Presentasi

Carles Octavianus

2022

Table of Contents

Ide Pengerjaan

- mengubah list busur (variable ROUTES) menjadi list ketetangaan

Ide Pengerjaan

- mengubah list busur (variable ROUTES) menjadi list ketetangaan
- melakukan breath first search dari node start pada graph (list ketetangaan) tersebut.

Ide Pengerjaan

- mengubah list busur (variable ROUTES) menjadi list ketetangaan
- melakukan breath first search dari node start pada graph (list ketetangaan) tersebut.
- Breadth-first search akan bergerak ke node berikutnya sesuai dengan kenaikan jaraknya (relatif terhadap node start)

Ide Pengerjaan

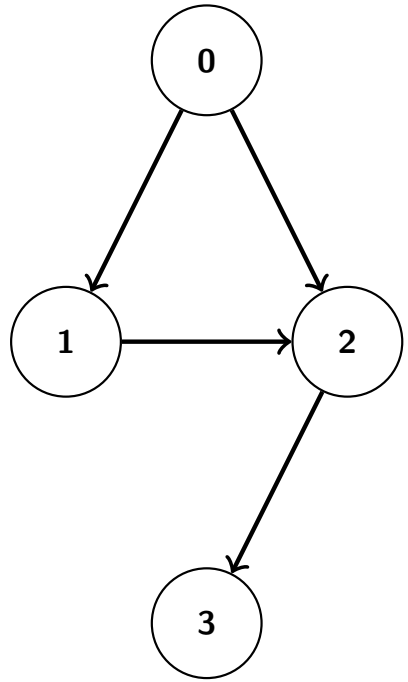
- mengubah list busur (variable ROUTES) menjadi list ketetangaan
- melakukan breath first search dari node start pada graph (list ketetangaan) tersebut.
- Breadth-first search akan bergerak ke node berikutnya sesuai dengan kenaikan jaraknya (relatif terhadap node start)
- dengan begitu, single breadth-first search dari start ke end akan menjamin lintasan tersebut adalah lintasan terpendeknya.

Ide Pengerjaan

- mengubah list busur (variable ROUTES) menjadi list ketetangaan
- melakukan breath first search dari node start pada graph (list ketetangaan) tersebut.
- Breadth-first search akan bergerak ke node berikutnya sesuai dengan kenaikan jaraknya (relatif terhadap node start)
- dengan begitu, single breadth-first search dari start ke end akan menjamin lintasan tersebut adalah lintasan terpendeknya.
- selama proses BFS kita bisa mencatat jarak relatif node lain terhadap node start.

List Busur

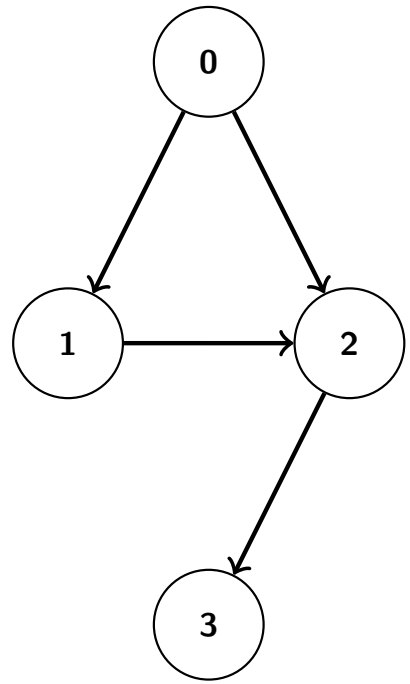
```
ROUTES=[[0,1],[0,2],[1,2],[2,3]]
```



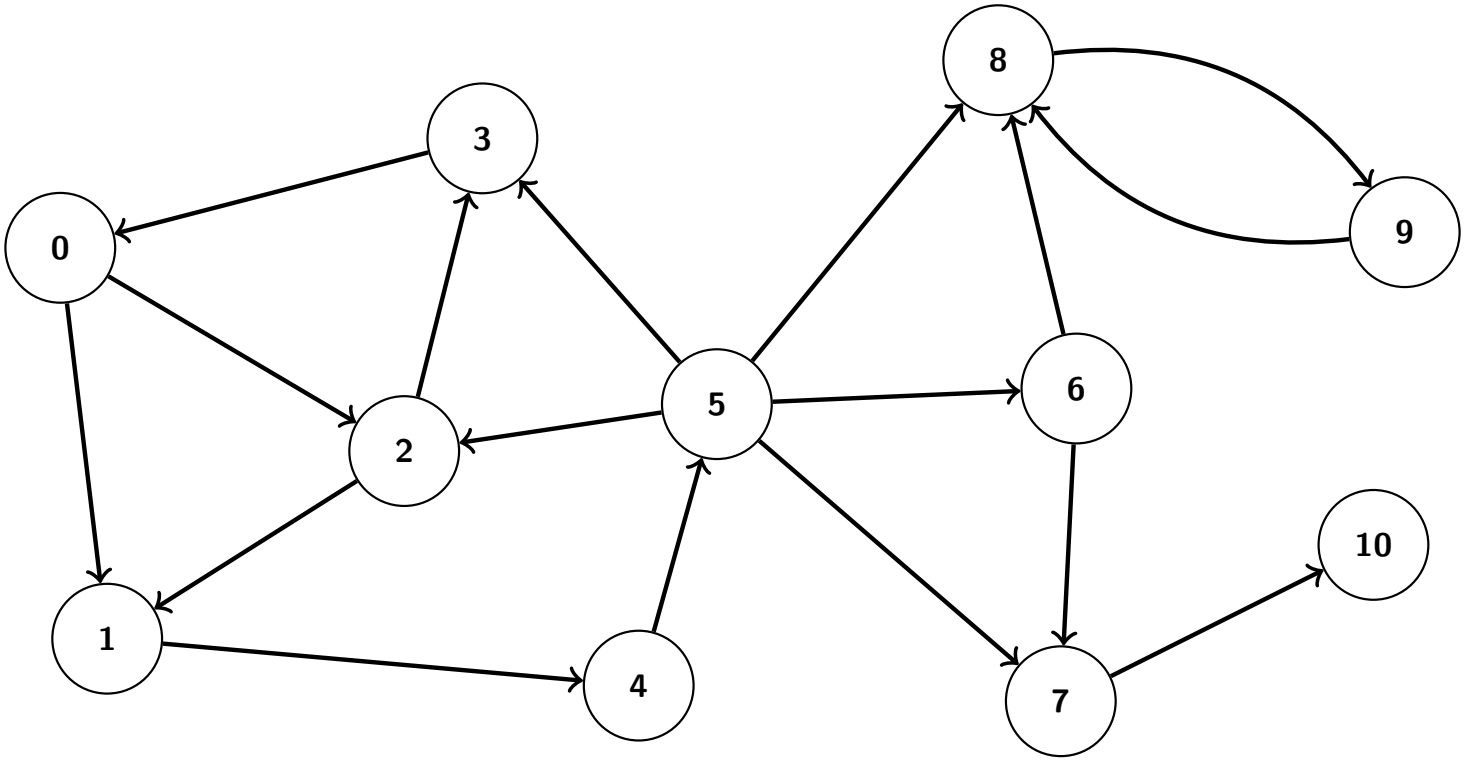
List Ketetangaan

```
0: 1,2
1: 2
2: 3
3:
```

```
adj=defaultdict(list)
for u,v in ROUTES:
    adj[u]+=[v]
```



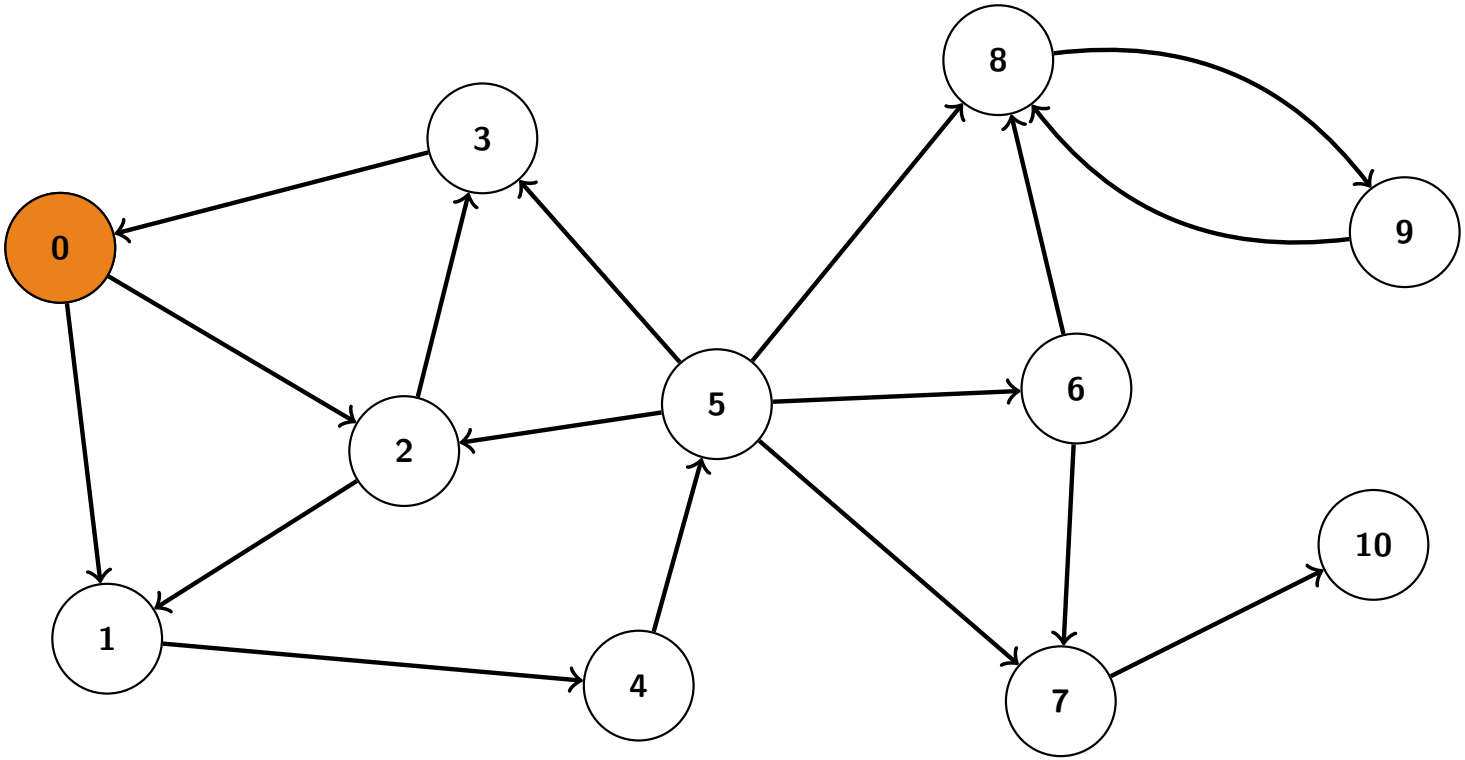
Breadth-first search



Queue:

