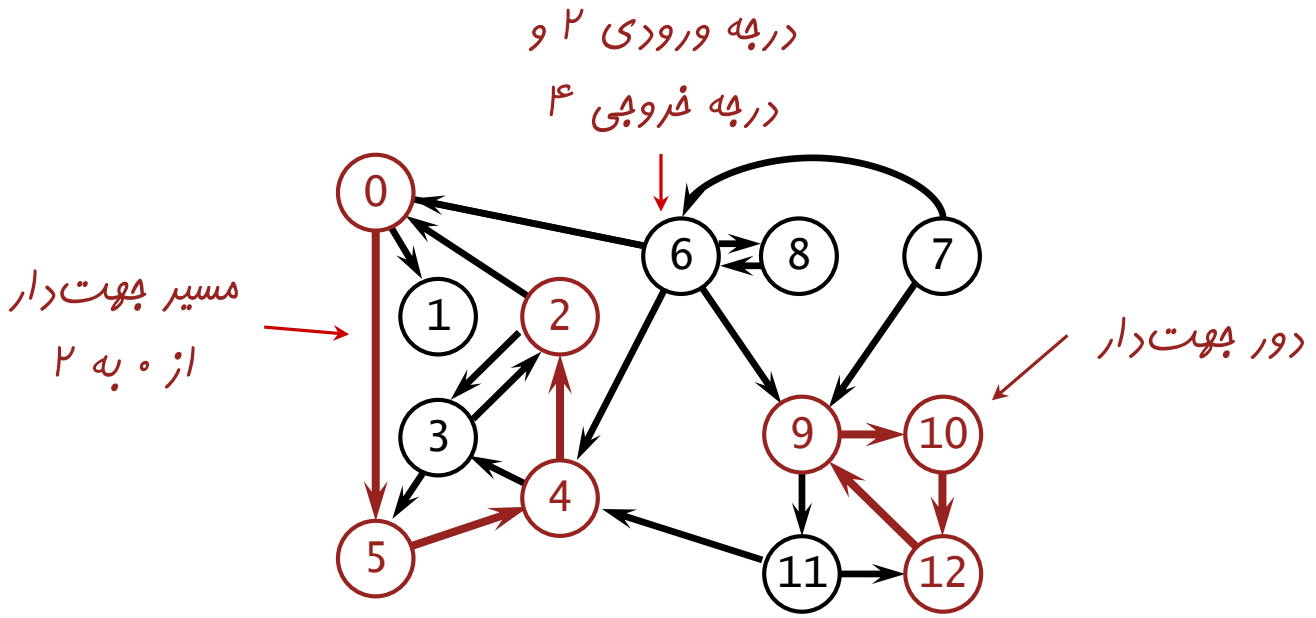
گراف (جهت‌دار)

سید ناصر رضوی www.snrazavi.ir
۱۳۹۵

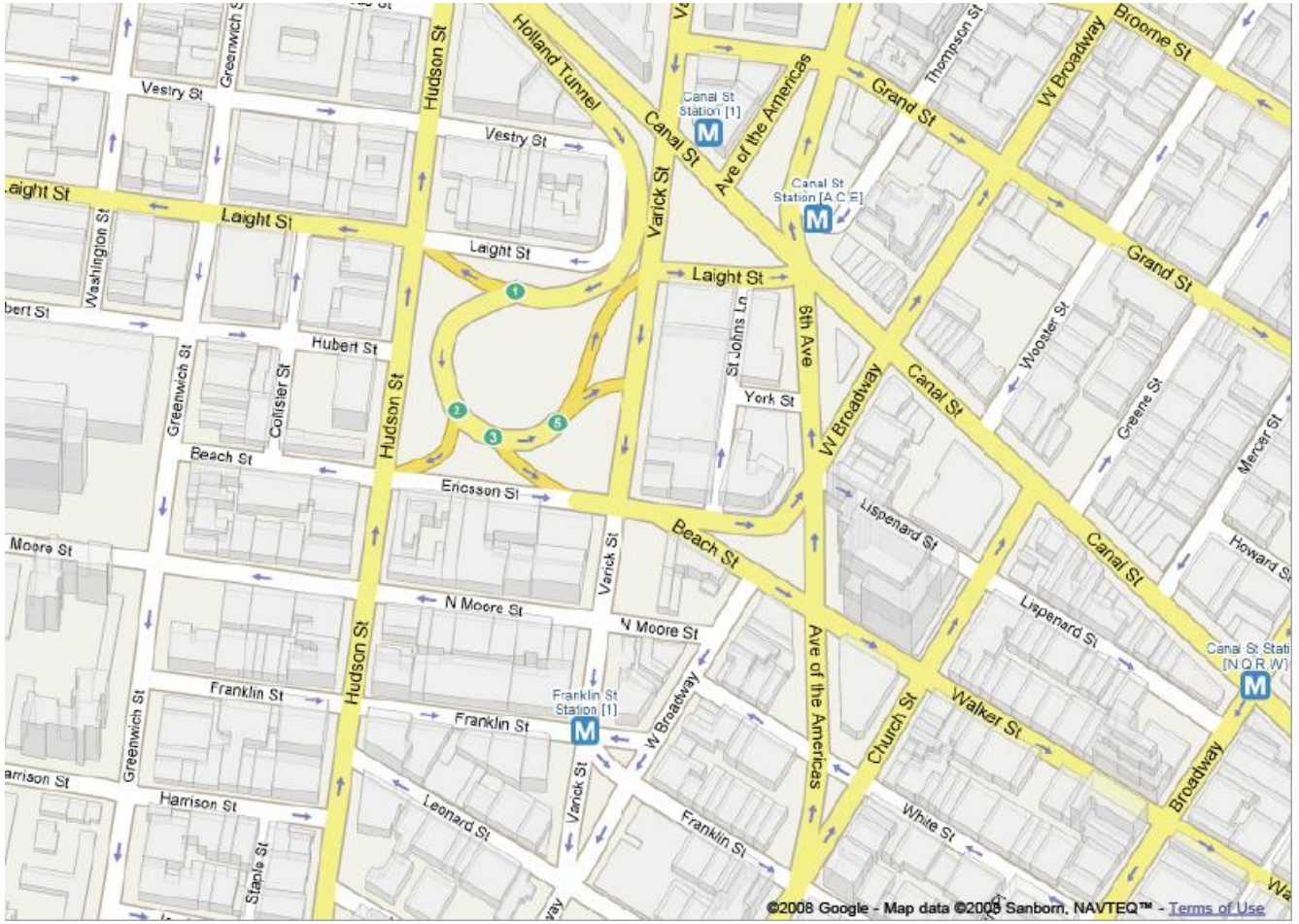
گراف جهت‌دار

- گراف جهت‌دار. مجموعه‌ای از رئوس که به وسیله‌ی تعدادی یال جهت‌دار به صورت دو به دو به هم وصل شده‌اند.



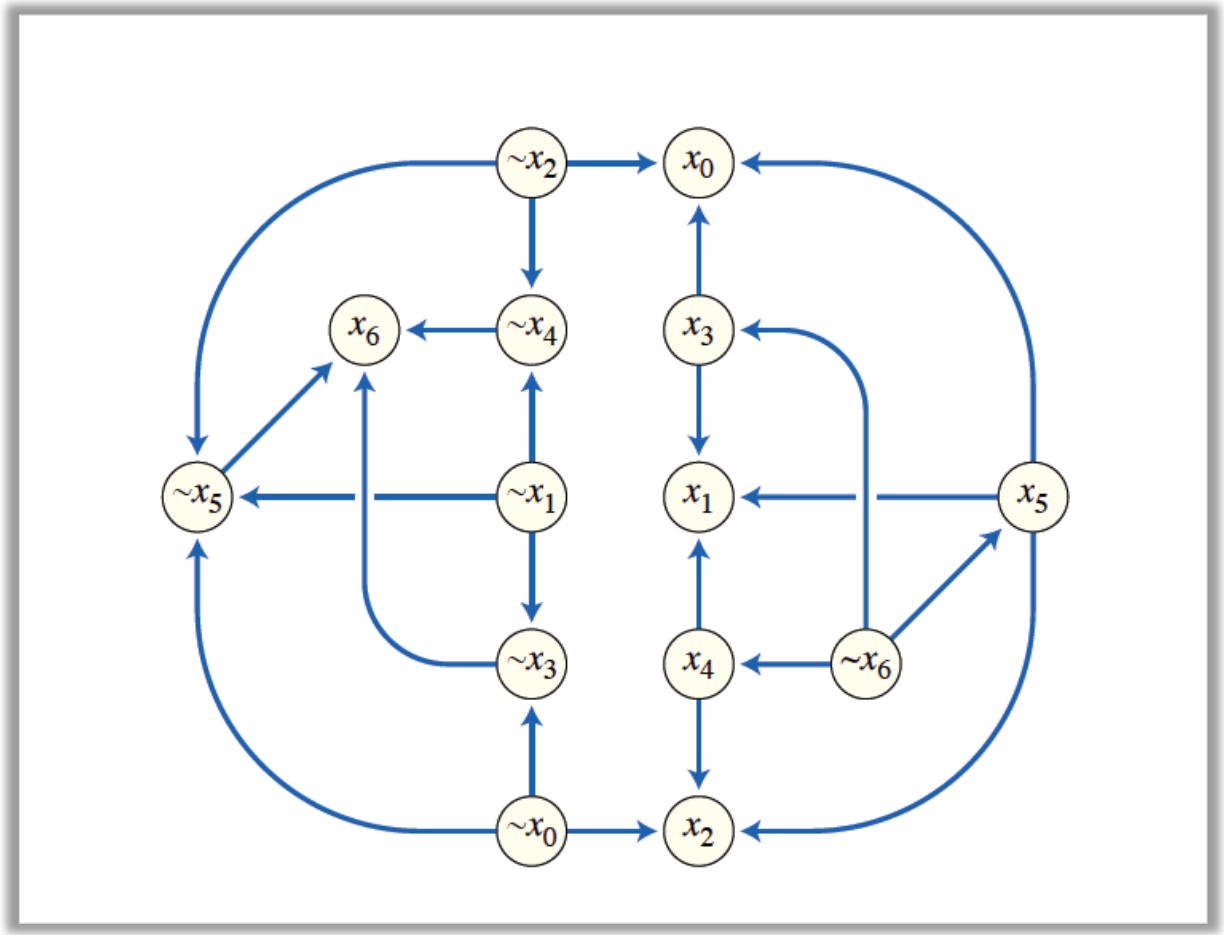
شبکه راه‌ها

- رئوس: تقاطع‌ها؛ یال‌ها: خیابان‌های یک طرفه



گراف استلزام

- رئوس: متغیرهای بولی؛ یال‌ها: استلزام منطقی



برخی از کاربردهای گراف جهت‌دار

گراف جهت‌دار	رأس	یال
حمل و نقل	تقاطع	خیابان یک‌طرفه
وب	صفحه‌ی وب	ابر پیوند
زنجیره‌ی غذایی	گونه‌ها	رابطه‌ی شکار و شکارچی
حوزه‌ی مالی	بانک	تعاملات بین بانکی
بیماری‌های مسری	شخص	سرایت
بازی	وضعیت صفحه	حرکات قانونی
زمان‌بندی	وظیفه	رابطه تقدم
گراف شی	شی	پیغام
سلسله مراتب وراثت	گونه‌ها	رابطه‌ی ارث‌بری

چند مسئله در رابطه با گراف‌های جهت‌دار

- مسیر. آیا بین دو رأس **s** و **t** یک مسیر جهت‌دار وجود دارد؟
- کوتاه‌ترین مسیر. کوتاه‌ترین مسیر جهت‌دار بین دو رأس **s** و **t** کدام است؟
- مرتب‌سازی توپولوژیکی. آیا می‌توان گراف جهت‌دار را به گونه‌ای ترسیم نمود که جهت همه‌ی یالها رو به بالا باشد؟
- همبندی قوی. آیا بین هر دو رأس دلخواه یک مسیر جهت‌دار وجود دارد؟
- رتبه‌بندی صفحات وب. اهمیت یک صفحه‌ی وب چه میزان است؟

واسط گراف جهت‌دار

گراف جهت‌دار

public class	Digraph	
	Digraph(int V)	ایجاد یک گراف تهی با V رأس
	Digraph(In in)	ایجاد یک گراف از جریان ورودی
void	addEdge(int v, int w)	اضافه‌کردن یال جهت‌دار v→w
Iterable<Integer>	adj(int v)	برگرداندن رئوس مجاور v
int	V()	برگرداندن تعداد رئوس
int	E()	برگرداندن تعداد یالها
Digraph	reverse()	برگرداندن معکوس گراف جهت‌دار
String	toString()	نمایش گراف جهت‌دار به صورت یک رشته

```
In in = new In(args[0]);
Digraph G = new Digraph(in);

for (int v = 0; v < G.V(); v++)
    for (int w : G.adj(V))
        StdOut.println(v + "->" + w);
```

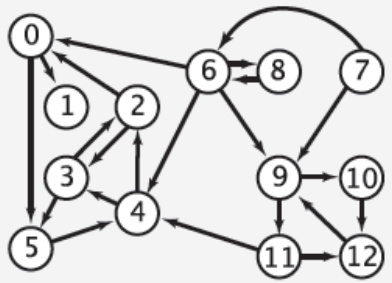
خواندن و ایجاد گراف از جریان ورودی

چاپ یالها (هر یال یک بار)

گراف جهت‌دار

tinyDG.txt

V → 13
E → 22
4 2
2 3
3 2
6 0
0 1
2 0
11 12
12 9
9 10
9 11
7 9
10 12
11 4
4 3
3 5
6 8
8 6
⋮



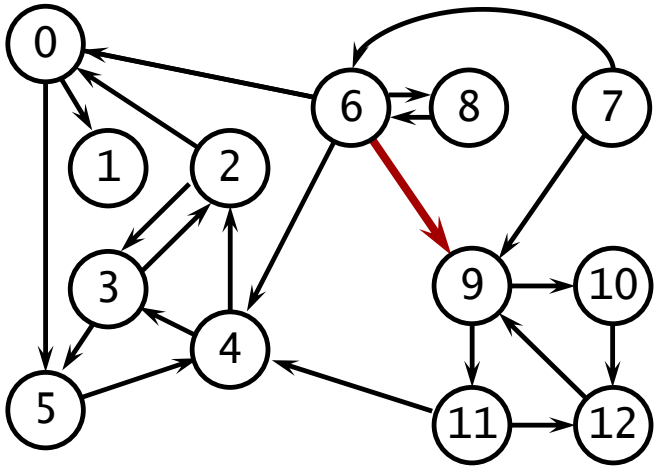

```
% java Digraph tinyDG.txt
0->5
0->1
2->0
2->3
3->5
3->2
4->3
4->2
5->4
⋮
11->4
11->12
12->9
```

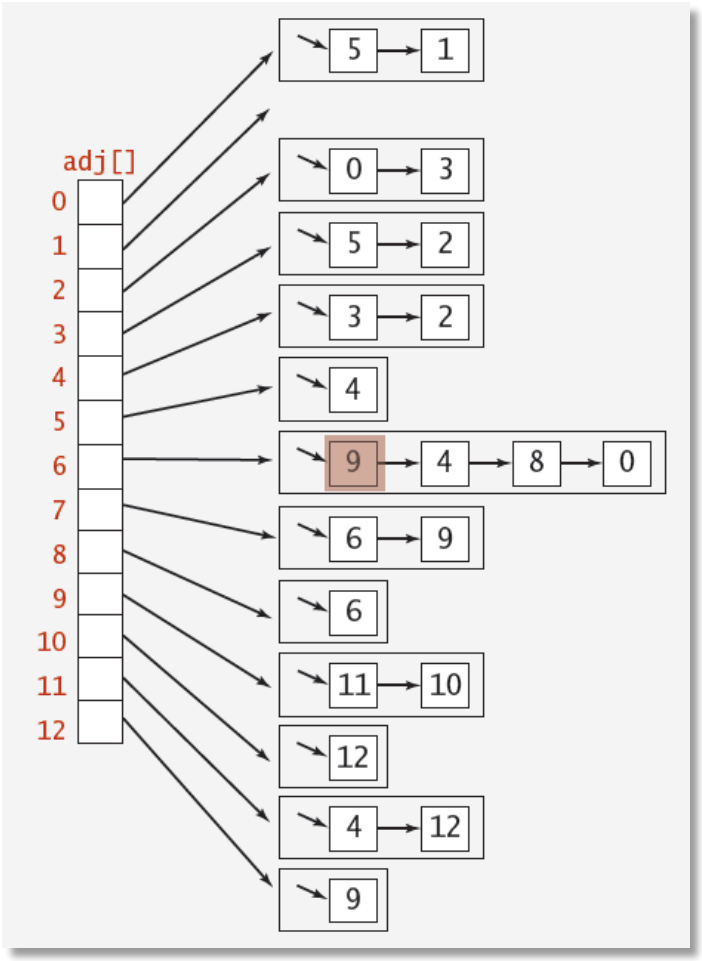
```
In in = new In(args[0]);
Digraph G = new Digraph(in);

for (int v = 0; v < G.V(); v++)
    for (int w : G.adj(V))
        StdOut.println(v + "->" + w);
```

خواندن و ایجاد گراف
از جریان ورودی

چاپ یالها
(هر یال یک بار)

نمایش گراف جهت‌دار: لیست همسایگی





لیست همسایگی: پیاده‌سازی در جاوا


```
public class Graph
{
    private final int V;
    private int E;
    private Bag<Integer>[] adj;

    public Graph(int V)
    {
       this.V = V;
       adj = (Bag<Integer>[]) new Bag[V];
       for (int v = 0; v < V; v++)
          adj[v] = new Bag<Integer>();
    }

    public void addEdge(int v, int w)
    {
       adj[v].add(w);
       adj[w].add(v);
       E++;
    }

    public Iterable<Integer> adj(int v)
    {  return adj[v];  }
}
```

لیست‌های همسایگی

ایجاد یک گراف تهی شامل V رأس

افزودن یال v-w

تکرارگر برای رئوس همسایه رأس v

لیست همسایگی: پیاده‌سازی در جاوا


```
public class Digraph
{
   private final int V;
   private int E;
   private Bag<Integer>[] adj;

   public Digraph(int V)
   {
      this.V = V;
      adj = (Bag<Integer>[]) new Bag[V];
      for (int v = 0; v < V; v++)
         adj[v] = new Bag<Integer>();
   }

   public void addEdge(int v, int w)
   {
      adj[v].add(w);

      E++;
   }

 public Iterable<Integer> adj(int v)
   {  return adj[v];  }
}
```

لیست‌های همسایگی

ایجاد یک گراف تهی
شامل V رأس

افزودن یال v→w

تکرارگر برای رئوس
همسایه رأس v

نمایش گراف جهت‌دار

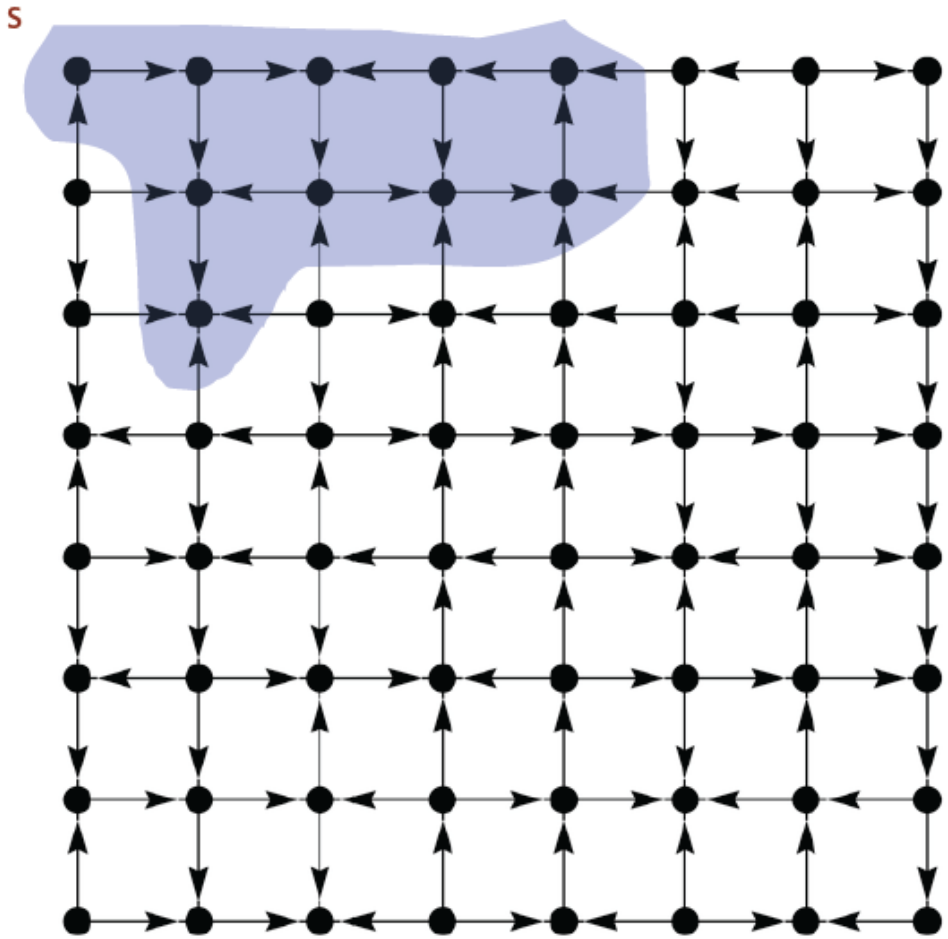
- در عمل. از لیست همسایگی استفاده کنید.
  - الگوریتم‌های گراف بر اساس حلقه زدن بر روی رئوس همسایه‌ی **v** کار می‌کنند.
  - گراف‌های جهت‌دار در دنیای واقعی بیشتر به سمت گراف‌های خلوت تمایل دارند.

روش نمایش	حافظه	افزودن یال	بررسی وجود یال بین رئوس **v** و **w**	چرخیدن روی رئوس مجاور **v**
لیست یال‌ها	E	1	E	E
ماتریس همسایگی	V^2	1	1	V
لیست همسایگی	E + V	1	outdegree(v)	indegree(v)

جستجو در گراف جهت‌دار

دسترس پذیری

- مسئله. تمام رئوس قابل دسترس از رأس **s** را پیدا کن.



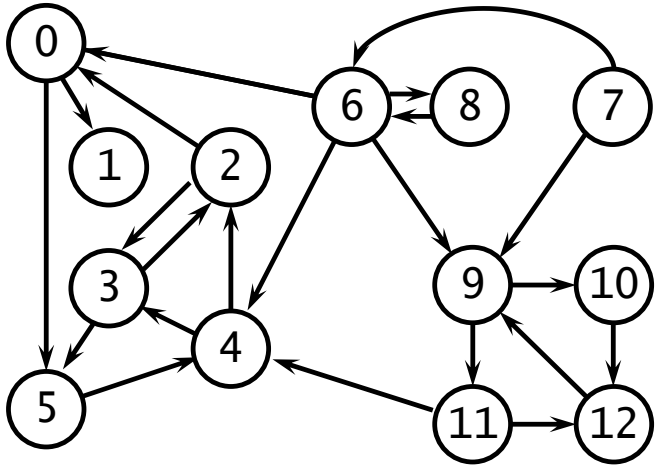
جستجوی عمقی در گراف جهت‌دار

- همانند جستجوی عمقی در گراف بدون جهت.
  - هر گراف بدون جهت یک گراف جهت‌دار است. (هر یال معادل دو یال جهت‌دار)
  - جستجوی DFS یک الگوریتم جستجو برای گراف‌های جهت‌دار است.

DFS (to visit a vertex v)

Mark v as visited.

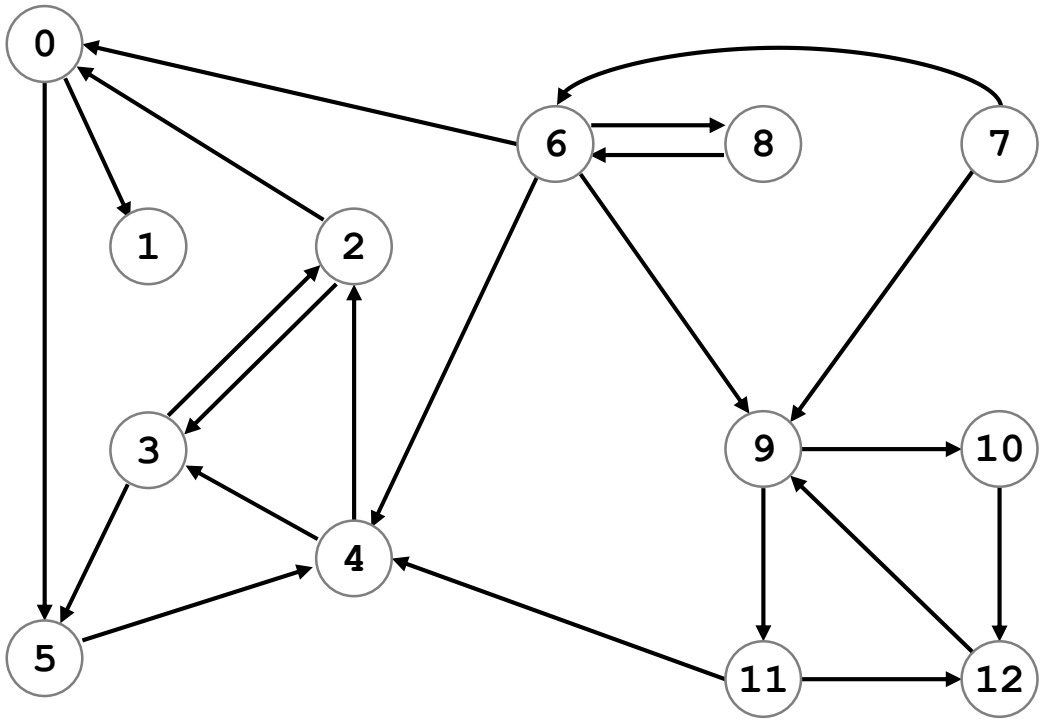
Recursively visit all unmarked vertices w adjacent to v.



جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	F	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

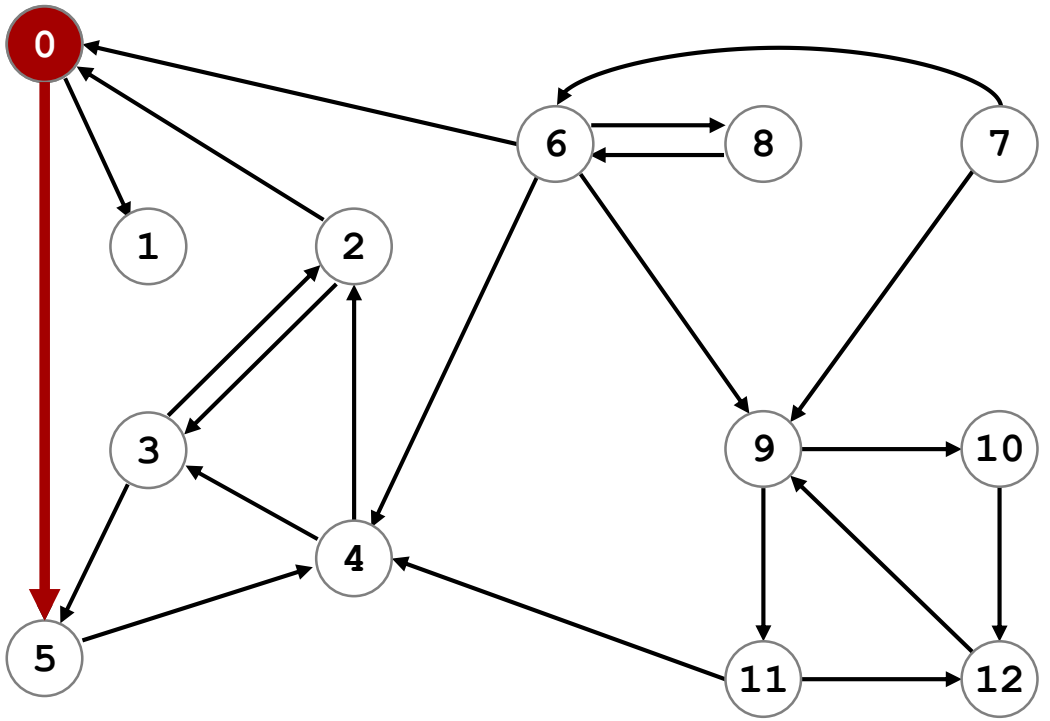


Graph G

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

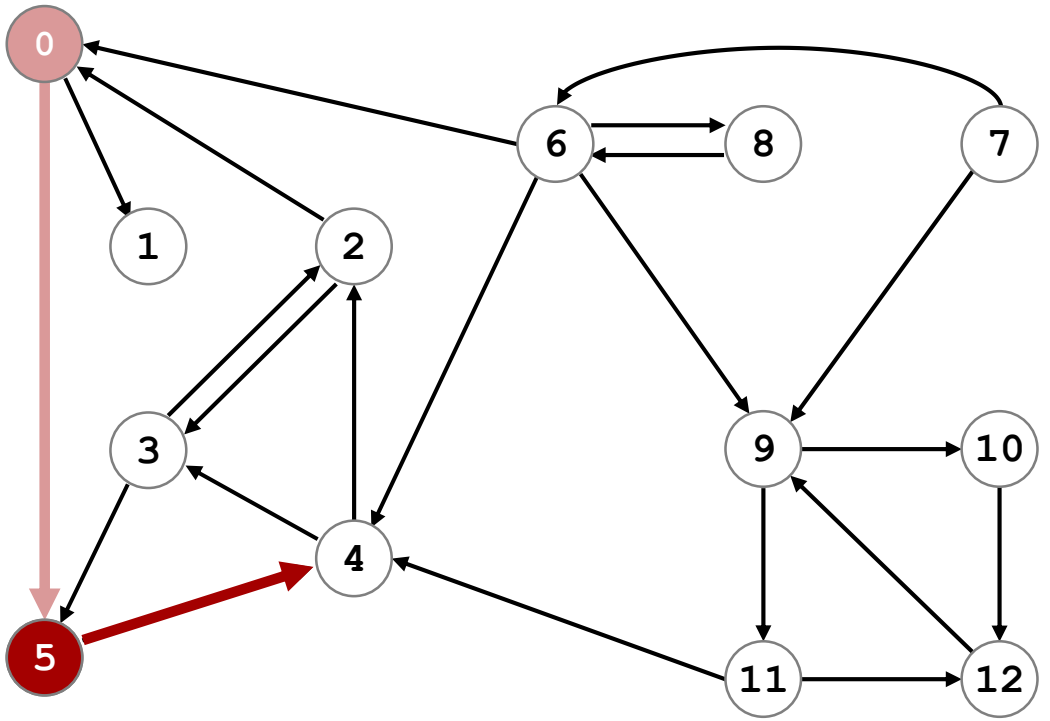


Visit 0: check 5, check 1

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	F	-
3	F	-
4	F	-
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

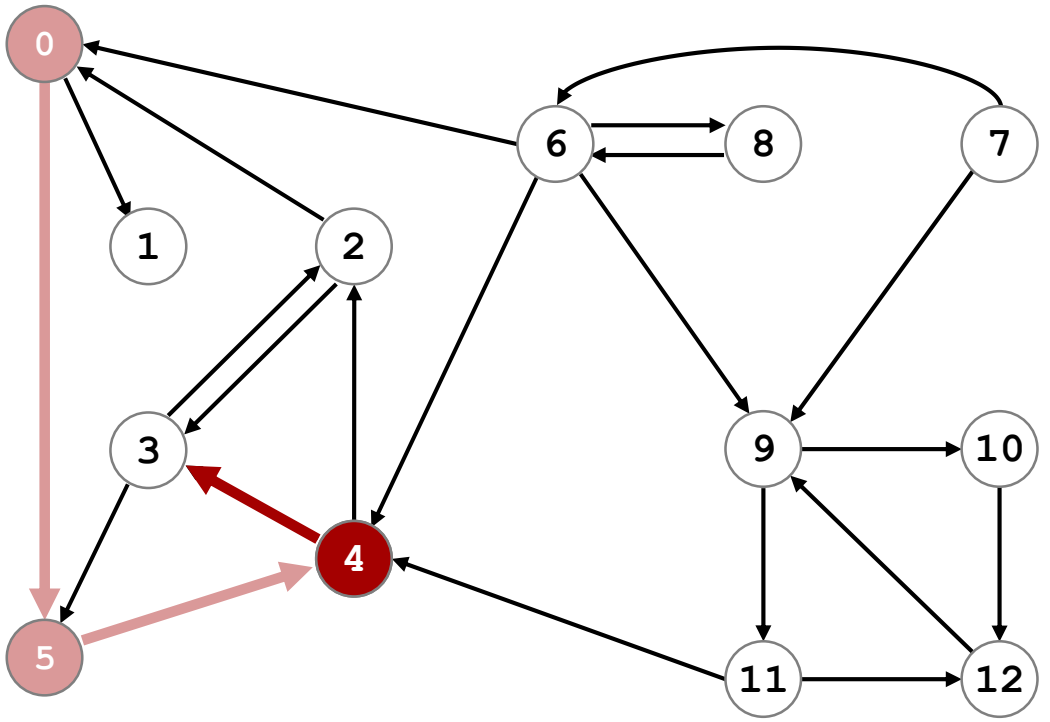


Visit 5: check 4

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	F	-
3	F	-
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

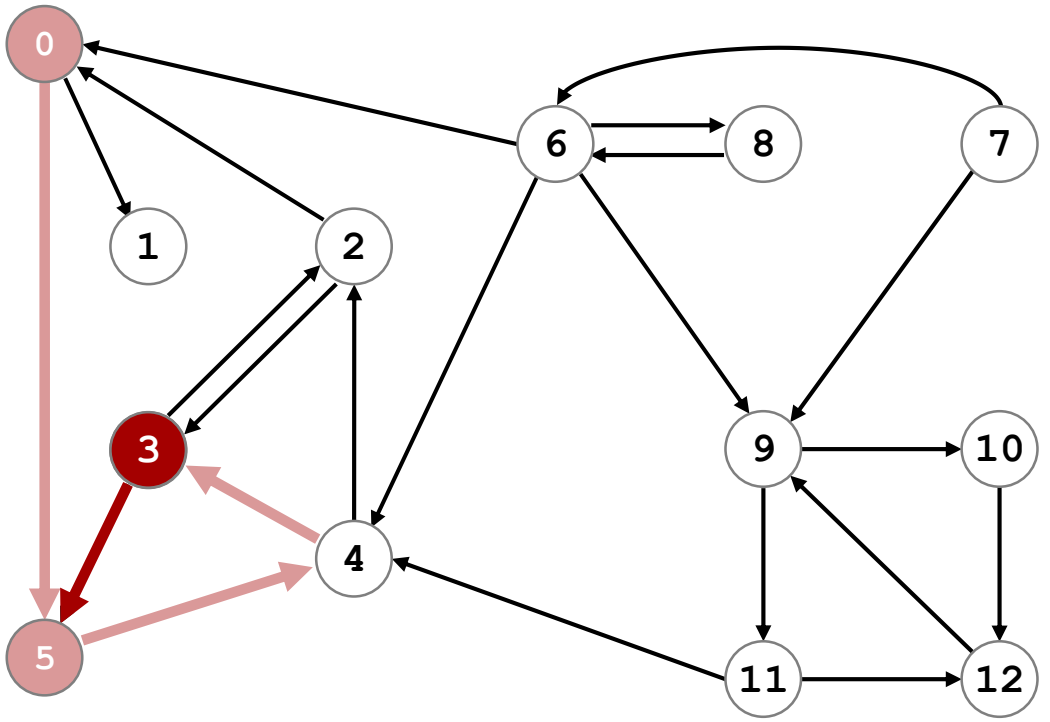


Visit 4: check 3, check 2

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	F	-
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

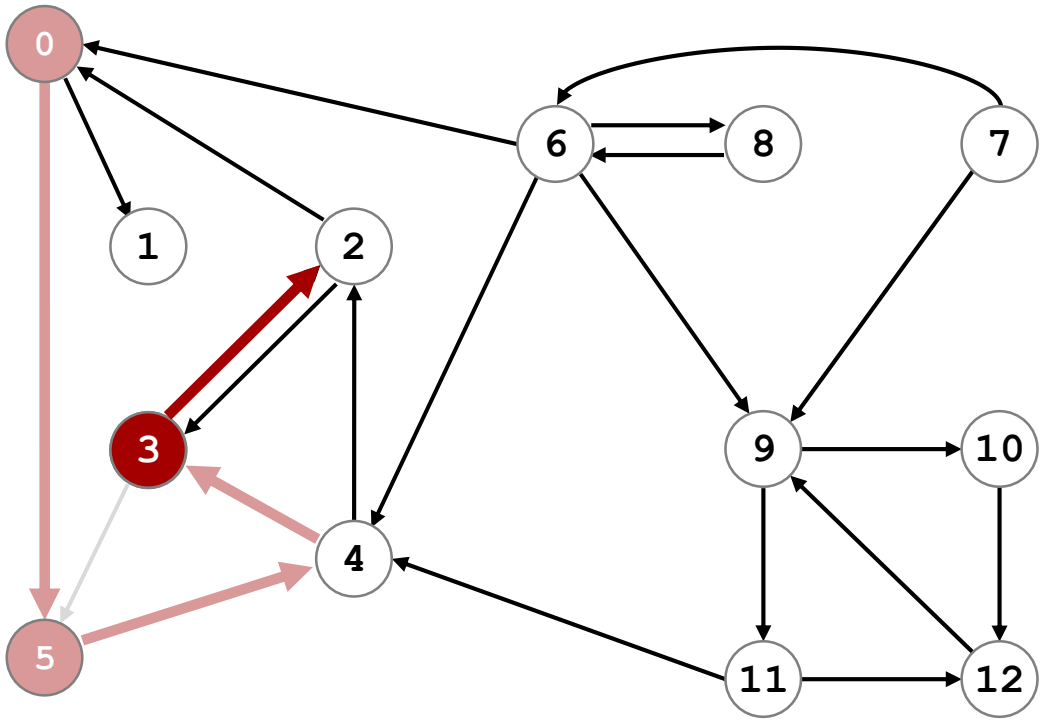


Visit 3: check 5, check 2

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	F	-
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

