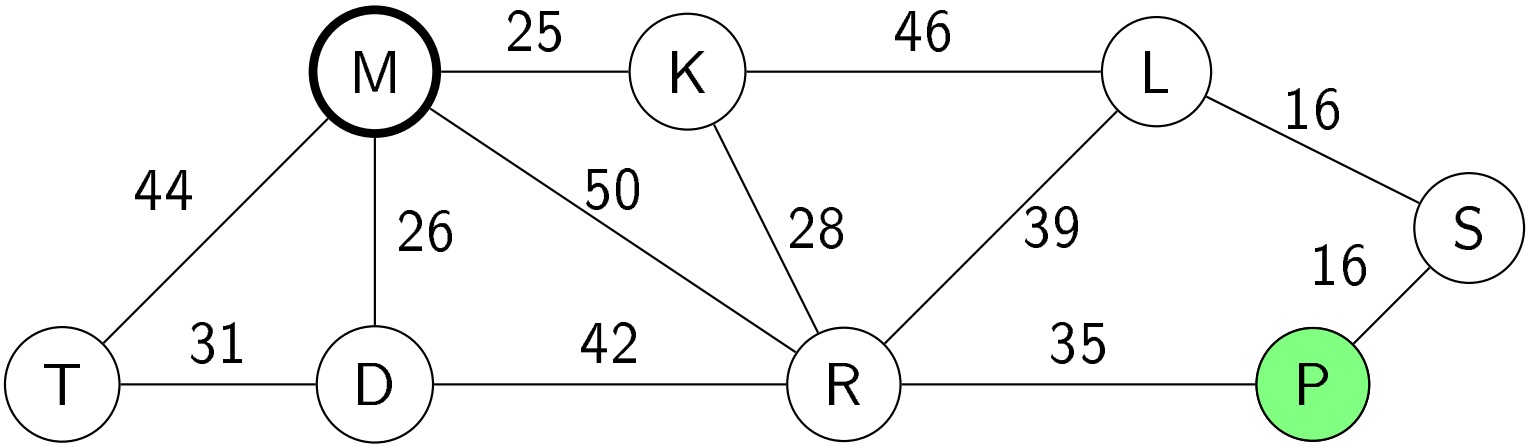
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



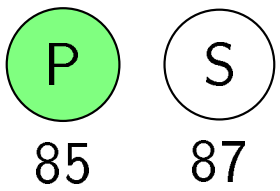
Dijkstra



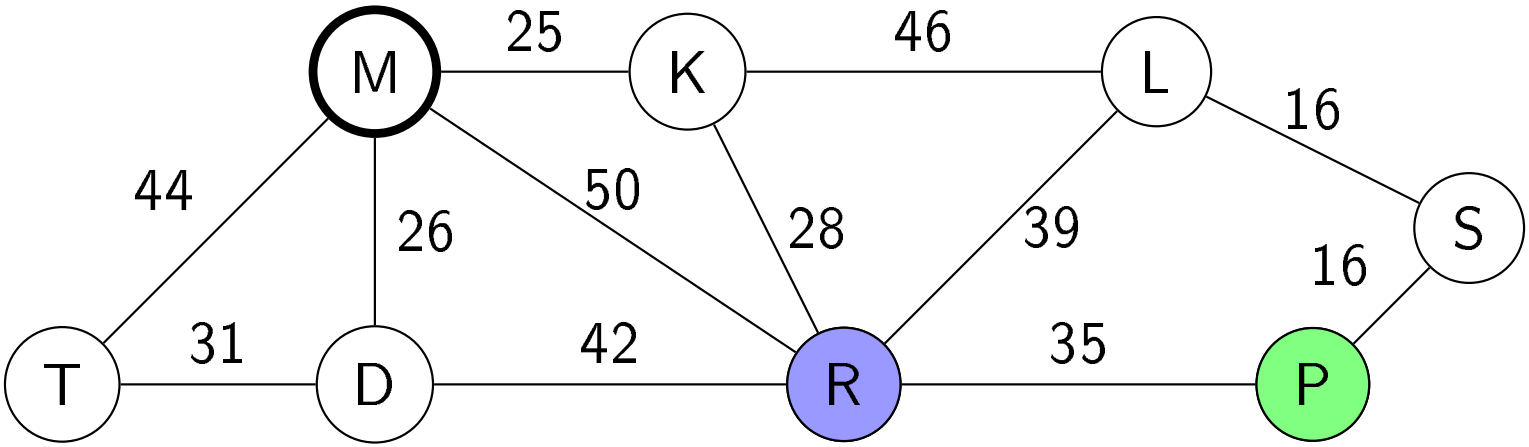
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