	0	1	2	3	4	5	6	7	8	9	10
marked	0	0	0	0	0	0	0	0	0	0	0

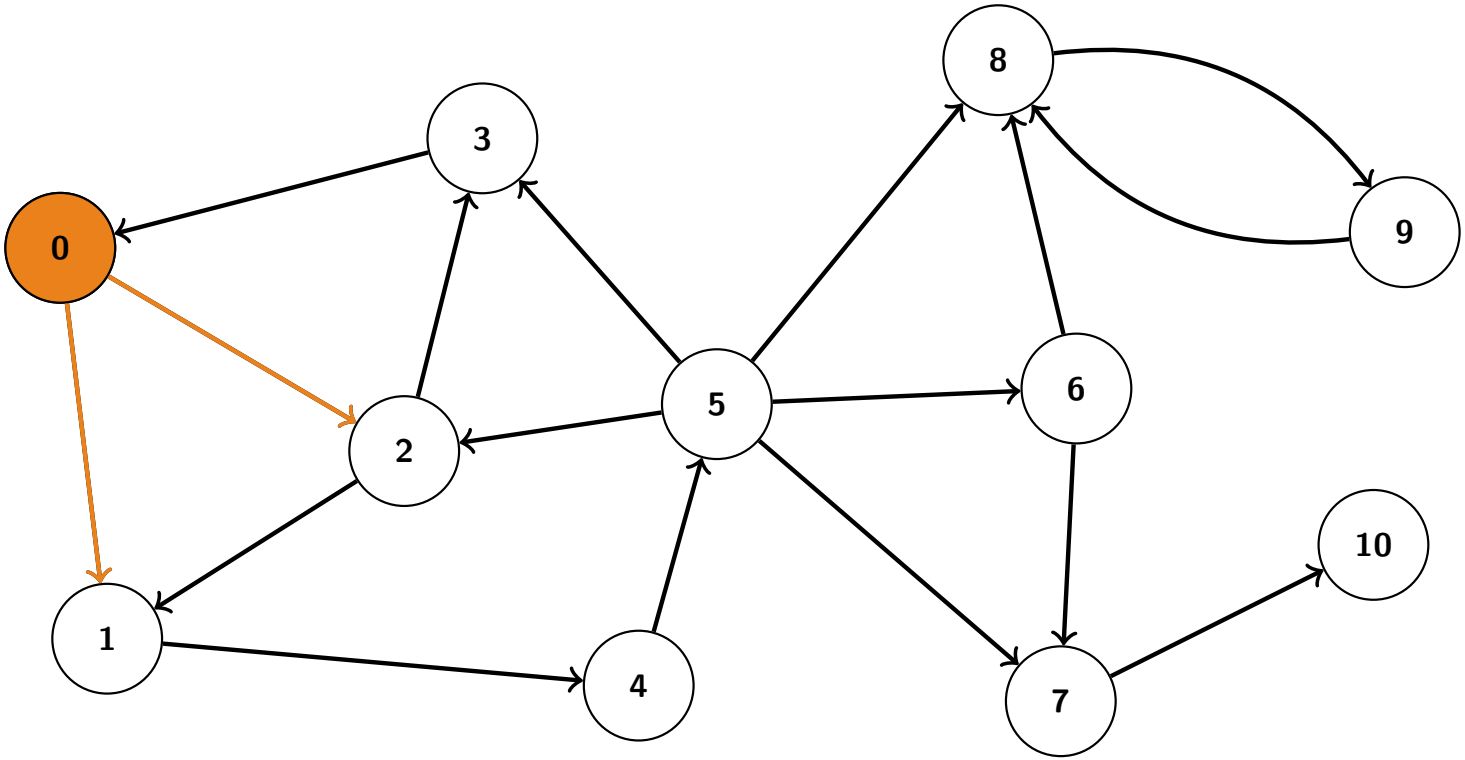
Breadth-first search



Queue: 0

	0	1	2	3	4	5	6	7	8	9	10
marked	1	0	0	0	0	0	0	0	0	0	0

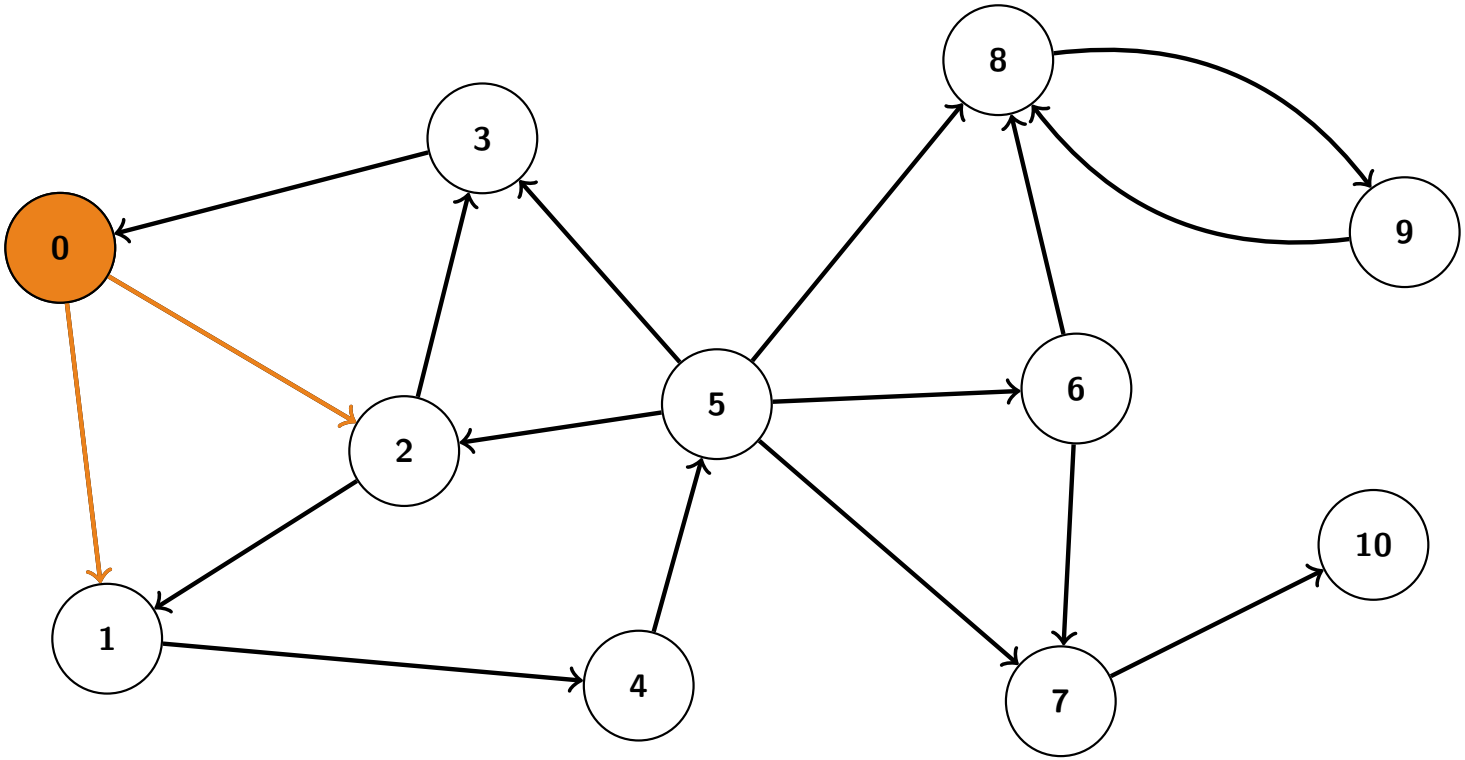
Breadth-first search



Queue: 0

	0	1	2	3	4	5	6	7	8	9	10
marked	1	0	0	0	0	0	0	0	0	0	0

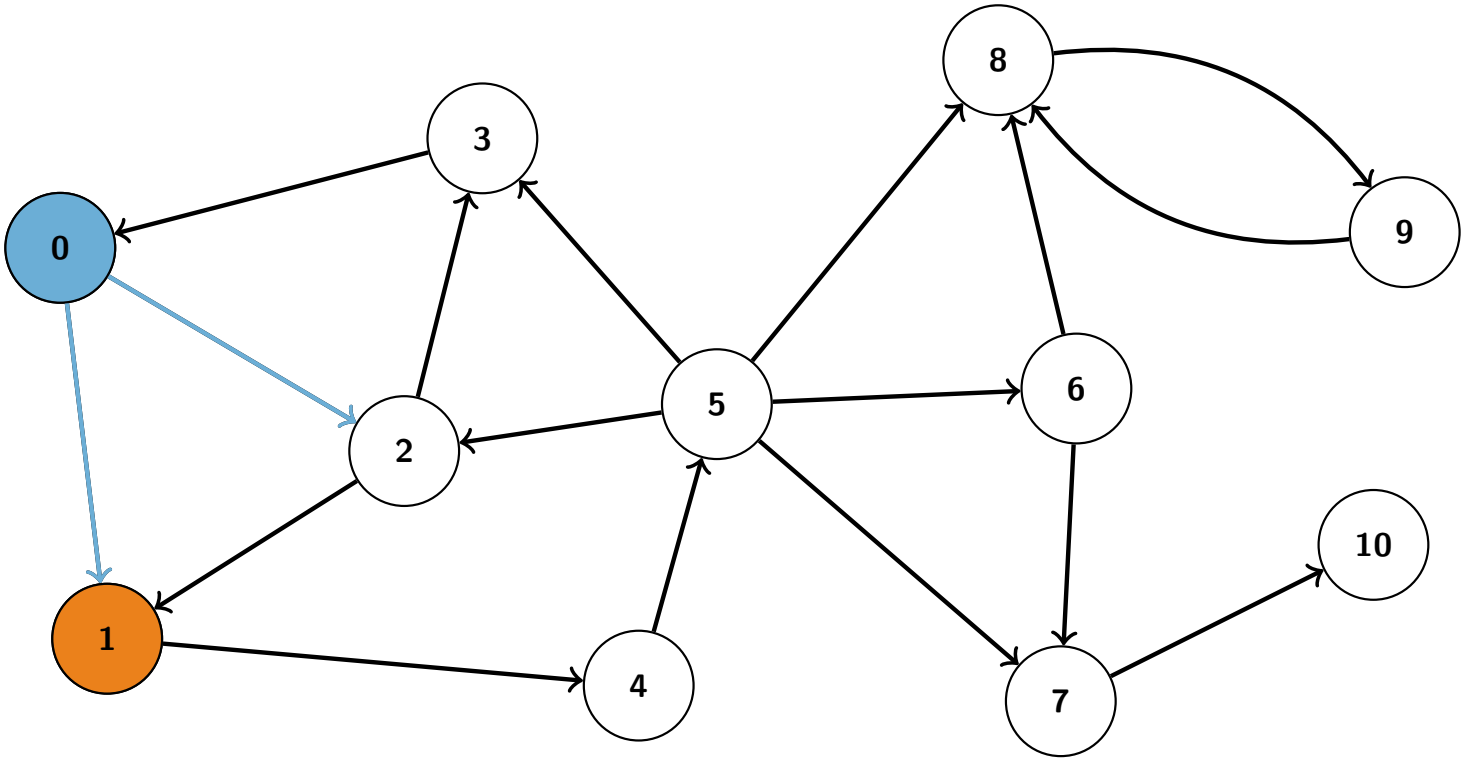
Breadth-first search



Queue: 0 1 2

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	0	0	0	0	0	0	0	0

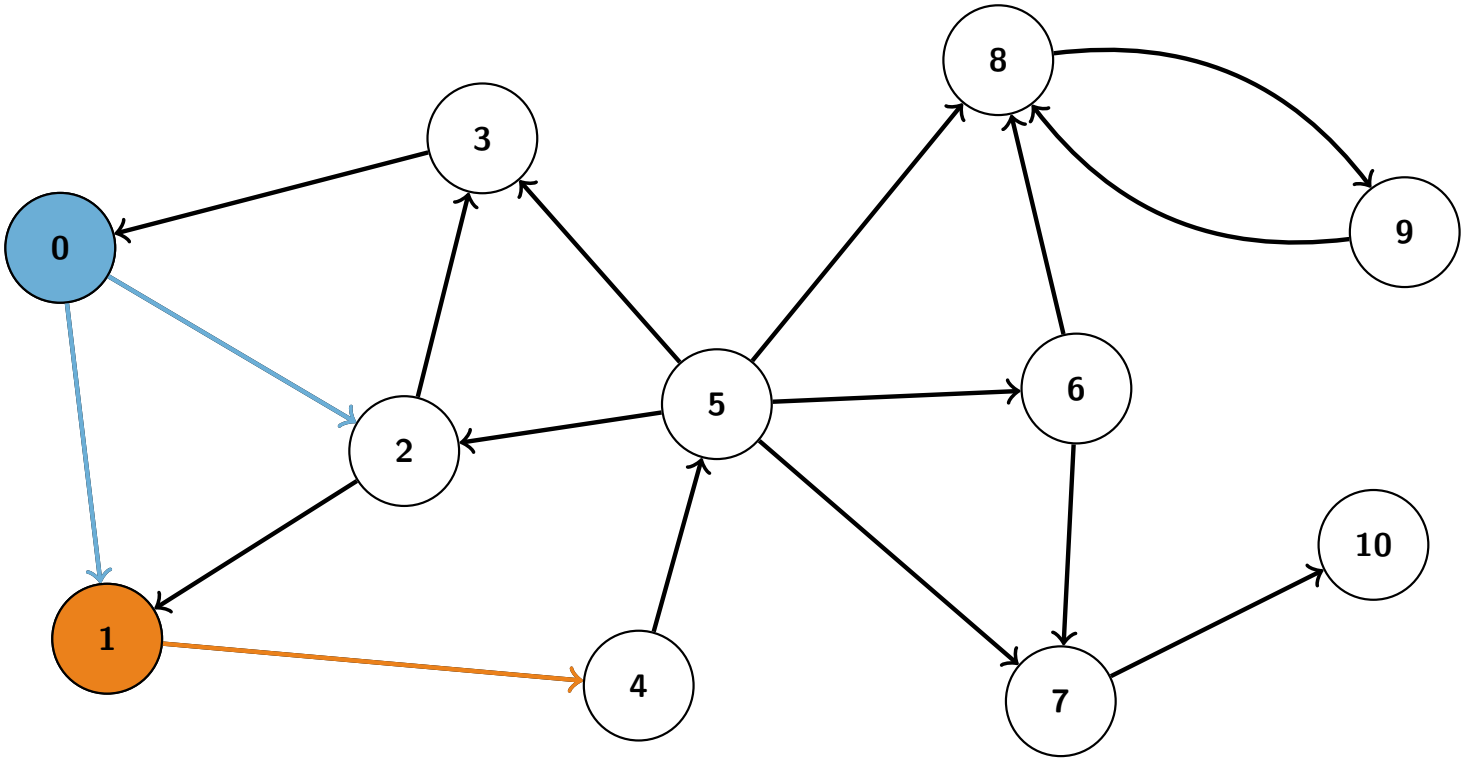
Breadth-first search



Queue: 1 2

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	0	0	0	0	0	0	0	0

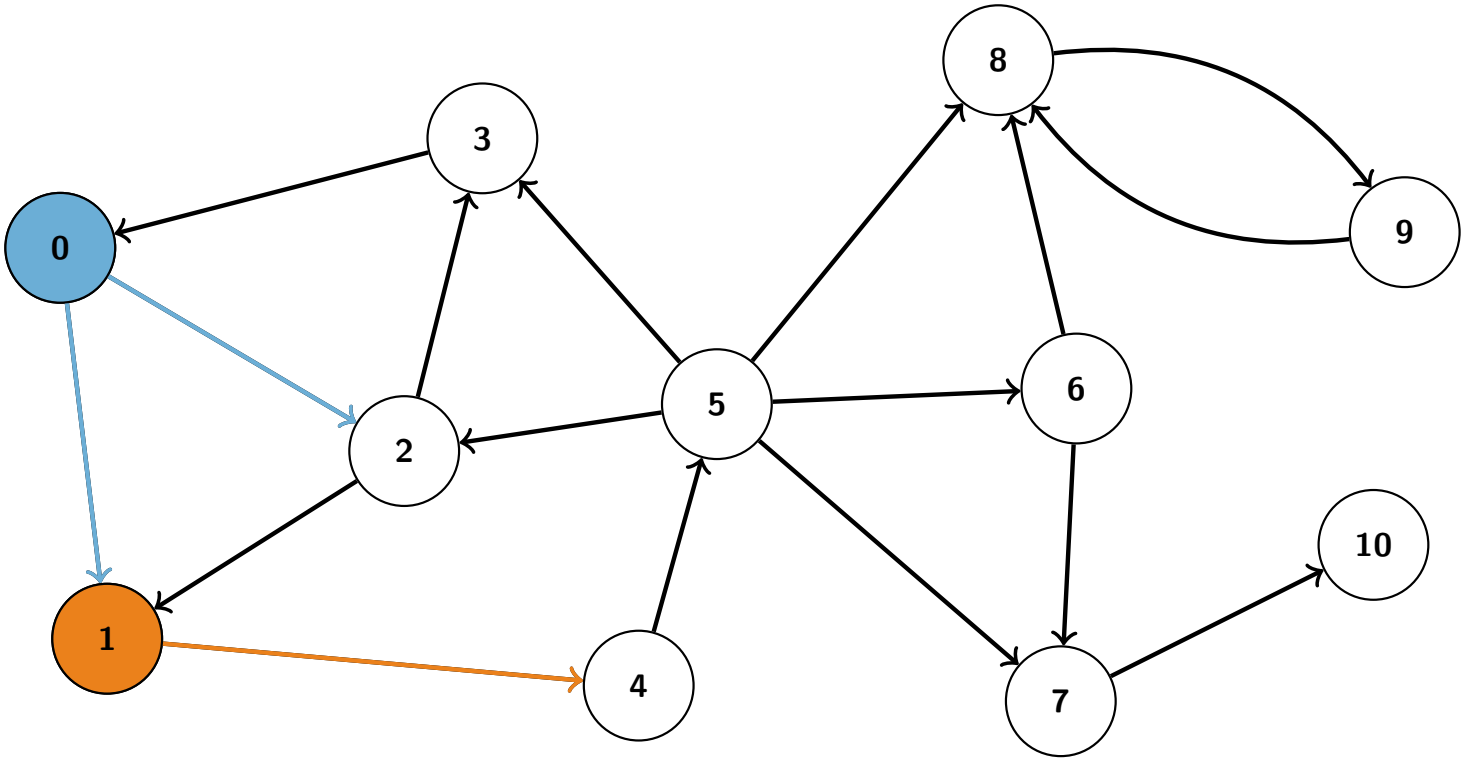
Breadth-first search



Queue: 1 2

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	0	0	0	0	0	0	0	0

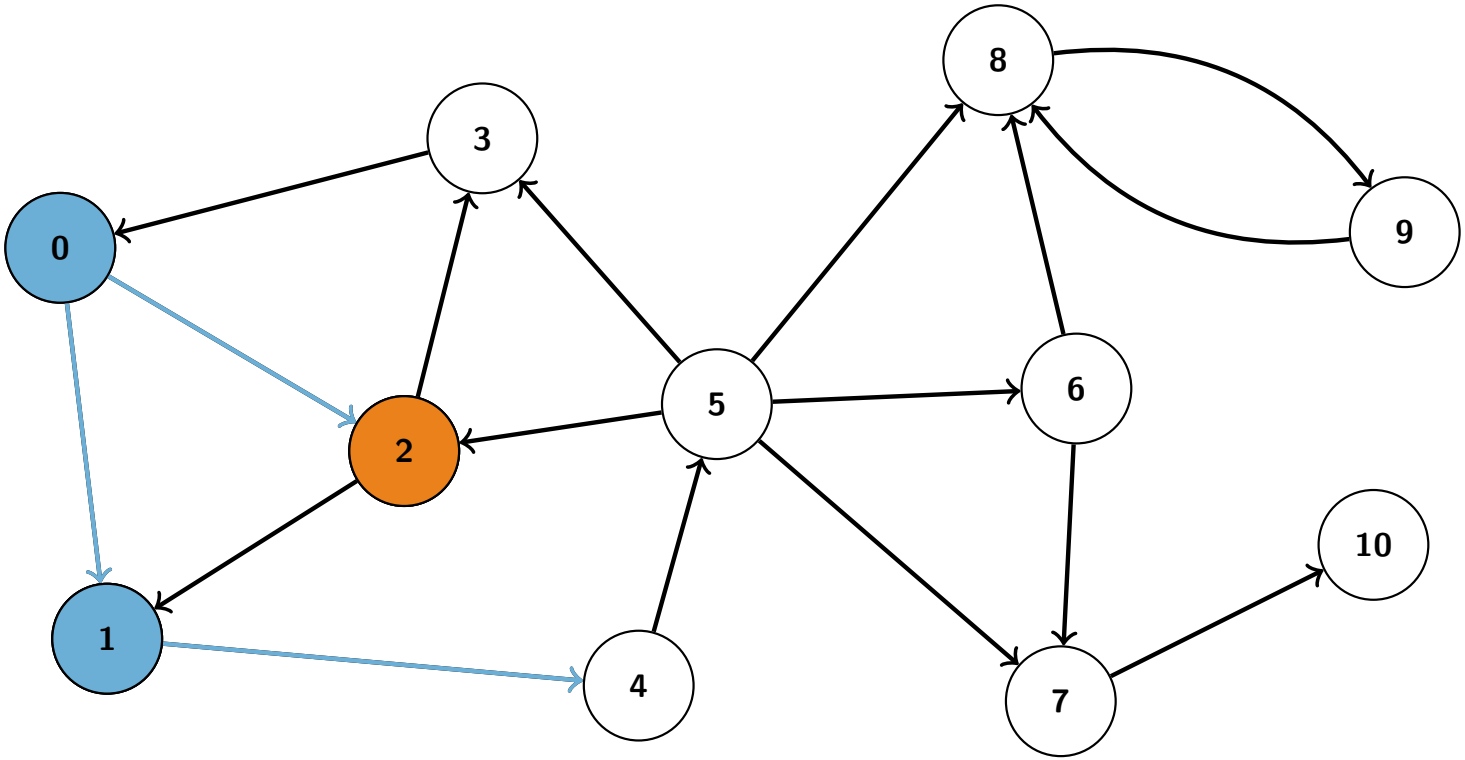
Breadth-first search



Queue: 1 2 4