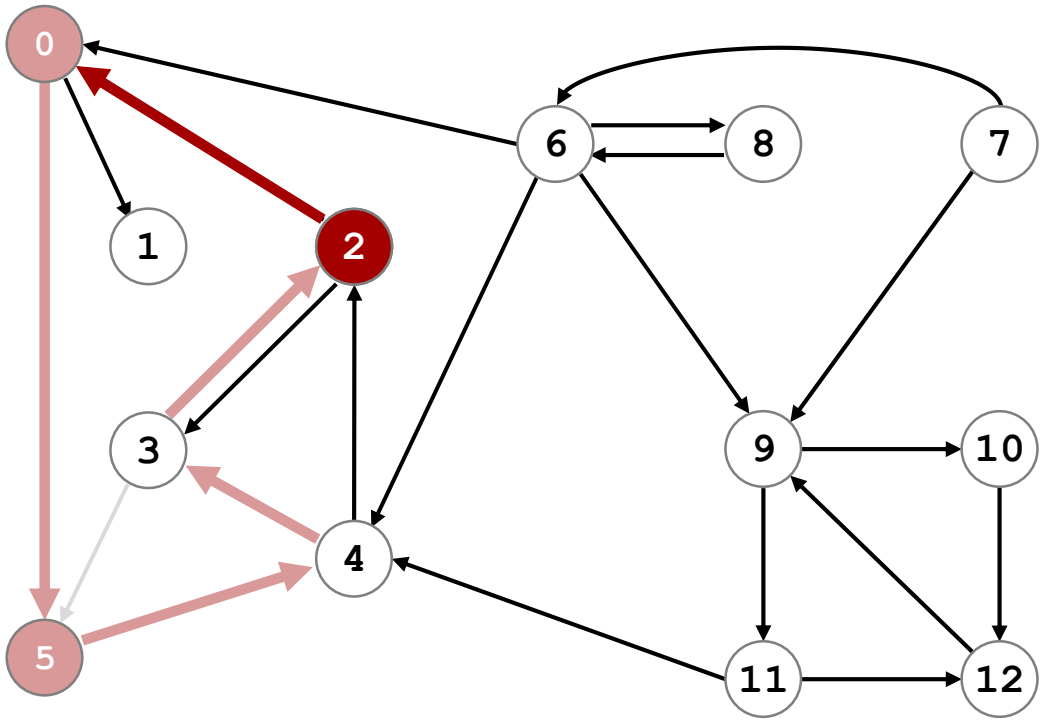


Visit 3: check 5, check 2

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

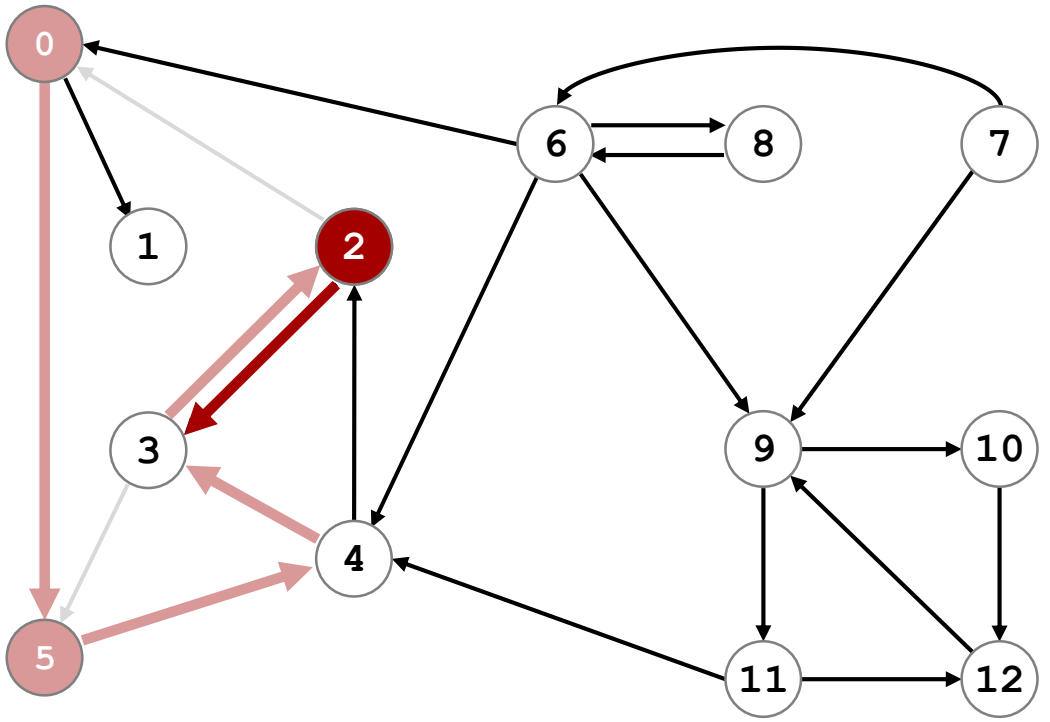


Visit 2: check 0, check 3

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

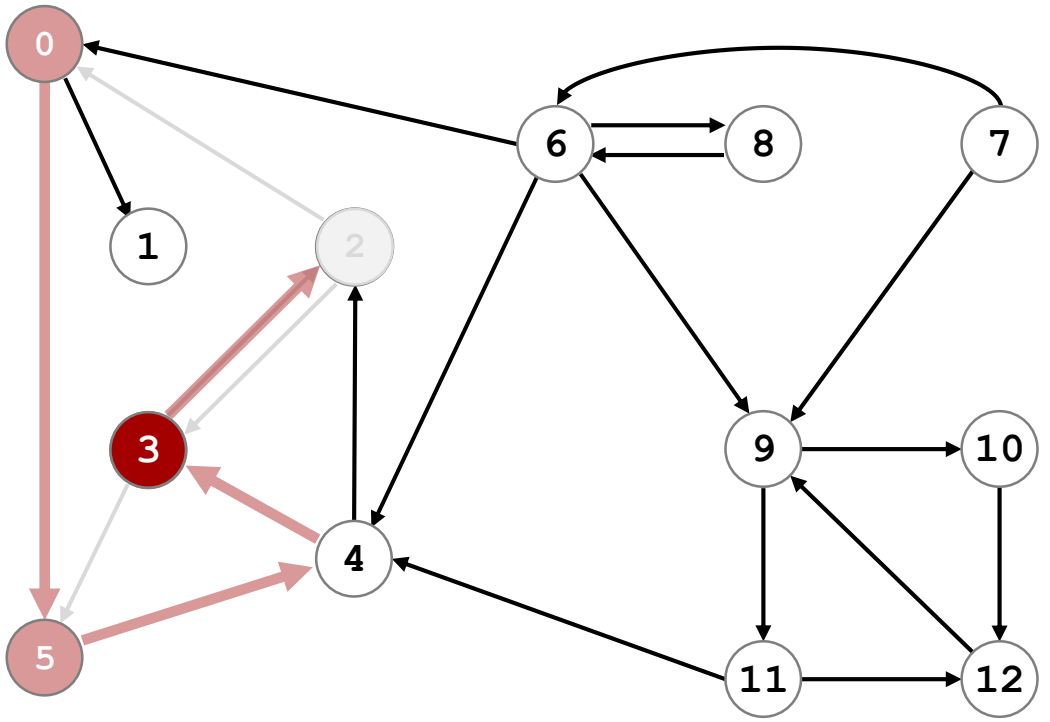


Visit 2: check 0, check 3

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

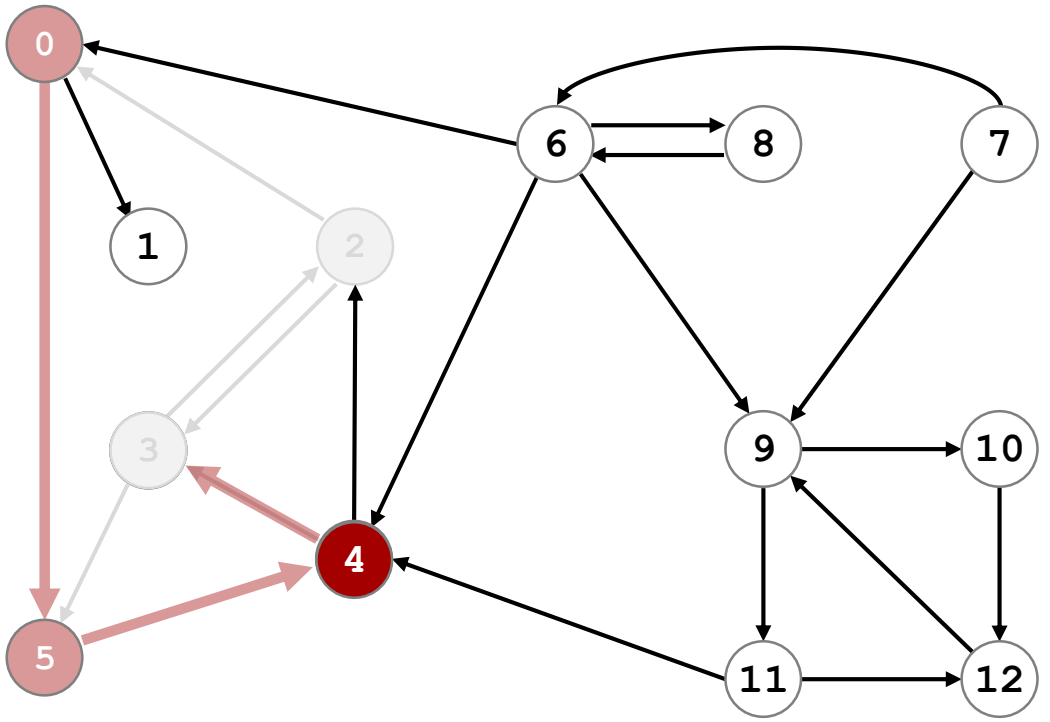


2 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

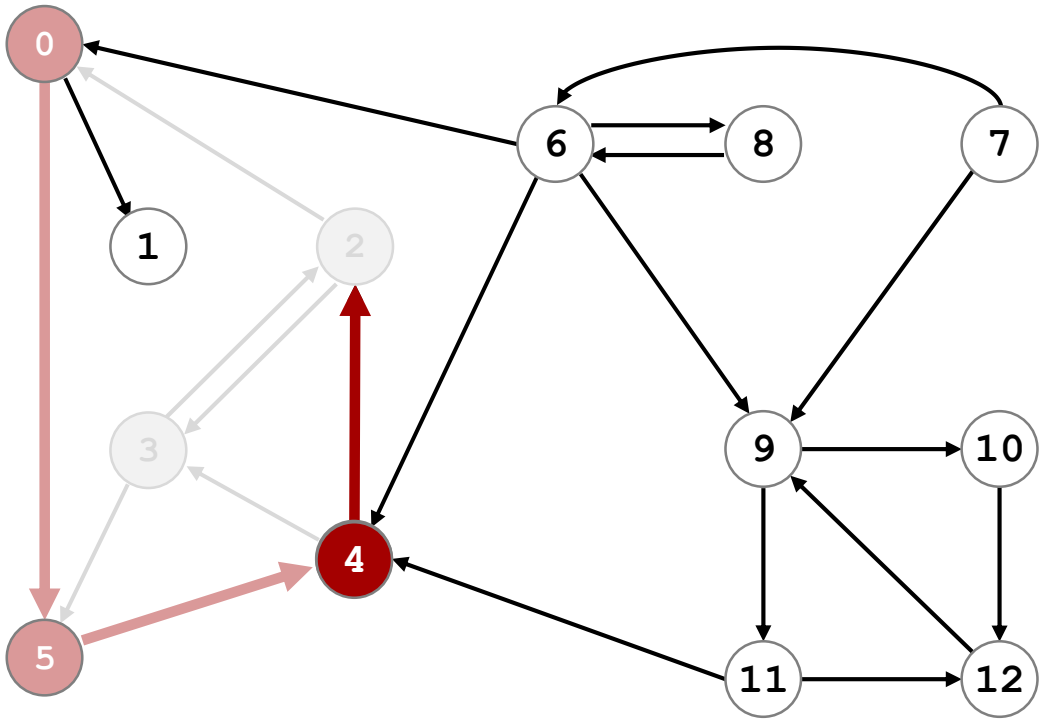


3 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

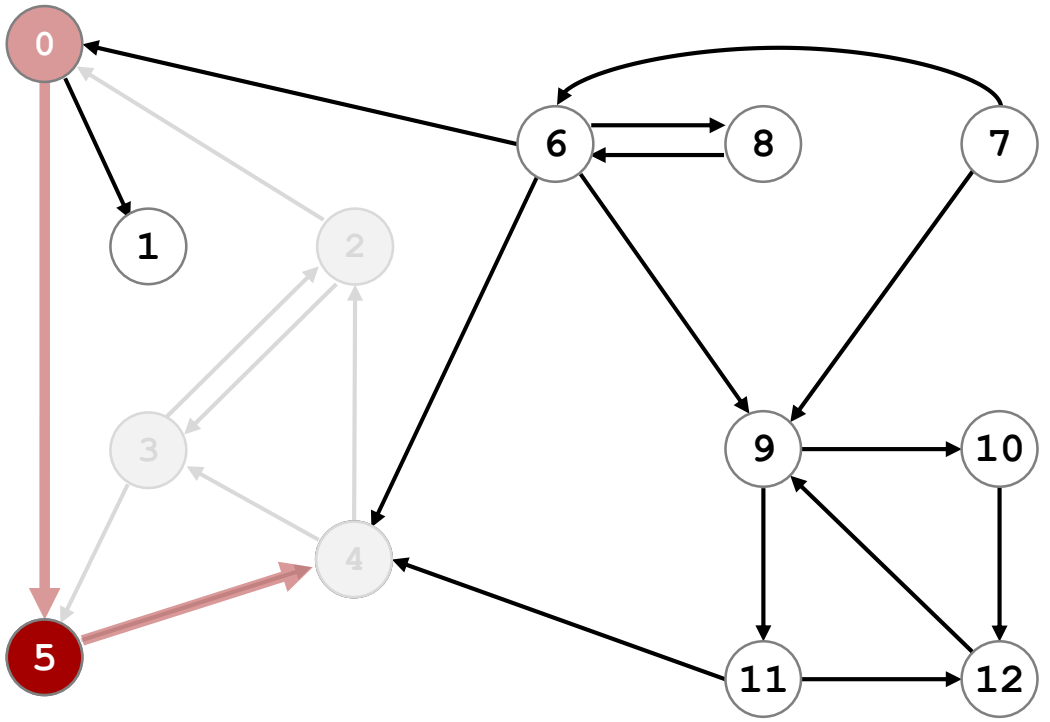


Visit 4: check 3, check 2

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

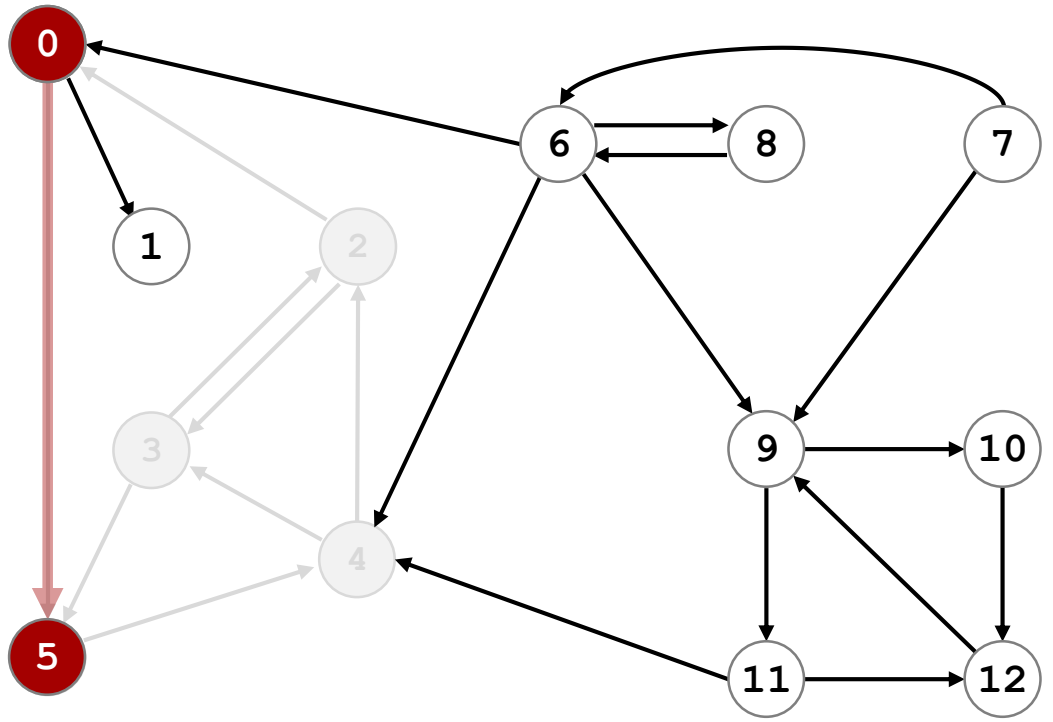


4 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

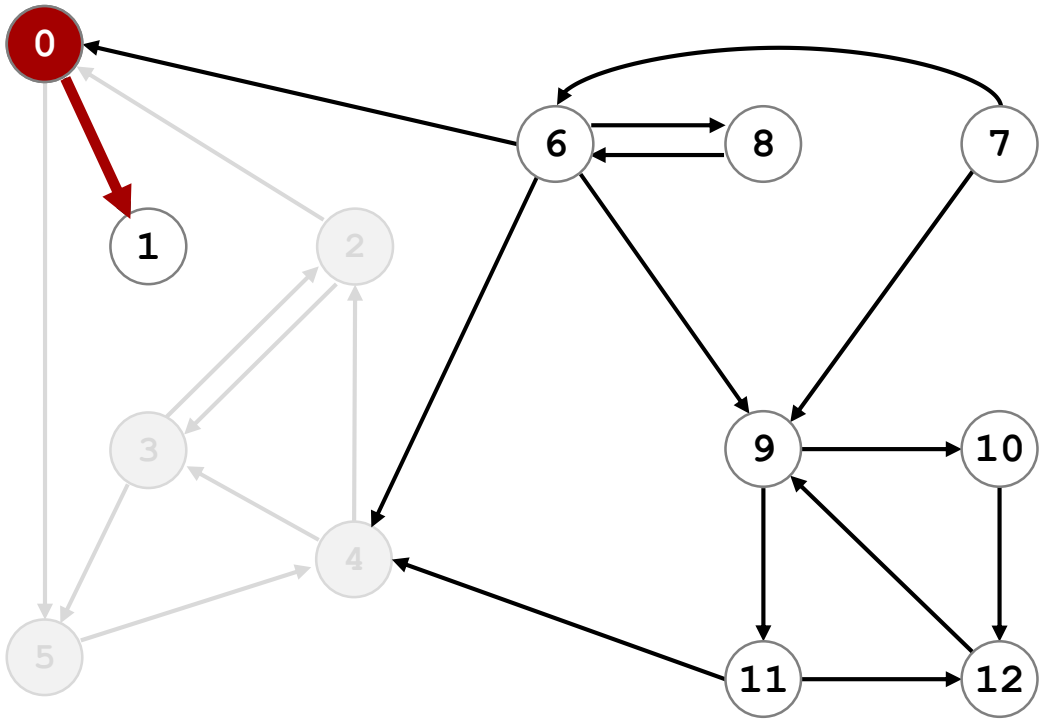


5 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	F	-
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

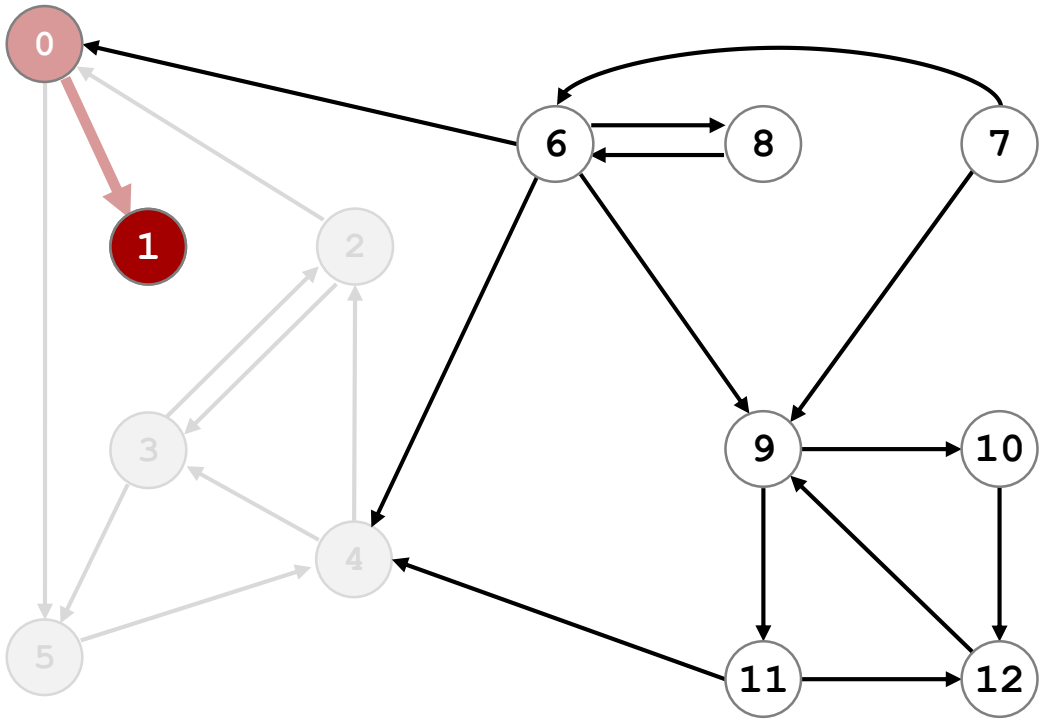


Visit 0: check 5, check 1

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	T	0
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

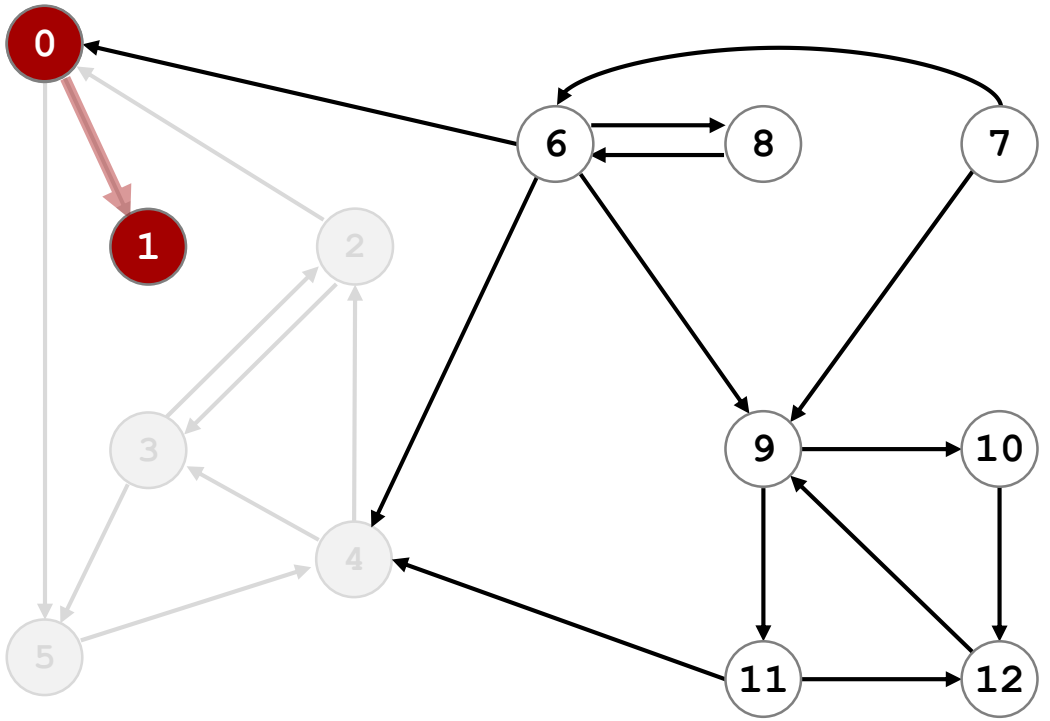


Visit 1

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	T	0
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

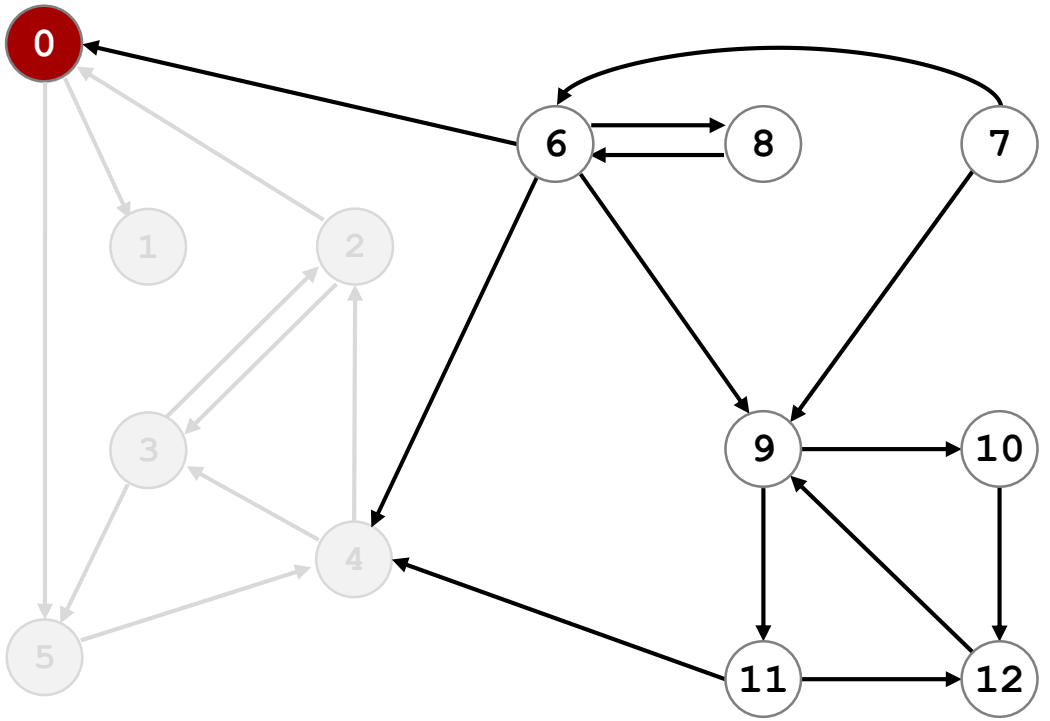


1 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	T	0
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

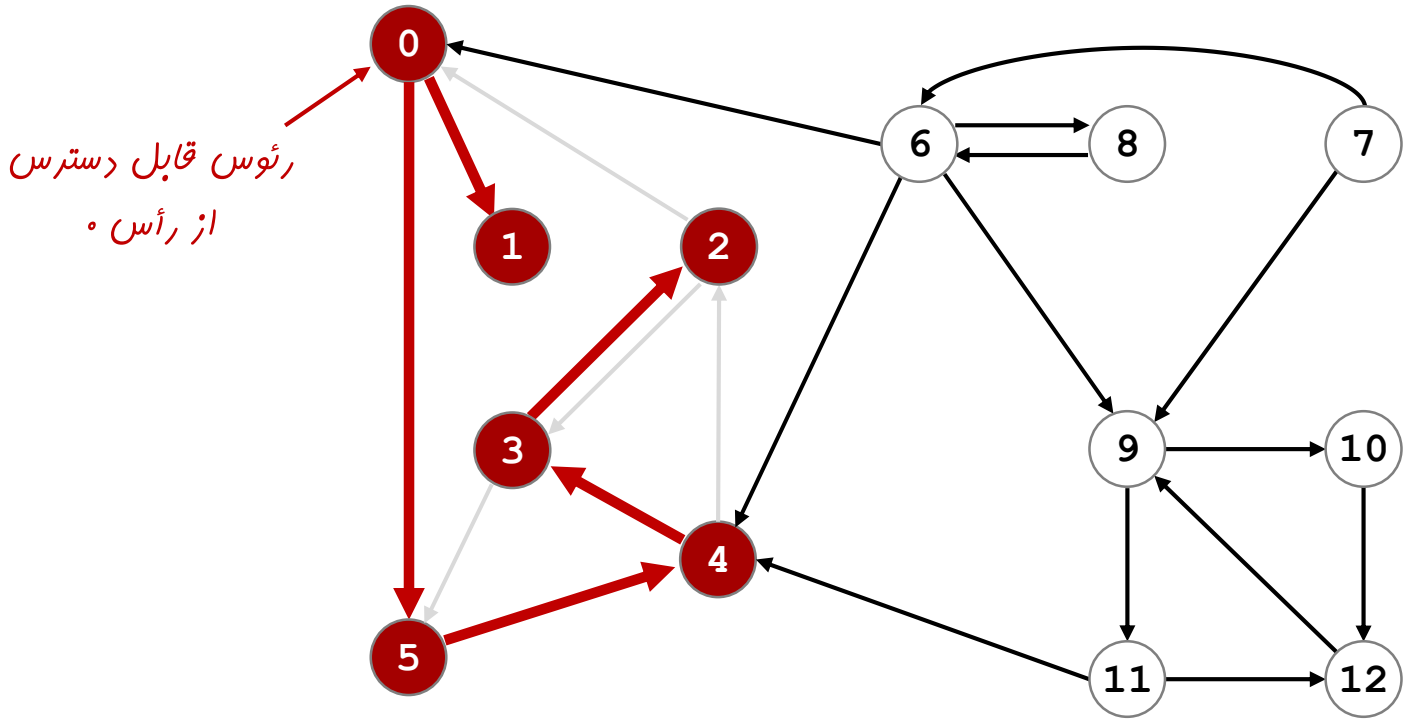


0 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را علامت گذاری کن.
  - تمام رئوس مجاور v را که علامت گذاری نشده‌اند، به صورت بازگشتی ملاقات کن.

v	marked[]	prev[]
0	T	-
1	T	0
2	T	3
3	T	4
4	T	5
5	T	0
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-



جستجوی عمقی: پیاده‌سازی در جاوا


```
public class DepthFirstPaths
{
   private boolean[] marked;

   public DepthFirstPaths(Graph G, int s)
   {
      marked = new boolean(G.V());
      dfs(G, s);
   }

   private void dfs(Graph G, int v)
   {
      marked[v] = true;
      for (int w : G.adj(v))
         if (!marked[w]) dfs(G, w);
   }

   public boolean visited(int v)
   {  return marked[v];  }
}
```

اگر v به s متصل باشد
marked[v] = true

مقداردهی اولیه

یافتن رئوس متصل به s

جستجوی بازگشتی DFS

آیا رأس v به s متصل است؟

جستجوی عمقی در گراف جهت‌دار


```
public class DirectedDFS
{
   private boolean[] marked;

   public DirectedDFS(Digraph G, int s)
   {
      marked = new boolean[G.V()];
      dfs(G, s);
   }

   public void dfs(Digraph G, int v)
   {
      marked[v] = true;
      for (int w : G.adj(v))
         if (!marked[w]) dfs(G, w);
   }

   public boolean visited(int v)
   {  return marked[v];  }
}
```

اگر v به s متصل باشد
marked[v] = true

مقداردهی اولیه

یافتن رئوس متصل به s

جستجوی بازگشتی DFS

آیا رأس v به s متصل است؟

کاربرد دسترس‌پذیری: جمع‌آوری زباله

هر ساختمان‌داده یک گراف جهت‌دار است.

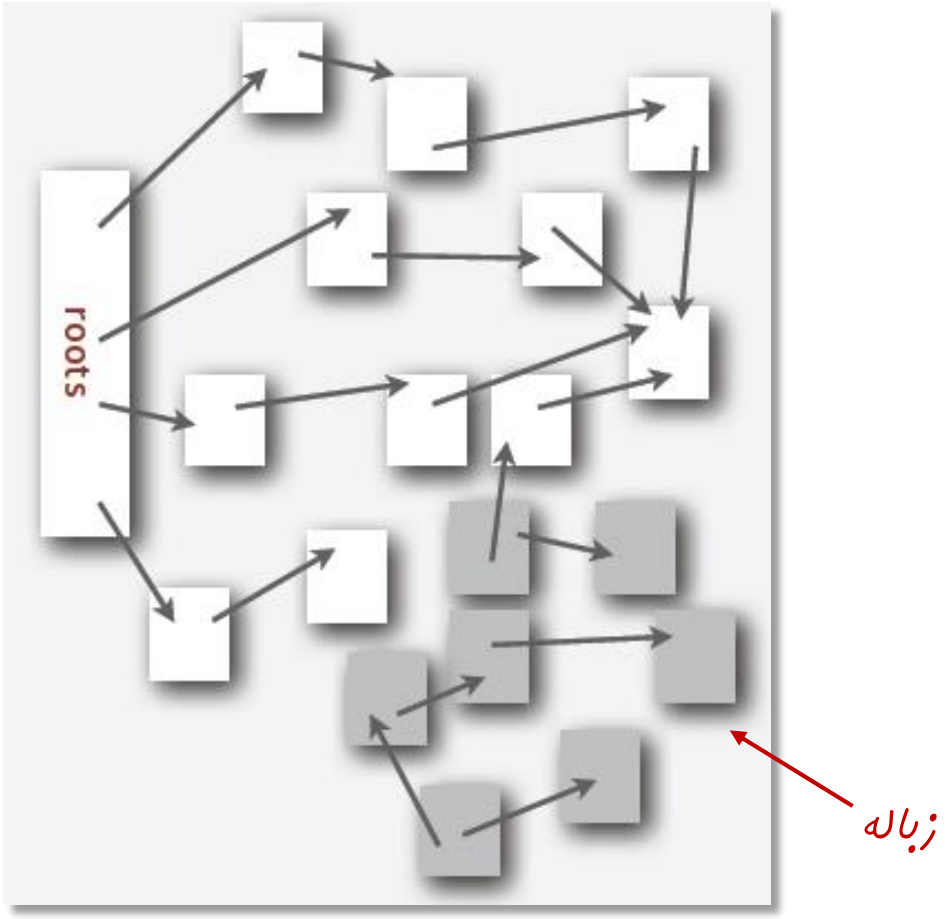
- رئوس: اشیا
- یال ها: ارجاع به اشیا

ریشه‌ها.

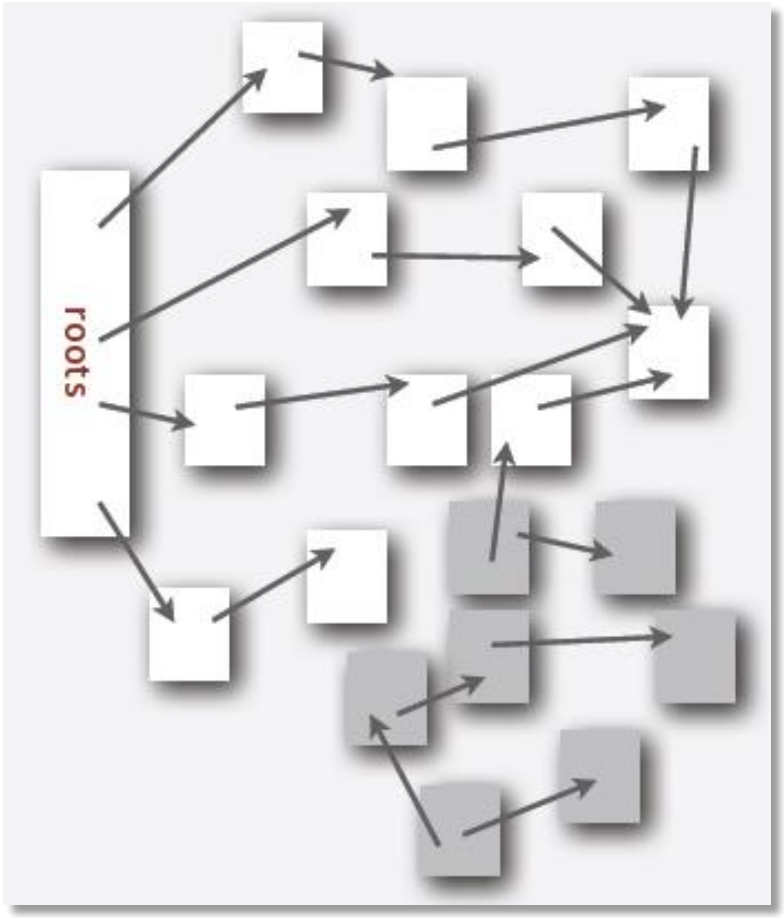
- اشیایی که به طور مستقیم قابل دسترس هستند.

اشیای دسترس‌پذیر.

- اشیایی که به طور غیر مستقیم قابل دسترس هستند.



کاربرد دسترس‌پذیری: جمع‌آوری زباله



الگوریتم نشانه‌گذاری-جاروب. [مک کارتی، ۱۹۶۰]

- نشانه‌گذاری: اشیای قابل دسترس را نشانه‌گذاری کن.
- جاروب: اگر یک شی نشانه‌گذاری نشده است، زباله است. [آن را به لیست فضای آزاد اضافه کن]

مصرف حافظه.