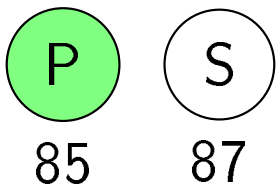
Dijkstra



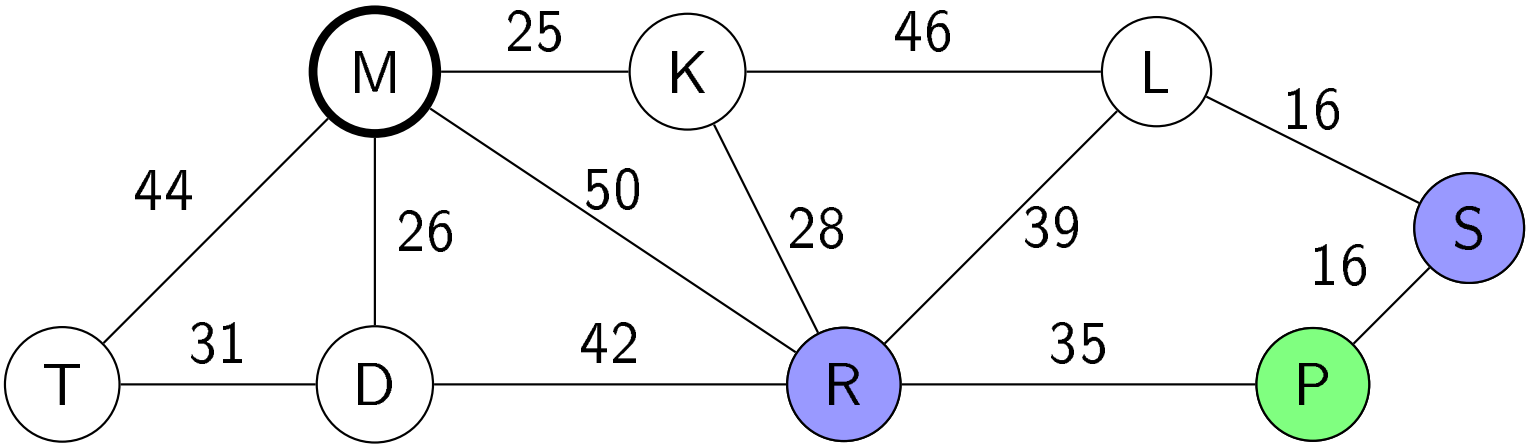
Długość aktualnej ścieżki: $85 + 35 = 120$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