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	0	1	0	0	0	0	0	0

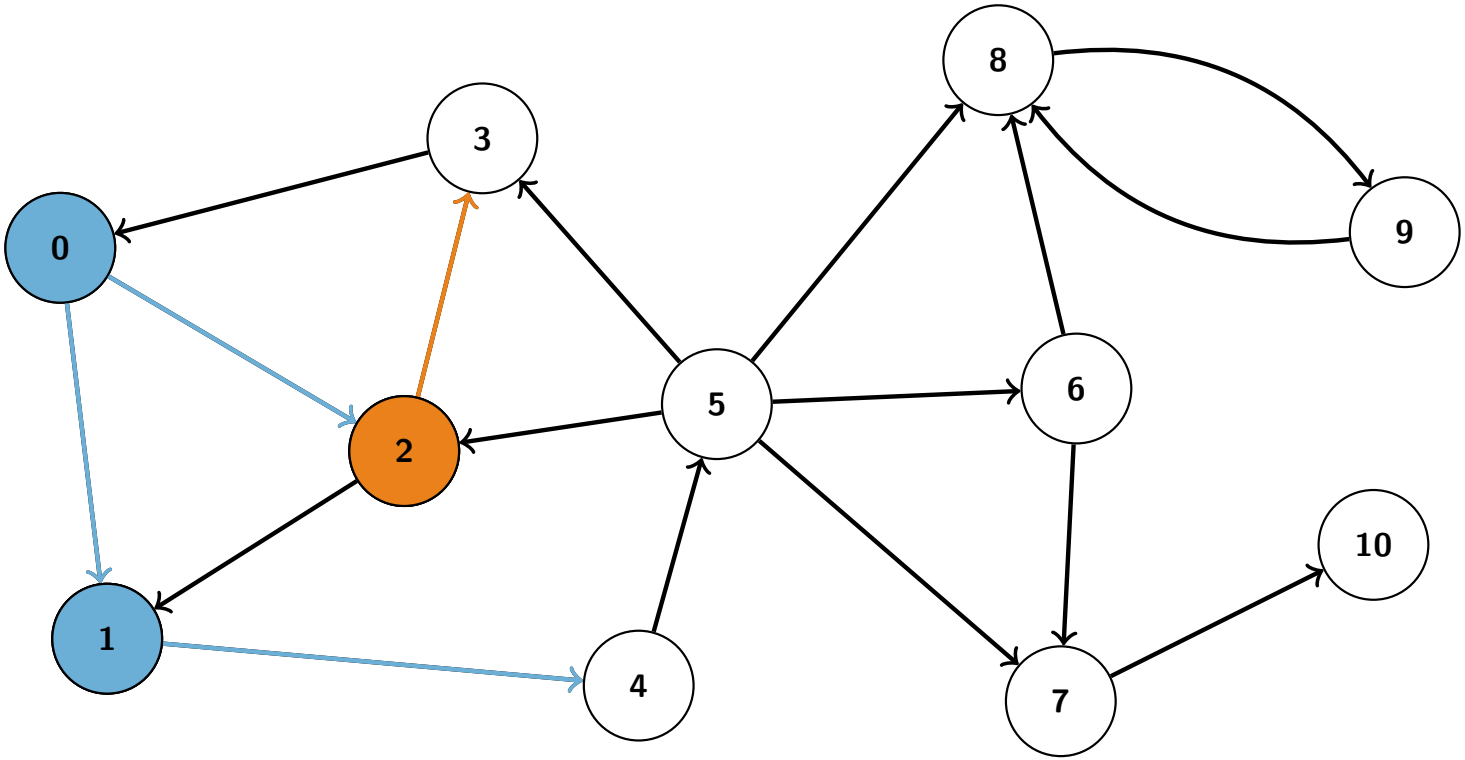
Breadth-first search



Queue: 2 4

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	0	1	0	0	0	0	0	0

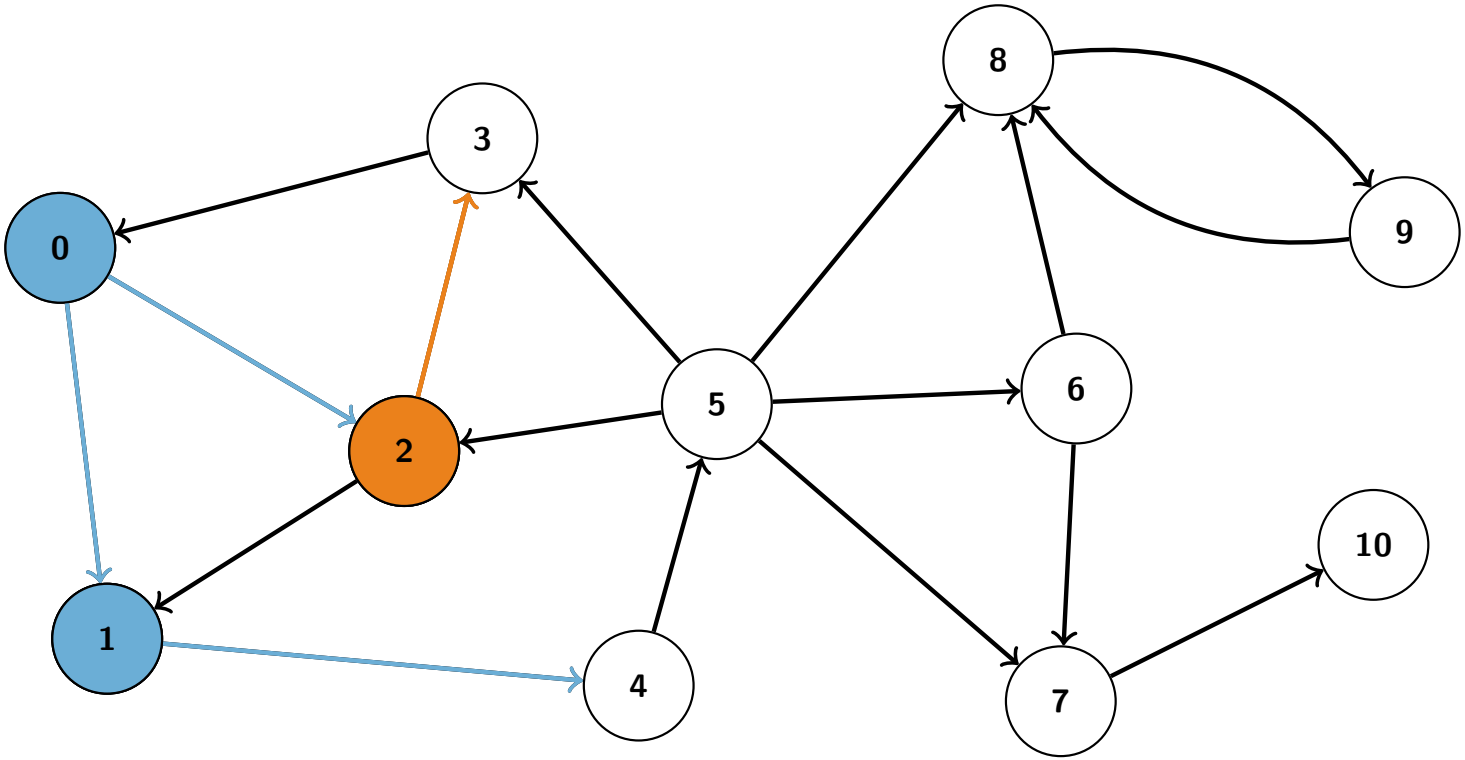
Breadth-first search



Queue: 2 4

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	0	1	0	0	0	0	0	0

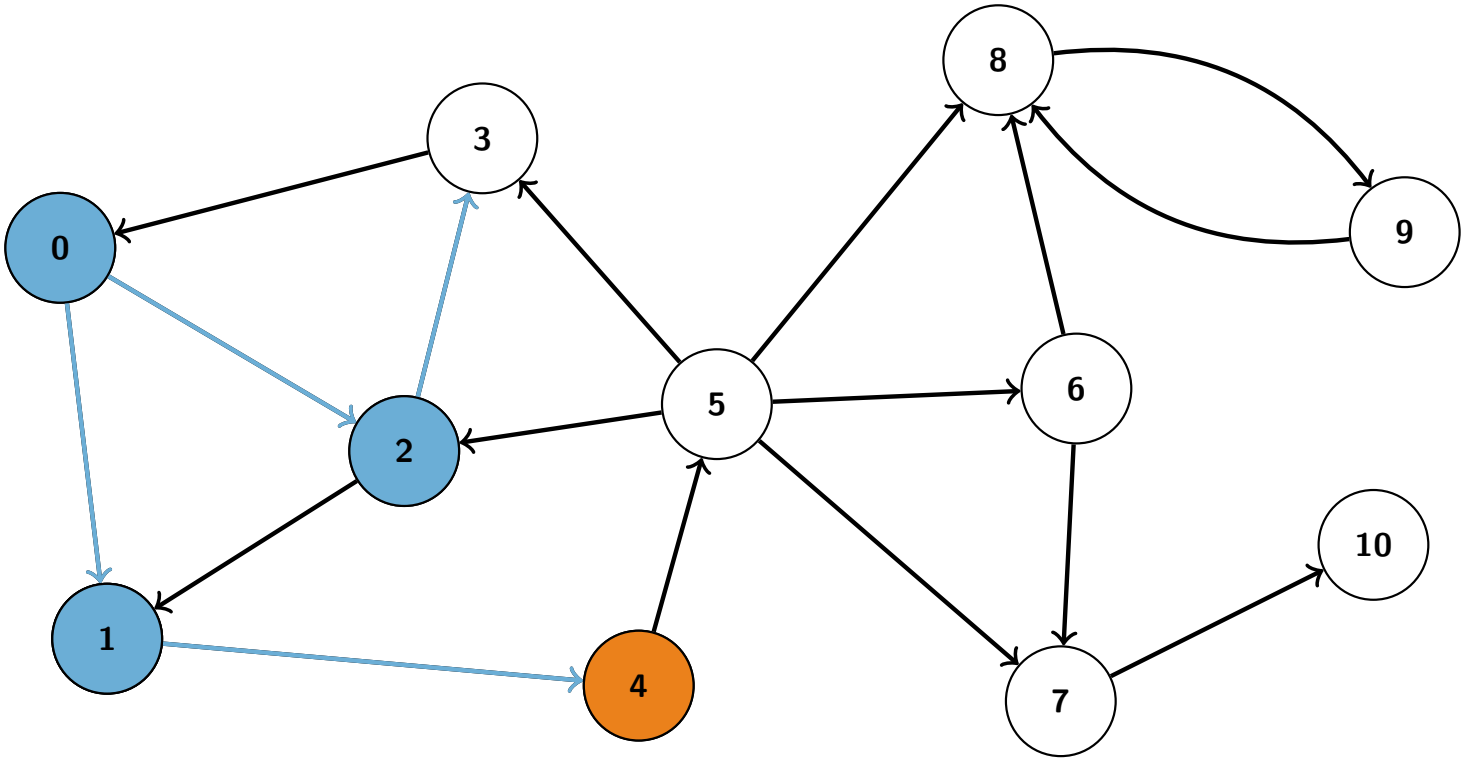
Breadth-first search



Queue: 2 4 3

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	0	0	0	0	0	0

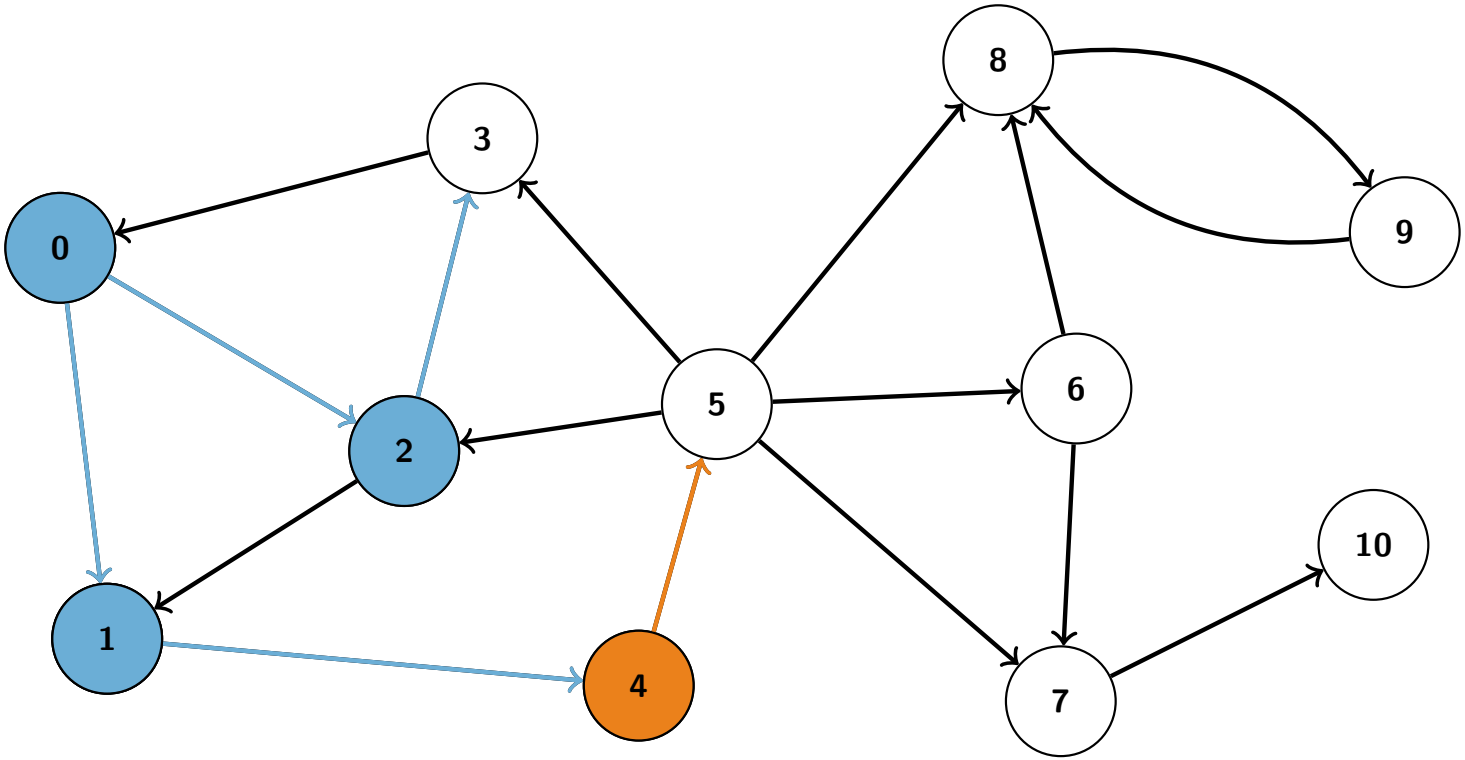
Breadth-first search



Queue: 4 3

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	0	0	0	0	0	0

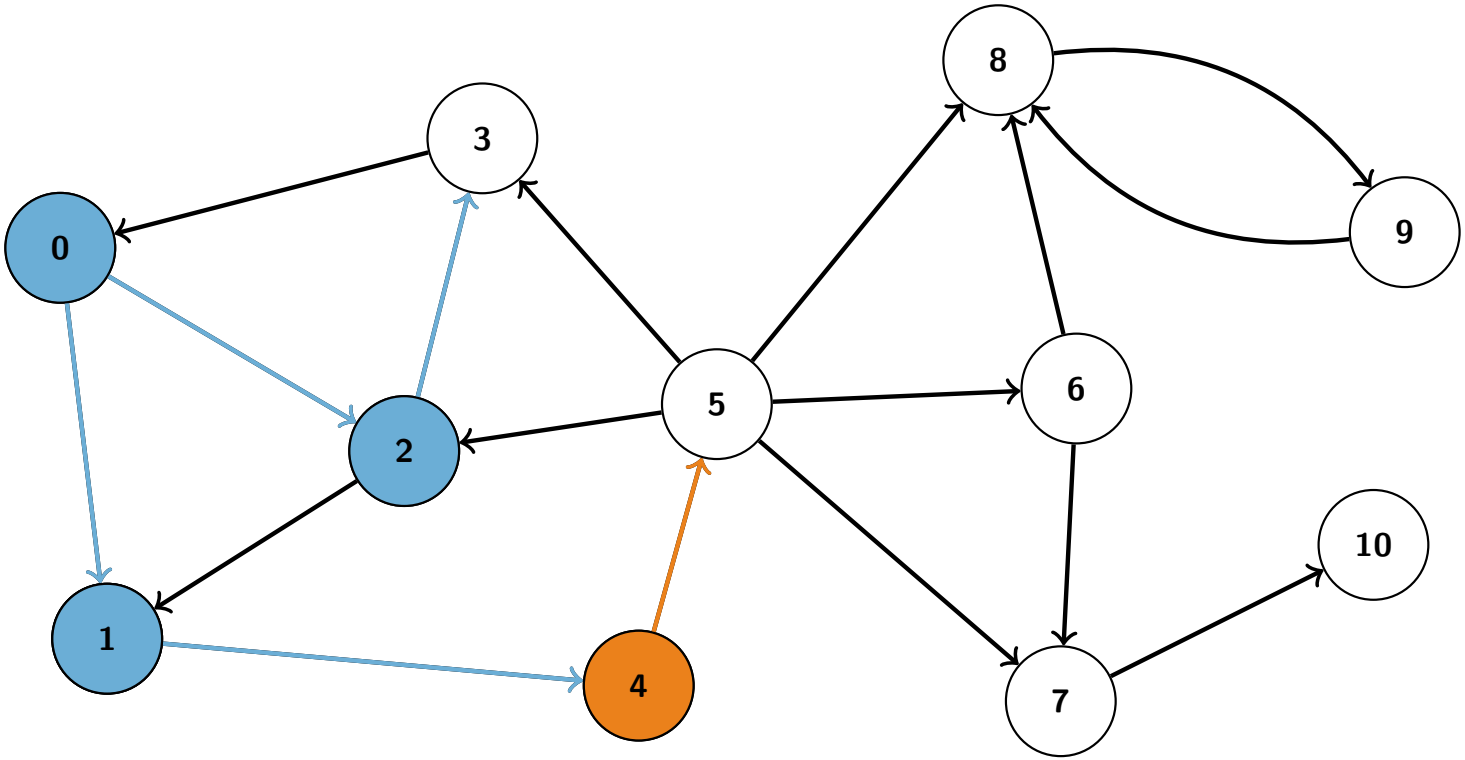
Breadth-first search



Queue: 4 3

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	0	0	0	0	0	0

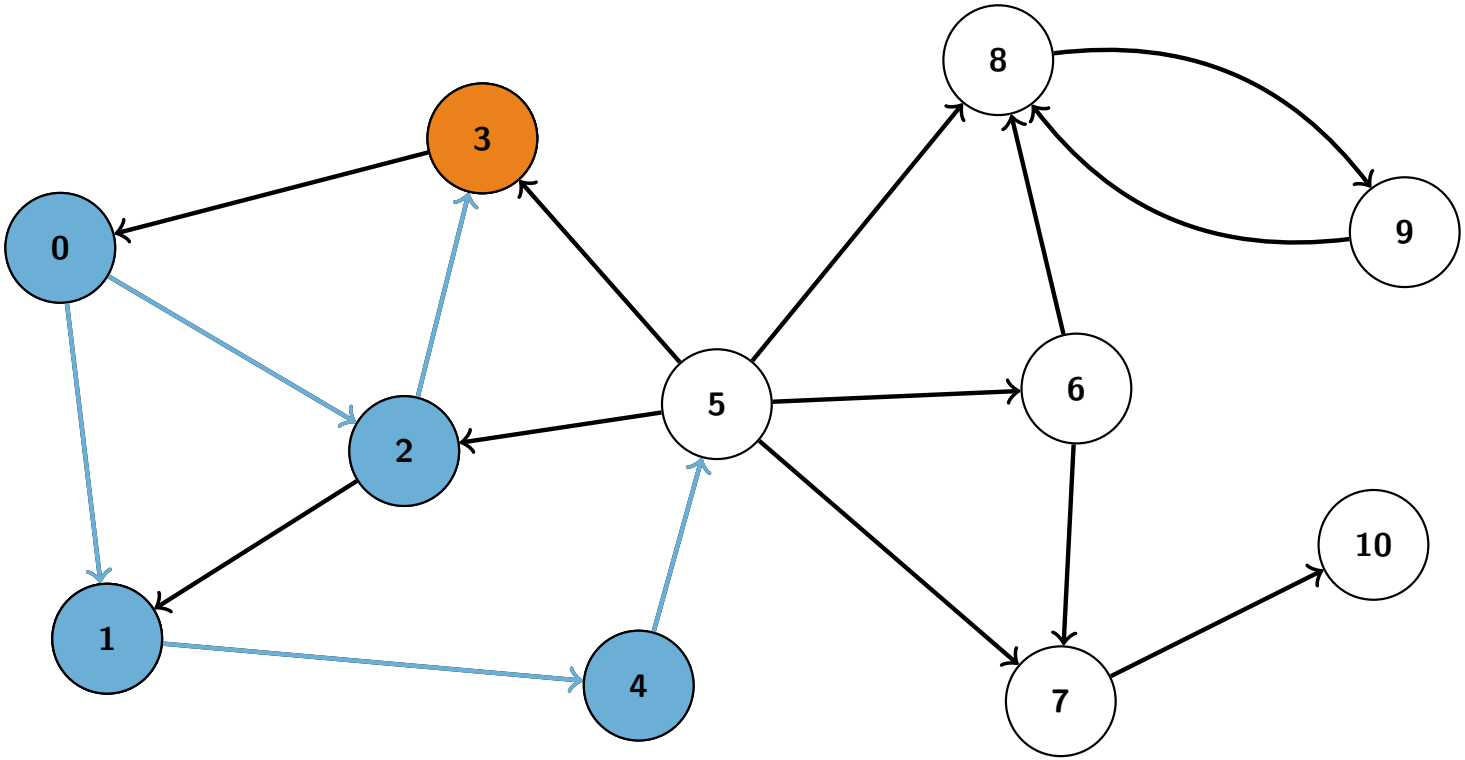