- هر شی به یک بیت برای نشانه‌گذاری نیاز دارد.
- به علاوه حافظه مورد نیاز برای پشته در DFS.

کاربردهای جستجوی عمقی در گراف جهت‌دار

- مسایل ساده.
  - دسترس‌پذیری
  - مسیریابی
  - مرتب‌سازی توپولوژیکی
  - تشخیص دور
- مسایل سخت‌تر.
  - مسیر اویلری جهت‌دار
  - یافتن مؤلفه‌های همبند قوی

جستجوی سطحی در گراف جهت‌دار

- همانند جستجوی سطحی در گراف بدون جهت.
  - هر گراف بدون جهت یک گراف جهت‌دار است. [هر یال معادل دو یال جهت‌دار]
  - جستجوی BFS یک الگوریتم جستجو برای گراف‌های جهت‌دار است.

BFS (from source vertex s)

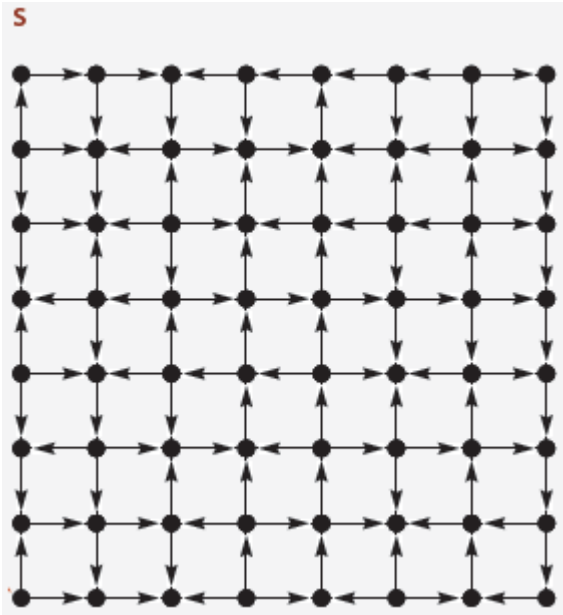
Put s onto a FIFO queue and mark s as visited.

Repeat until the queue is empty:

- remove the least recently added vertex v

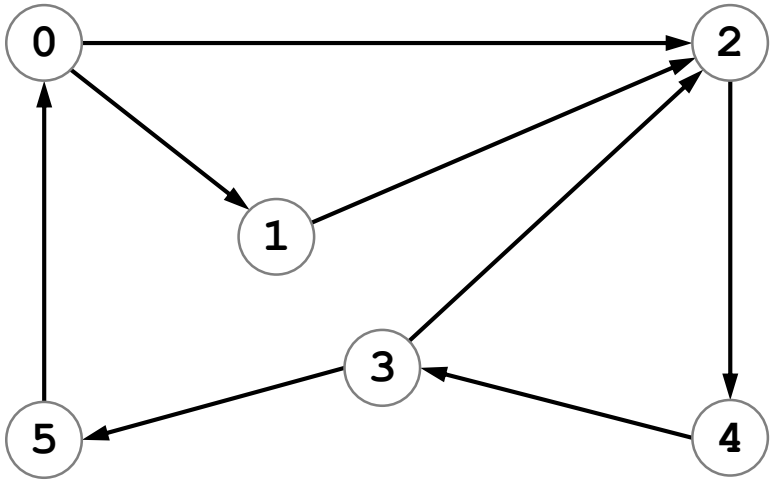
- add each unmarked vertex w pointing from v:

add w to queue and mark it as visited.



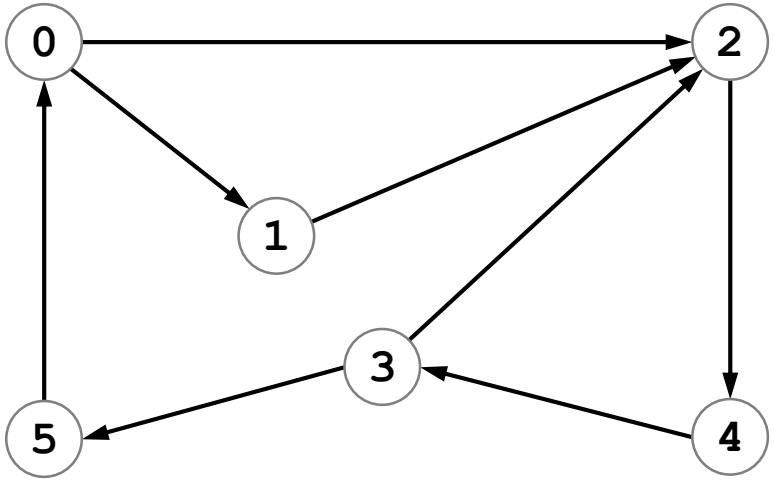
جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.



جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

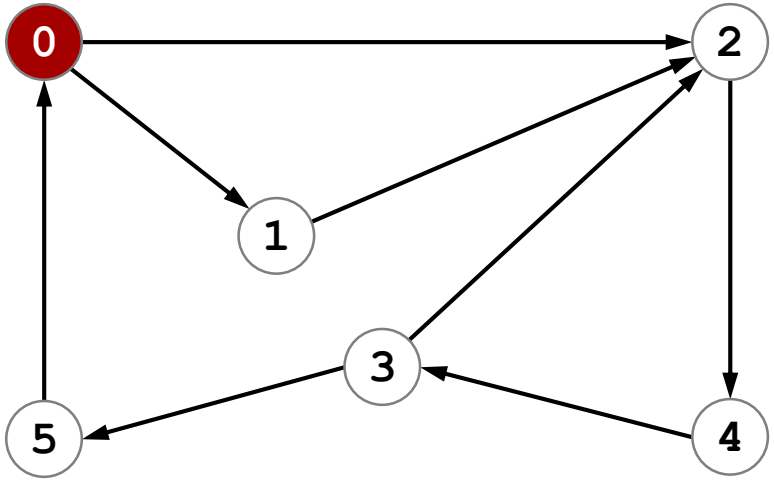


enqueue(0)

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	3	-	-
	4	-	-
	5	-	-
0			

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

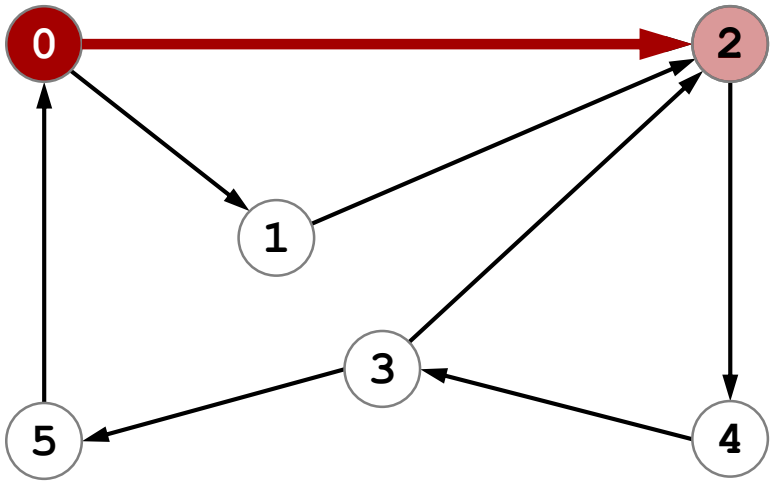


dequeue 0

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	3	-	-
	4	-	-
	5	-	-
0			

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

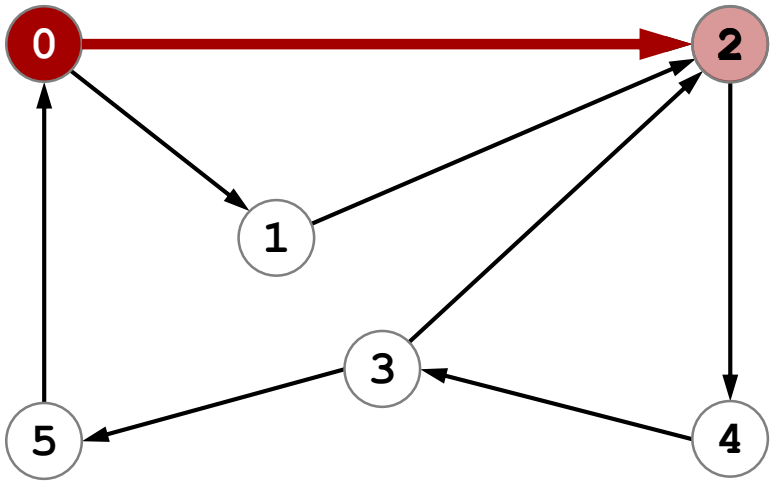


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	3	-	-
	4	-	-
	5	-	-

```
dequeue 0: check 2, check 1
```

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

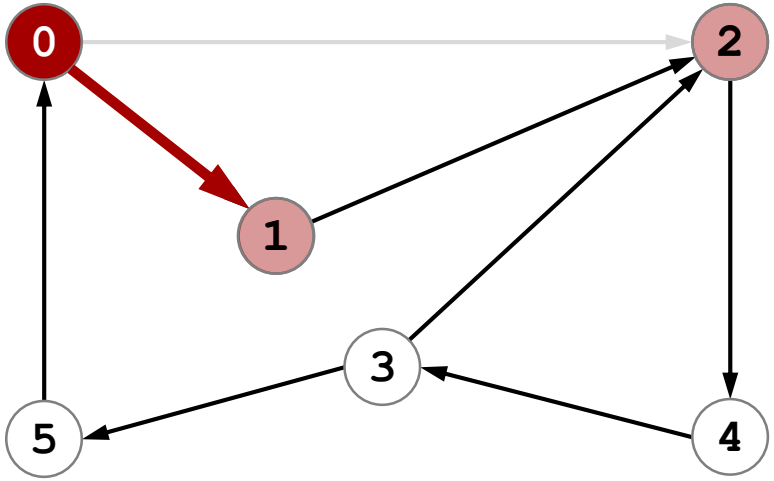


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	0	1
	3	-	-
	4	-	-
	5	-	-

```
dequeue 0: check 2, check 1
```

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

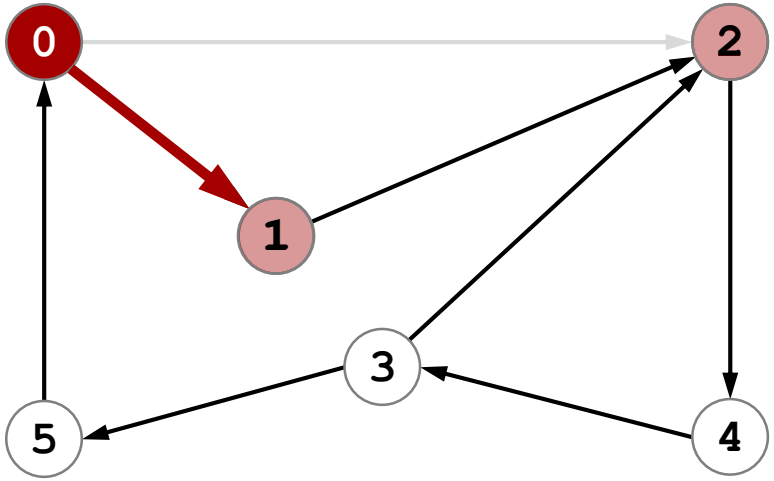


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	0	1
	3	-	-
	4	-	-
	5	-	-
2			

```
dequeue 0: check 2, check 1
```

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

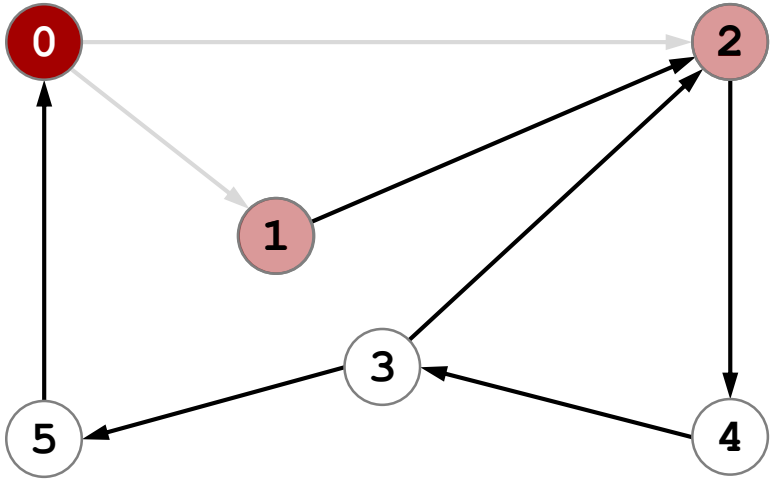


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	-	-
	5	-	-
2			

```
dequeue 0: check 2, check 1
```

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

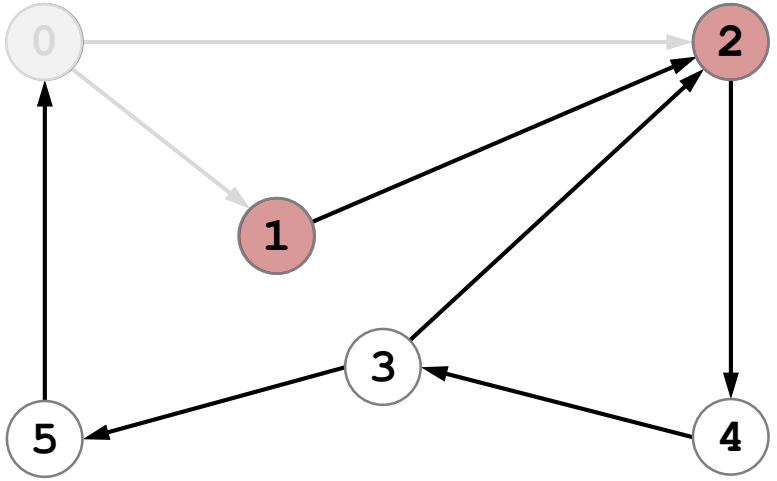


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	-	-
1	5	-	-
2			

```
dequeue 0: check 2, check 1
```

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

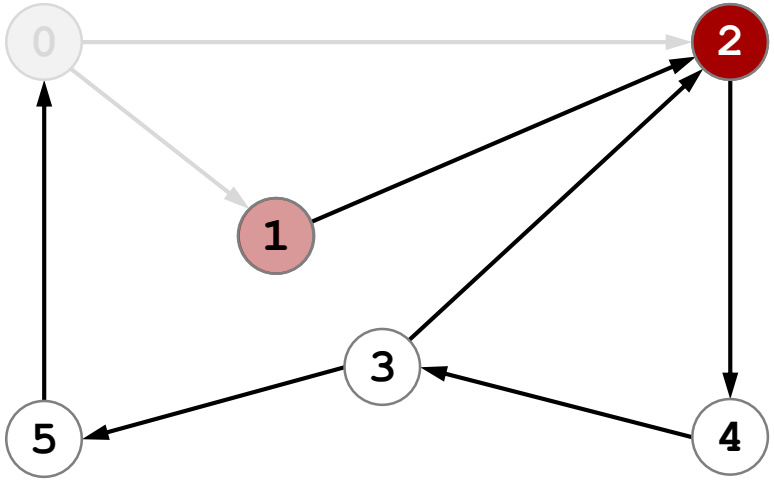


0 done

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	-	-
1	5	-	-
2			

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

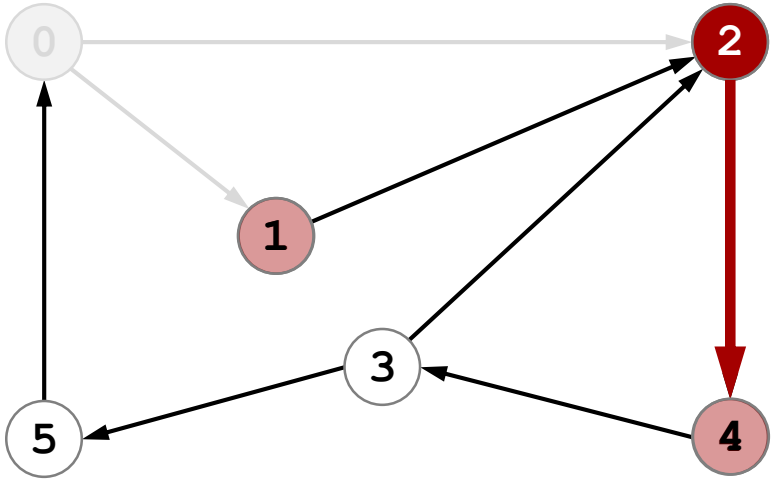


dequeue 2

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	-	-
1	5	-	-
2			

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

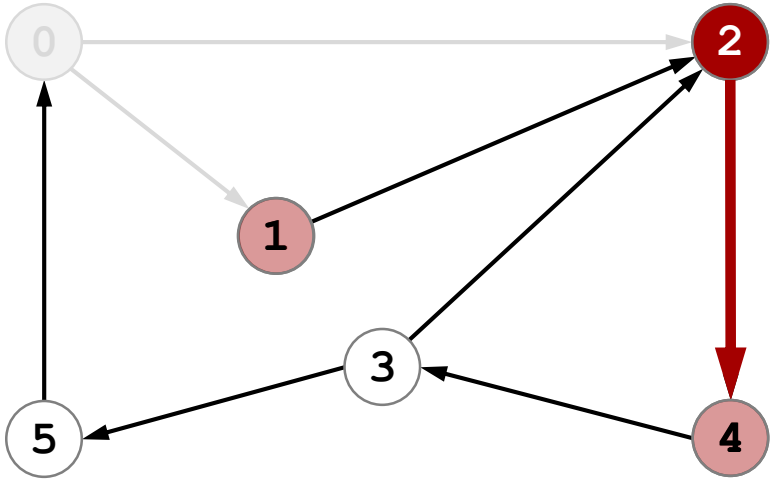


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	-	-
1	5	-	-
2			

dequeue 2: check 4

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

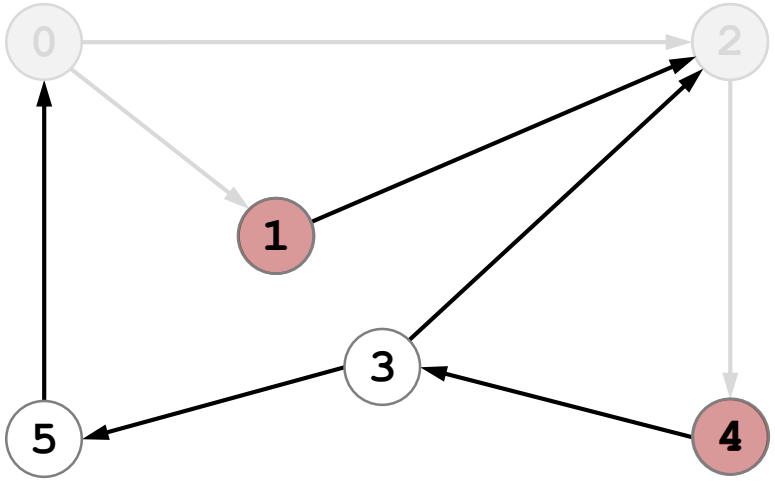


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	2	2
	5	-	-
1			

dequeue 2: check 4

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

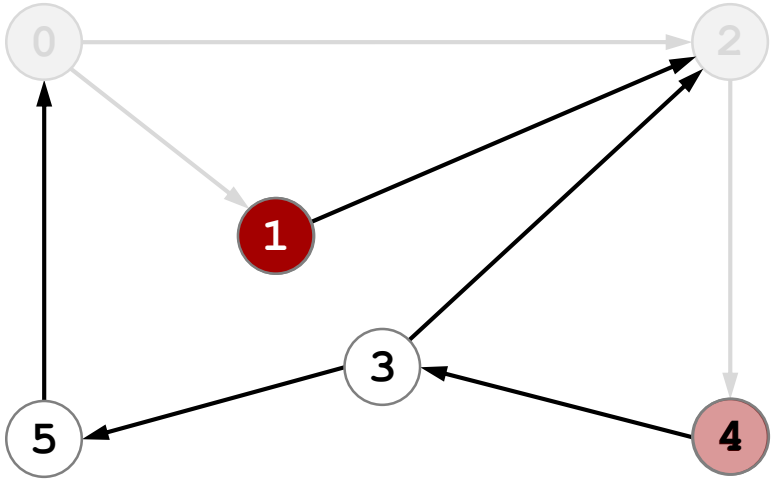


2 done

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	2	2
4	5	-	-
1			

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

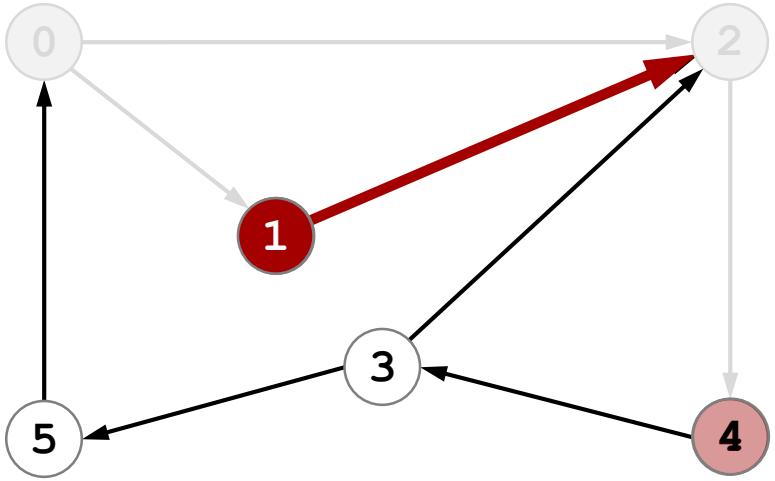


dequeue 1

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	2	2
4	5	-	-
1			

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

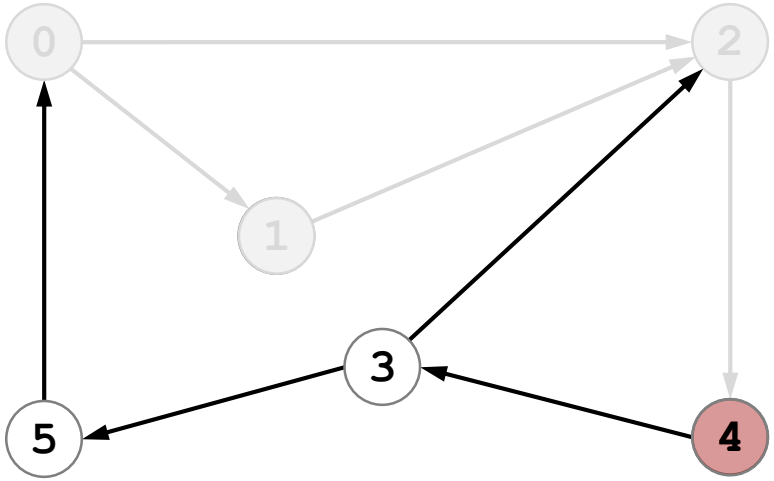


dequeue 1: check 2

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	2	2
	5	-	-
4			

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.



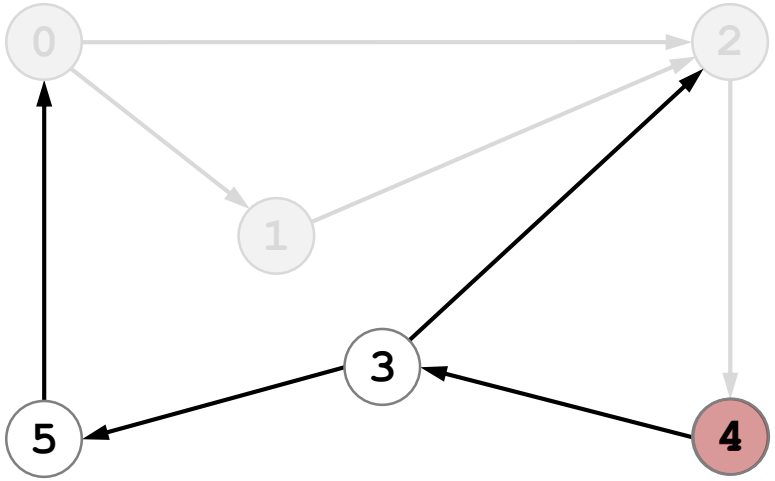
1 done

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	2	2
	5	-	-

4

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.



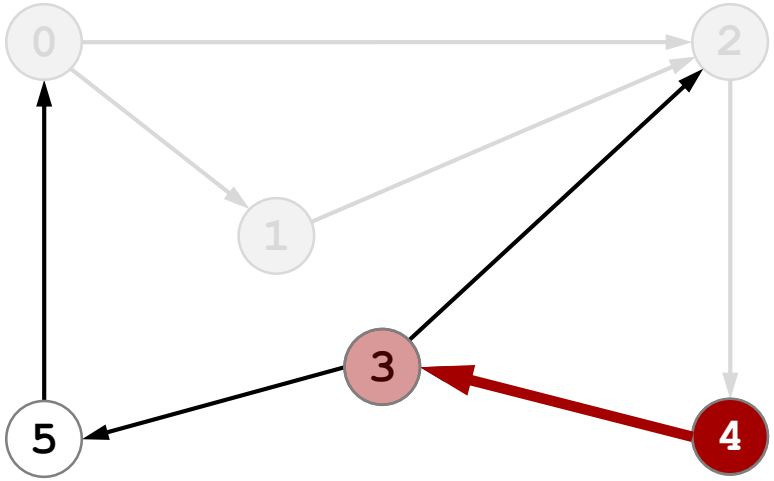
dequeue 4

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	2	2
	5	-	-

4

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

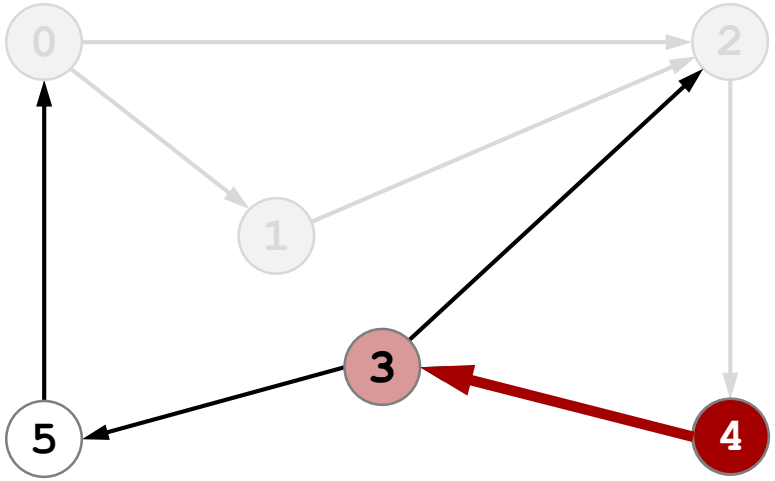


dequeue 4: check 3

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	-	-
	4	2	2
	5	-	-

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

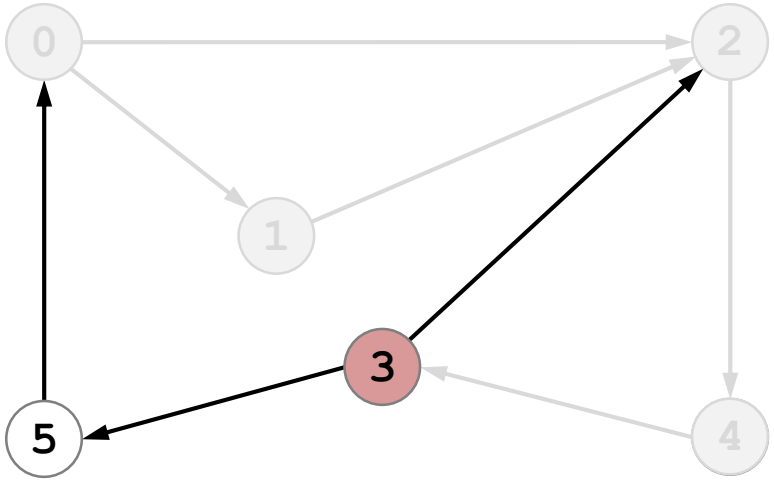


dequeue 4: check 3

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	-	-

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

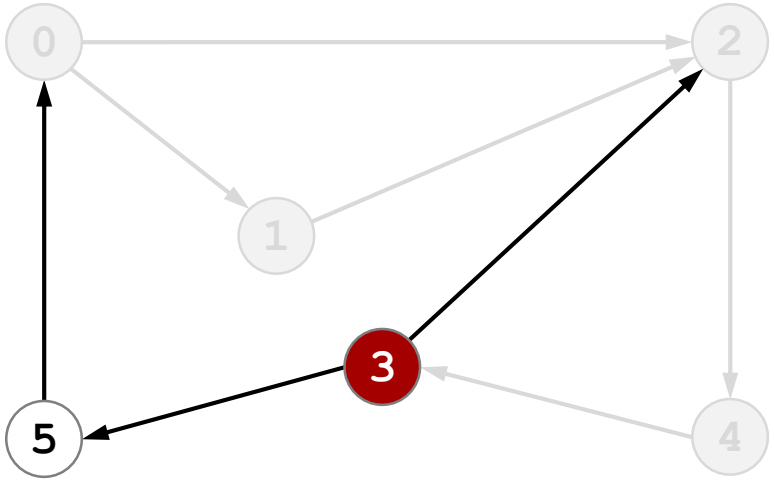


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	-	-
3			

4 done

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.



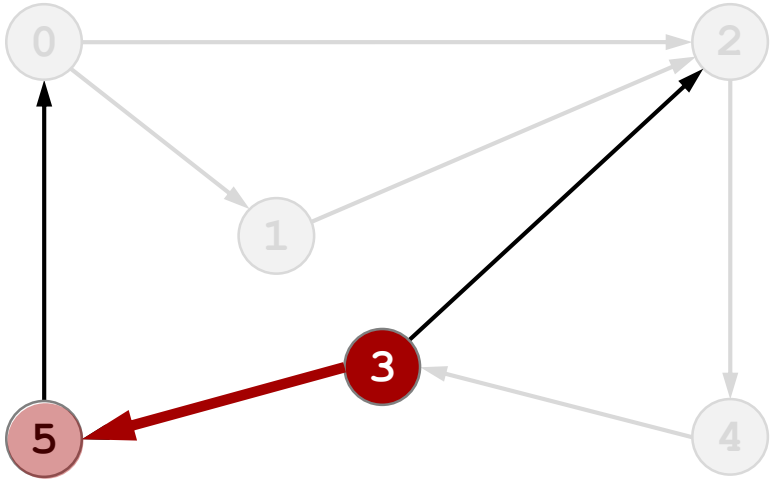
dequeue 3

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	-	-

3

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

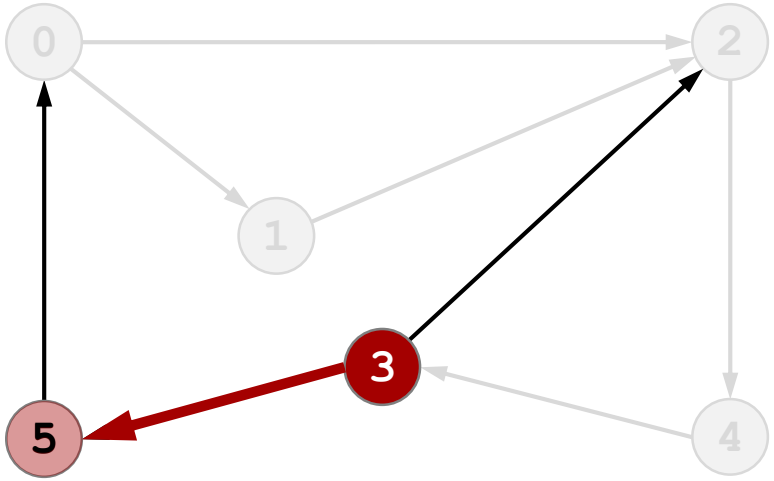


dequeue 3: check 5, check 2

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	3	4

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

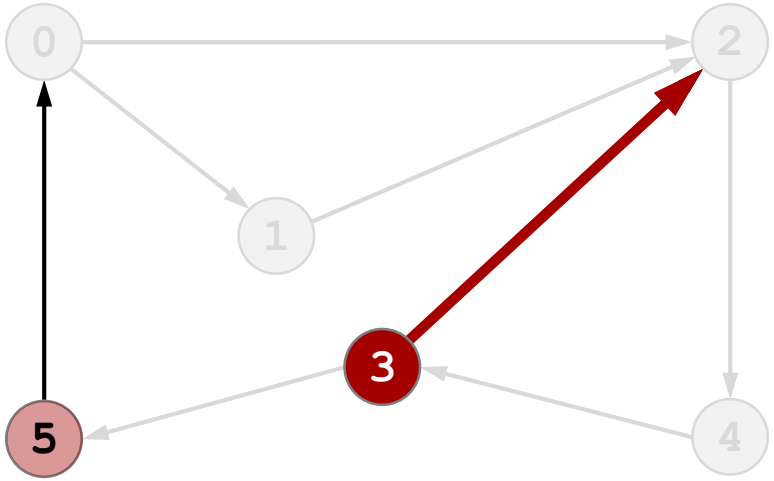


dequeue 3: check 5, check 2

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	3	4

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

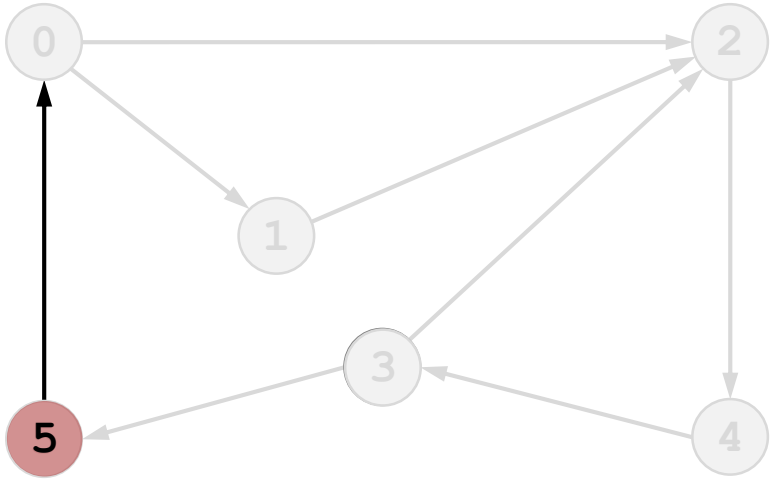


dequeue 3: check 5, **check 2**

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	3	4
5			

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

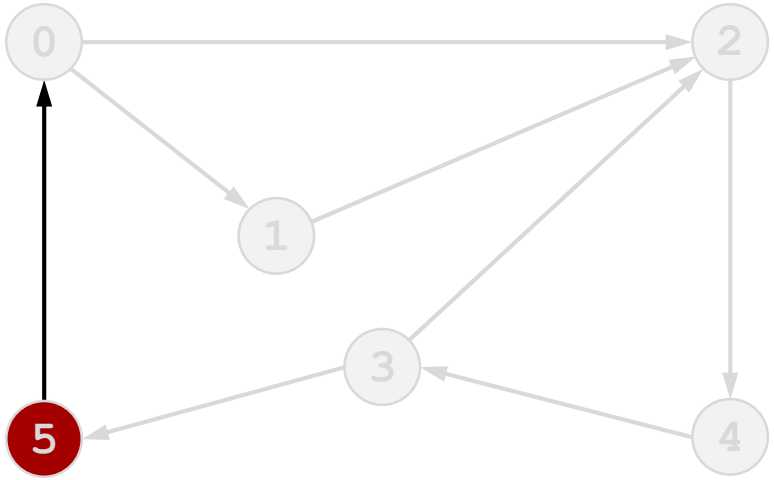


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	3	4
5			

3 done

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.