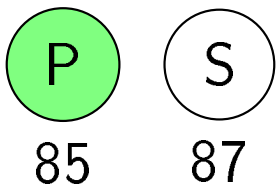
Dijkstra



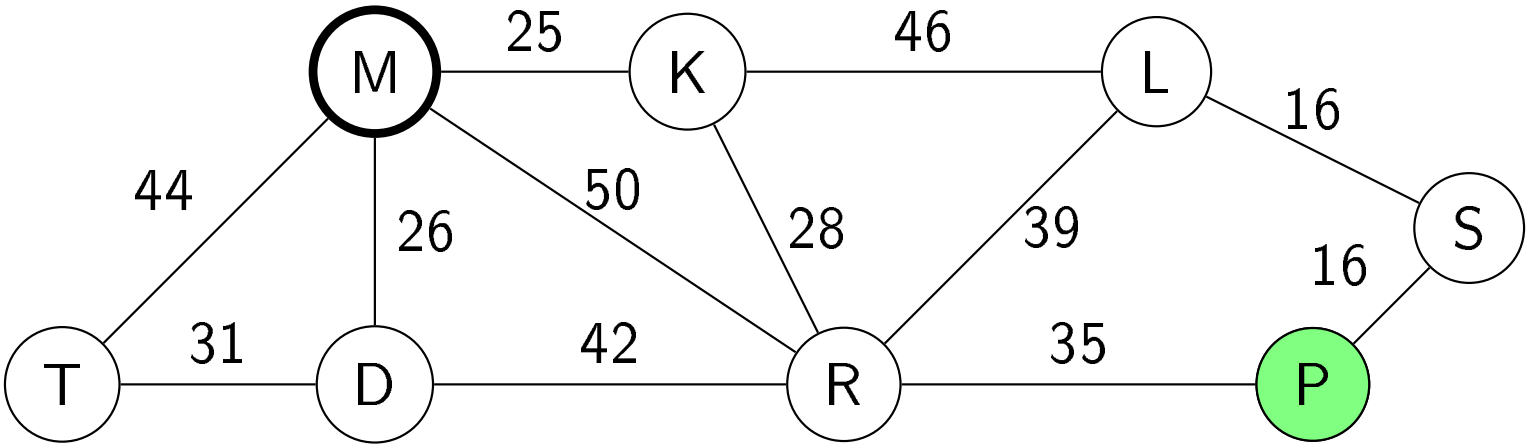
Długość aktualnej ścieżki: $85 + 16 = 101$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



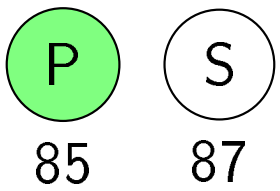
Dijkstra



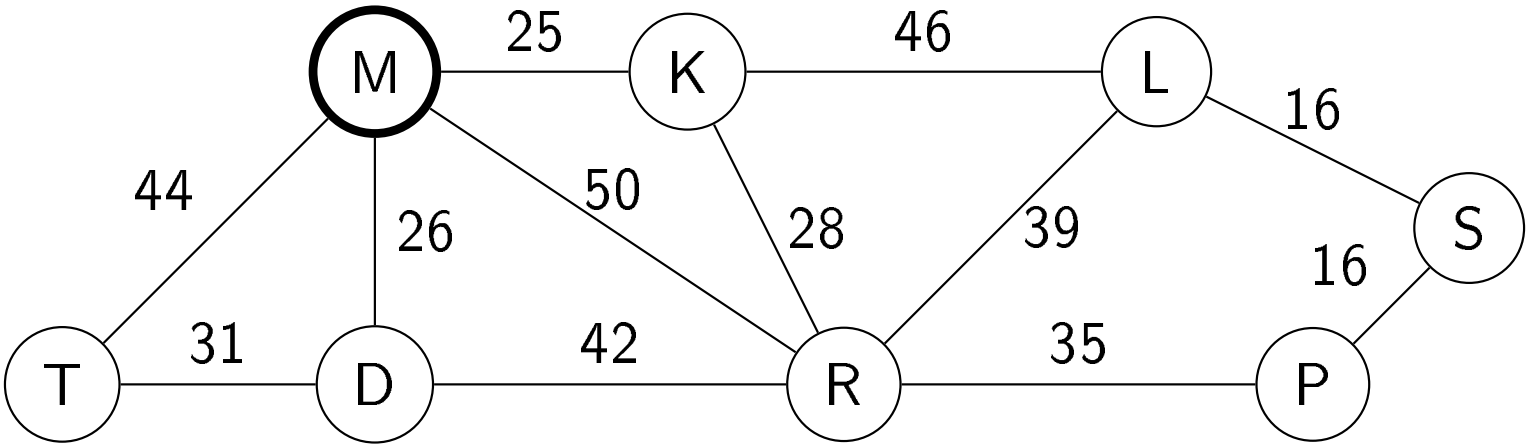
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