Breadth-first search



Queue: 4 3 5

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	0	0	0	0	0

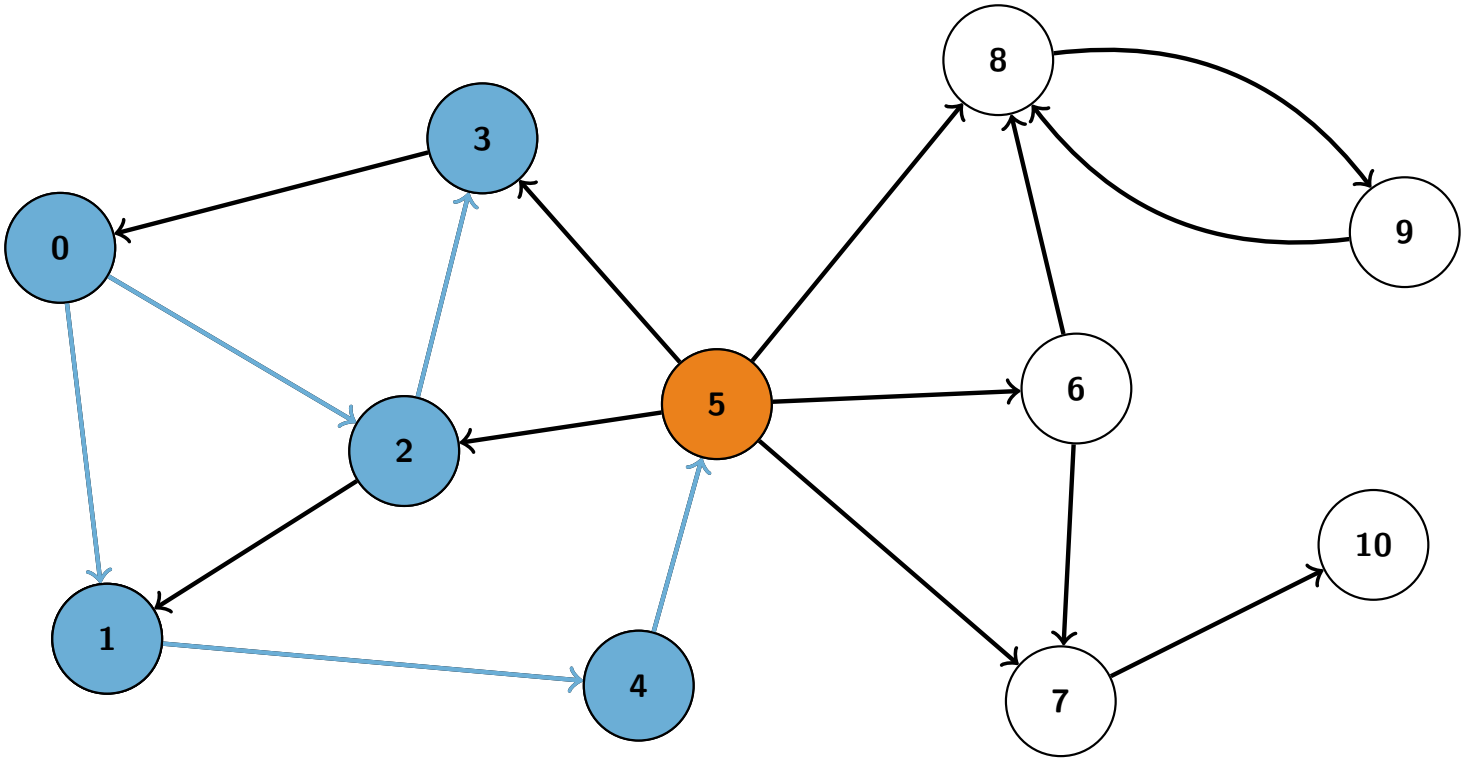
Breadth-first search



Queue: 3 5

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	0	0	0	0	0

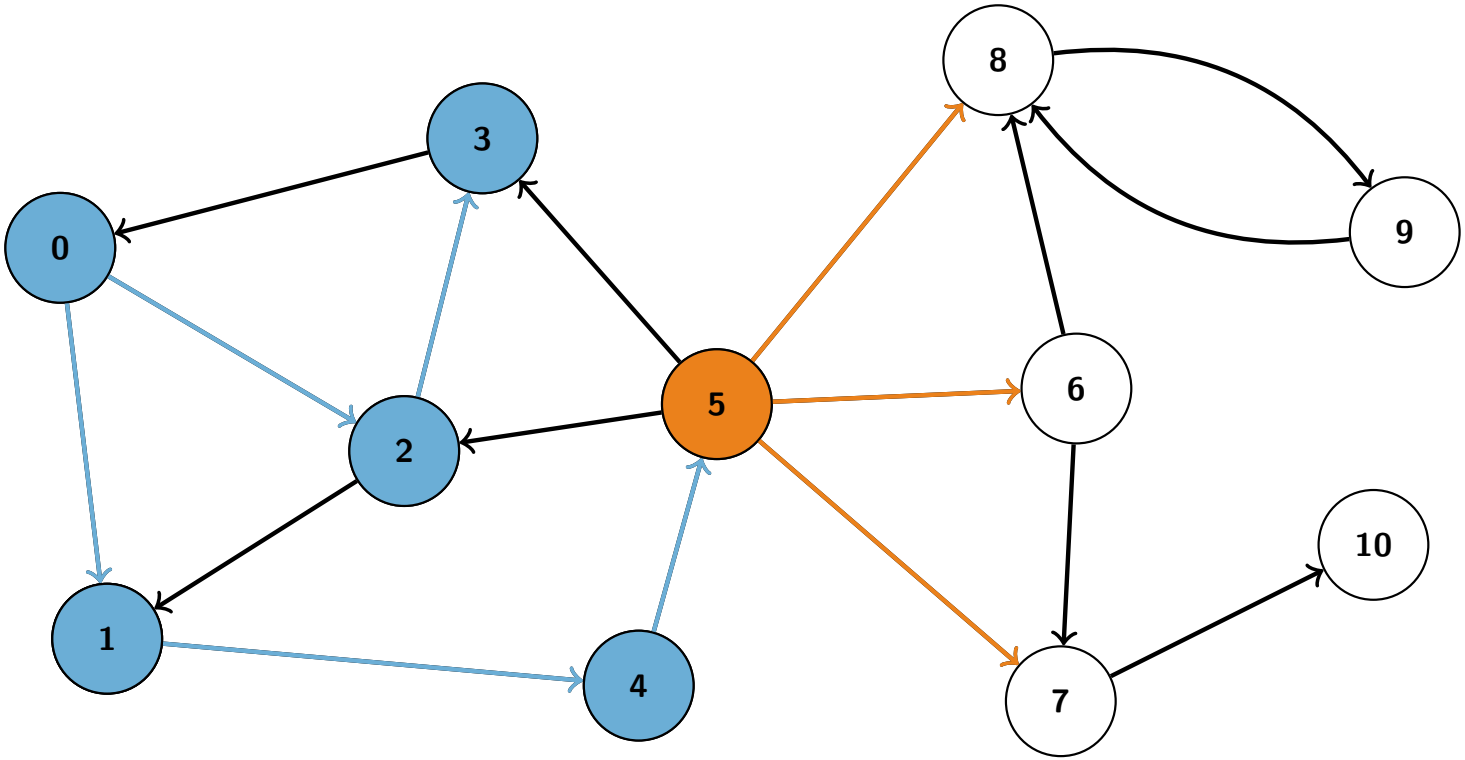
Breadth-first search



Queue: 5

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	0	0	0	0	0

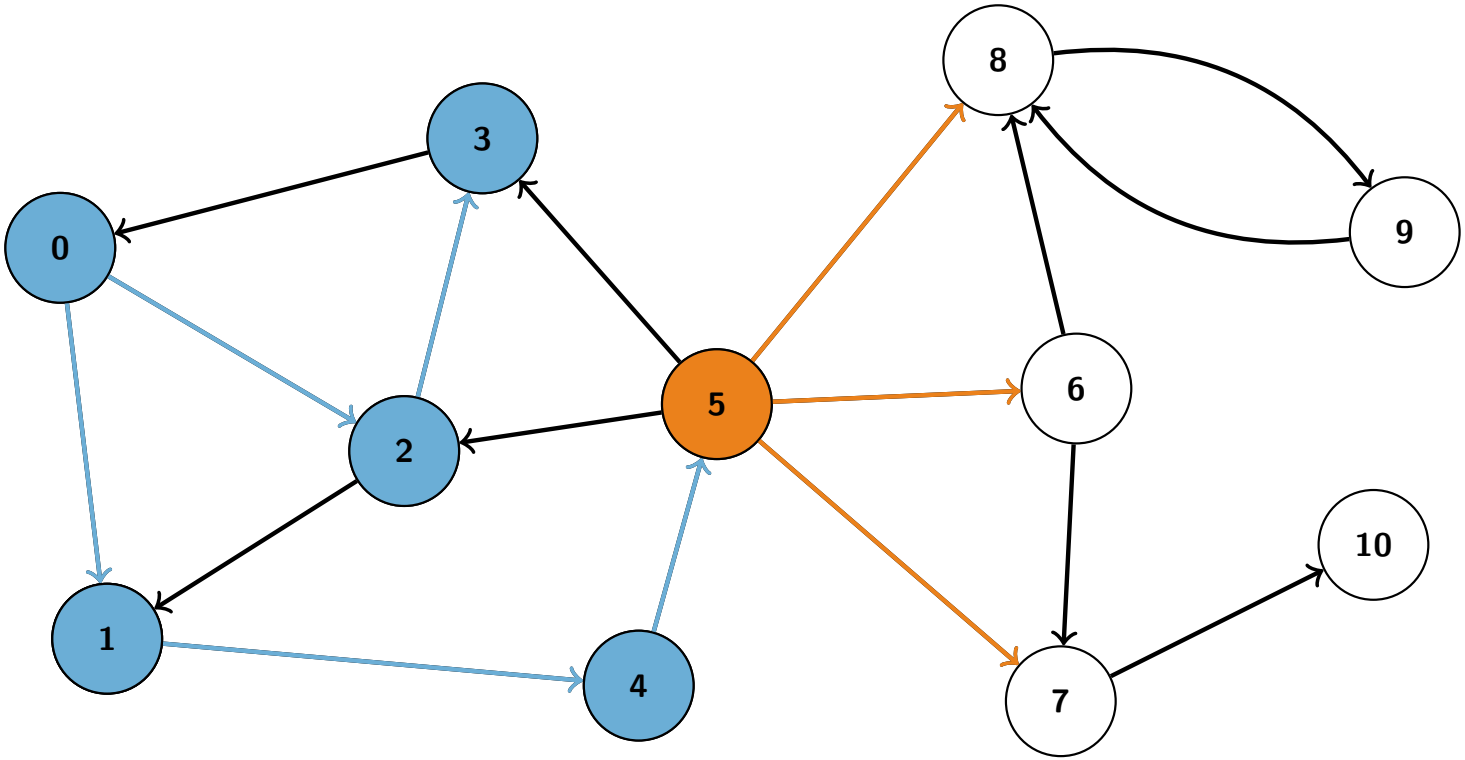
Breadth-first search



Queue: 5

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	0	0	0	0	0

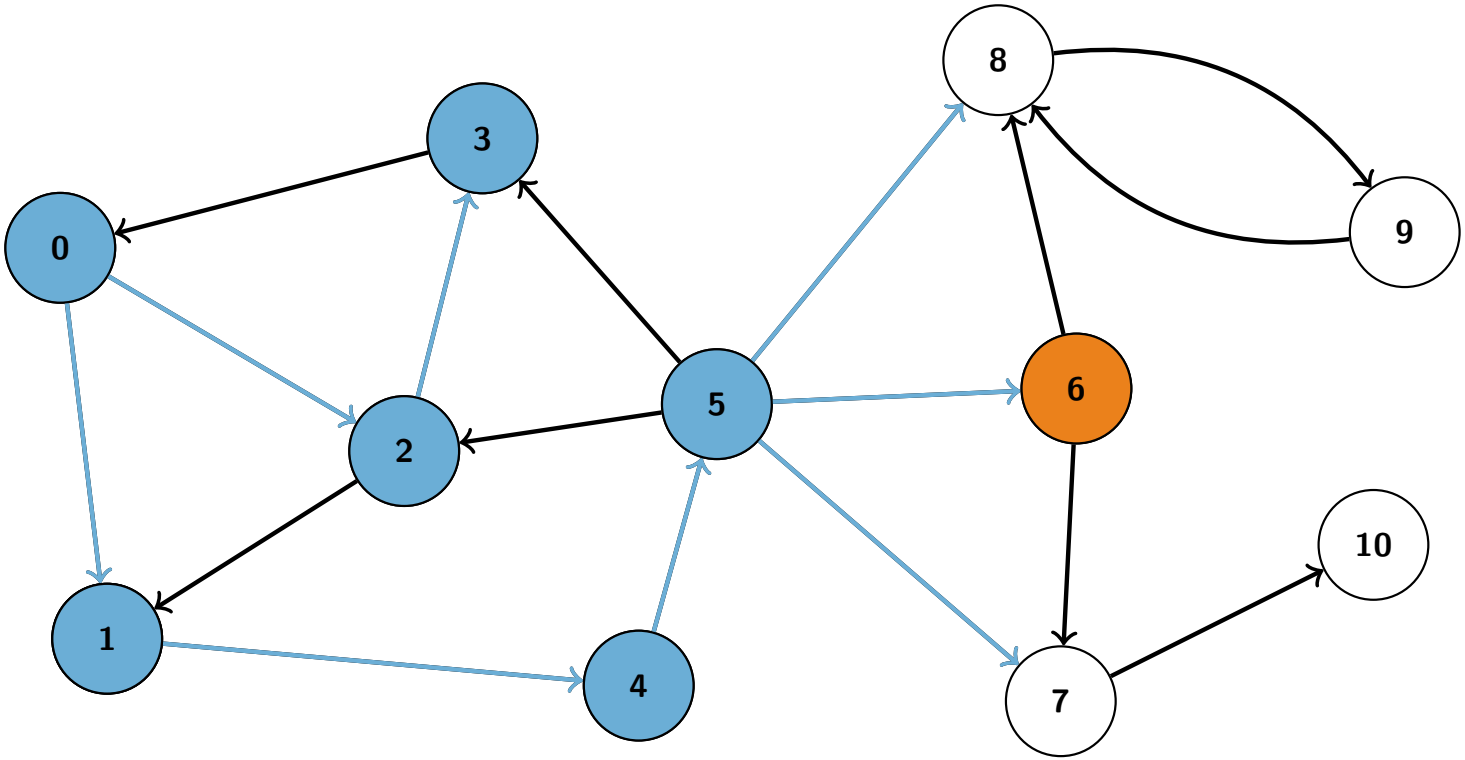
Breadth-first search



Queue: 5 6 7 8

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	0	0

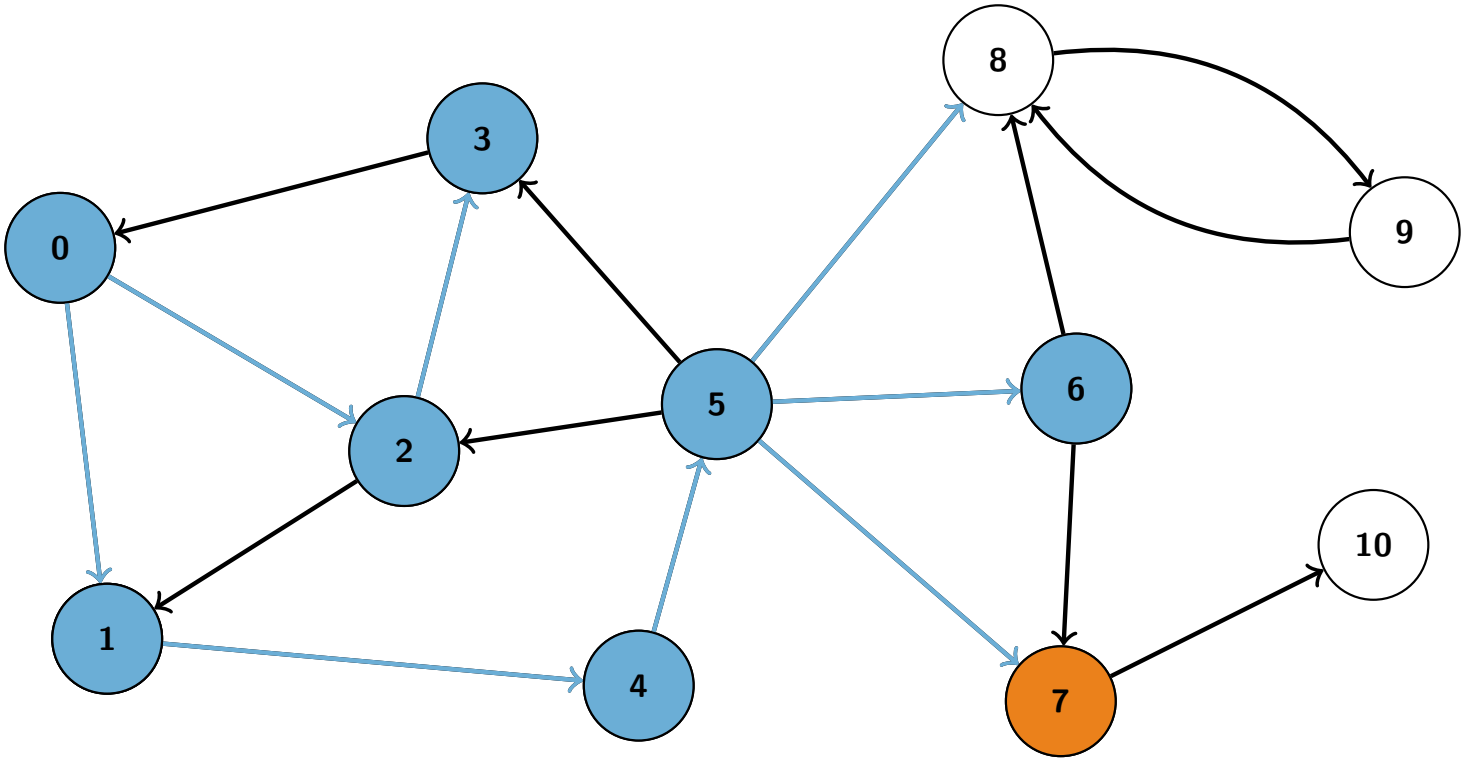
Breadth-first search



Queue: 6 7 8

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	0	0

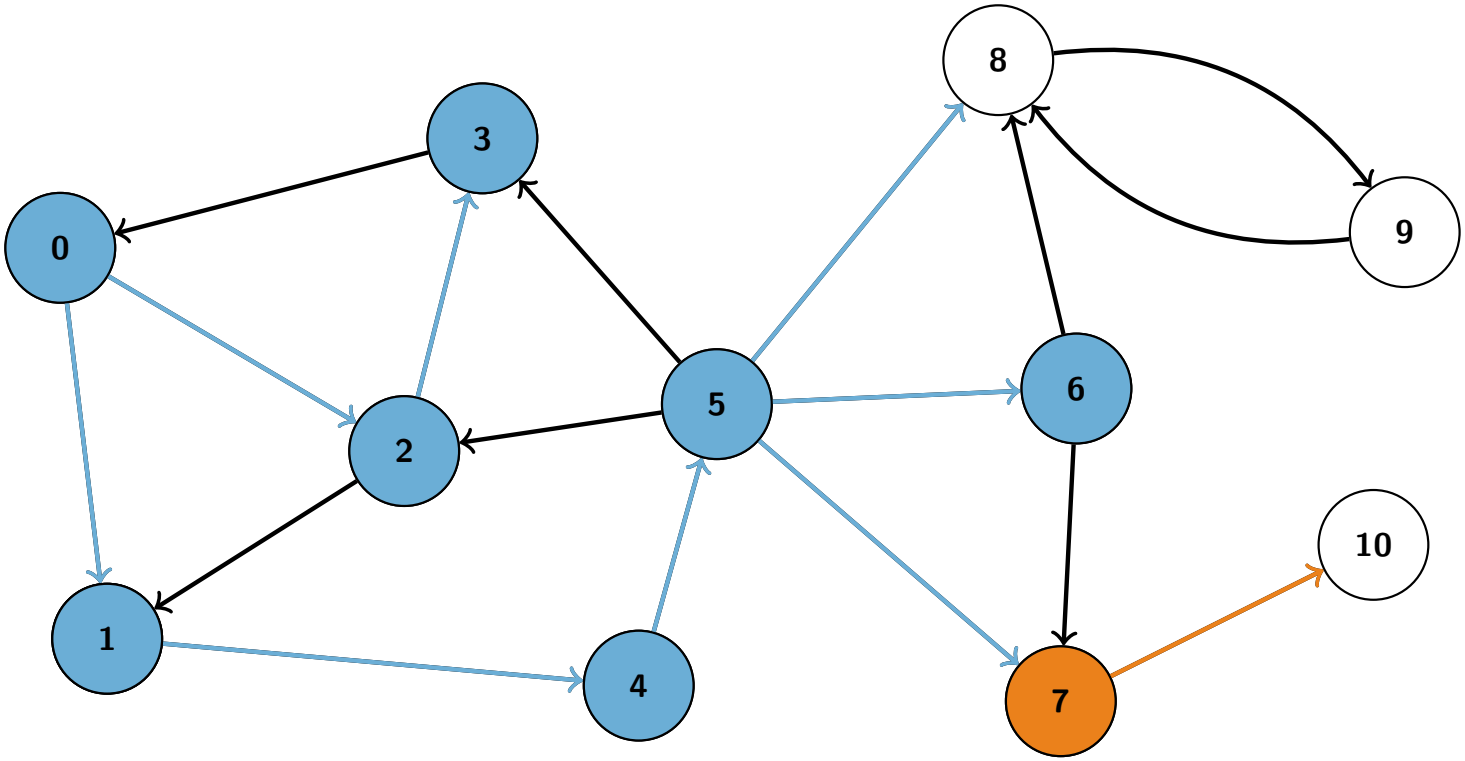