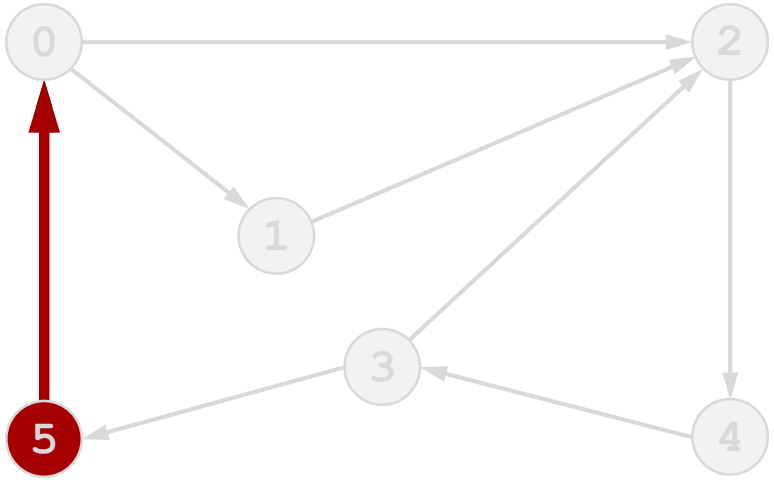
dequeue 5

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	3	4

5

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.



queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	3	4

dequeue 5: check 0

جستجوی سطحی: اجرای نمایشی

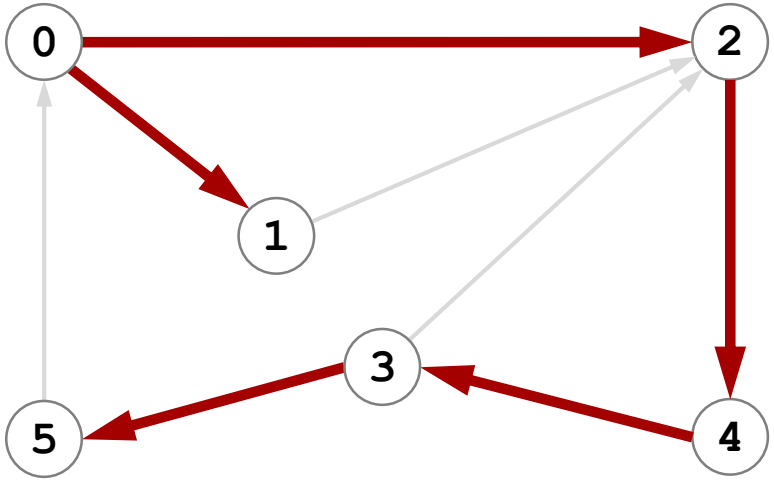
- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	3	4

5 done

جستجوی سطحی: اجرای نمایشی

- تا زمانی که صف که خالی نشده، مراحل زیر را تکرار کن:
  - رأس v را از ابتدای صف بردار.
  - تمام رئوس مجاور v را که علامت‌گذاری نشده‌اند به صف اضافه کن و آنها را علامت‌گذاری کن.



done

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	0	1
	2	0	1
	3	4	3
	4	2	2
	5	3	4

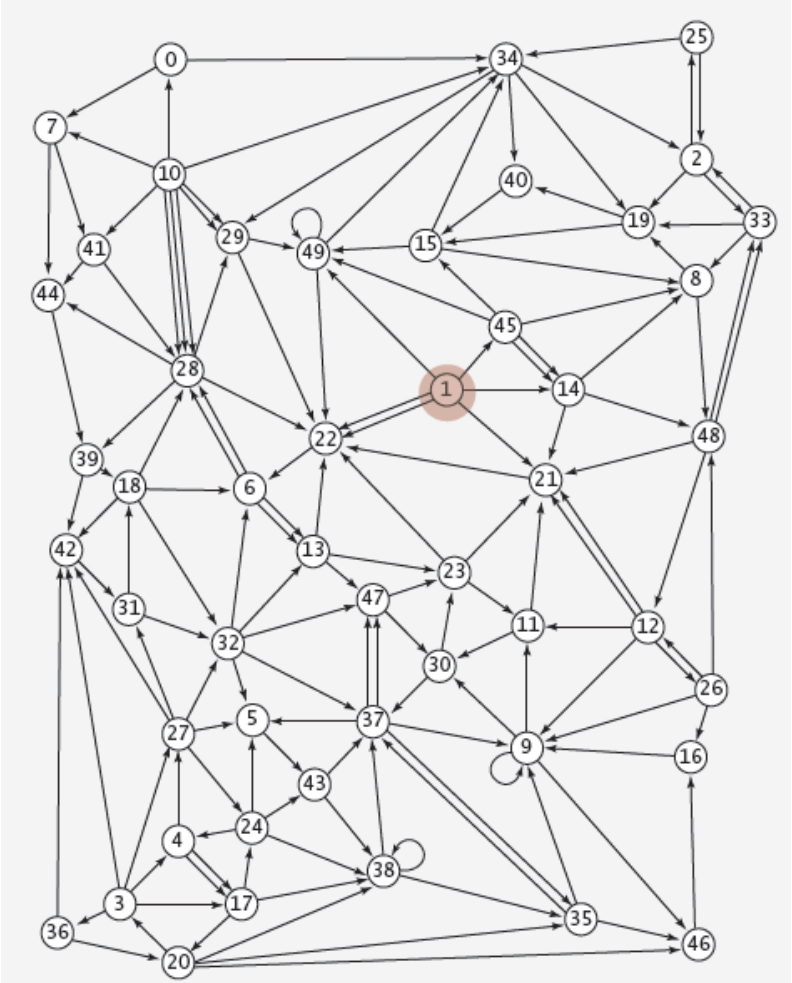
جستجوی سطحی: پیاده‌سازی جاوا


```
public class BreadthFirstPaths
{
    private boolean[] marked;
    private int[] prev;
    private int s;

    ...

    public void bfs(Graph G, int s) {
       Queue<Integer> q = new Queue<integer>();
       q.enqueue(s);
       marked[s] = true;
       while (!q.isEmpty()) {
          int v = q.dequeue();
          for (int w : G.adj(v))
             if (!marked[w])
             {
                marked[w] = true;
                prev[w] = v;
                q.enqueue(w);
              }
        }
    }
}
```

جستجوی سطحی: پیاده‌سازی جاوا


```
public class BreadthFirstDirectedPaths
{
    private boolean[] marked;
    private int[] prev;
    private int s;

    ...

    public void bfs(Digraph G, int s) {
       Queue<Integer> q = new Queue<integer>();
       q.enqueue(s);
       marked[s] = true;
       while (!q.isEmpty()) {
          int v = q.dequeue();
          for (int w : G.adj(v))
             if (!marked[w])
             {
                marked[w] = true;
                prev[w] = v;
                q.enqueue(w);
              }
        }
    }
}
```

کاربرد جستجوی سطحی در گراف‌های جهت‌دار



کاوش صفحات وب.

- یافتن تمام صفحات وب با شروع از یک صفحه‌ی دلخواه.

جستجوی سطحی.

- صفحه‌ی شروع را به عنوان مبدأ **s** انتخاب کن.
- یک صف برای کاوش صفحات وب در نظر بگیر.
- یک مجموعه از صفحات کشف شده نگهدار.
- صفحه‌ی وب بعدی را از صف حذف و تمام صفحاتی را که از این صفحه به آنها پیوند وجود دارد به صف اضافه کن.

س. چرا جستجوی عمقی نه؟

کاوش صفحات وب: پیاده‌سازی در جاوا


```
Queue<String> queue = new Queue<String>();
SET<String> discovered = new SET<String>();

String root = "www.tabrizu.ac.ir";
queue.enqueue(root);
discovered.add(root);

while (!queue.isEmpty()) {
   String v = queue.dequeue();
   StdOut.println(v);
   In in = new In(v);
   String input = in.readAll();

   String regexp = "http://(\\w+\\.)*(\\w+)";
   Pattern pattern = Pattern.compile(regexp);
   Matcher matcher = pattern.matcher(input);
   while (matcher.find())
   {
      String w = matcher.group();

      if (!discovered.contains(w)) {
         discovered.add(w);
         queue.enqueue(w);
      }

  }
}
```

خواندن محتویات صفحه وب بعدی

یافتن رشته‌های موجود در این صفحه وب به شکل: http://xxx.yyy.zzz

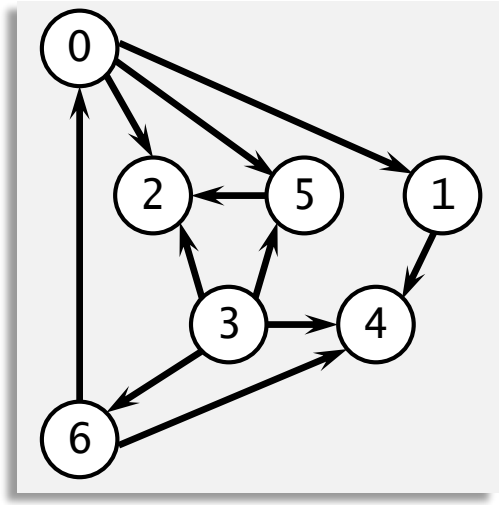
مرتب‌سازی توپولوژیکی

زمان‌بندی وظایف

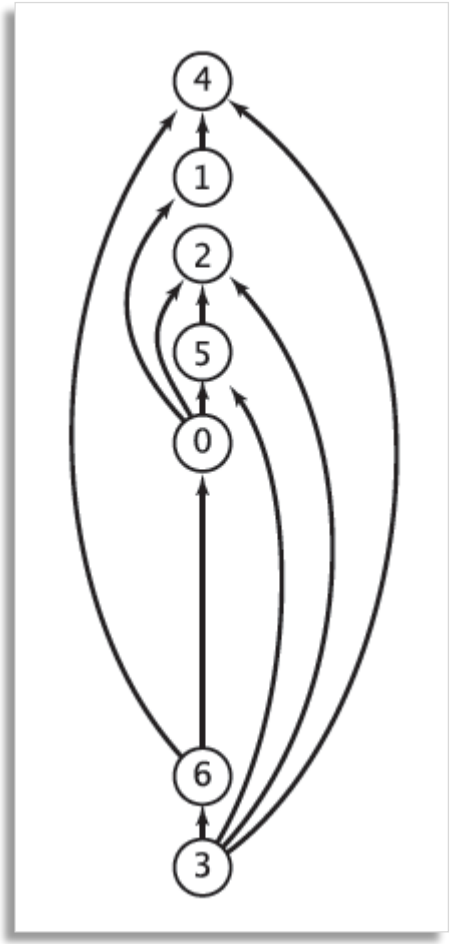
- هدف. با داشتن مجموعه‌ای از وظایف که بر روی آنها یک رابطه‌ی تقدم تعریف شده است، وظایف باید به چه ترتیبی زمانبندی شوند؟
- مدل‌سازی با گراف جهت‌دار. رأس = وظیفه؛ یال = رابطه‌ی تقدم

۰. الگوریتم‌ها
۱. نظریه پیچیدگی محاسباتی
۲. هوش مصنوعی
۳. مبانی کامپیوتر
۴. رمزنگاری
۵. محاسبات علمی
۶. برنامه‌سازی پیشرفته

وظایف



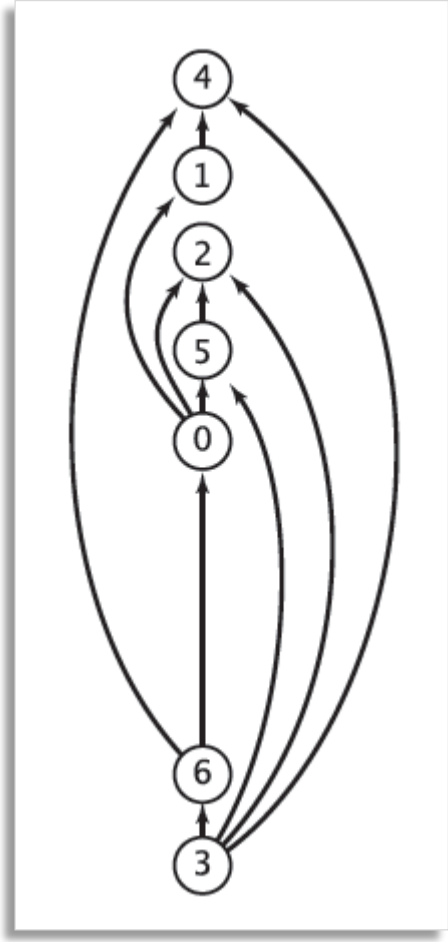
گراف تقدم

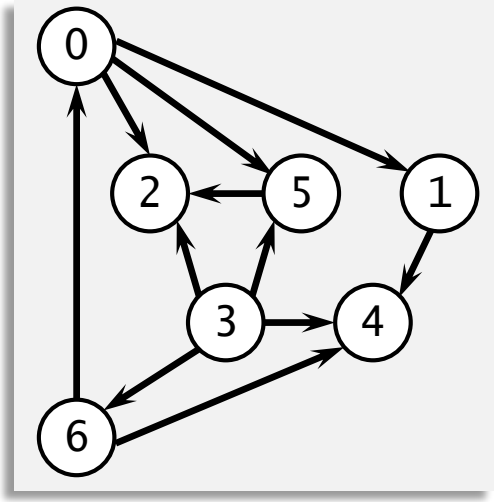


زمانبندی امکان‌پذیر

مرتب‌سازی توپولوژیکی

- DAG. گراف جهت‌دار بدون دور.
- مرتب‌سازی توپولوژیکی. ترسیم مجدد DAG به گونه‌ای که جهت همه‌ی یالها رو به بالا باشد.

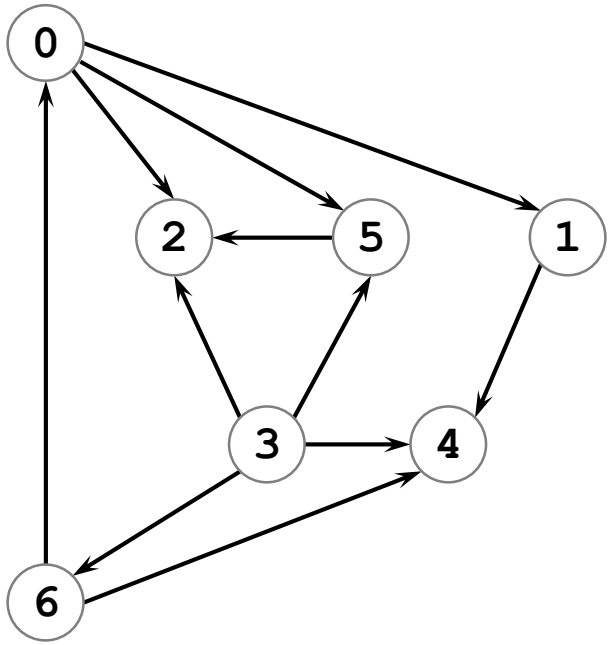




- راه حل. جستجوی عمقی.

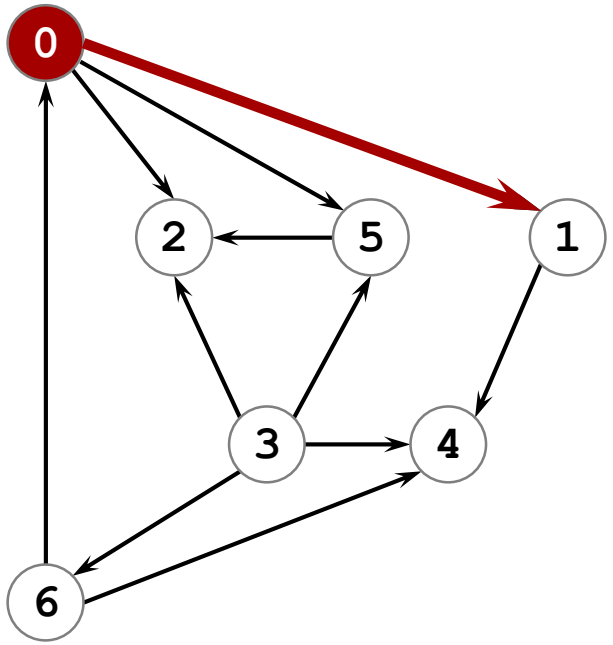
مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.



مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.



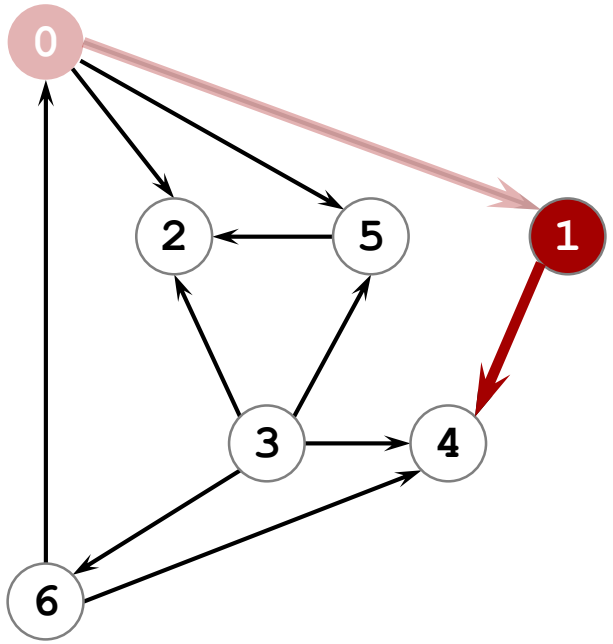

```
postorder
```

```
Visit 0: check 1, check 2, check 5
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

```
postorder
```

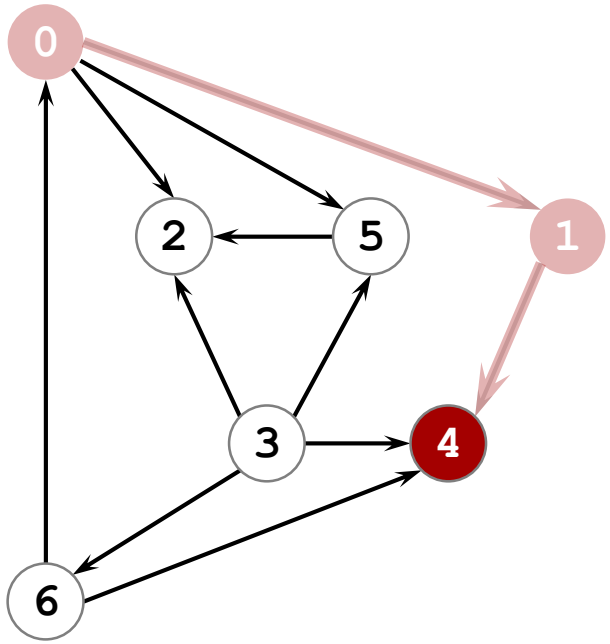



```
Visit 1: check 4
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

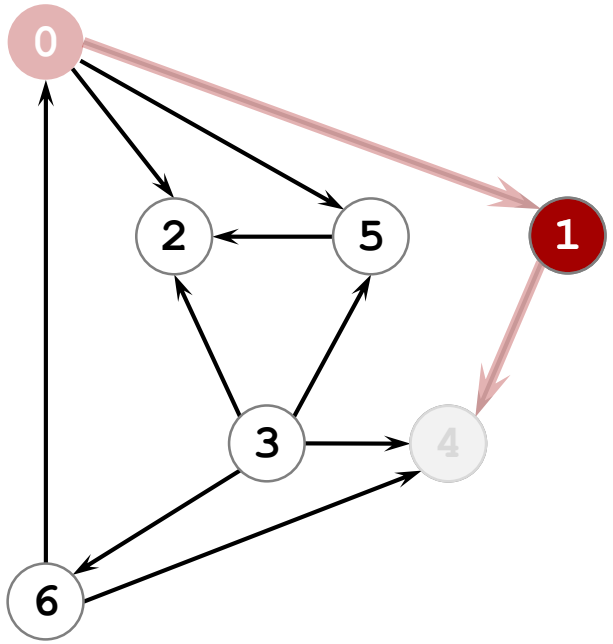
```
postorder
```




```
Visit 1: check 4
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

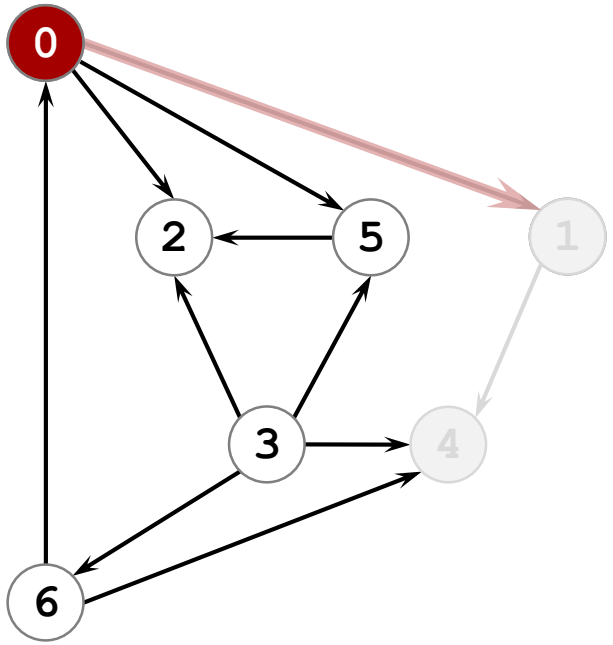
- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.




```
Postorder
4
4 done
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

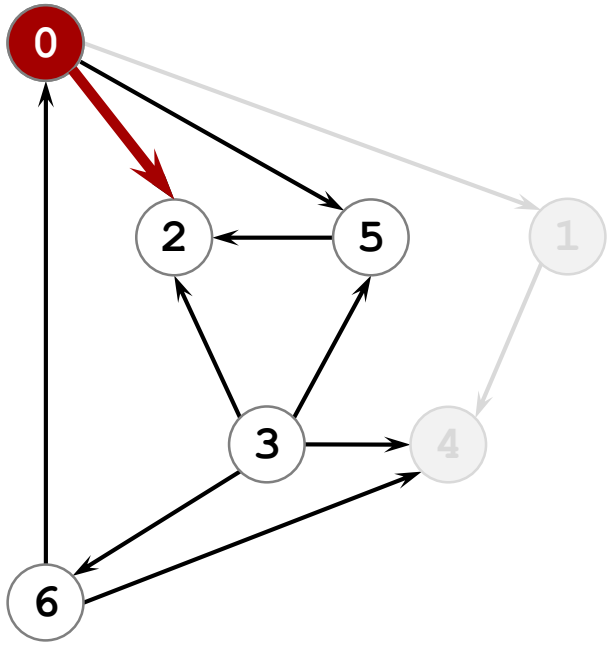
- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.




```
Postorder
4 1
1 done
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

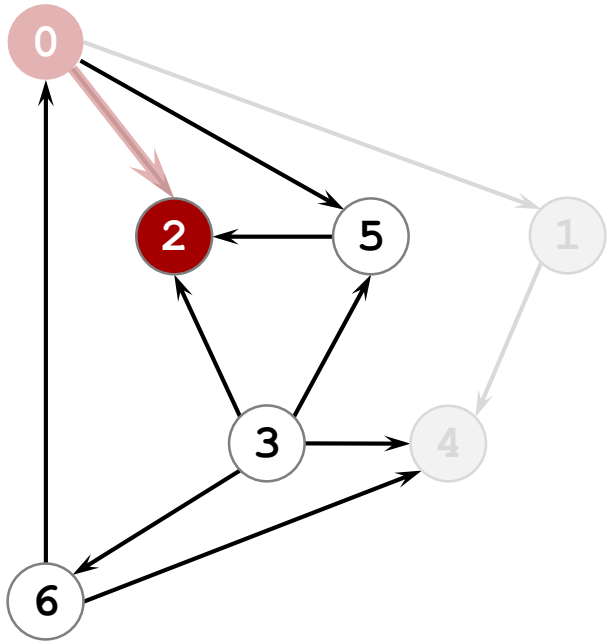



```
Visit 0: check 1, check 2, check 5
```

```
Postorder
4 1
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

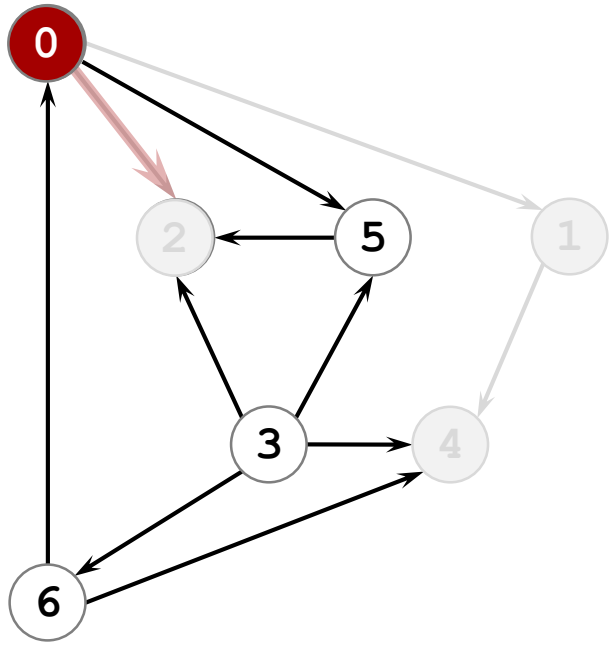



```
Visit 2
```

```
Postorder
4 1
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

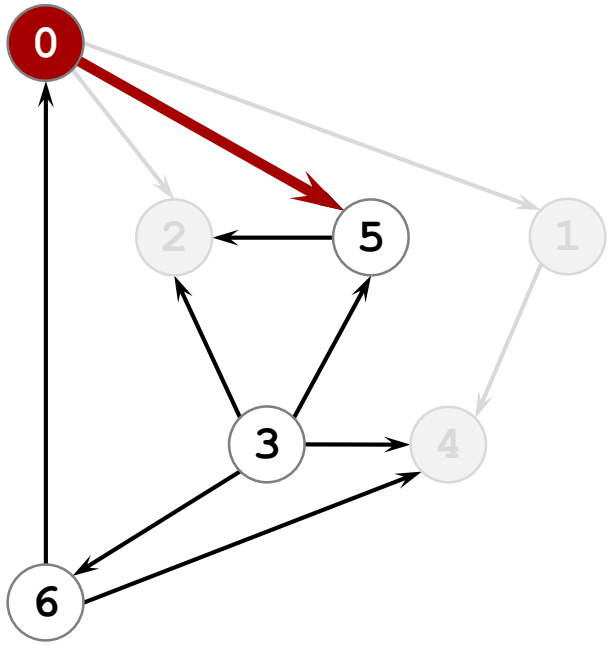



```
Postorder
4 1 2
```

```
2 done
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

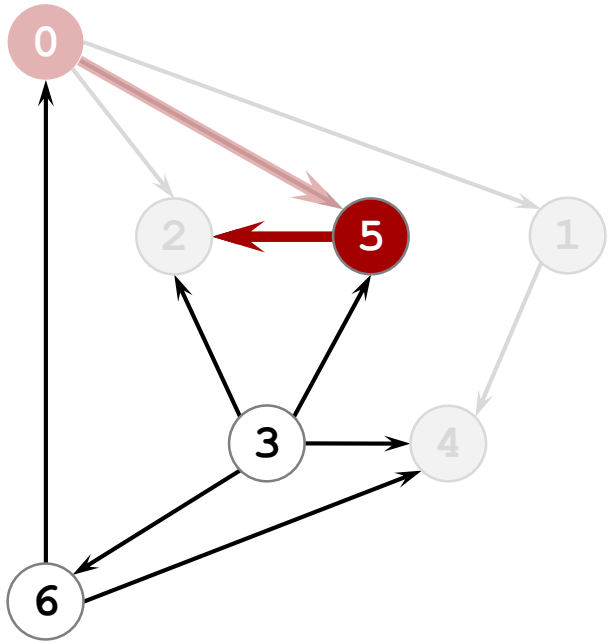



```
Postorder
4 1 2
```

```
Visit 0: check 1, check 2, check 5
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

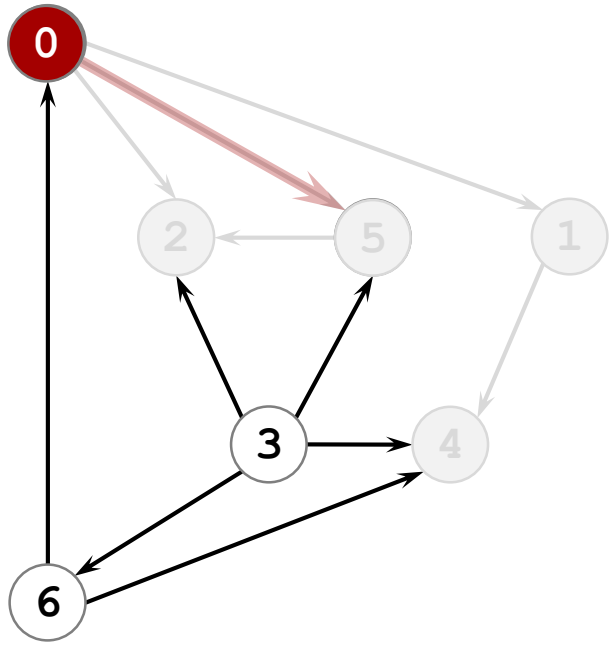



```
Visit 5: check 2
```

```
Postorder
4 1 2
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

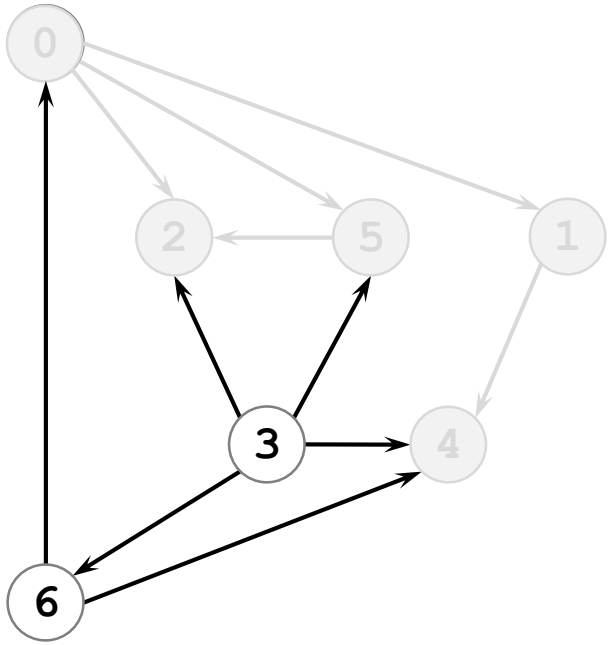



```
Postorder
4 1 2 5
```

```
5 done
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

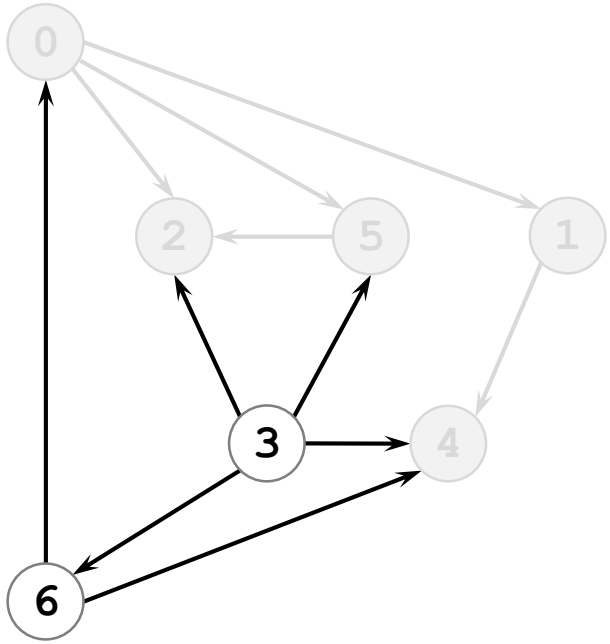



```
Postorder
4 1 2 5 0
```

```
0 done
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

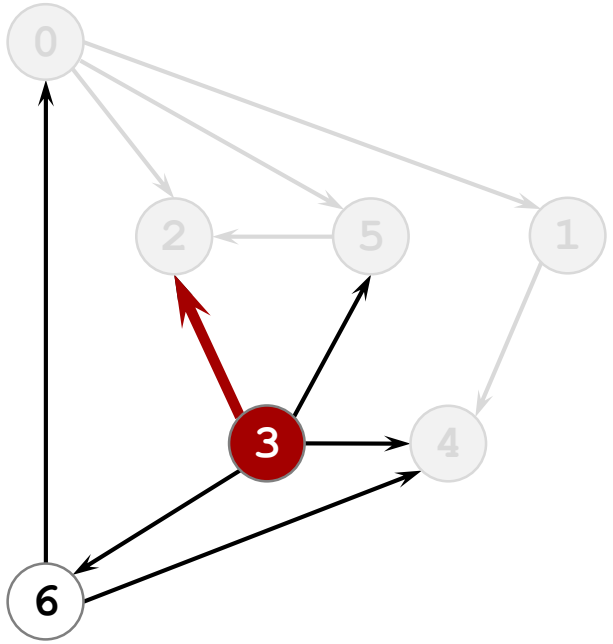



```
Postorder
4 1 2 5 0
```

```
Check 2
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

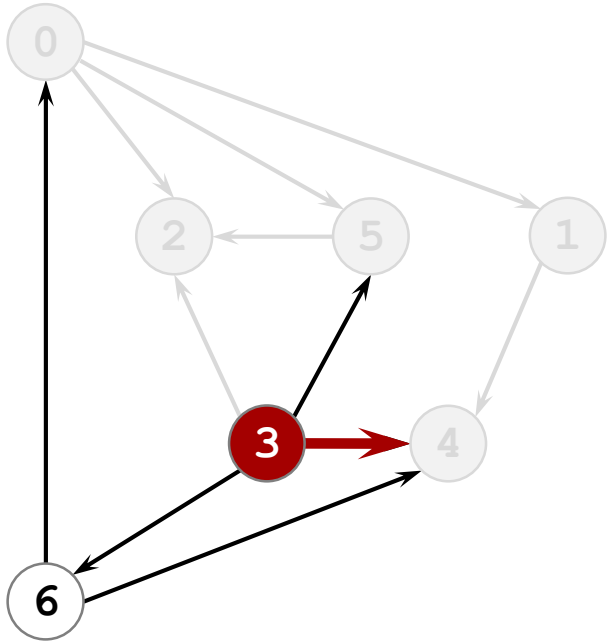



```
Postorder
4 1 2 5 0
```

```
Visit 3: check 2, check 4, check 5, check 6
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

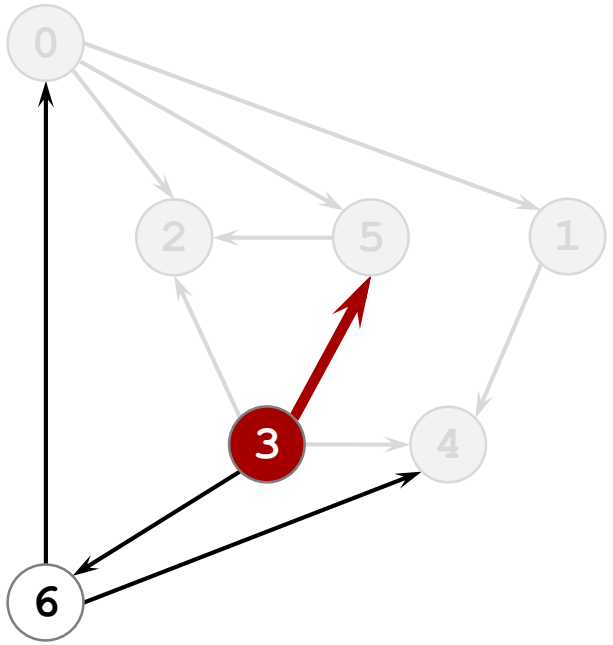


Postorder
4 1 2 5 0

Visit 3: check 2, **check 4**, check 5, check 6

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

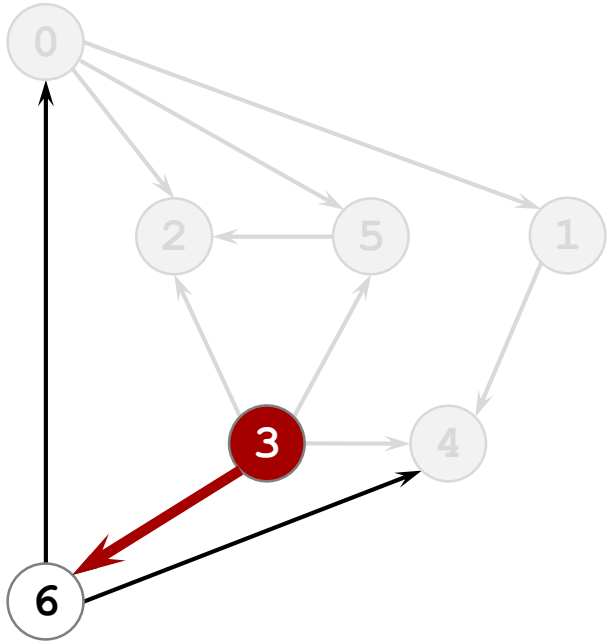


Postorder
4 1 2 5 0

Visit 3: check 2, check 4, **check 5**, check 6

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

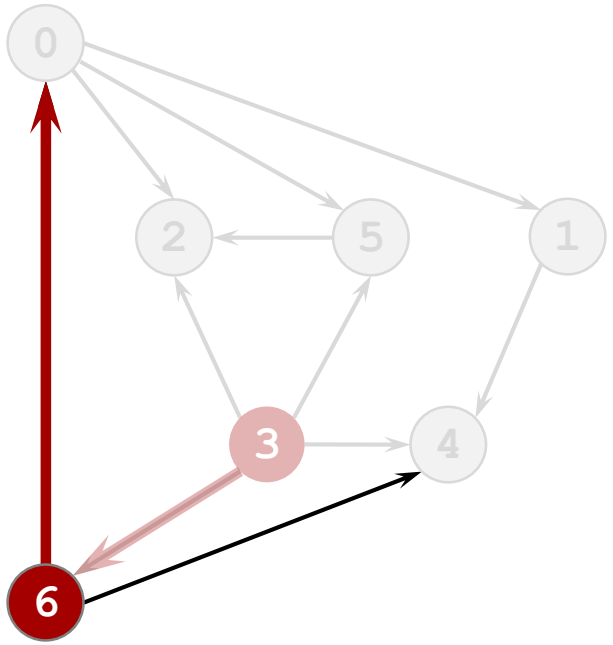


Postorder
4 1 2 5 0

Visit 3: check 2, check 4, check 5, **check 6**

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

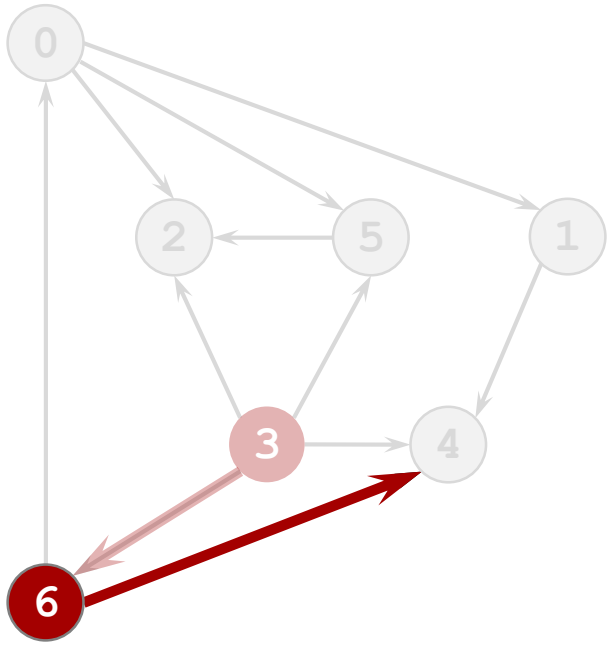


Visit 6: check 0, check 4

Postorder
4 1 2 5 0

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

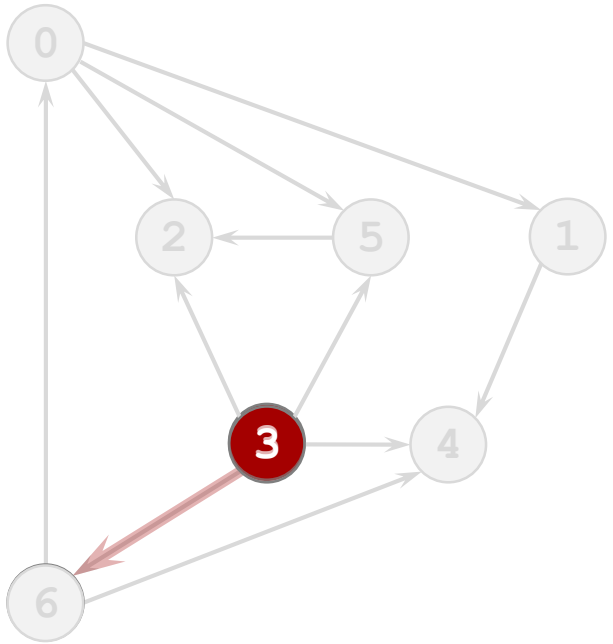


Visit 6: check 0, **check 4**

Postorder
4 1 2 5 0

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

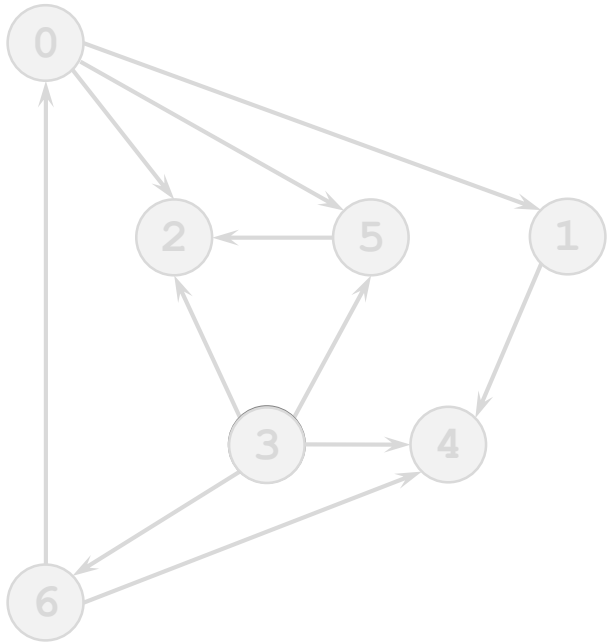



```
Postorder
4 1 2 5 0 6
```

```
6 done
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.

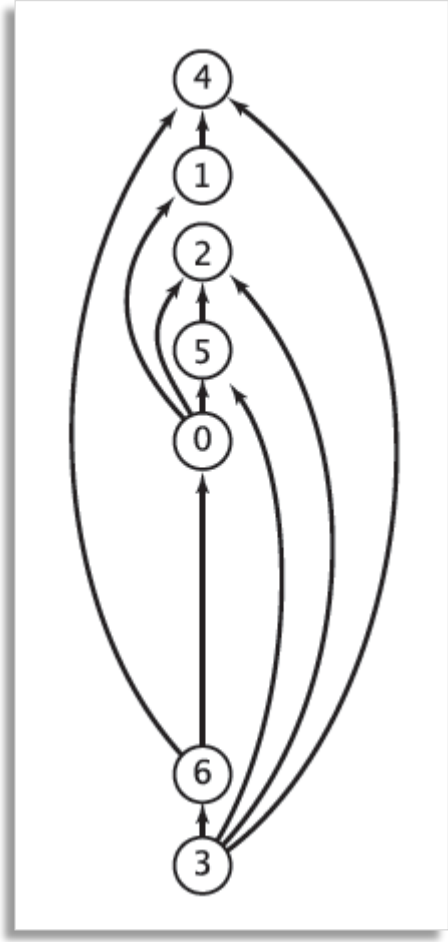


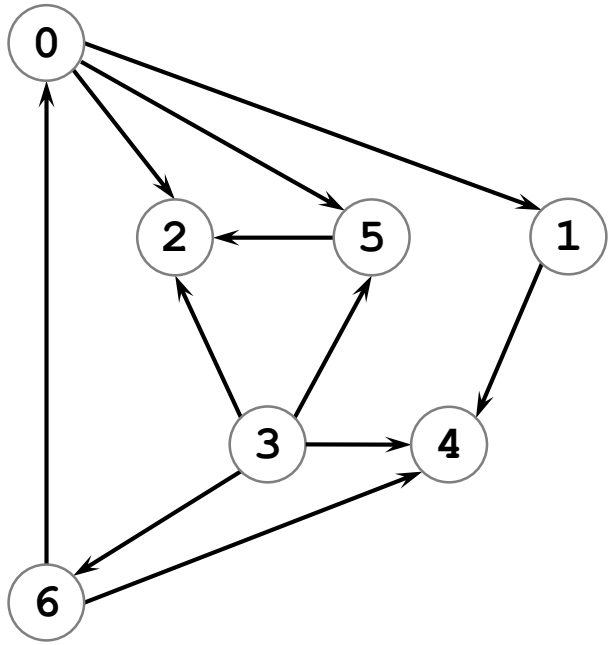

```
Postorder
4 1 2 5 0 6 3
```

```
3 done
```

مرتب‌سازی توپولوژیکی: اجرای نمایشی

- مرتب‌سازی توپولوژیکی.
  - یک جستجوی عمقی انجام بده.
  - رئوس را به ترتیب معکوس پس‌ترتیب برگردان.