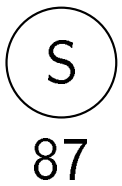
Dijkstra



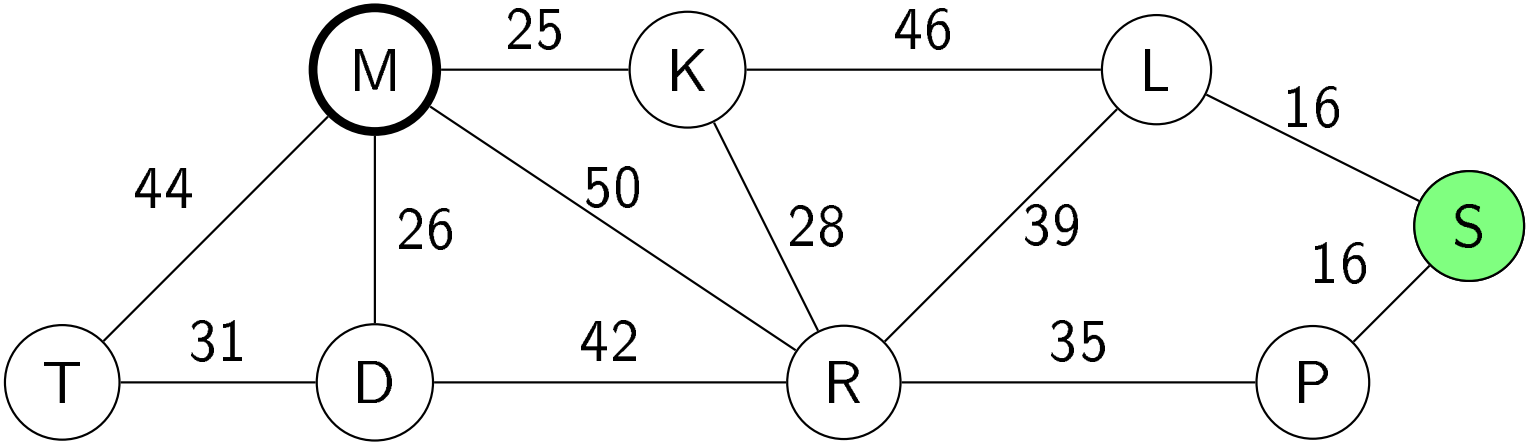
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



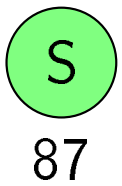
Dijkstra



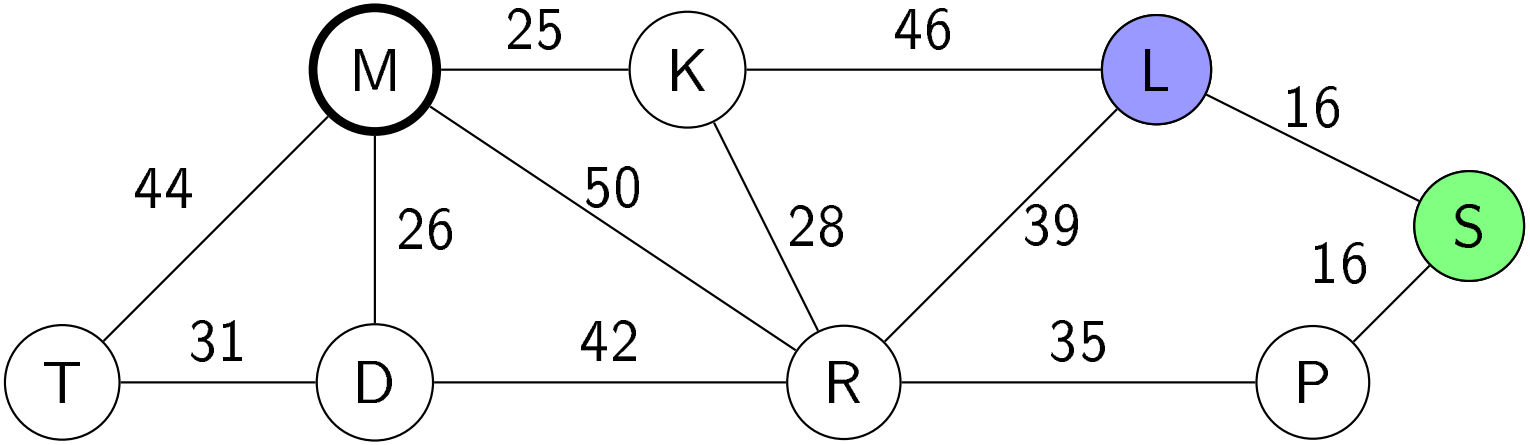
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