Breadth-first search



Queue: 7 8

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	0	0

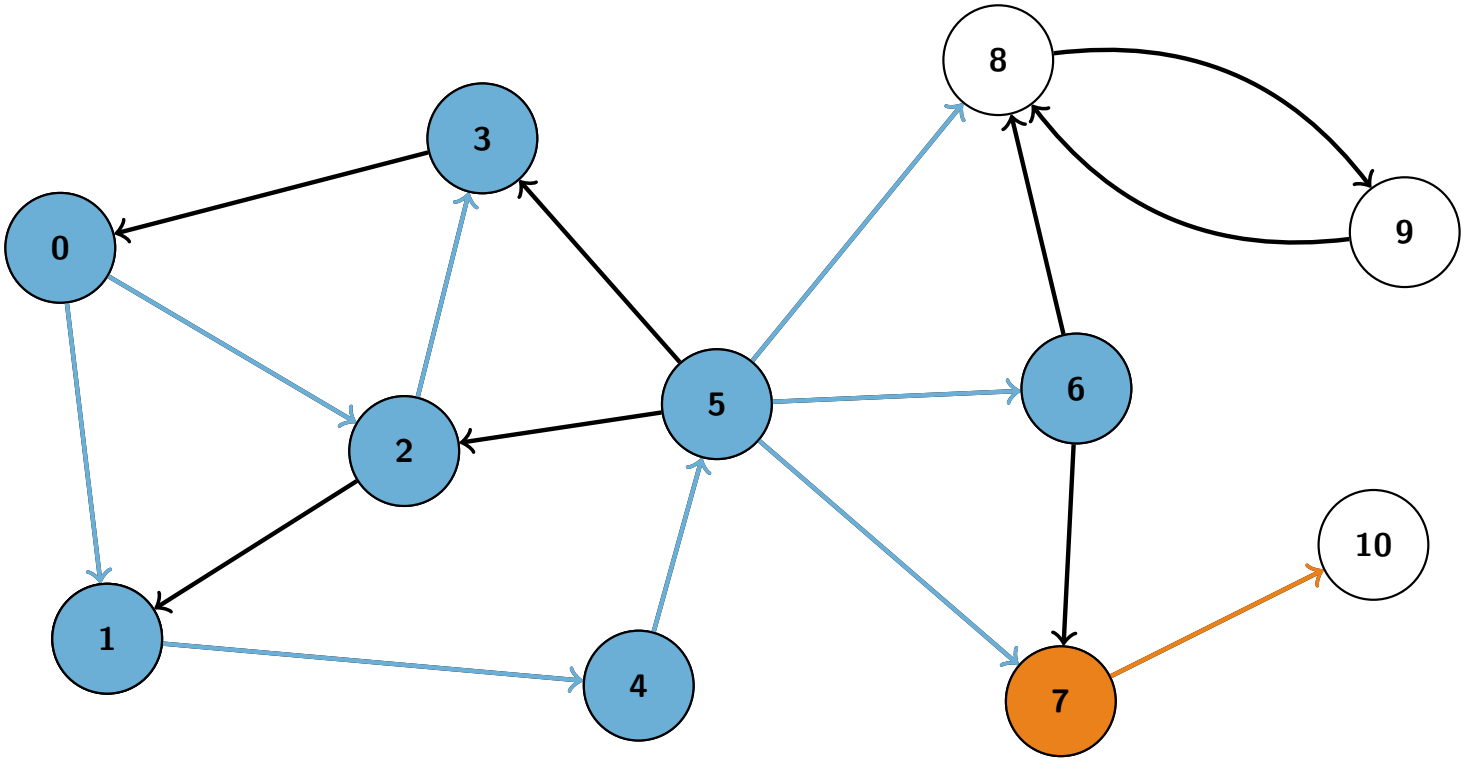
Breadth-first search



Queue: 7 8

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	0	0

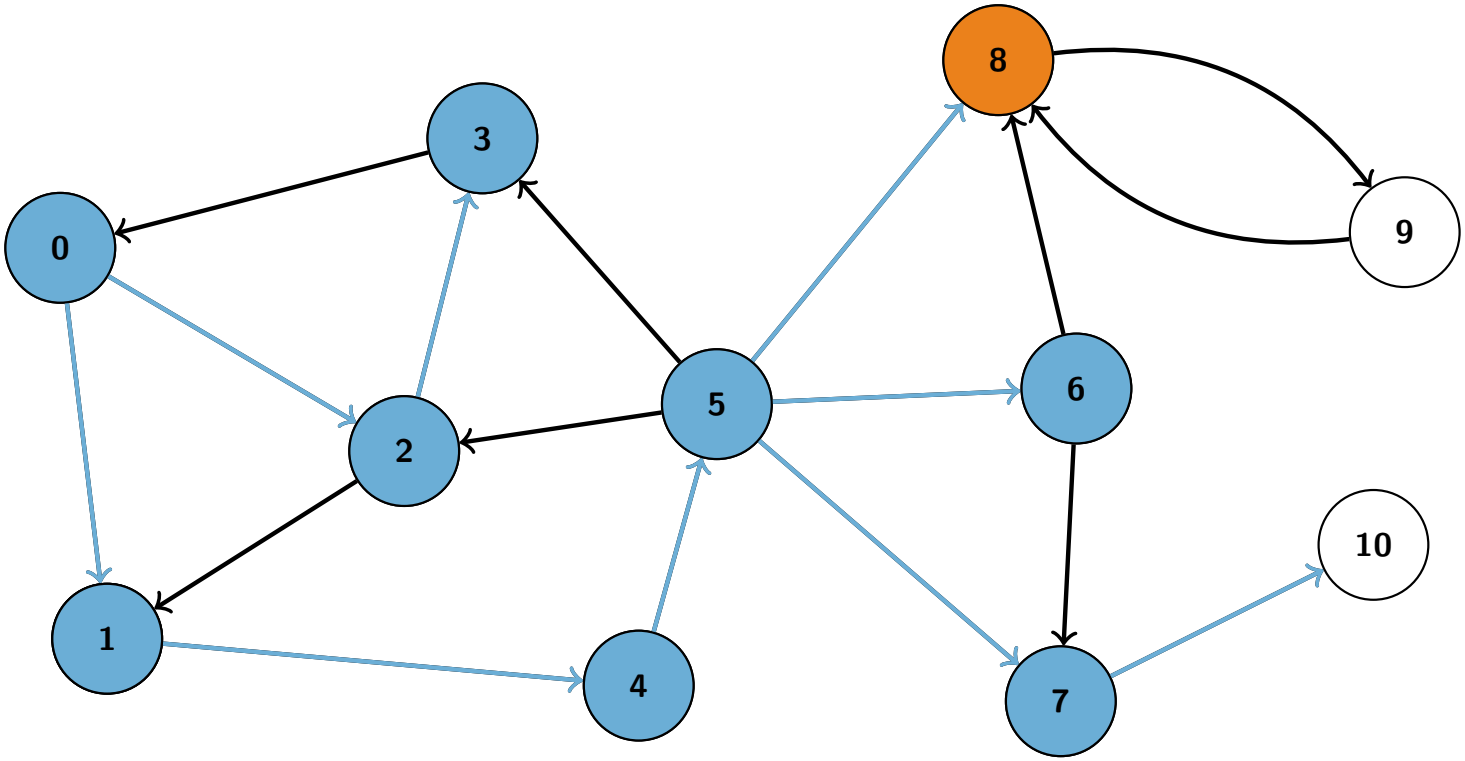
Breadth-first search



Queue: 7 8 10

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	0	1

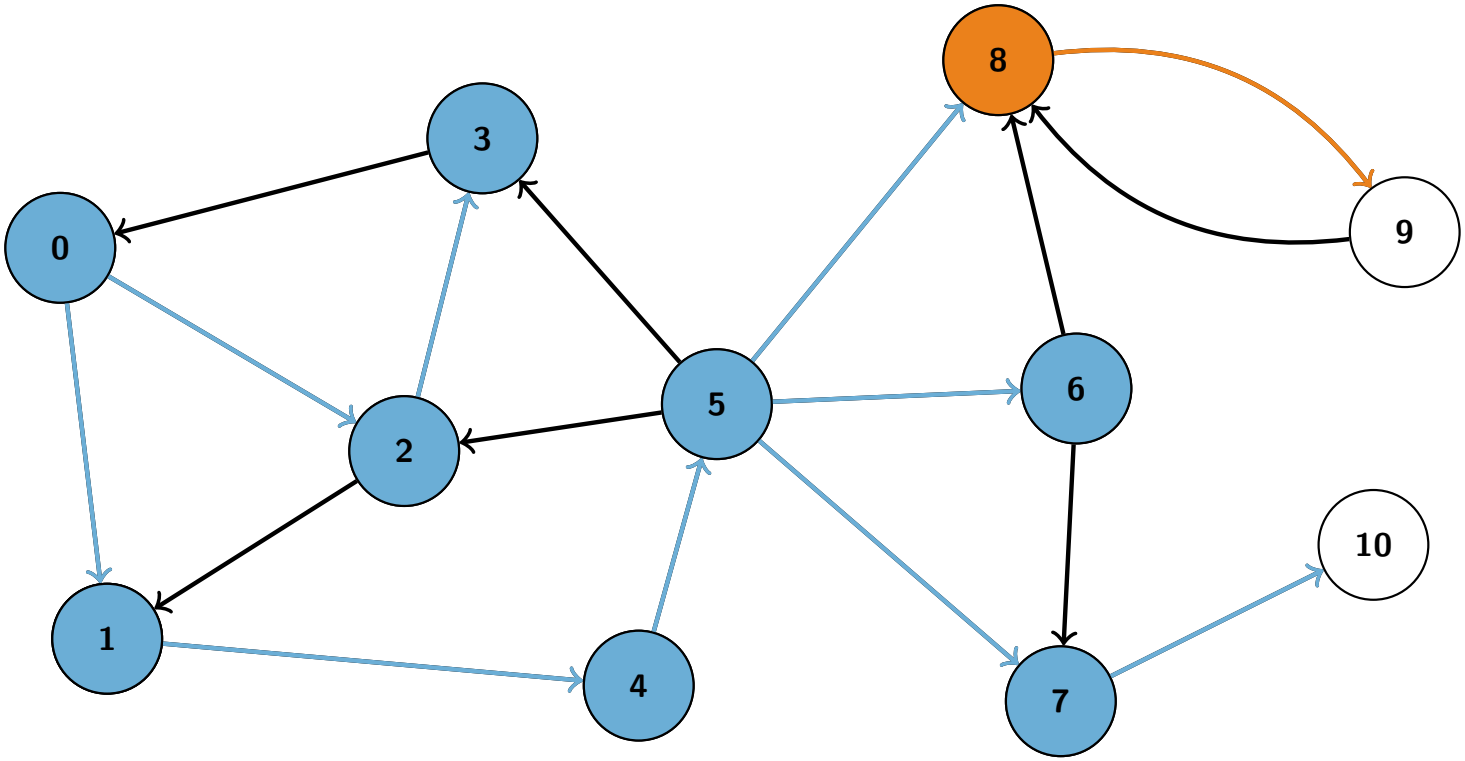
Breadth-first search



Queue: 8 10

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	0	1

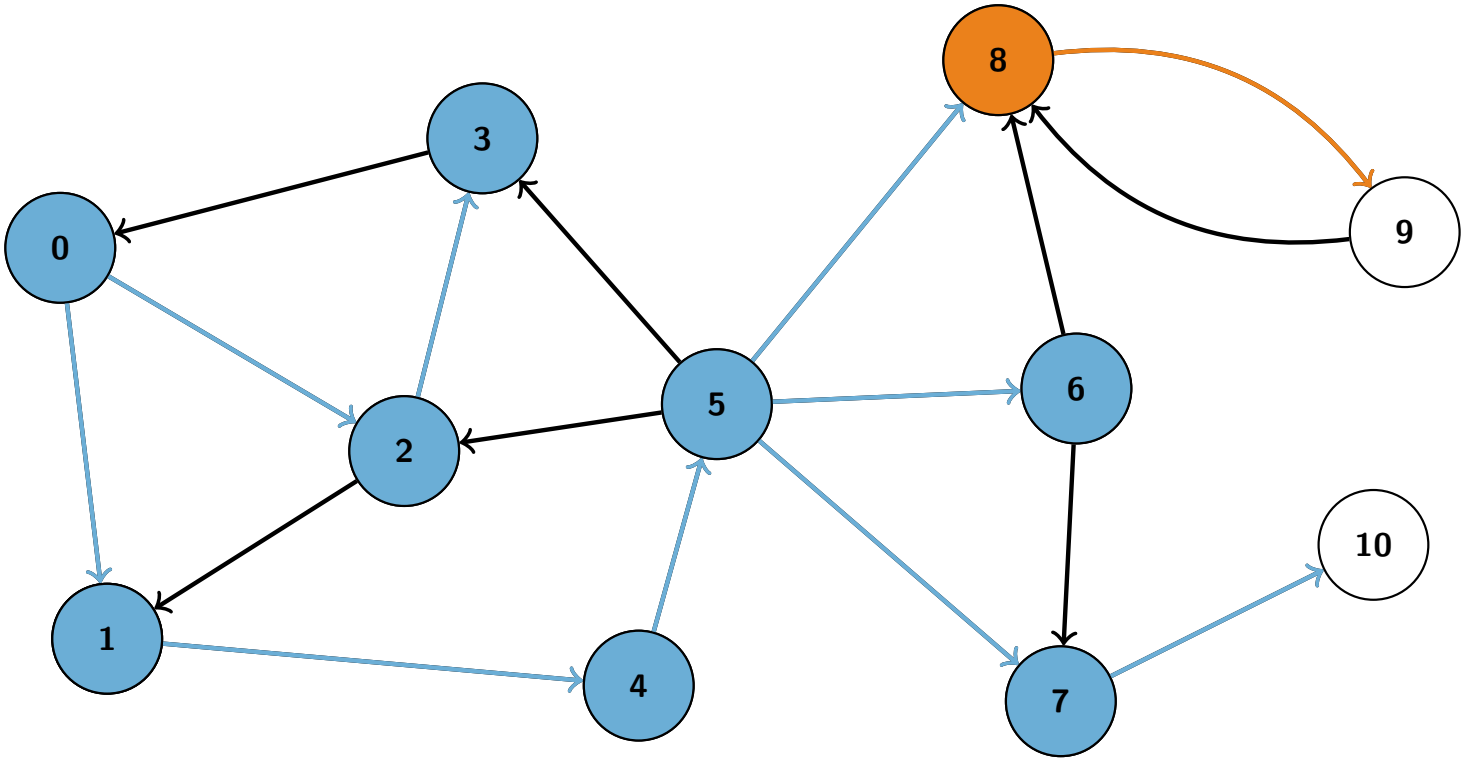
Breadth-first search



Queue: 8 10

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	0	1

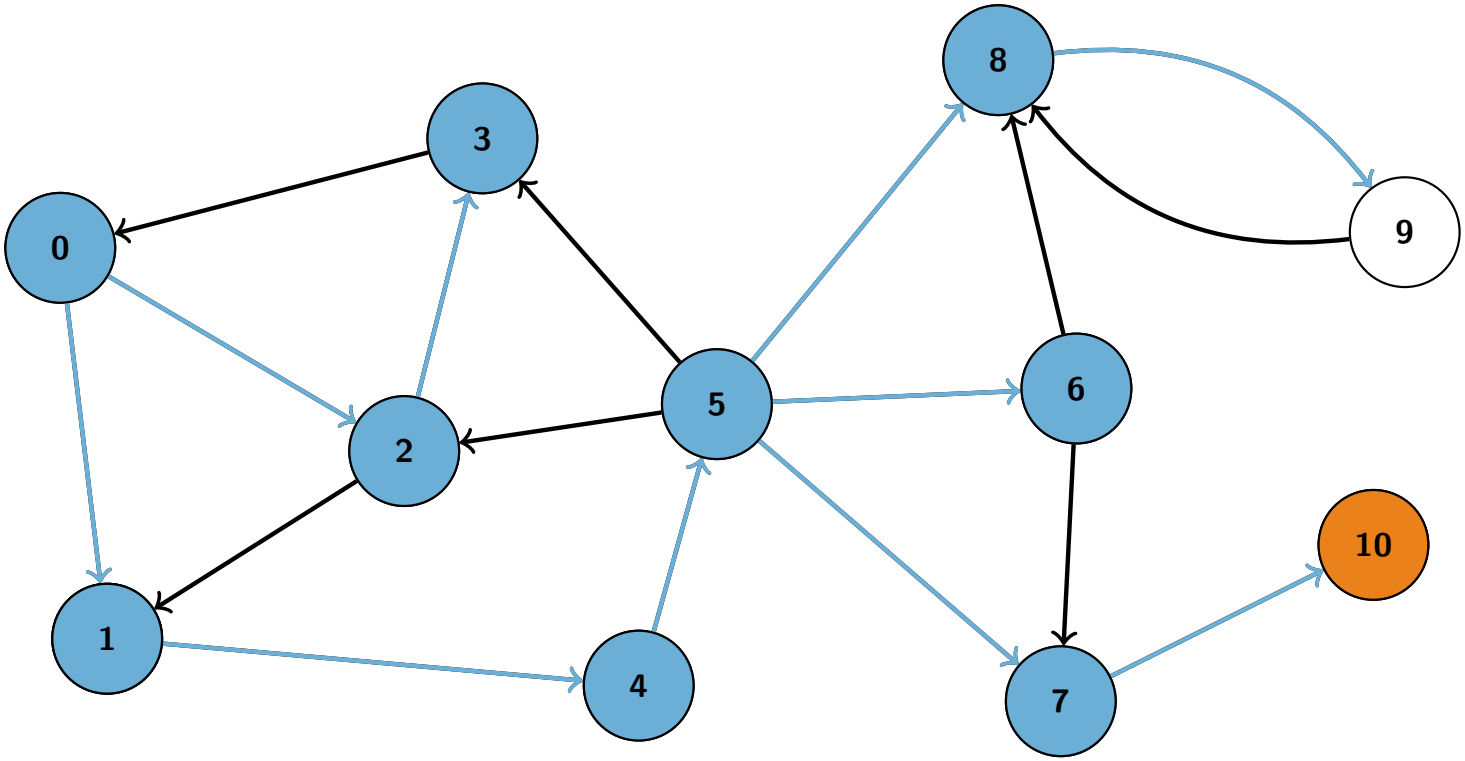
Breadth-first search



Queue: 8 10 9

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	1	1

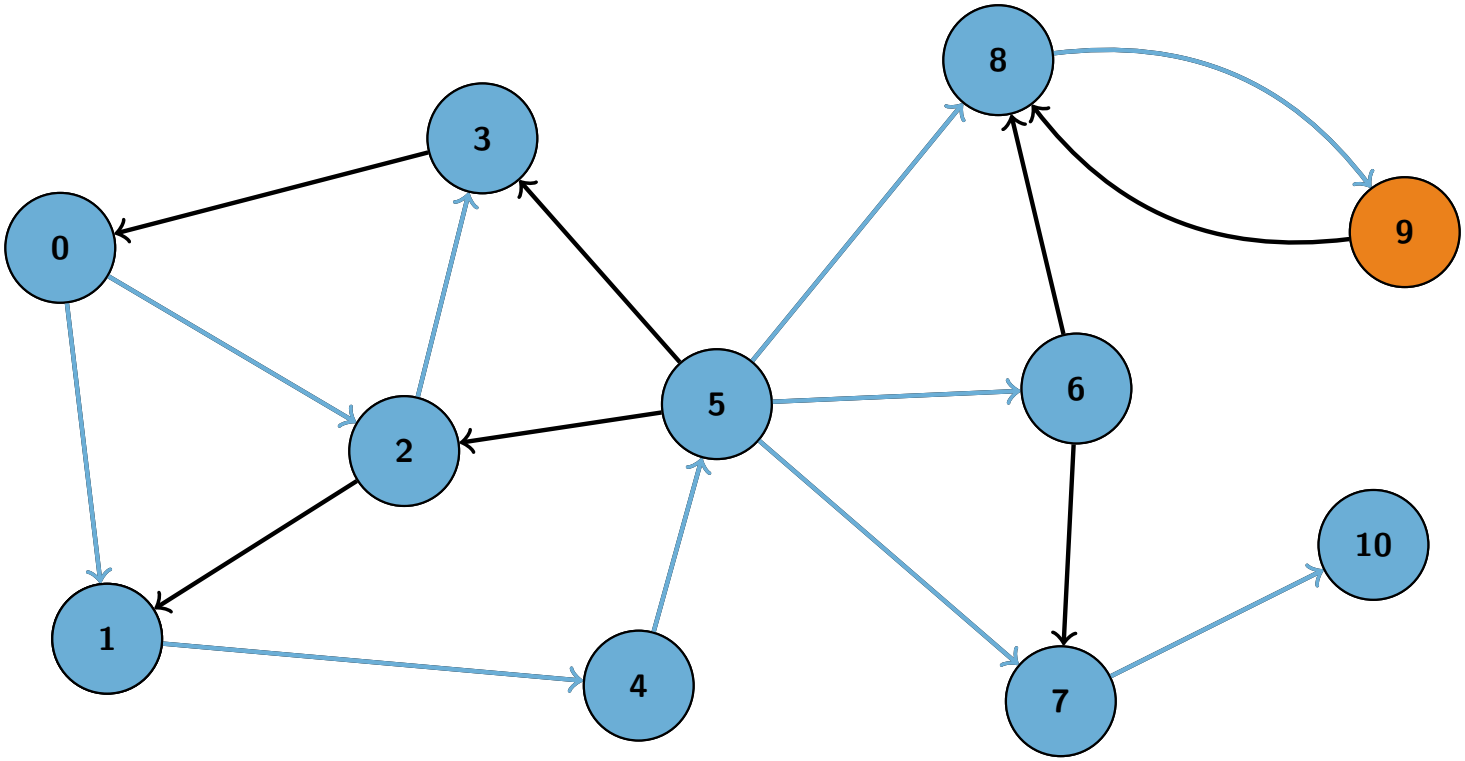