```
Postorder
4 1 2 5 0 6 3
```

مرتب‌سازی توپولوژیکی: پیاده‌سازی


```
public class DepthFirstOrder
{
   private boolean[] marked;
   private Stack<Integer> reversePost;

   public DepthFirstOrder(Digraph G)
   {
      reversePost = new Stack<Integer>();
      marked = new boolean[G.V()];
      for (int v = 0; v < G.V(); v++)
         if (!marked[v]) dfs(G, v);
   }

   private void dfs(Digraph G, int v) {
      marked[v] = true;
      for (int w : G.adj(v))
         if (!marked[w]) dfs(G, w);
      reversePost.push(v);
   }

   public Iterable<Integer> reversePost()
   {  return reversePost;  }
}
```

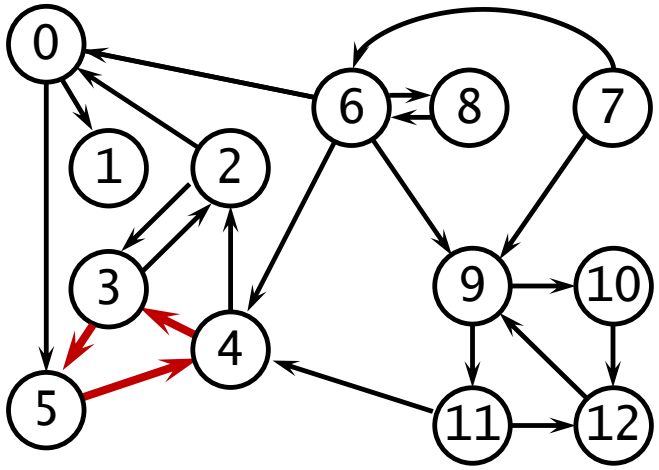
برگرداندن رئوس به ترتیب عکس پس‌ترتیب در جستجوی عمقی

تشخیص دور جهت‌دار

گزاره. یک گراف جهت‌دار دارای ترتیب توپولوژیکی است اگر و فقط اگر شامل دور جهت‌دار نباشد.

اثبات.

- اگر دور جهت‌دار وجود داشته باشد، ترتیب توپولوژیکی غیر ممکن است.
- اگر دور جهت‌دار وجود نداشته باشد، الگوریتم مبتنی بر DFS یک ترتیب توپولوژیکی پیدا می‌کند.



- هدف. تشخیص دور در گراف جهت‌دار.
- راه حل. استفاده از DFS

کاربرد تشخیص دور: وراثت چرخه‌ای

- کامپایلر جاوا از الگوریتم تشخیص دور استفاده می‌کند.

```
public class A extends B
{
    ...
}
```

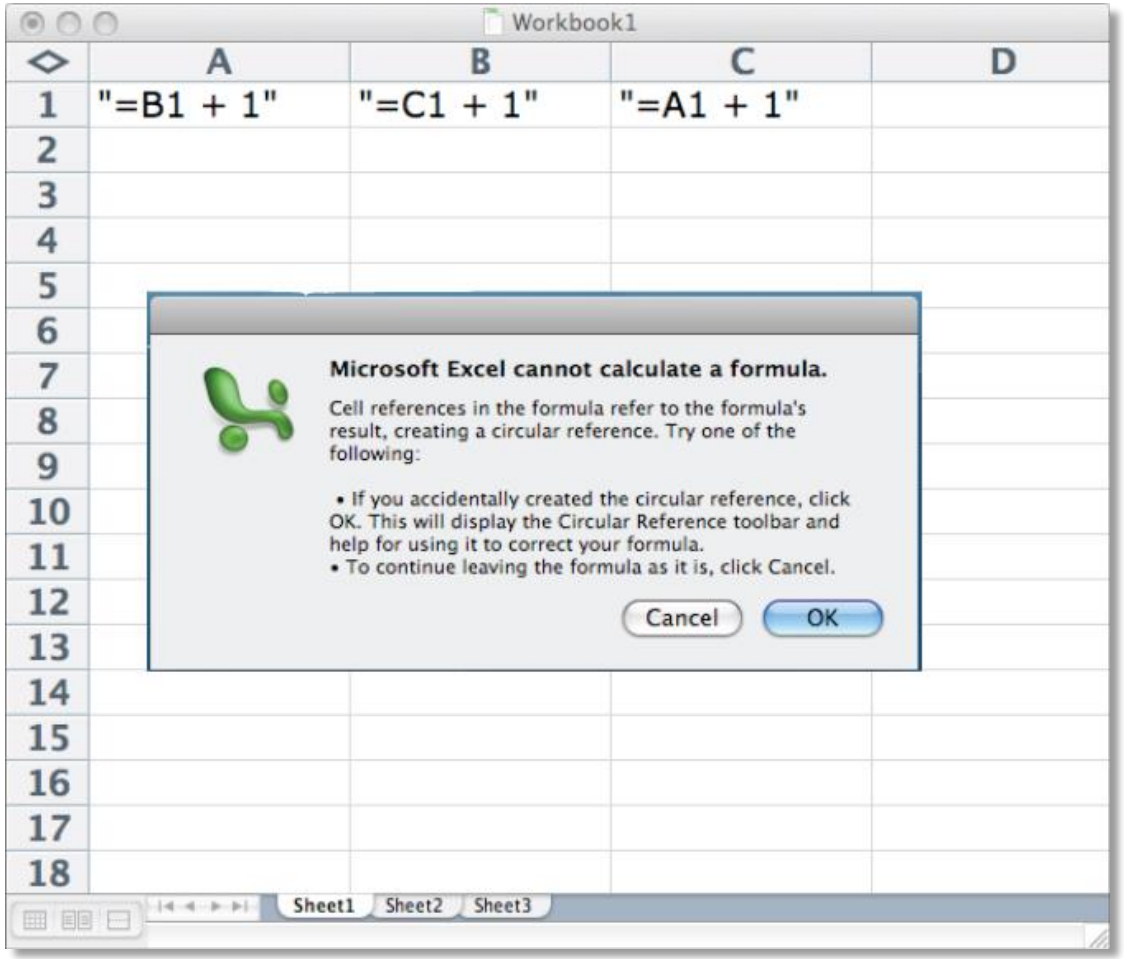
```
public class B extends C
{
    ...
}
```

```
public class C extends A
{
    ...
}
```

```
% javac A.java

A.java:1: cyclic inheritance involving A
public class A extends B { }
       ^
1 error
```

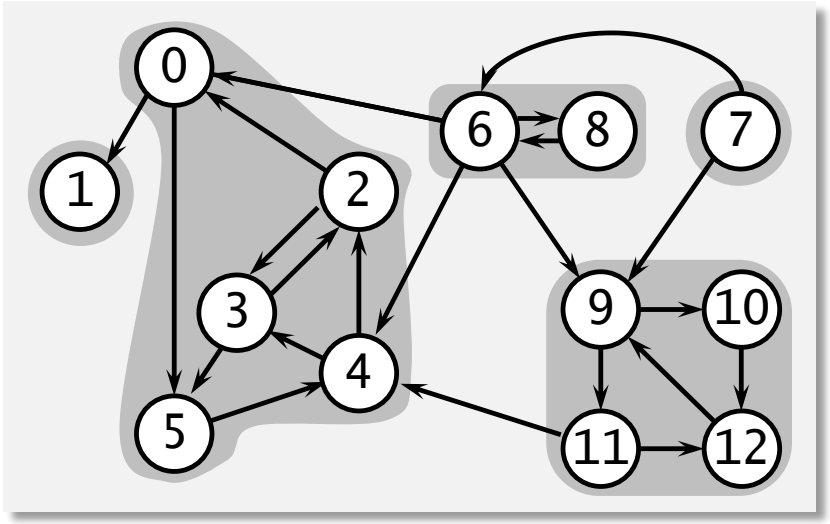
کاربرد تشخیص دور: محاسبات در اکسل

- برنامه‌ی اکسل میکروسافت از الگوریتم تشخیص دور استفاده می‌کند.



مؤلفه‌های همبند قوی

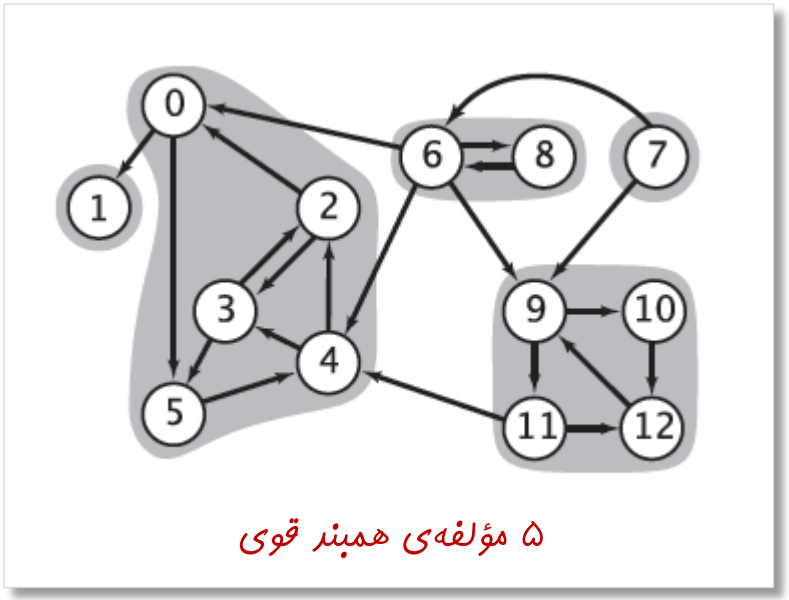
مؤلفه‌های همبند قوی



- تعریف. رئوس **v** و **w** همبند قوی هستند اگر یک مسیر جهت‌دار از **v** به **w** و یک مسیر جهت‌دار از **w** به **v** وجود داشته باشد.
- ویژگی کلیدی. همبندی قوی یک رابطه‌ی هم ارزی است.
- تعریف. یک مؤلفه‌ی همبند قوی یک زیرمجموعه‌ی بیشینه از رئوس همبند قوی است.

مؤلفه‌های همبند و مؤلفه‌های همبند قوی

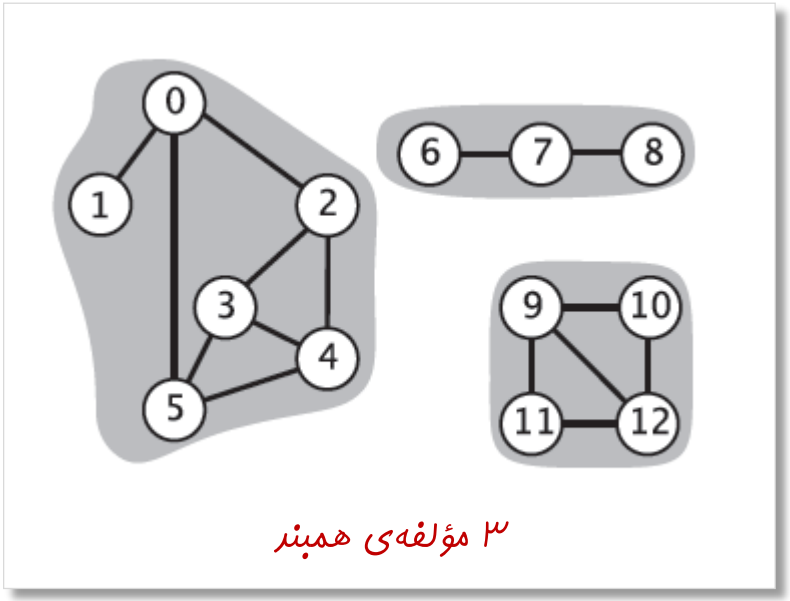
رئوس **V** و **W** همبند قوی هستند، اگر یک مسیر جهت‌دار از **V** به **W** و یک مسیر جهت‌دار از **W** به **V** وجود داشته باشد



۵ مؤلفه‌ی همبند قوی

	0	1	2	3	4	5	6	7	8	9	10	11	12
scc	1	0	1	1	1	1	3	4	3	2	2	2	2

رئوس **V** و **W** همبند هستند، اگر یک مسیر بین **V** و **W** وجود داشته باشد



۳ مؤلفه‌ی همبند

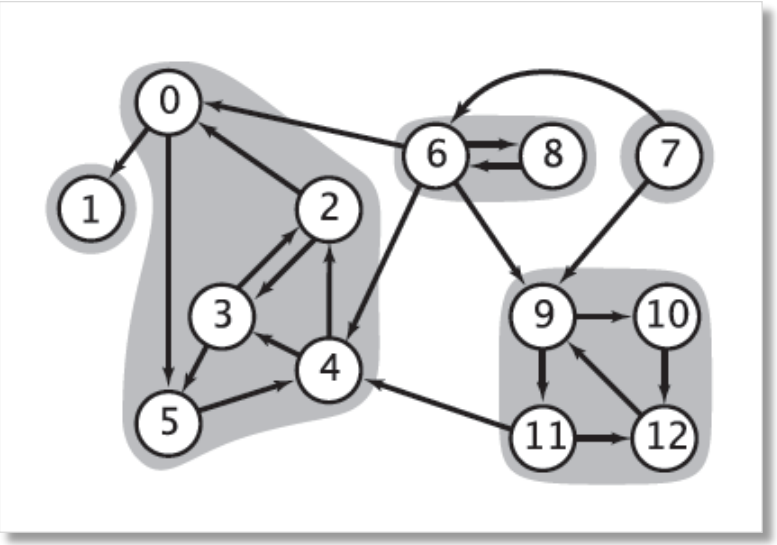
	0	1	2	3	4	5	6	7	8	9	10	11	12
cc	0	0	0	0	0	0	1	1	1	2	2	2	2

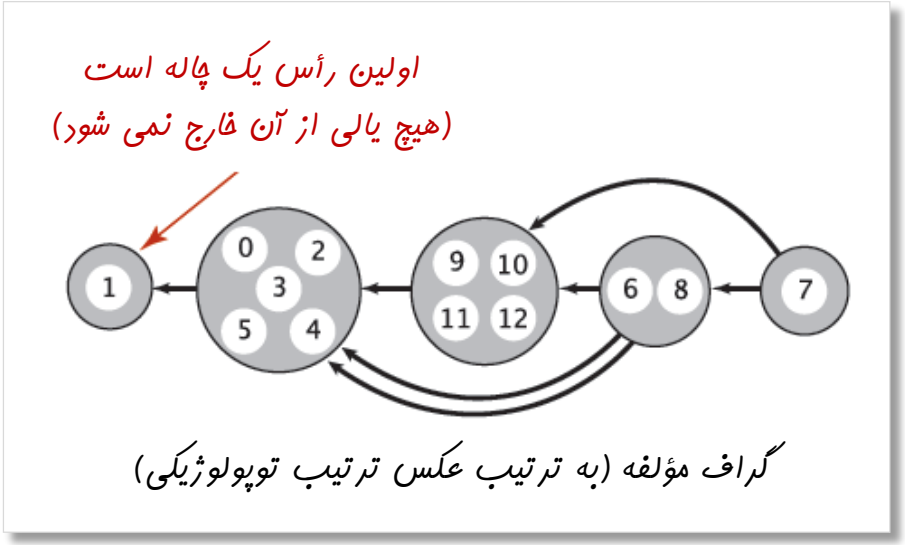
الگوریتم کاساراجو

- گراف معکوس. مؤلفه‌های همبند قوی در G و G^R یکسان هستند.
- گراف مؤلفه. قرار دادن مؤلفه‌های همبند قوی در یک رأس.

ایده.

- محاسبه‌ی ترتیب توپولوژیکی در گراف مؤلفه.
- اجرای DFS با در نظر گرفتن رئوس به ترتیب عکس ترتیب توپولوژیکی.

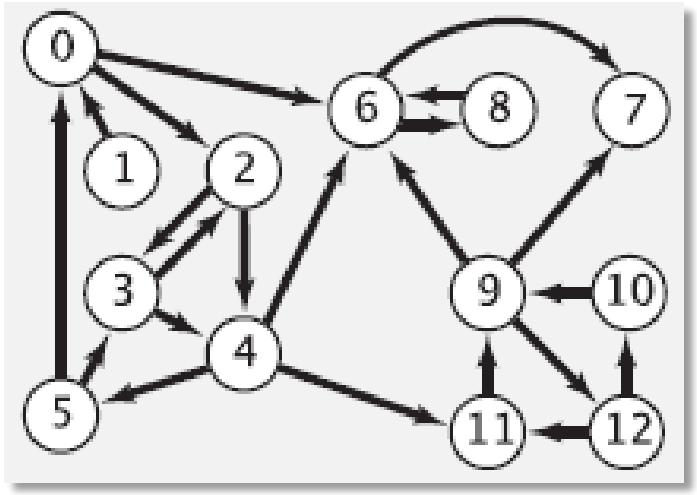




الگوریتم کاساراجو

یک الگوریتم ساده برای محاسبه مؤلفه‌های همبند قوی.

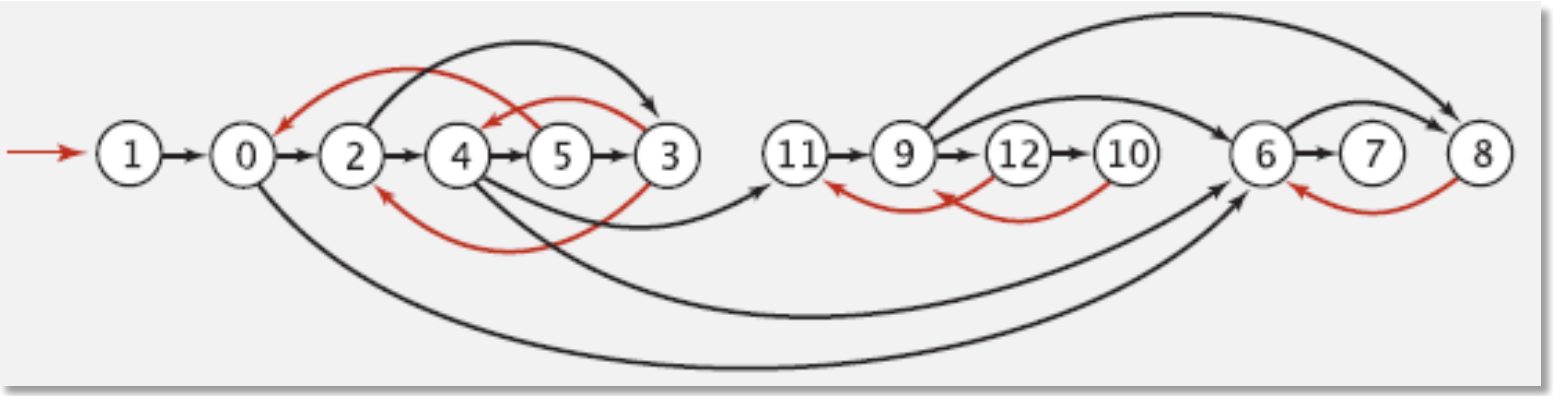
- اجرای DFS بر روی G^R برای محاسبه عکس پس‌ترتیب



عکس پس‌ترتیب برای استفاده در دومین اجرای DFS:

1 0 2 4 5 3 11 9 12 10 6 7 8

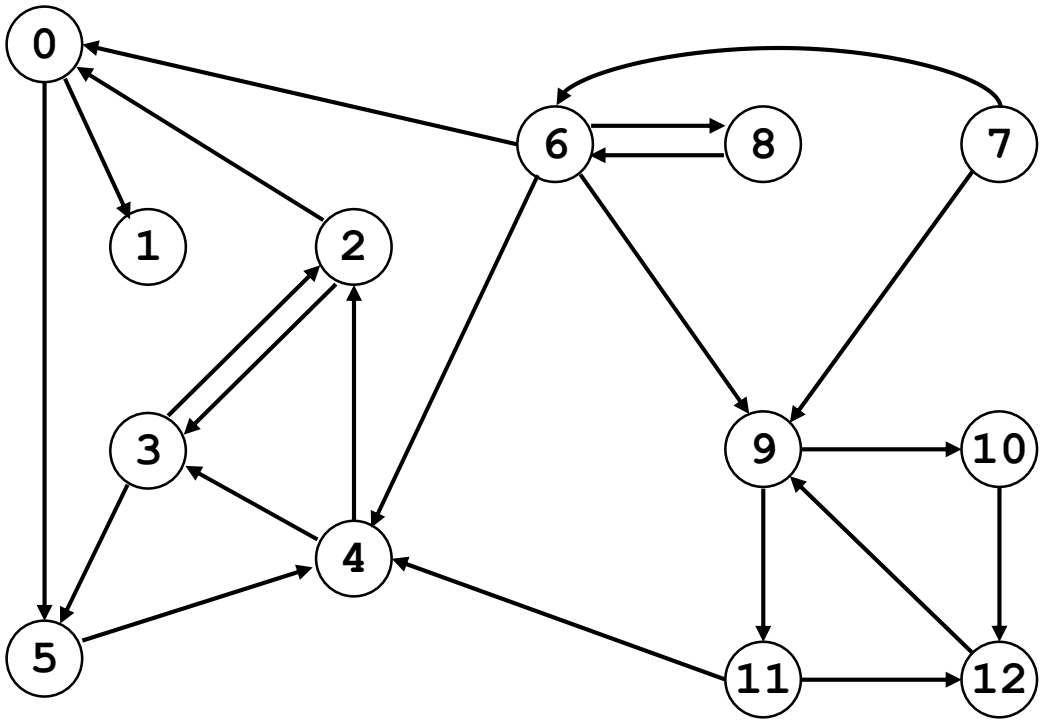
- اجرای DFS بر روی G با در نظر گرفتن رئوس به ترتیب تولید شده در اولین DFS



الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

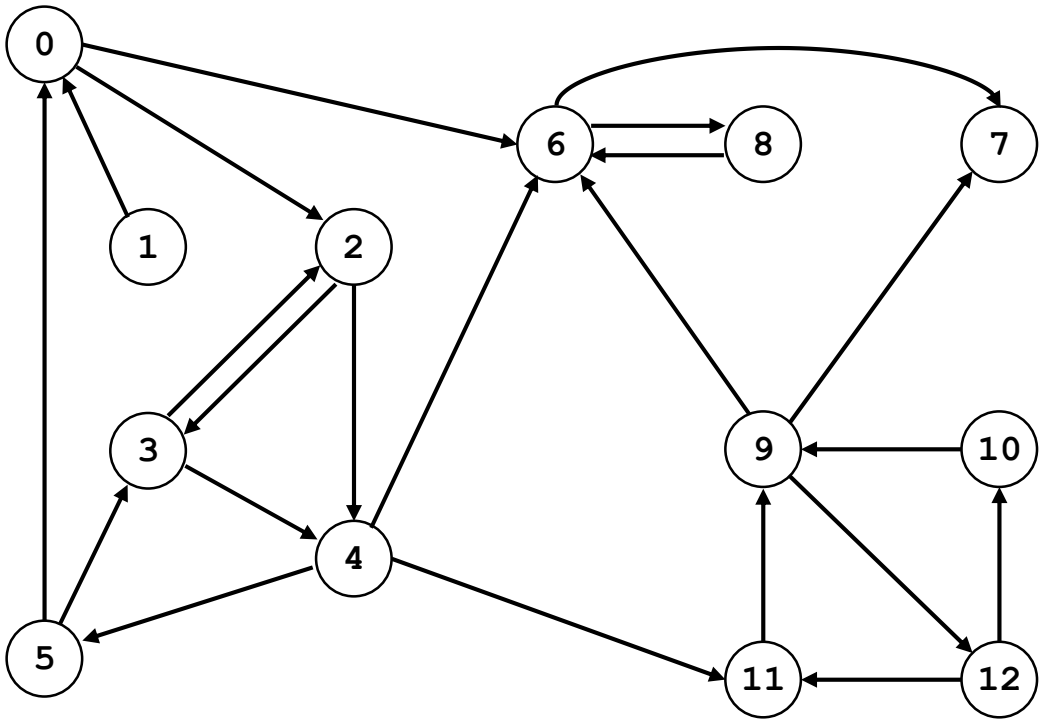
گراف جهت‌دار



الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

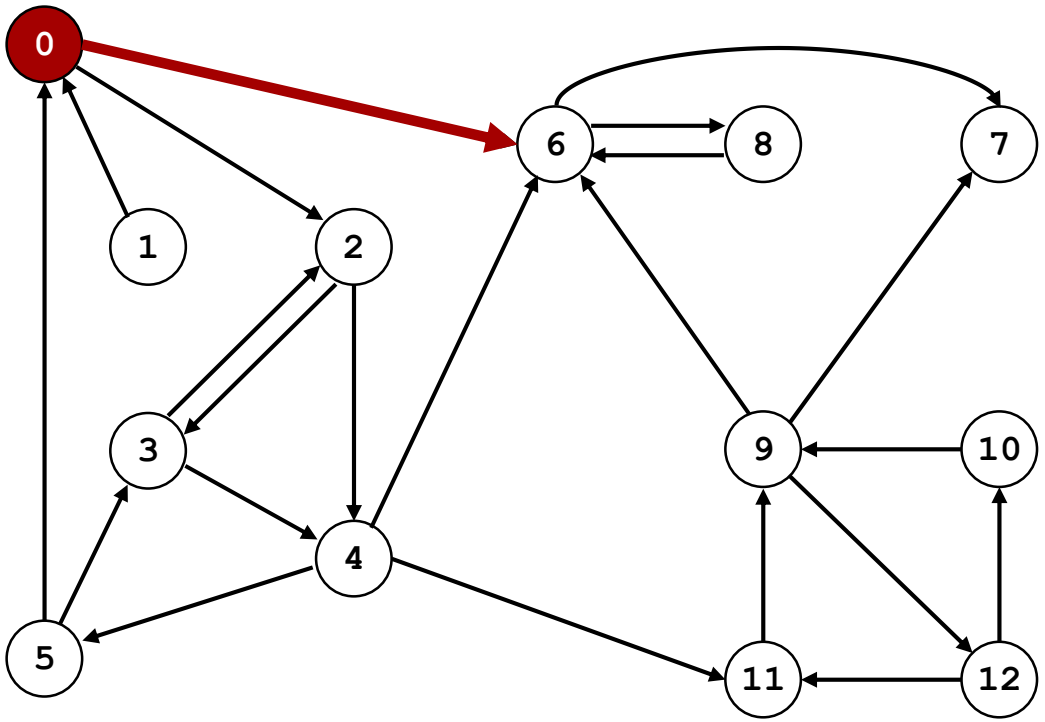
معکوس گراف جهت‌دار



v	marked[]
0	F
1	F
2	F
3	F
4	F
5	F
6	F
7	F
8	F
9	F
10	F
11	F
12	F

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

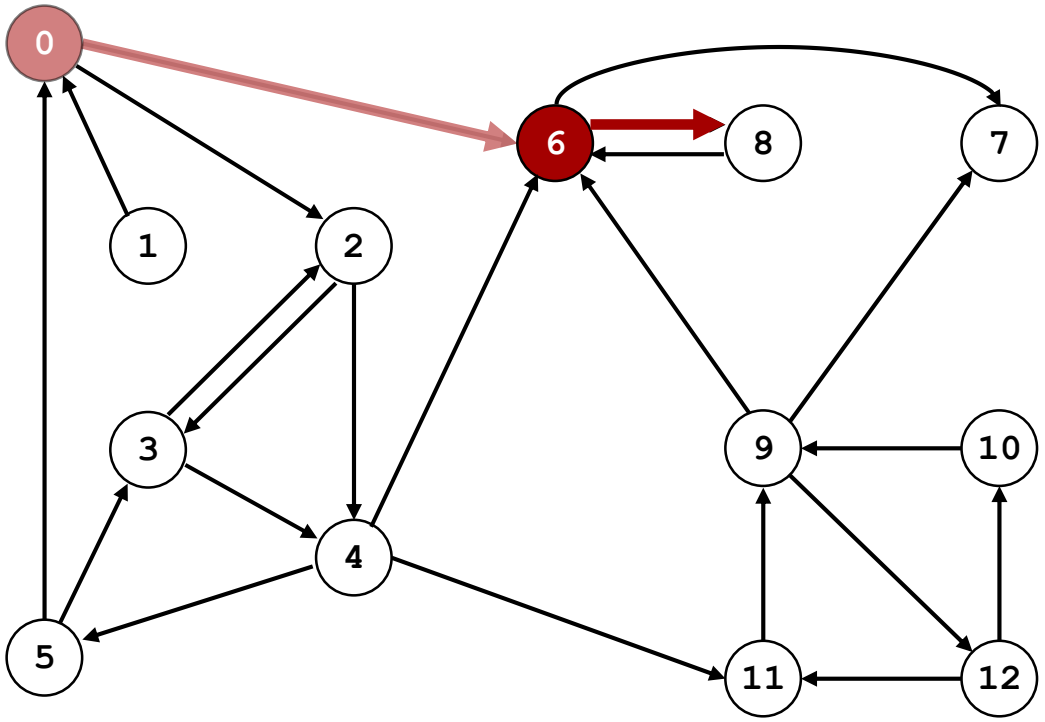


v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	F
7	F
8	F
9	F
10	F
11	F
12	F