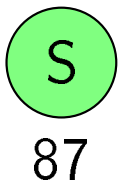
Dijkstra



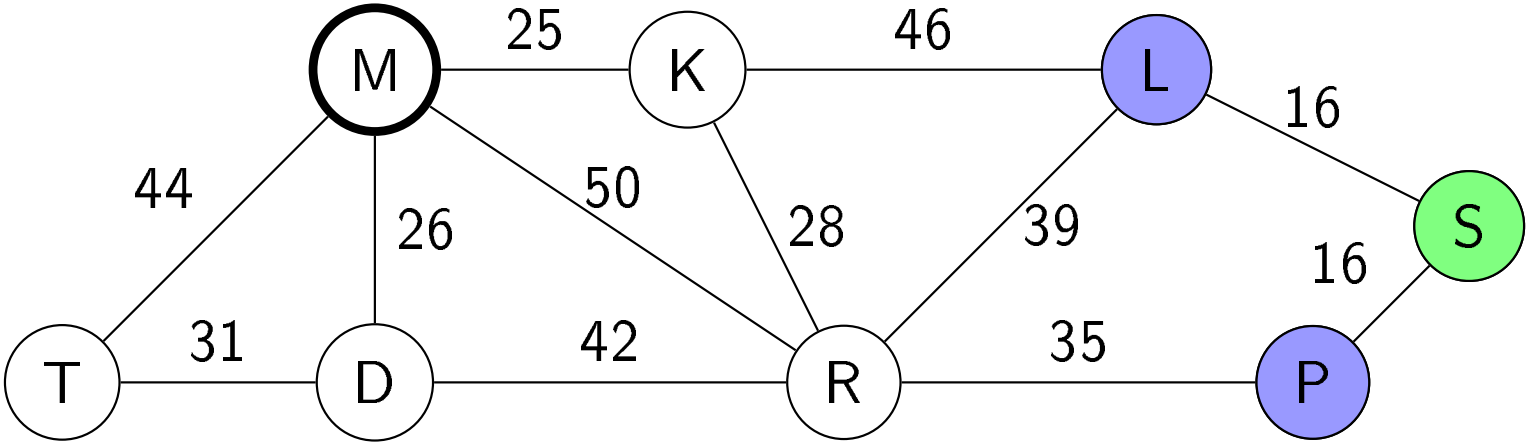
Długość aktualnej ścieżki: $87 + 16 = 103$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



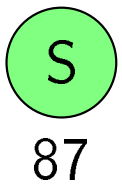
Dijkstra



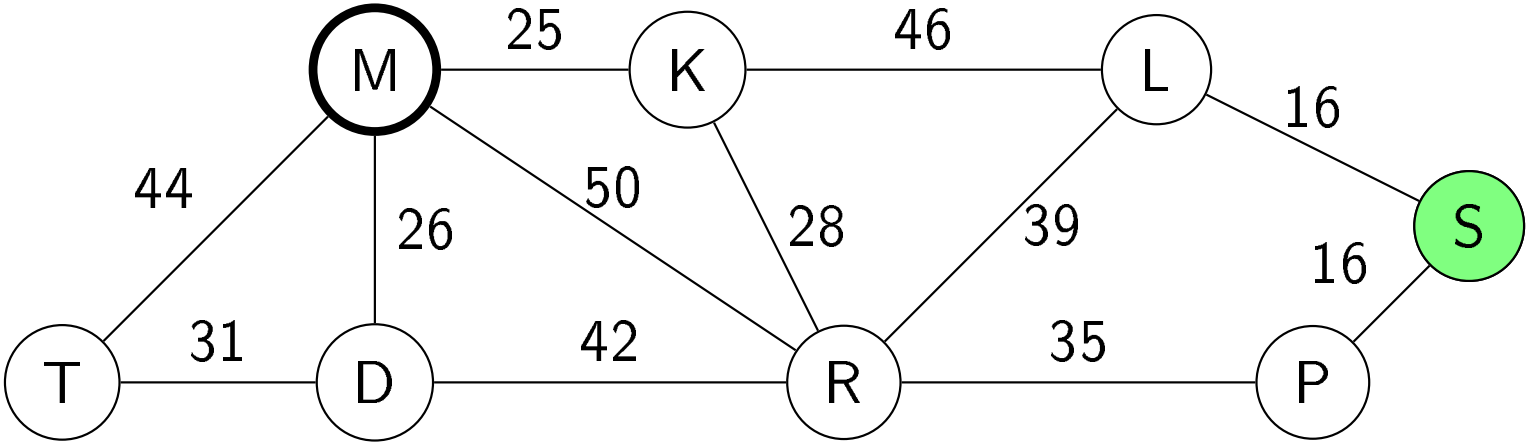
Długość aktualnej ścieżki: $87 + 16 = 103$