Breadth-first search



Queue: 10 9

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	1	1

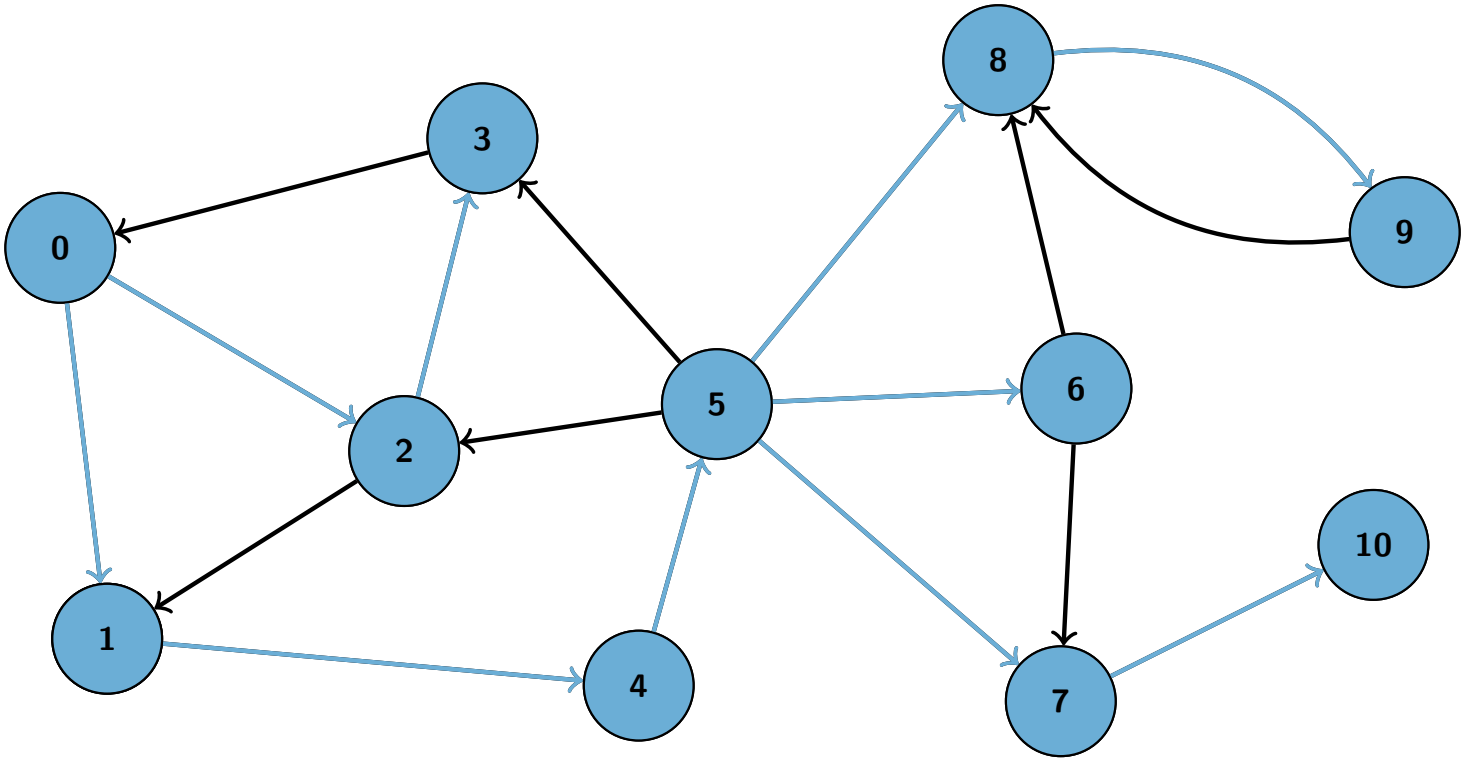
Breadth-first search



Queue: 9

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	1	1

Breadth-first search



Queue:

	0	1	2	3	4	5	6	7	8	9	10
marked	1	1	1	1	1	1	1	1	1	1	1

Ide Pengerjaan

Kita akan melakukan traversal pada grid dengan pergerakan spiral, lalu mengisi cell tersebut dengan angka dari 1 sampai n^2.

Ide Pengerjaan

Kita akan melakukan traversal pada grid dengan pergerakan spiral, lalu mengisi cell tersebut dengan angka dari 1 sampai n^2.
Tinjau bahwa, secara berurutan pergerakannya hanyalah $\{\downarrow, \rightarrow, \uparrow, \leftarrow \ \ldots\}$.

Ide Pengerjaan

Kita akan melakukan traversal pada grid dengan pergerakan spiral, lalu mengisi cell tersebut dengan angka dari 1 sampai n^2.