```
Visit 0: check 6, check 2
```

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

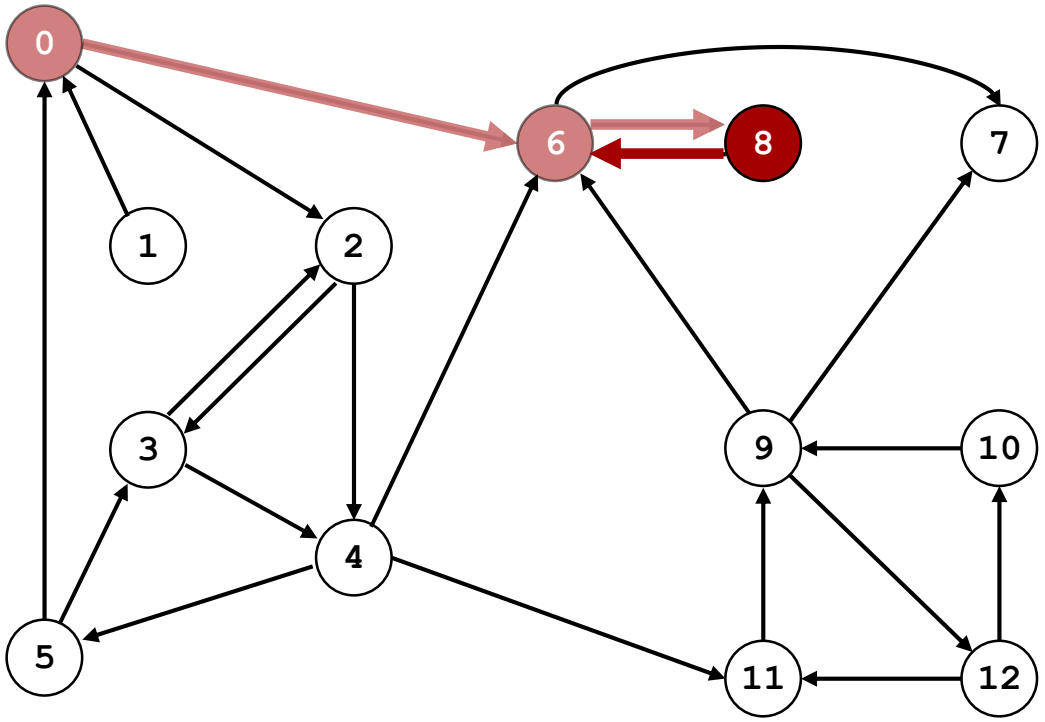


v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	F
8	F
9	F
10	F
11	F
12	F

Visit 6: check 8, check 7

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R



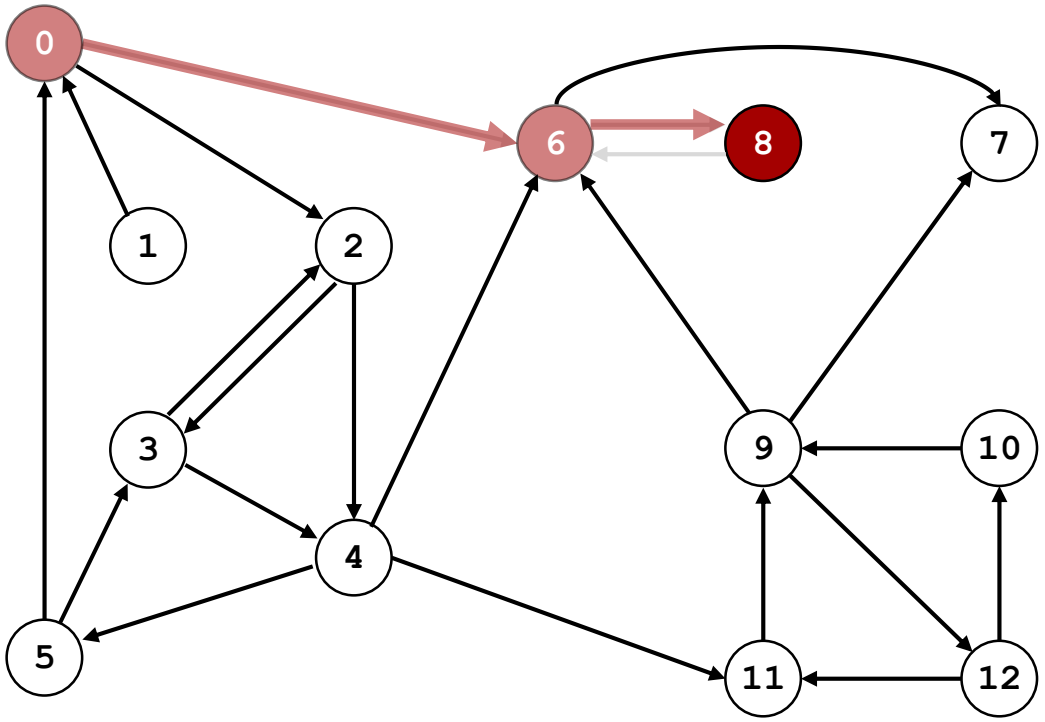
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	F
8	T
9	F
10	F
11	F
12	F

Visit 8: check 6

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

8



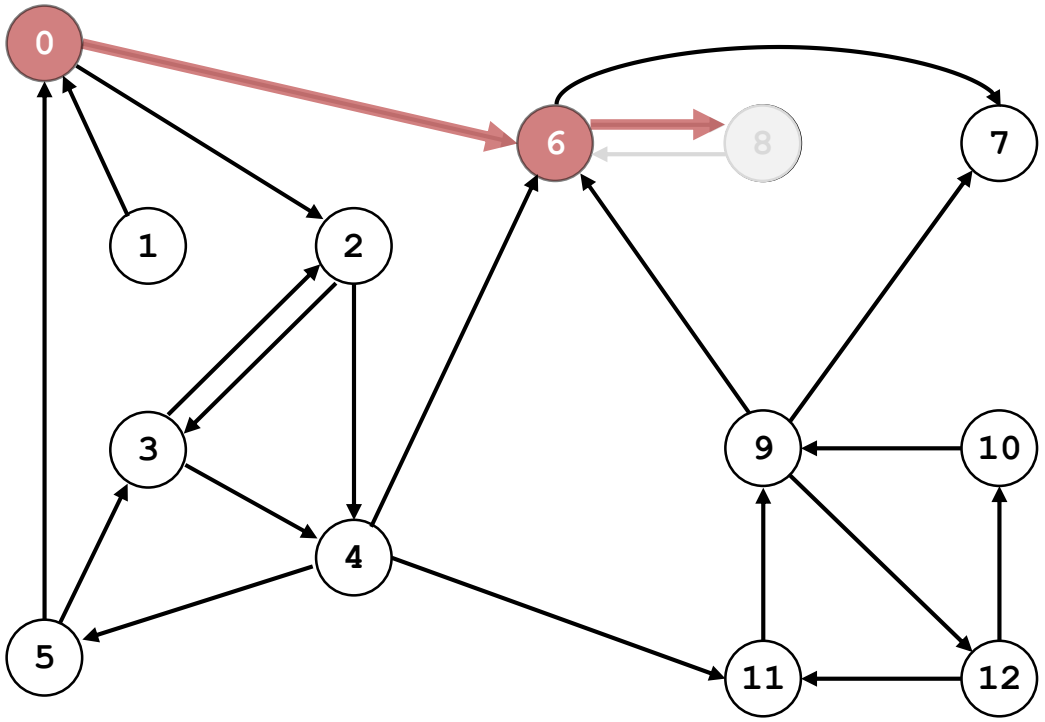
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	F
8	T
9	F
10	F
11	F
12	F

8 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

8



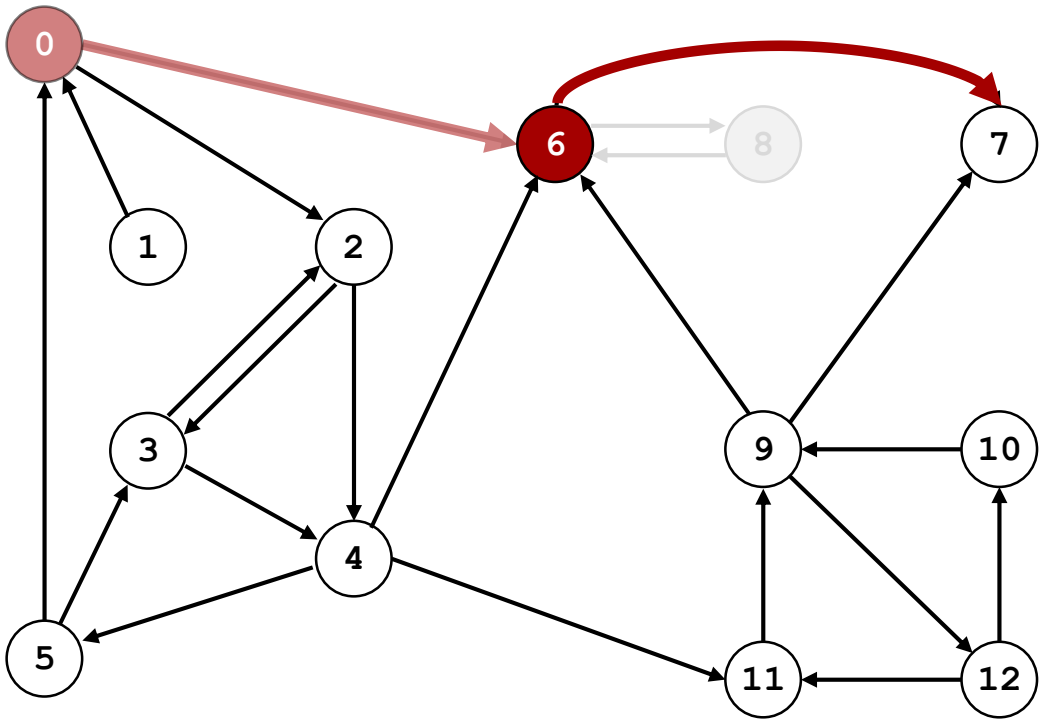
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	F
8	T
9	F
10	F
11	F
12	F

8 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

8



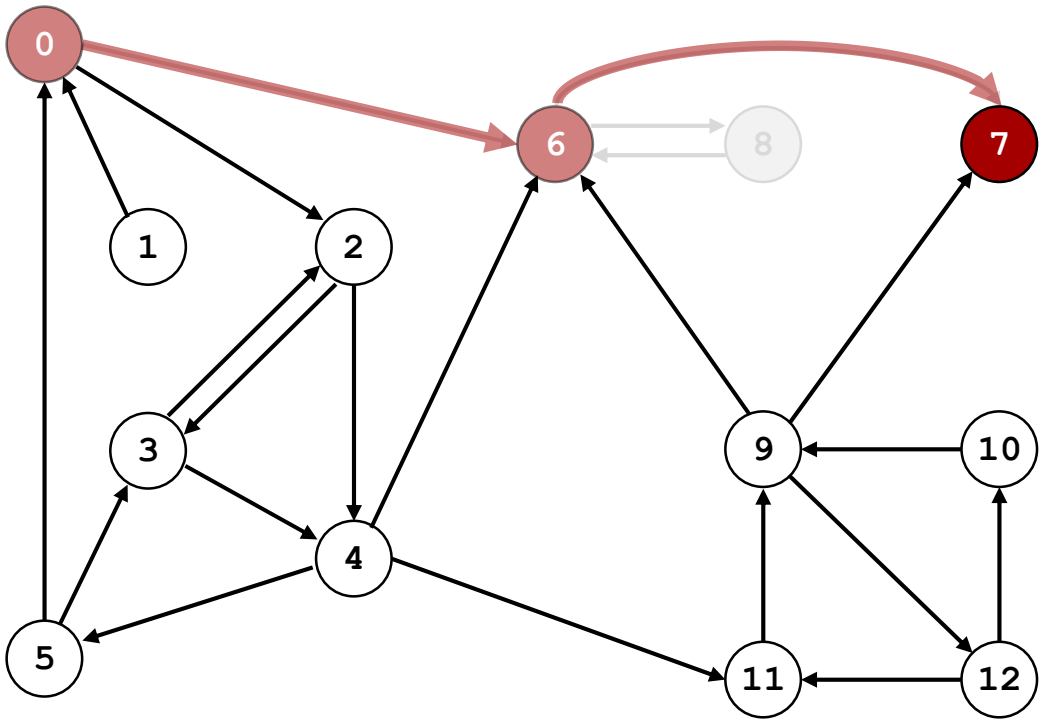
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	F
8	T
9	F
10	F
11	F
12	F

Visit 6: check 8, **check 7**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

7 8



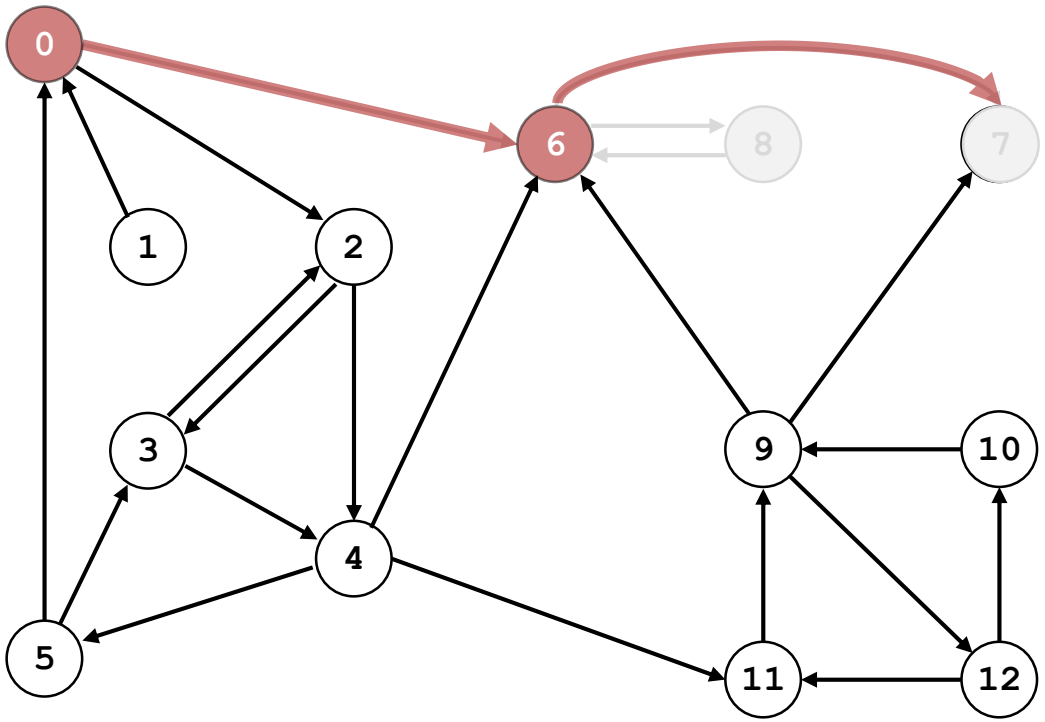
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	T
8	T
9	F
10	F
11	F
12	F

Visit 7

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

7 8



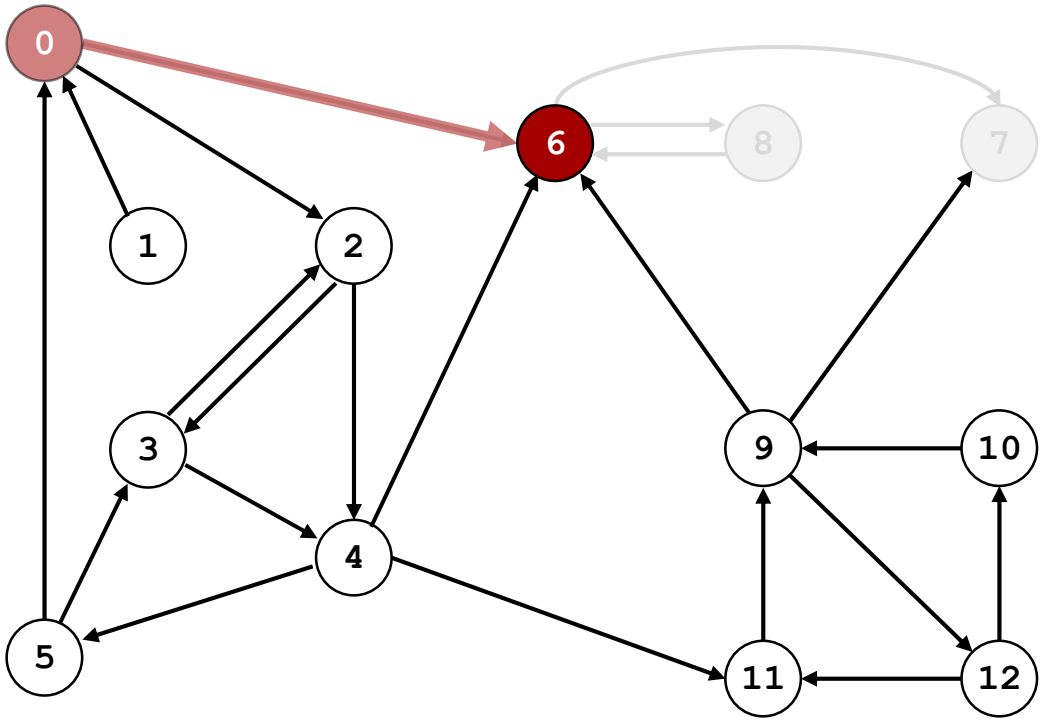
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	T
8	T
9	F
10	F
11	F
12	F

7 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

```
7  8
```



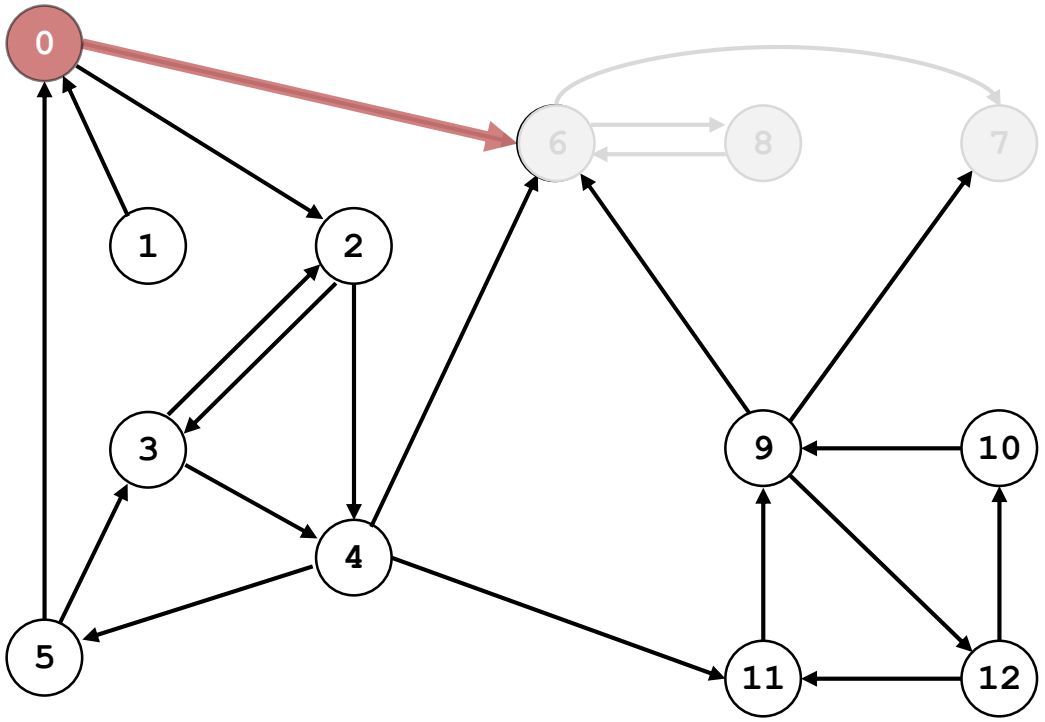
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	T
8	T
9	F
10	F
11	F
12	F