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



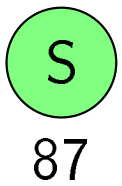
Dijkstra



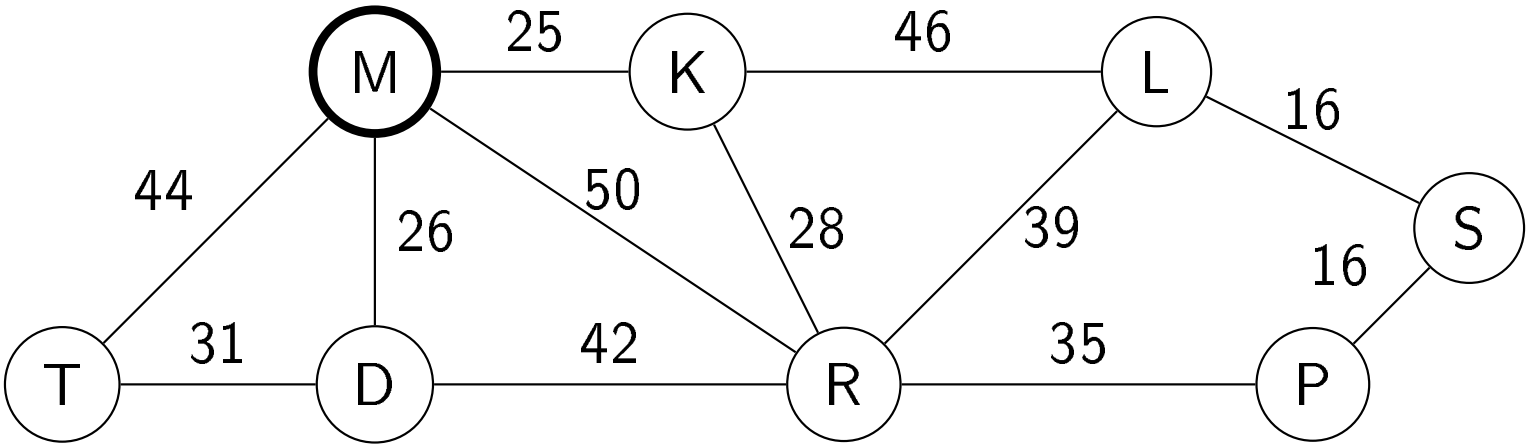
Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:



Dijkstra



Długość aktualnej ścieżki:

wierzchołek	M	T	D	R	K	L	P	S
odległość	0	44	26	50	25	71	85	87

Kolejka:

Algorytm Dijkstry

Wejście: Ważony graf $\mathtt{G}$, wierzchołek startowy $\mathtt{s}$

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona lub `odległość[u] > odległość[w] + krawędź[w, u]`:

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona lub `odległość[u] > odległość[w] + krawędź[w, u]`:
      2. Ustaw ją: `odległość[u] = odległość[w] + krawędź[w, u]`.

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona lub `odległość[u] > odległość[w] + krawędź[w, u]`:
      2. Ustaw ją: `odległość[u] = odległość[w] + krawędź[w, u]`.
      3. Wrzuć `u` do kolejki lub zaktualizuj kolejkę.

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona lub `odległość[u] > odległość[w] + krawędź[w, u]`:
      2. Ustaw ją: `odległość[u] = odległość[w] + krawędź[w, u]`.
      3. Wrzuć `u` do kolejki lub zaktualizuj kolejkę.
      4. W przeciwnym wypadku nic nie rób.