Tinjau bahwa, secara berurutan pergerakannya hanyalah $\{\downarrow, \rightarrow, \uparrow, \leftarrow \ \ldots\}$. Kita hanya perlu kondisi kapan kita akan berganti gerakan (contoh: dari gerakan kebawah menjadi gerakan kekanan) Tinjau beberapa kasus general berikut:

Kasus 1

Table: Kasus 1: mengganti gerakan ketika akan keluar dari grid-nya

1	-	-	-
2	-	-	-
3	-	-	-
X	-	-	-

Tinjau tabel 1. Pada kasus ini, kita sudah traverse kebawah hingga posisi X. untuk mengganti ke gerakan selanjutnya, kita gunakan conditional ketika gerakan awal (sebelum berganti) mengakibatkan posisi keluar dari grid. Dengan begitu, untuk contoh kasus di atas kita punya kondosi berikut:

Kasus 1 Cont

```
IF posisi_baris+current_movement_baris>n
    current_movement_baris= next_movement_baris
    current_movement_kolom= next_movement_kolom
END IF
```

Untuk ketiga kasus out of grid lainnya similar.

Kasus 2

Table: mengganti gerakan ketika posisi selanjutnya sudah disi

1	-	-	-
2	-	-	-
3	10	9	8
4	5	6	7

Tinjau tabel 2. pada kasus ini kita berada di posisi 10 dan sedang bergerak ke kiri. Tinjau bahwa jika kita bergerak ke kanan, sel selanjutnya sudahlah diisi. oleh karena itu, kita bisa mengganti gerakannya dengan kondisikondisi sel selanjutnya sudah diisi. dalam psudocode:

kasus 2 cont

```
IF grid[posisi_baris+current_movement+baris]
[posisi_kolom+current_movement_kolom] not empty
current_movement_baris= next_movement_baris
current_movement_kolom= next_movement_kolom
END IF
```

Nomor 2 Mathematica

Selanjutnya, Nilai terbesar selalulah n^2 dan posisinya akan ditengah jika n ganjil. lalu sumasi dari elemen pada grid adalah
$1 + 2 + \cdots + n^2 = \frac{n^2(n^2+1)}{2}$