```
7 done
```

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



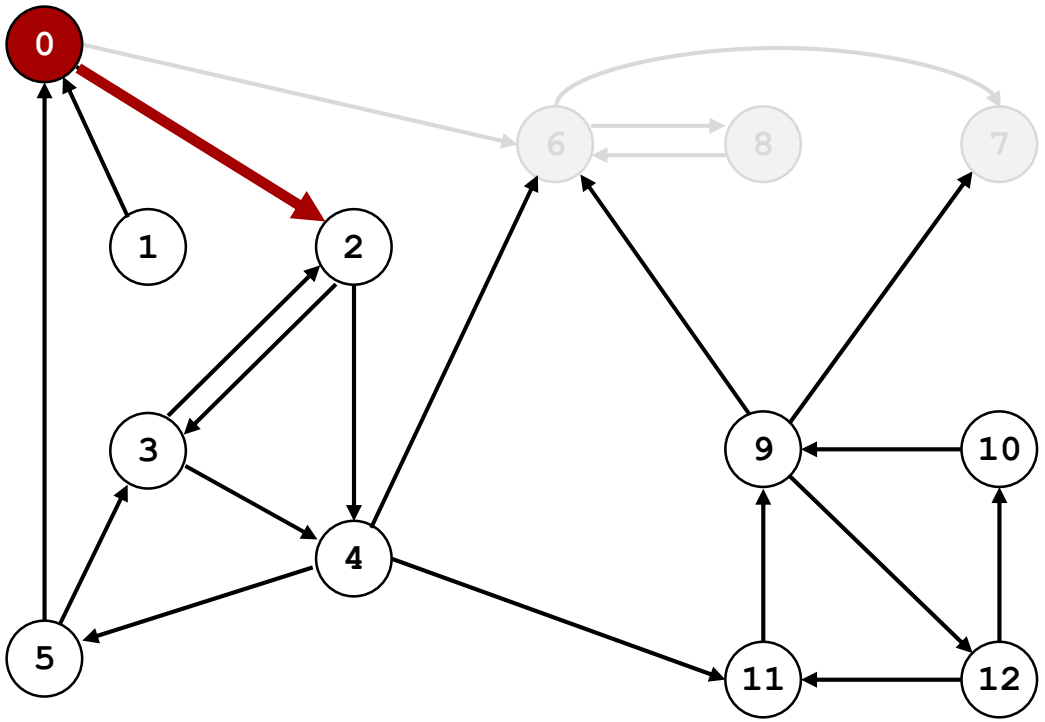
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	T
8	T
9	F
10	F
11	F
12	F

6 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



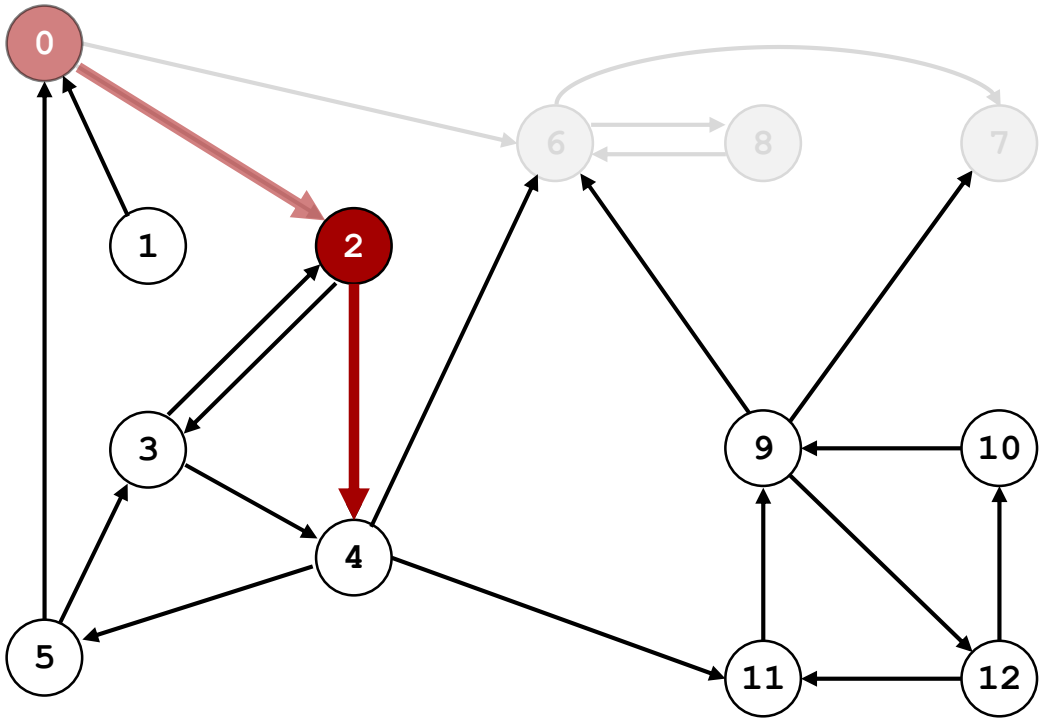
v	marked[]
0	T
1	F
2	F
3	F
4	F
5	F
6	T
7	T
8	T
9	F
10	F
11	F
12	F

Visit 0: check 6, check 2

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



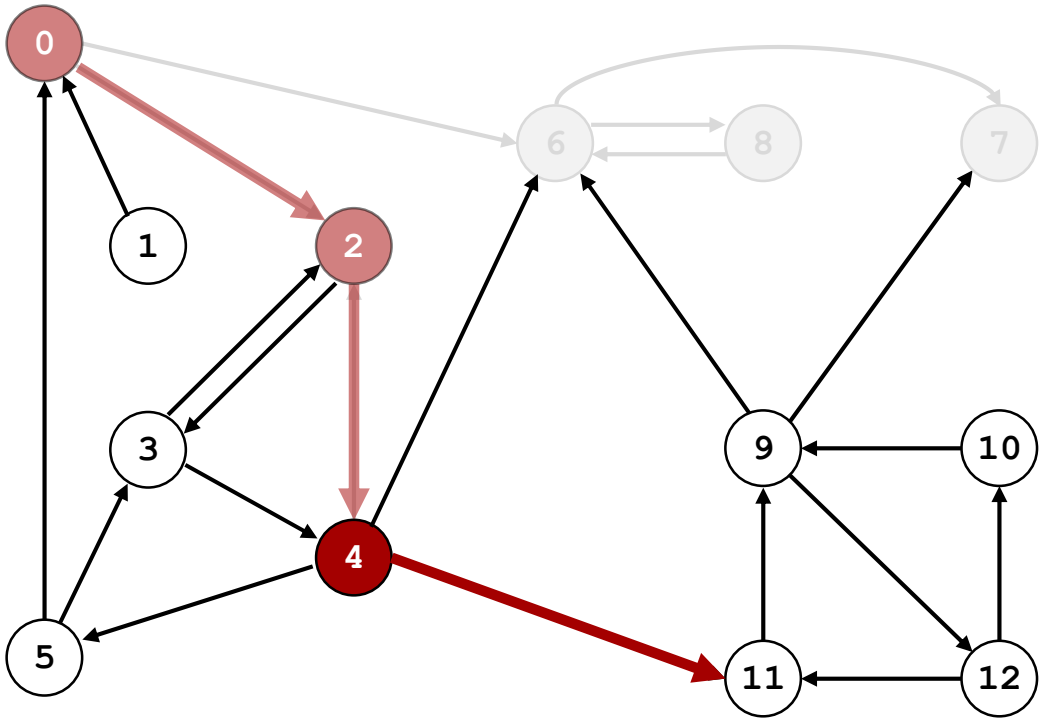
v	marked[]
0	T
1	F
2	T
3	F
4	F
5	F
6	T
7	T
8	T
9	F
10	F
11	F
12	F

Visit 2: check 4, check 3

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



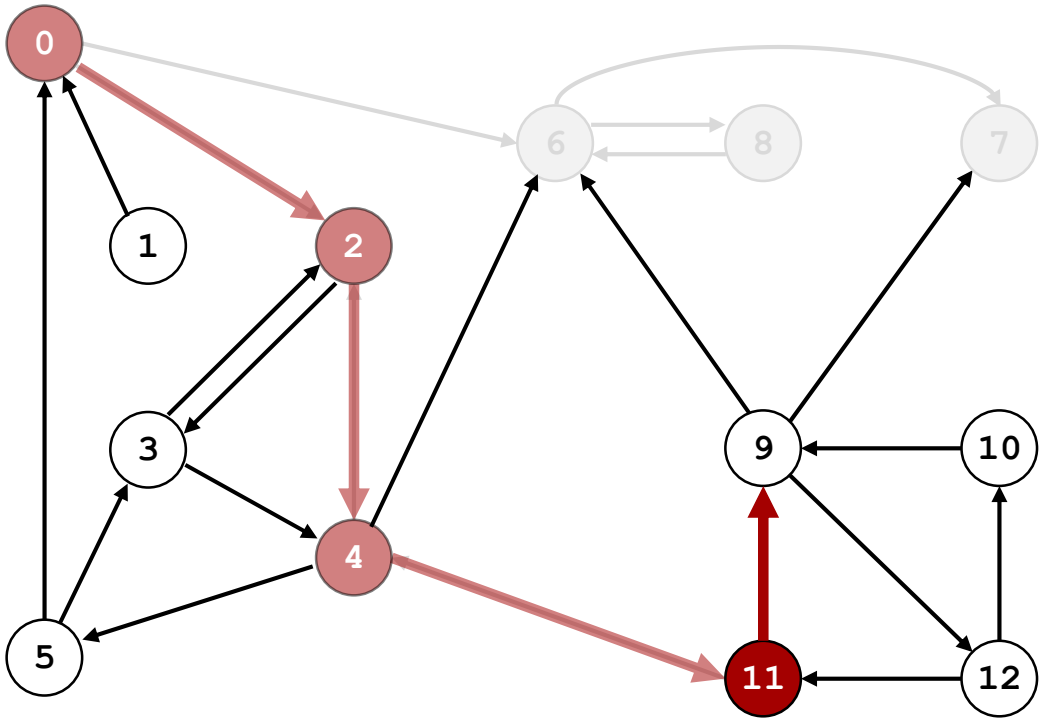
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	F
10	F
11	F
12	F

Visit 4: check 11, check 6, check 5

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



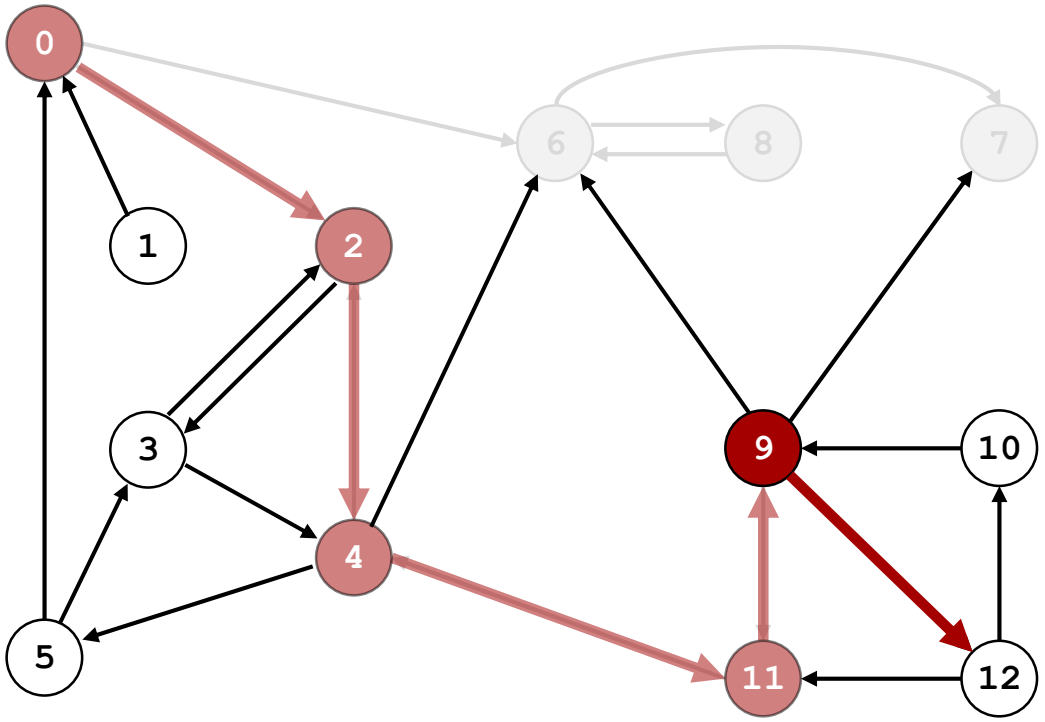
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	F
10	F
11	T
12	F

Visit 11: check 9

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



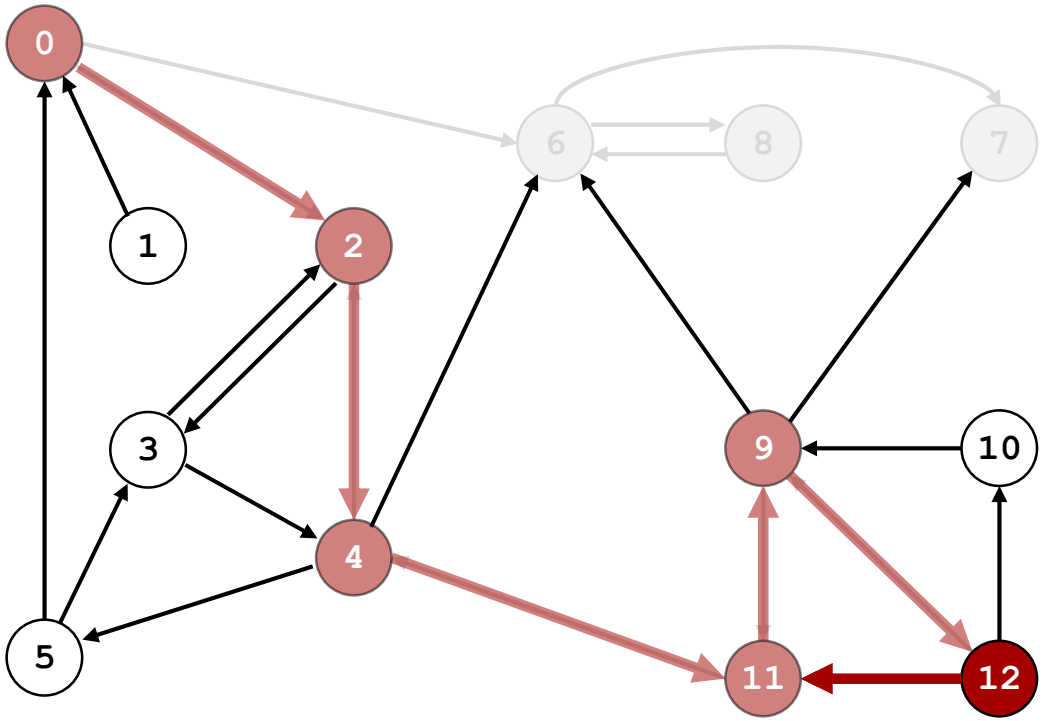
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	F
11	T
12	F

Visit 9: check 12, check 7, check 6

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



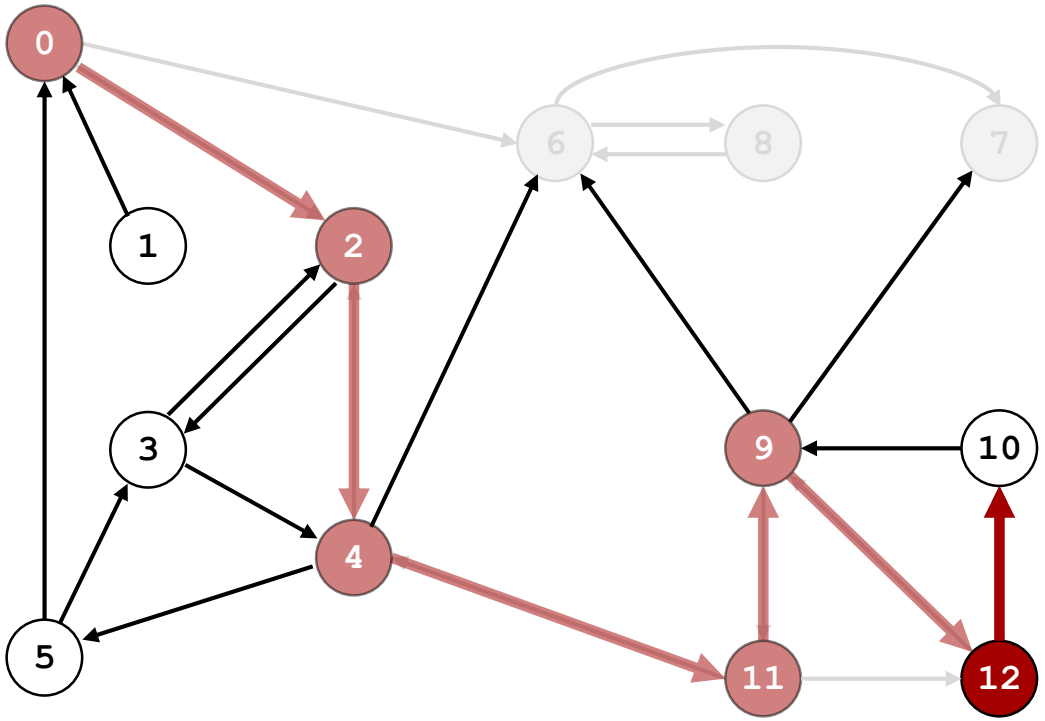
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	F
11	T
12	T

Visit 12: check 11, check 10

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



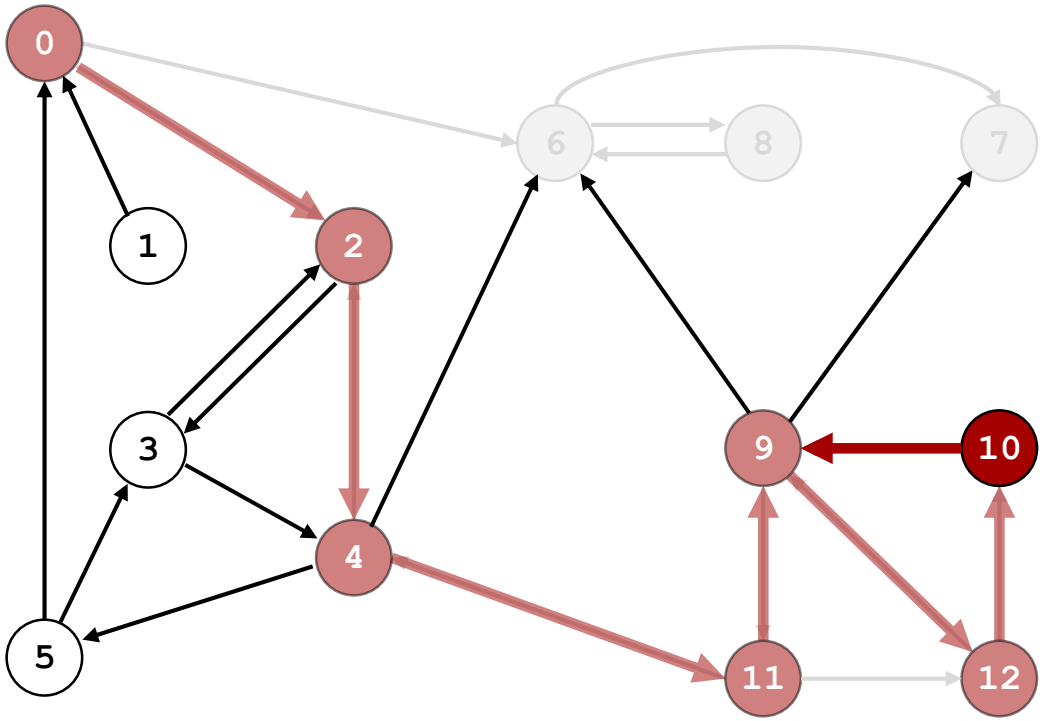
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	F
11	T
12	T

Visit 12: check 11, check 10

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

6 7 8



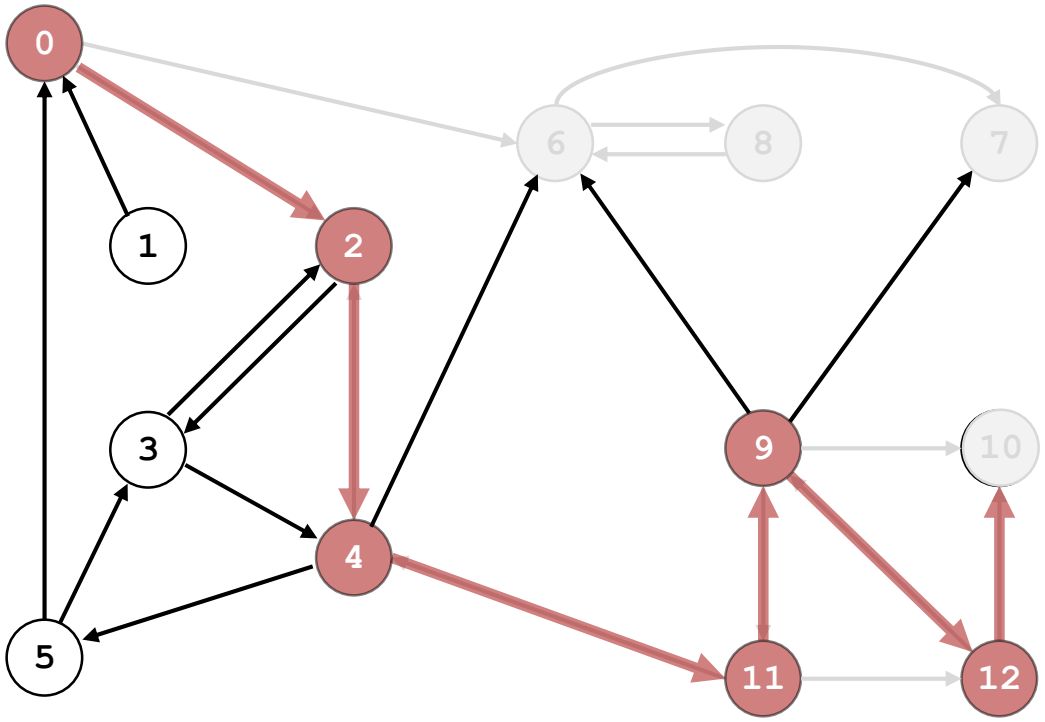
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 10: check 9

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

10 6 7 8



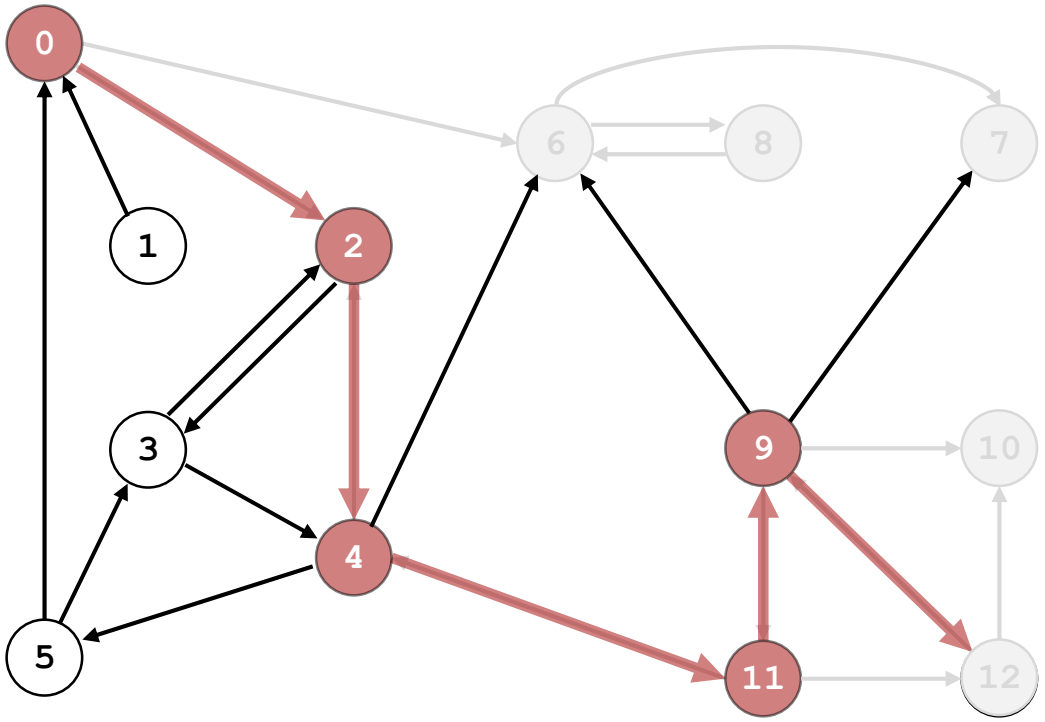
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

10 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

12 10 6 7 8



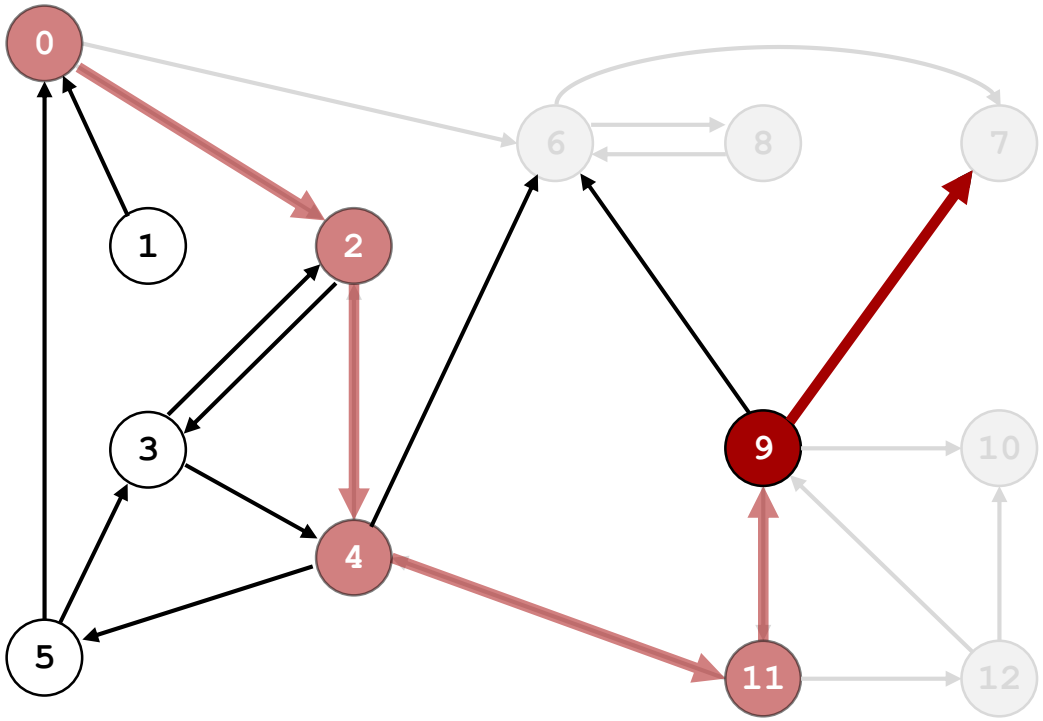
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

12 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

12 10 6 7 8



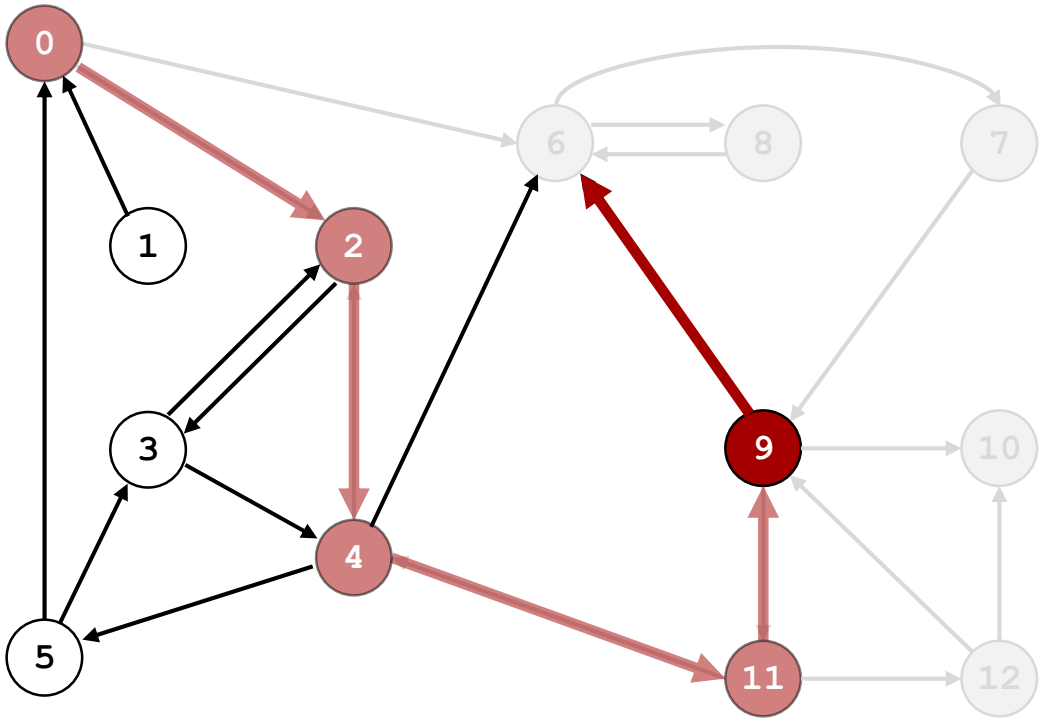
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 9: check 12, **check 7**, check 6

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

12 10 6 7 8



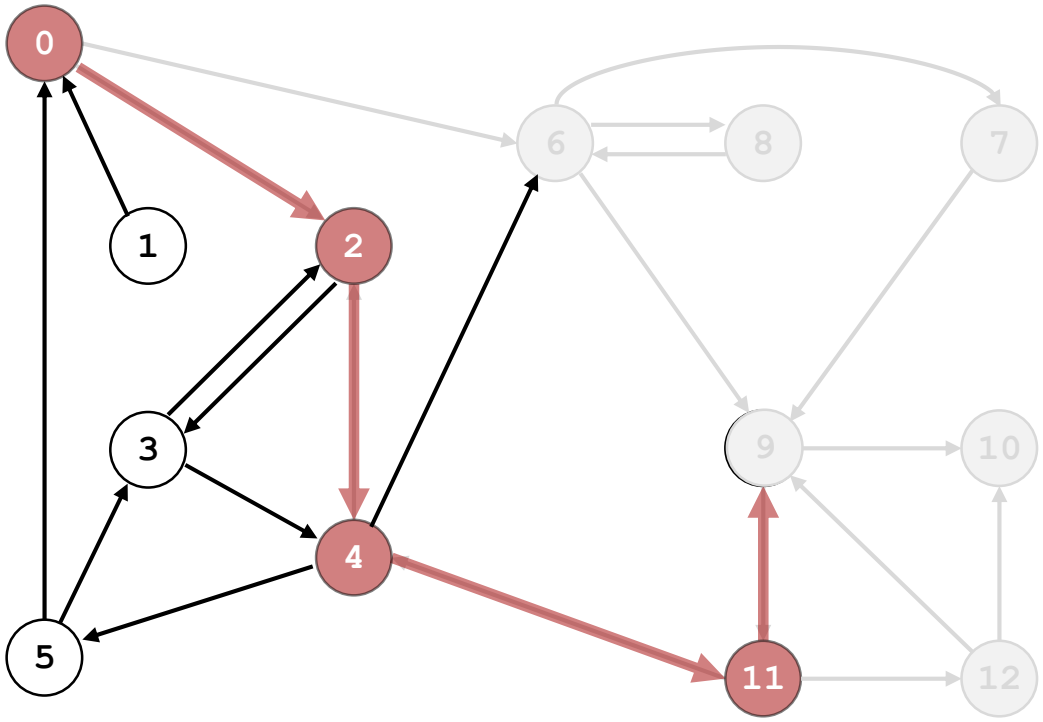
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 9: check 12, check 7, **check 6**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

9 12 10 6 7 8



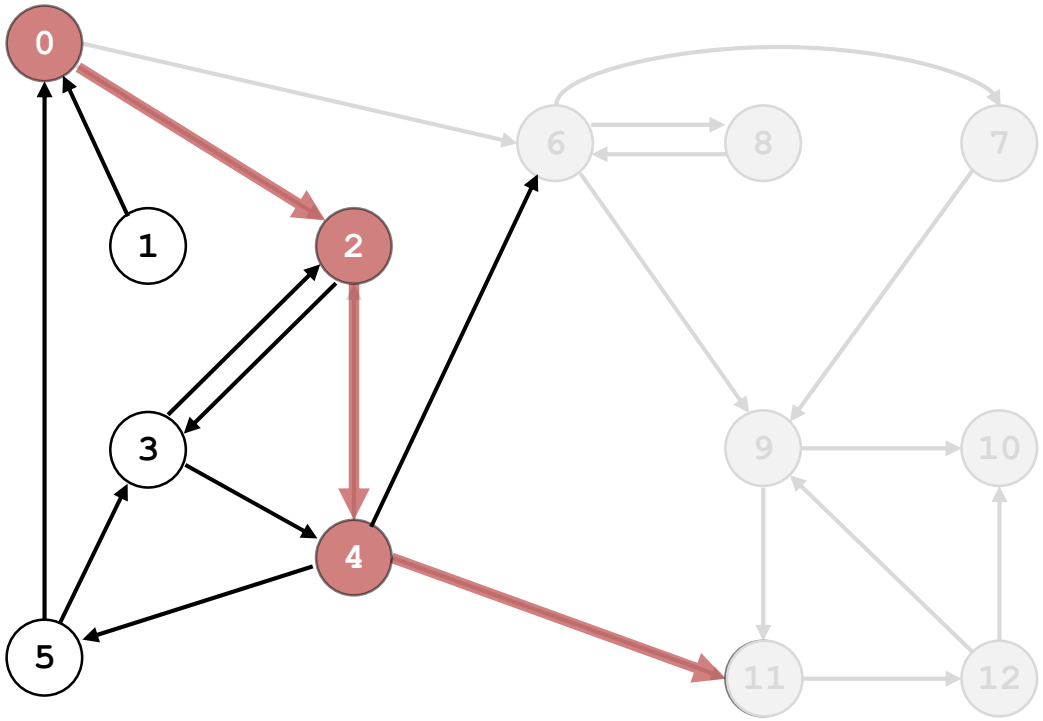
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

9 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

11 9 12 10 6 7 8



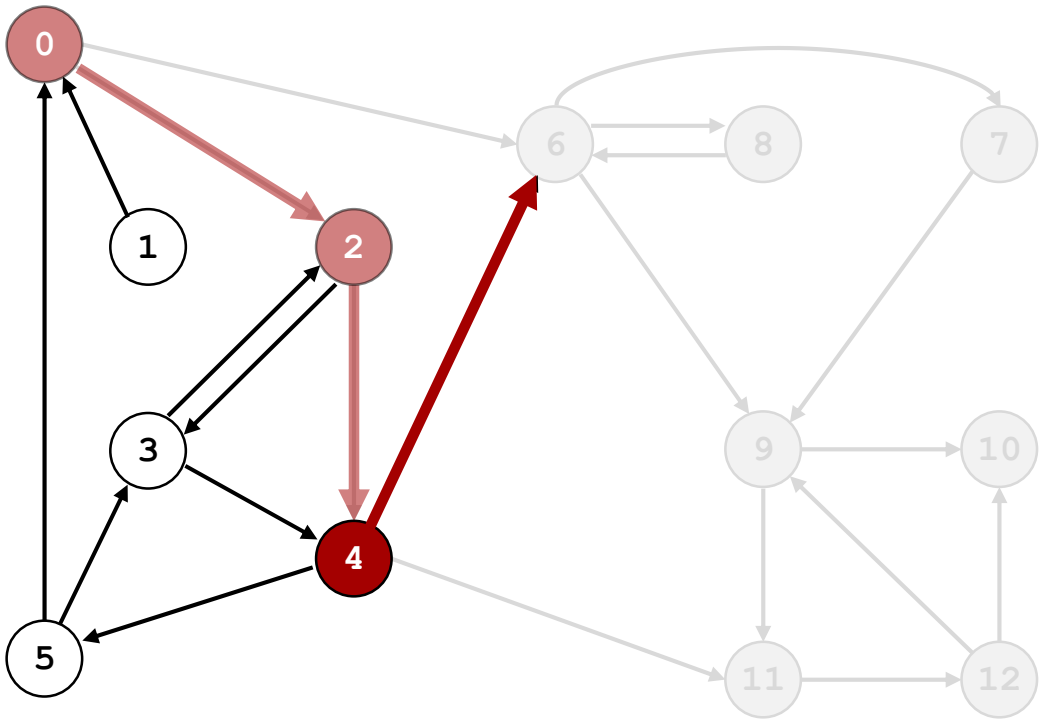
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

11 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

11 9 12 10 6 7 8



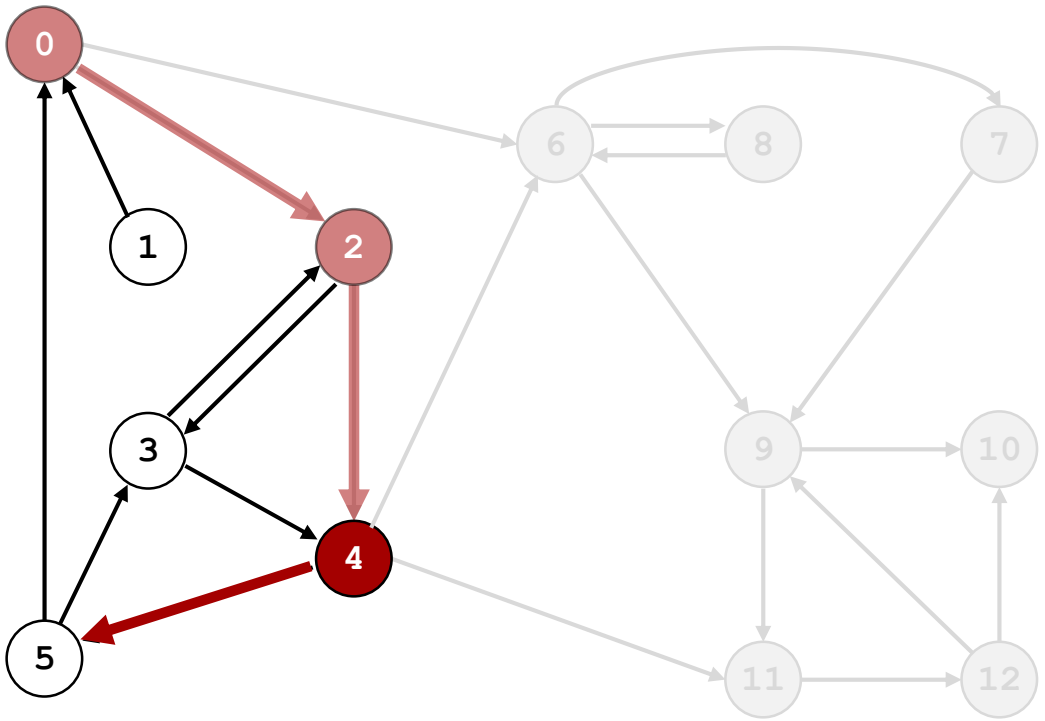
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 4: check 11, **check 6**, check 5

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

11 9 12 10 6 7 8



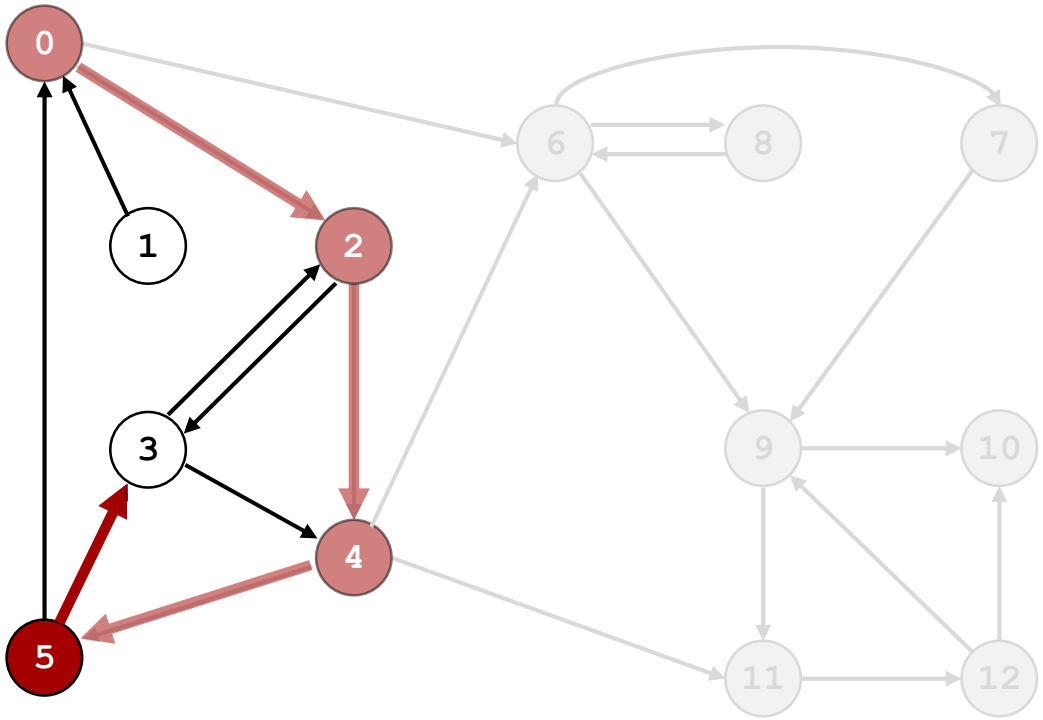
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	F
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 4: check 11, check 6, **check 5**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

11 9 12 10 6 7 8



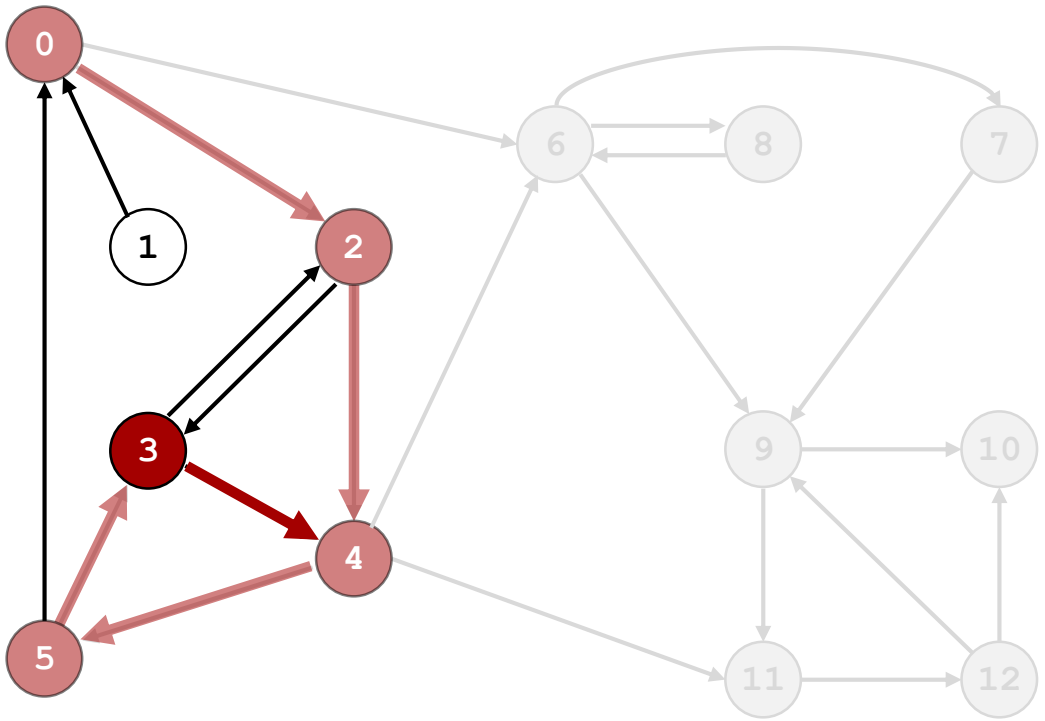
v	marked[]
0	T
1	F
2	T
3	F
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 5: check 3, check 0

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

11 9 12 10 6 7 8



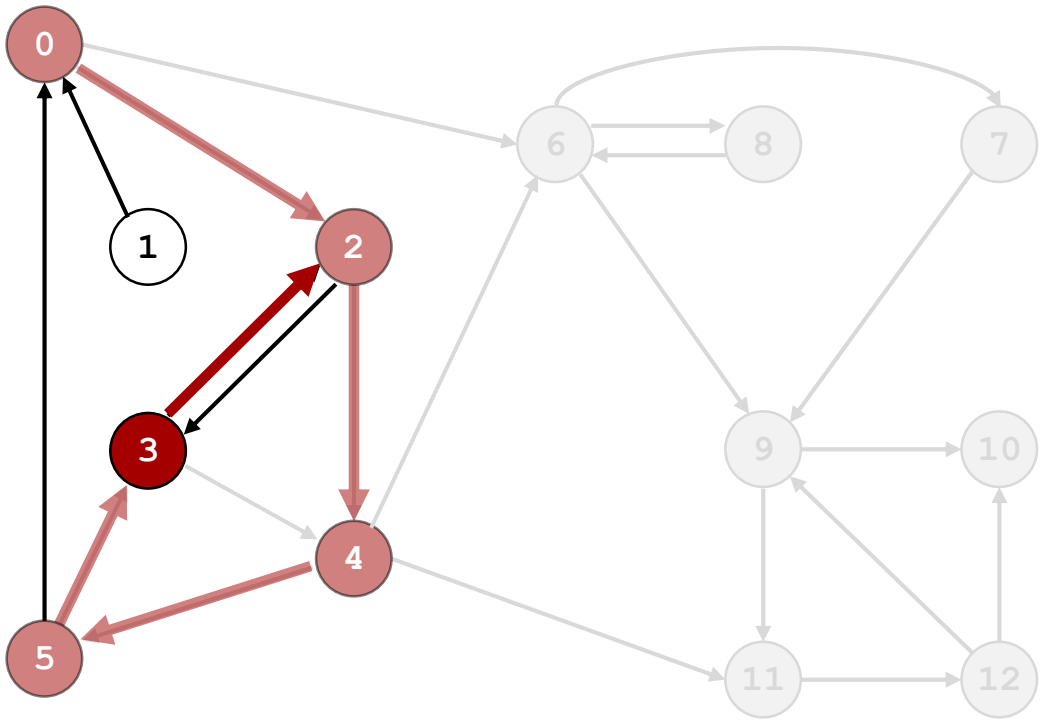
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 3: check 4, check 2

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

11 9 12 10 6 7 8



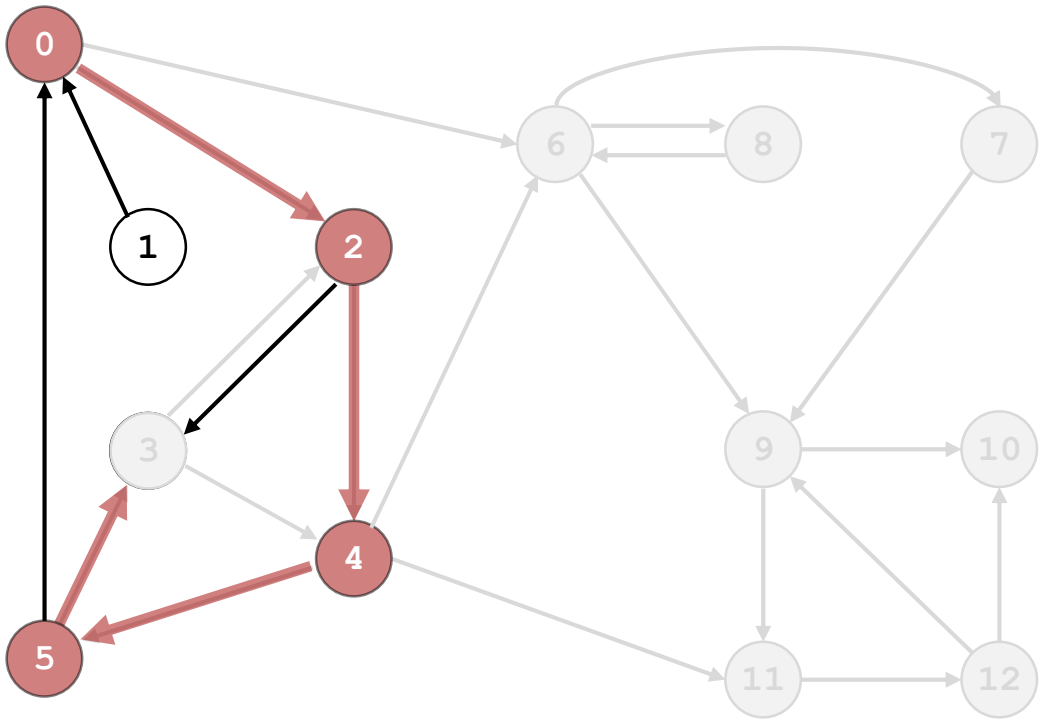
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 3: check 4, check 2