Algorytm Dijkstry

Wejście: Ważony graf `G`, wierzchołek startowy `s`

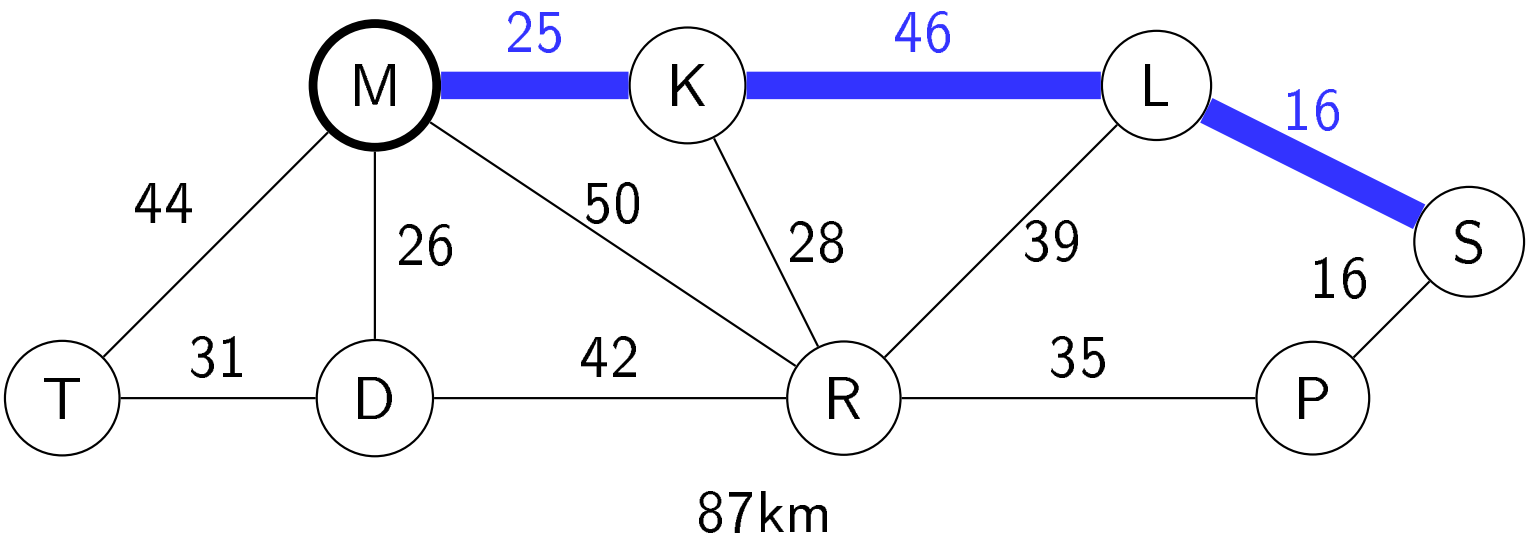
1. Wrzuć `s` do kolejki priorytetowej `Q`.
2. Ustaw odległość `s` na 0: `odległość[s] = 0`
3. Dopóki jest coś w kolejce `Q`:
   1. Weź wierzchołek z kolejki, nazwijmy go `w`:
   2. Dla każdego `u` - sąsiada wierzchołka `w` w grafie `G`:
      1. Jeśli odległość `u` nie jest ustawiona lub `odległość[u] > odległość[w] + krawędź[w, u]`:
      2. Ustaw ją: `odległość[u] = odległość[w] + krawędź[w, u]`.
      3. Wrzuć `u` do kolejki lub zaktualizuj kolejkę.
      4. W przeciwnym wypadku nic nie rób.
4. W tablicy `odległość` znajdują się wyniki.

Algorytm Dijkstry

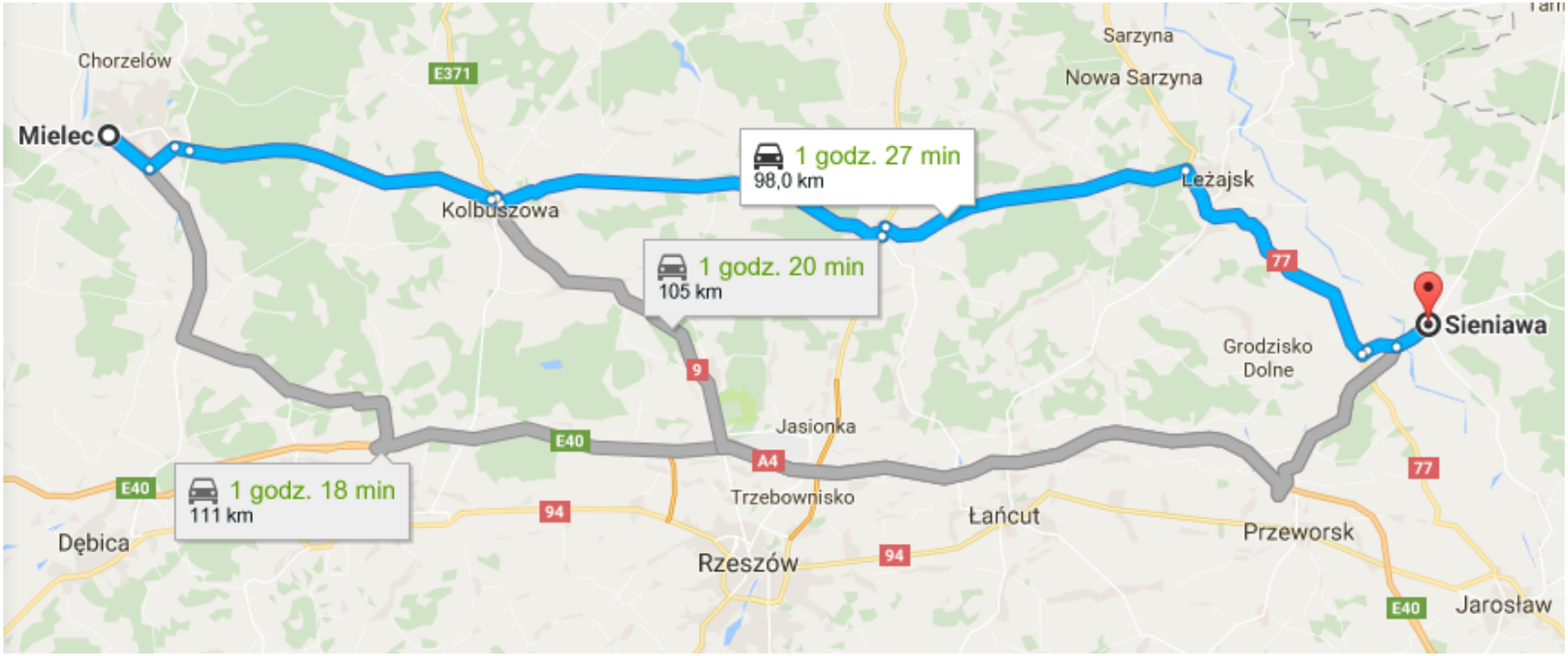
```
def dijkstra(G, s):
    odleglosc = {}
    Q = PriorityQueue()
    Q.put((0,s))
    while len(Q) > 0:
        rank, w = Q.get()
        if rank != odleglosc[w]:
            continue
        for u, koszt_krawedzi in G[w]:
            if u not in odleglosc or odleglosc[u] >
                odleglosc[w] + koszt_krawedzi:
                odleglosc[u] = odleglosc[w] + koszt_krawedzi
                Q.put((odleglosc[u], u))
    return odleglosc
```

Czy mamy rację?

Czy mamy rację?



TAK


Chorzelów
E371
Sarzyna
Nowa Sarzyna
Mielec
1 godz. 27 min
98,0 km
Leżajsk
Kolbuszowa
1 godz. 20 min
105 km
77
Sieniawa
Grodzisko
Dolne
9
Jasionka
E40
A4
77
E40
1 godz. 18 min
111 km
94
Trzebownisko
Łańcut
Przeworsk
Dębica
Rzeszów
94
E40
Jarosław

Czy Google Maps jednak tak działają?

Czy Google Maps jednak tak działają?

1. dodatkowe usprawnienia

Czy Google Maps jednak tak działają?

1. dodatkowe usprawnienia
2. więcej komputerów

Dziękuję za uwagę!