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

3 11 9 12 10 6 7 8



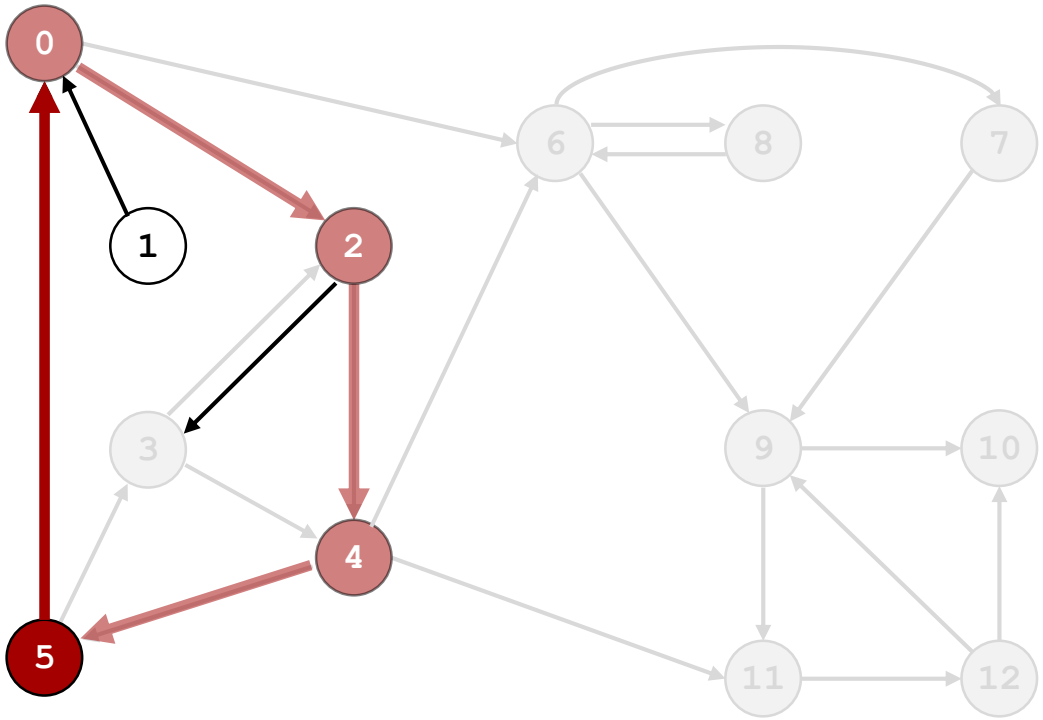
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

3 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

3 11 9 12 10 6 7 8



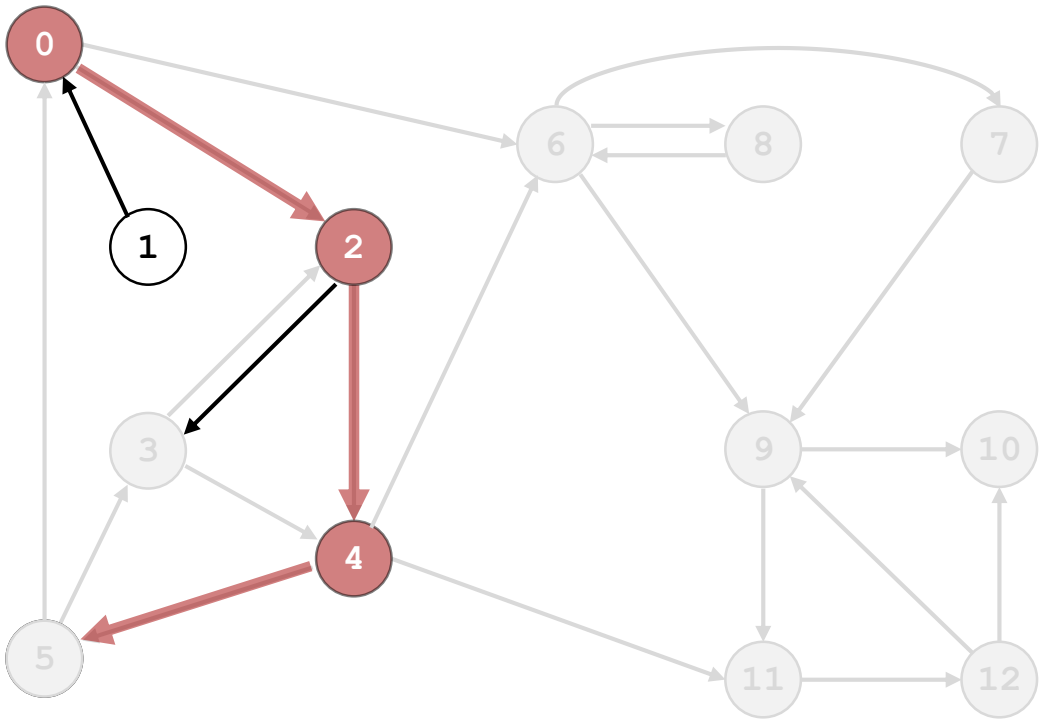
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 5: check 3, check 0

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

5 3 11 9 12 10 6 7 8



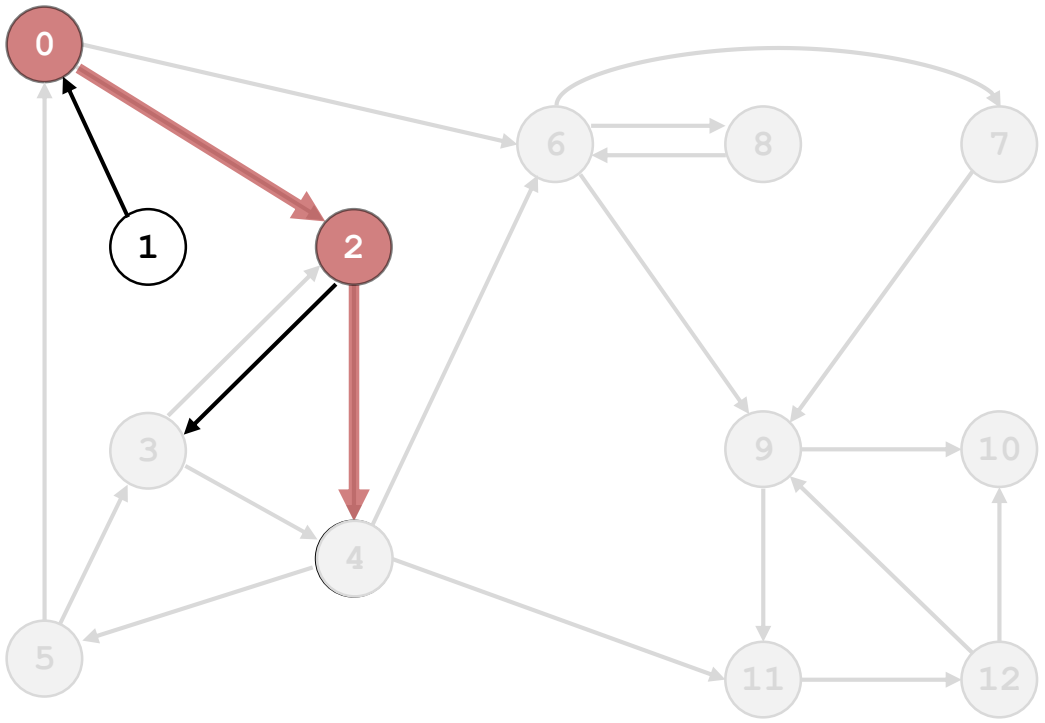
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

5 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

4 5 3 11 9 12 10 6 7 8



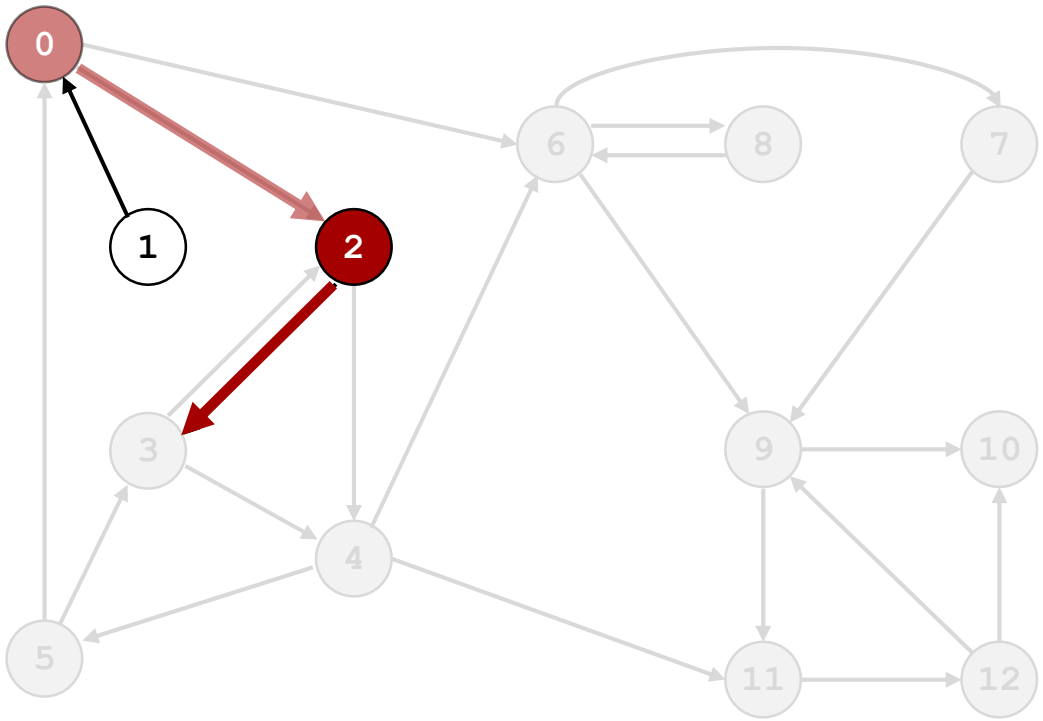
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

4 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

4 5 3 11 9 12 10 6 7 8



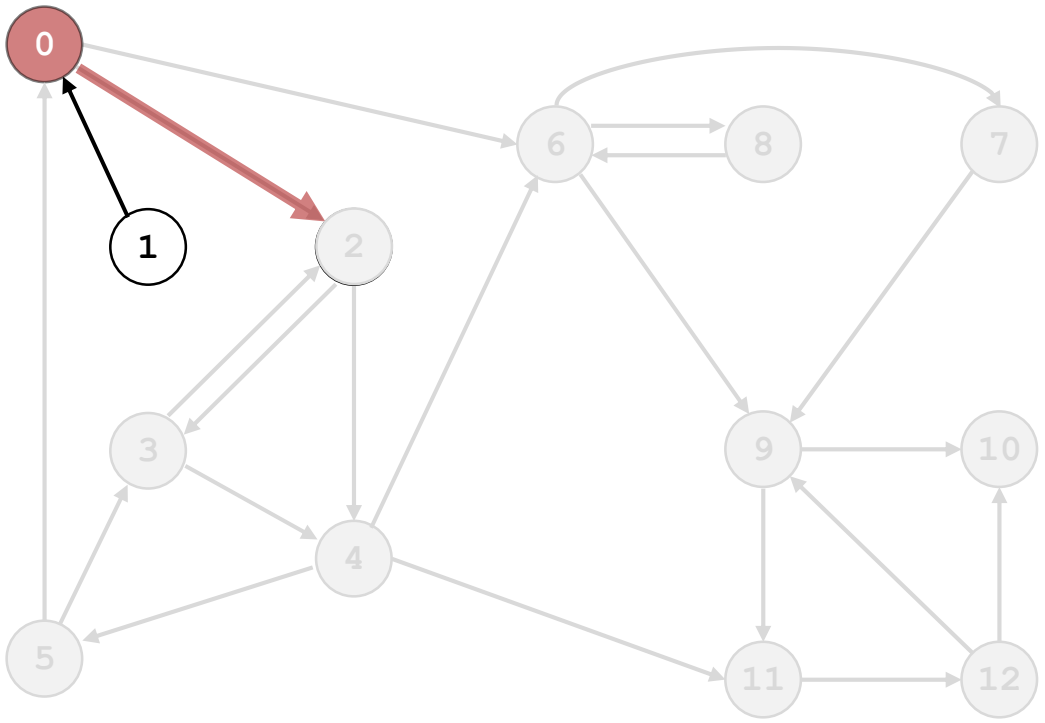
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 2: check 4, **check 3**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

2 4 5 3 11 9 12 10 6 7 8



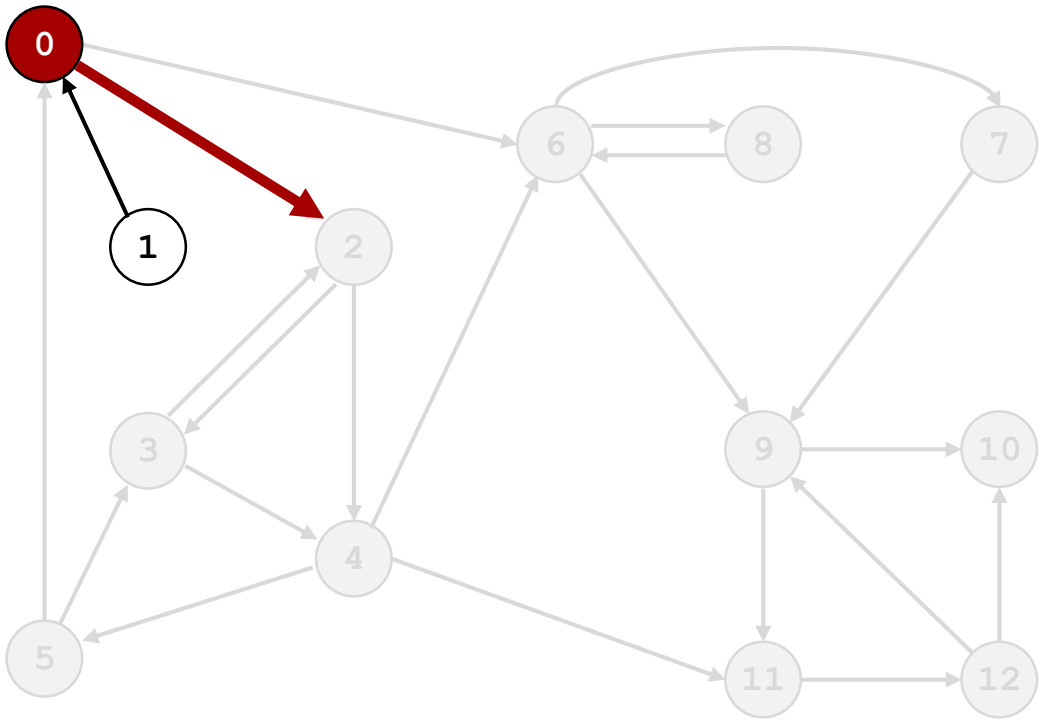
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

2 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

2 4 5 3 11 9 12 10 6 7 8



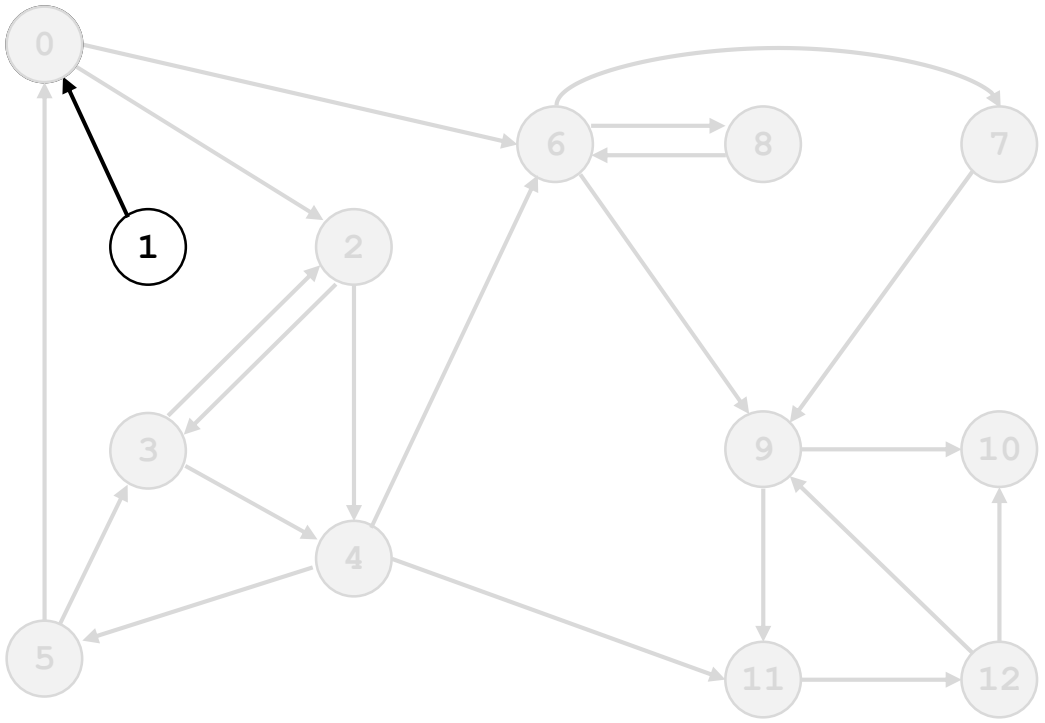
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 0: check 6, **check 2**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

0 2 4 5 3 11 9 12 10 6 7 8



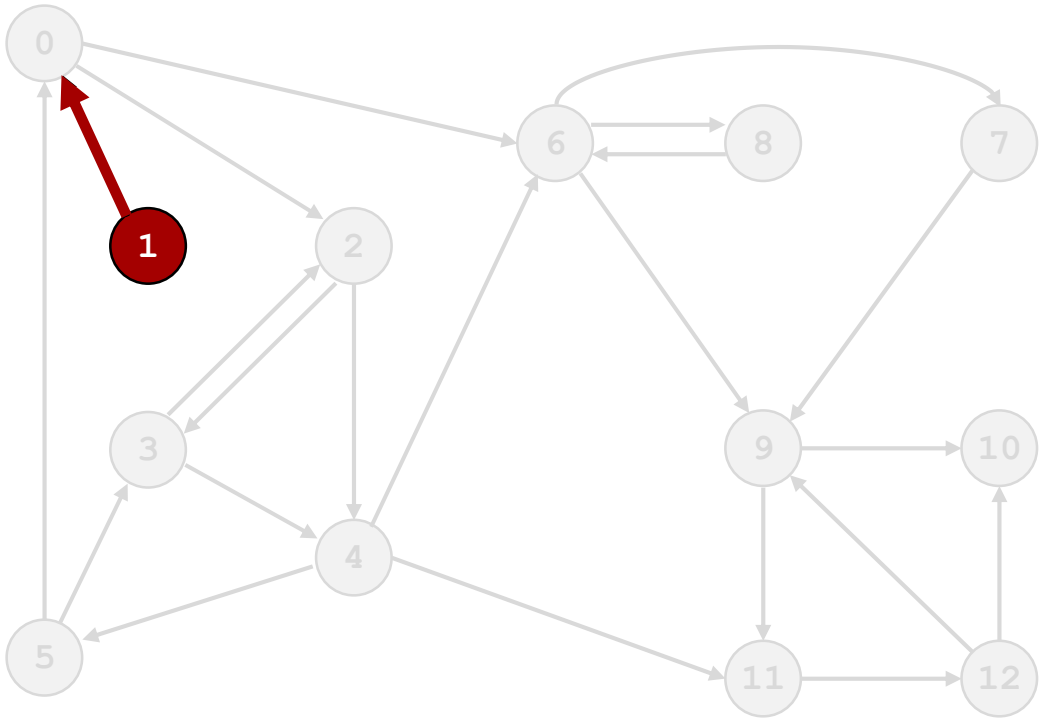
v	marked[]
0	T
1	F
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

0 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

0 2 4 5 3 11 9 12 10 6 7 8



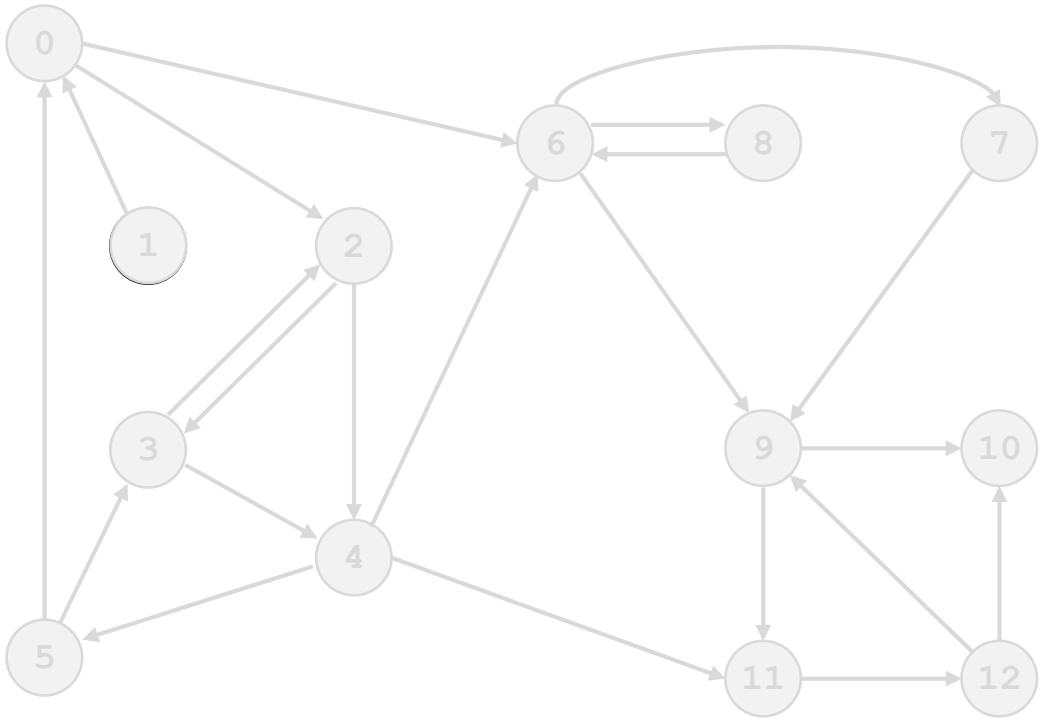
v	marked[]
0	T
1	T
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

Visit 1: check 0

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



v	marked[]
0	T
1	T
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

1 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۱. محاسبه‌ی عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8

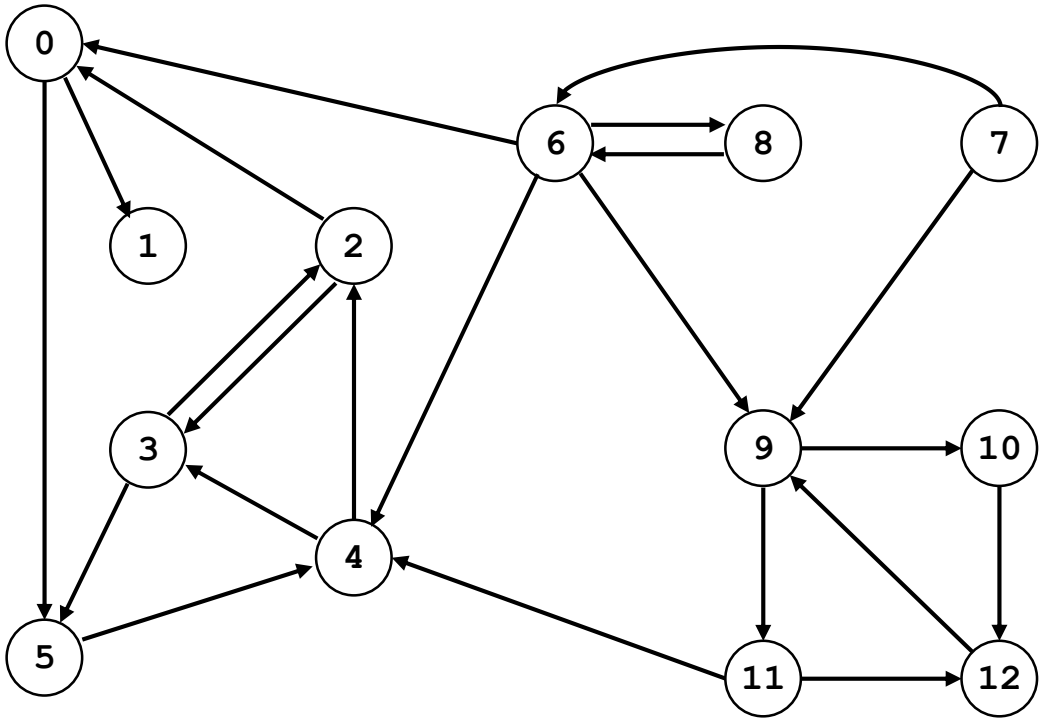
v	marked[]
0	T
1	T
2	T
3	T
4	T
5	T
6	T
7	T
8	T
9	T
10	T
11	T
12	T

done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



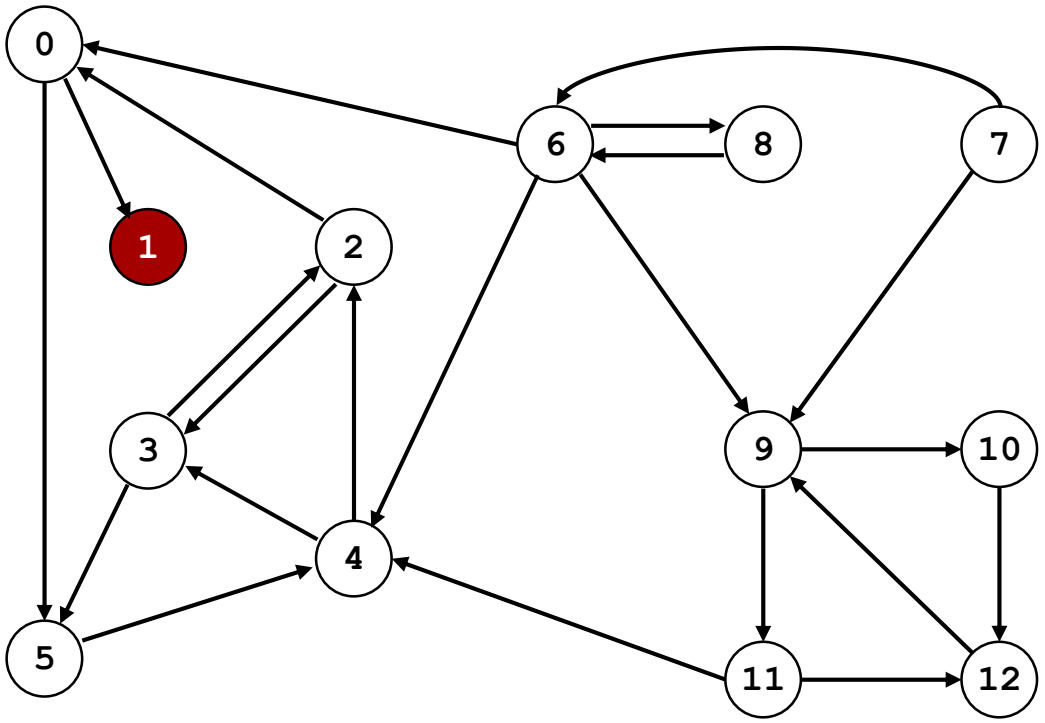
v	scc[]
0	-
1	-
2	-
3	-
4	-
5	-
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Graph G

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



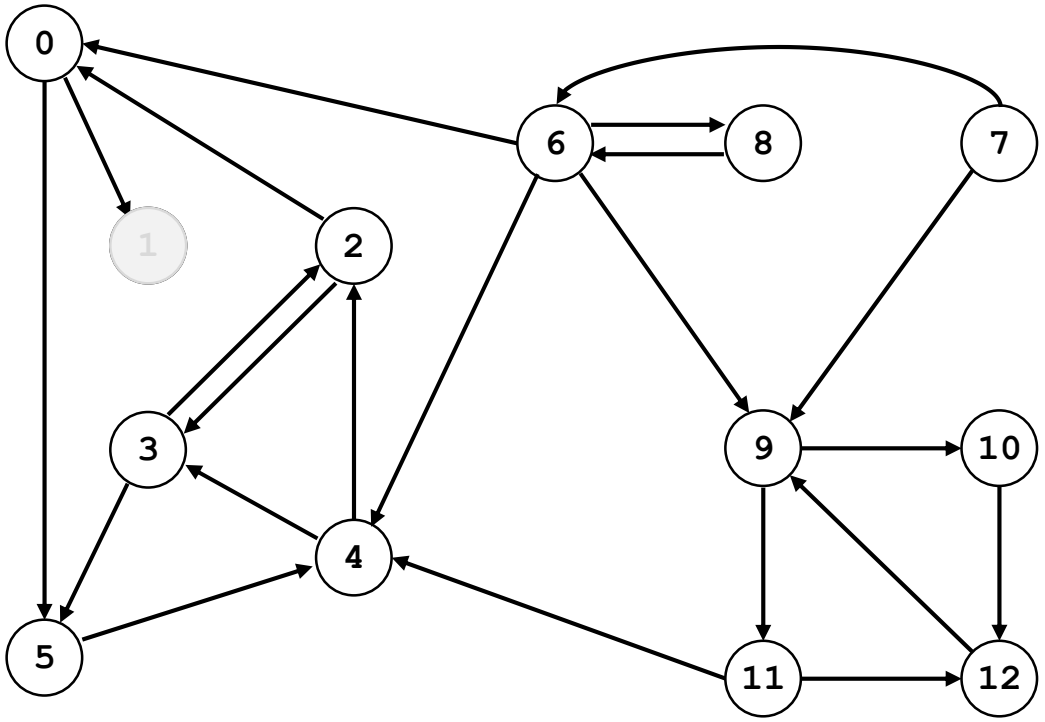
v	scc[]
0	-
1	0
2	-
3	-
4	-
5	-
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 1

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



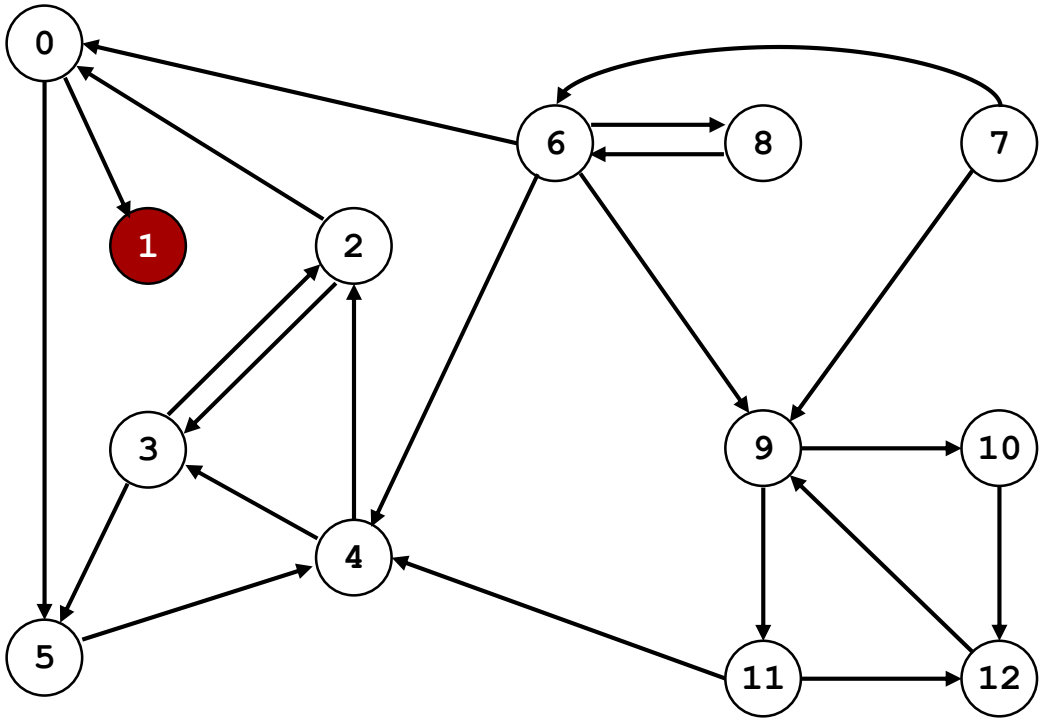
v	scc[]
0	-
1	0
2	-
3	-
4	-
5	-
6	-
7	-
8	-
9	-
10	-
11	-
12	-

1 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



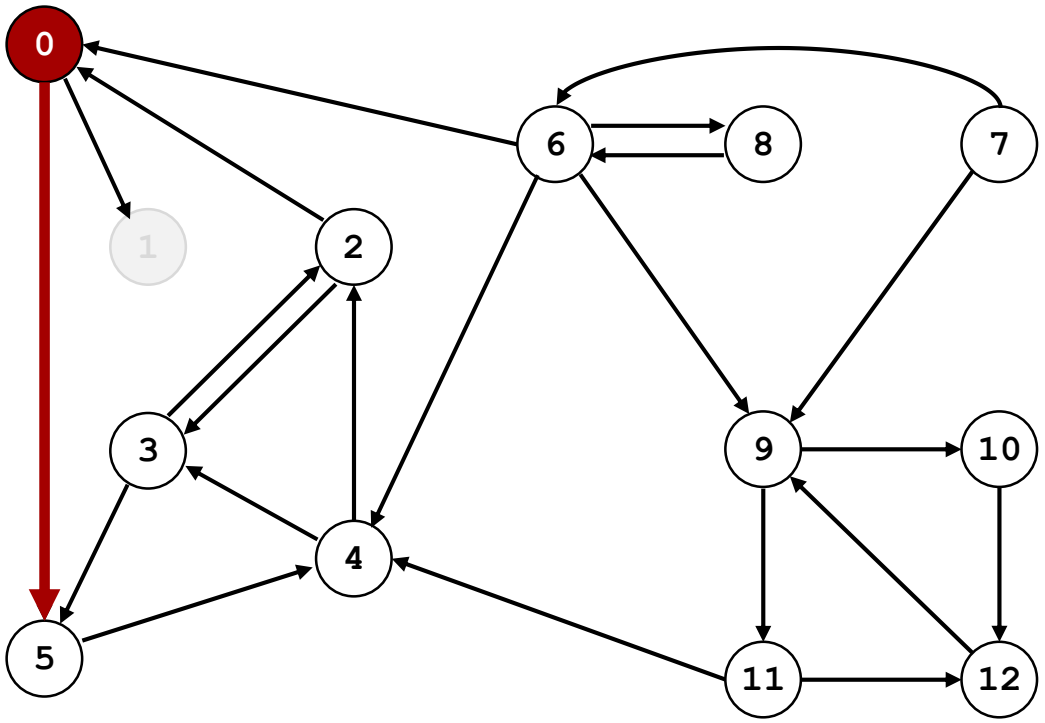
v	scc[]
0	-
1	0
2	-
3	-
4	-
5	-
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Strong component: 1

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



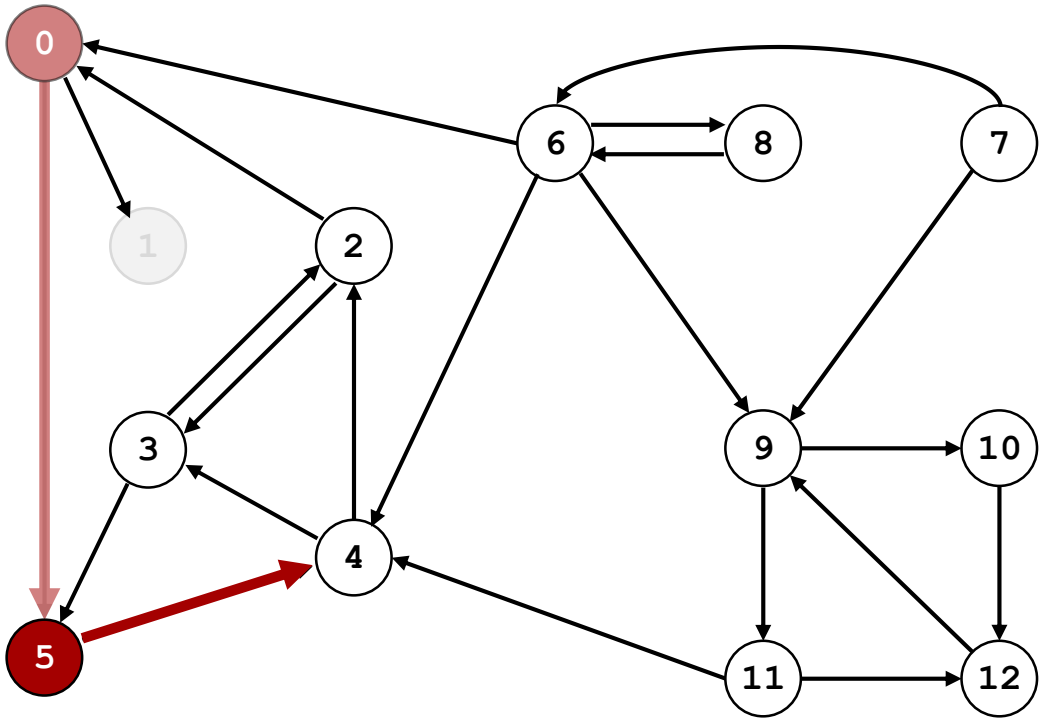
v	scc[]
0	1
1	0
2	-
3	-
4	-
5	-
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 0: check 5, check 1

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



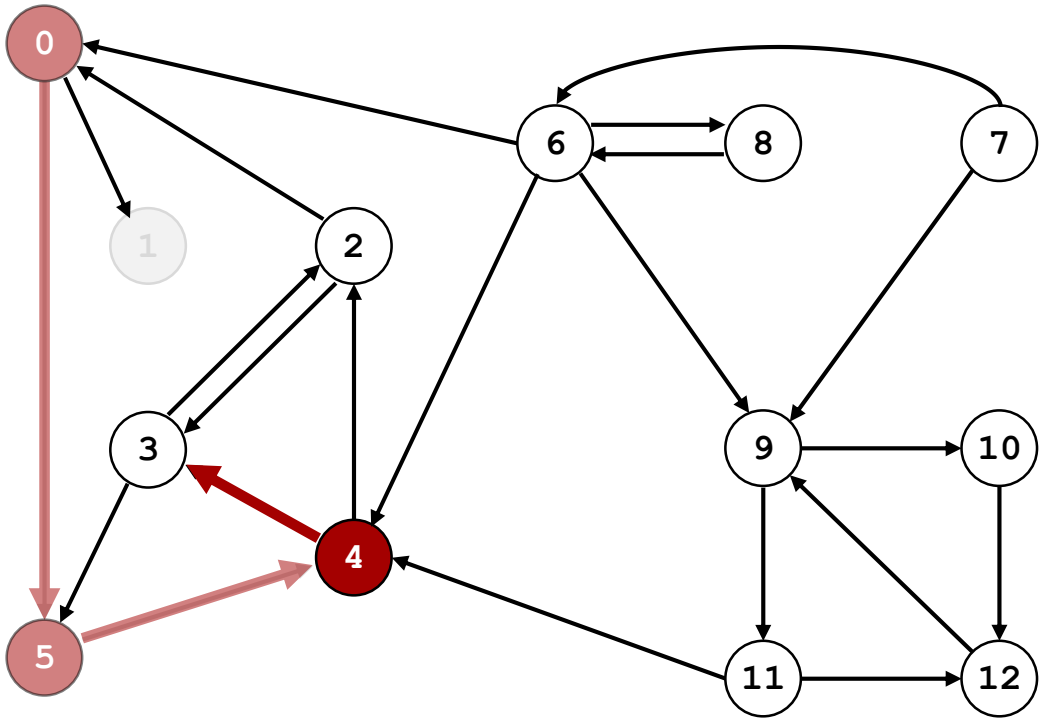
v	scc[]
0	1
1	0
2	-
3	-
4	-
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 5: check 4

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



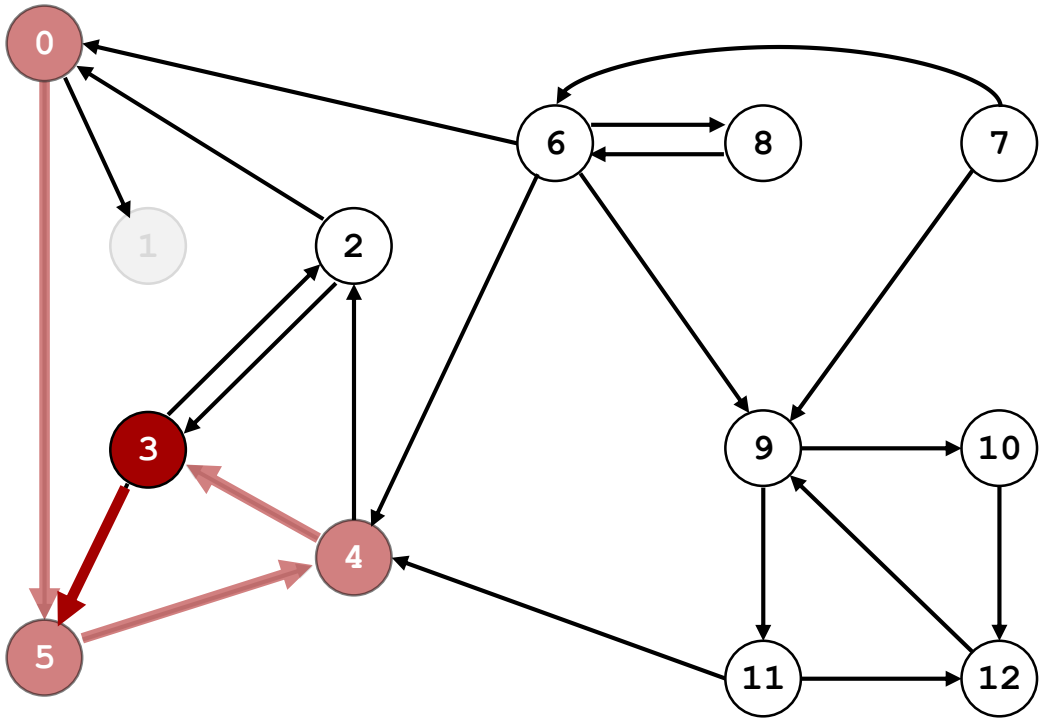
v	scc[]
0	1
1	0
2	-
3	-
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 4: check 3, check 2

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



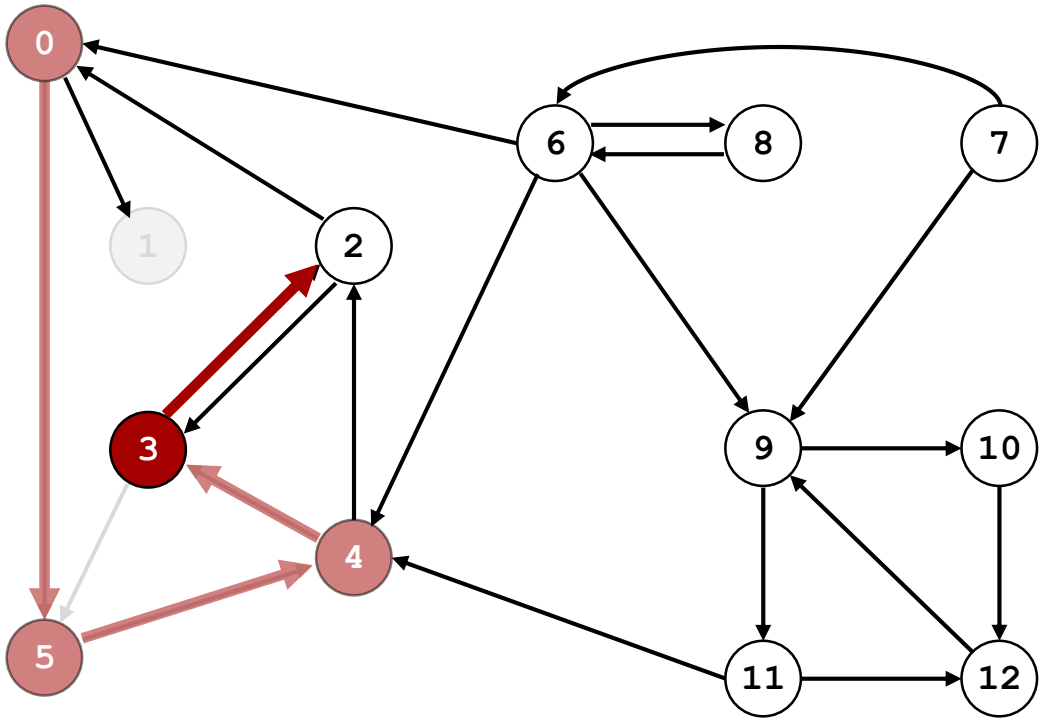
v	scc[]
0	1
1	0
2	-
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 3: check 5, check 2

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



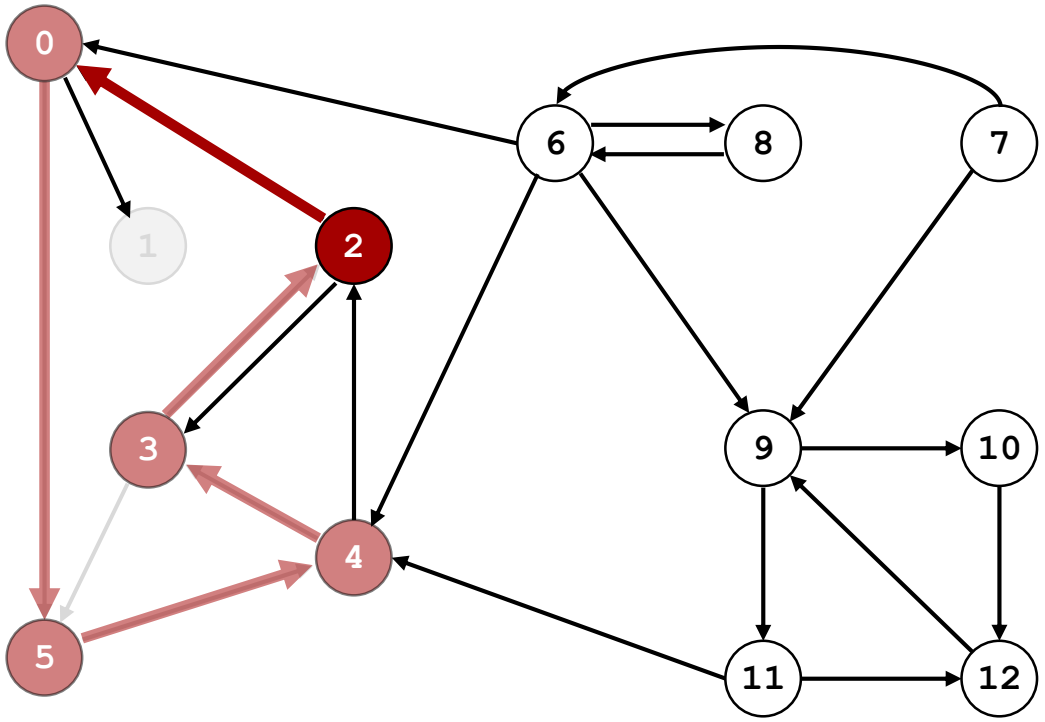
v	scc[]
0	1
1	0
2	-
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 3: check 5, **check 2**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



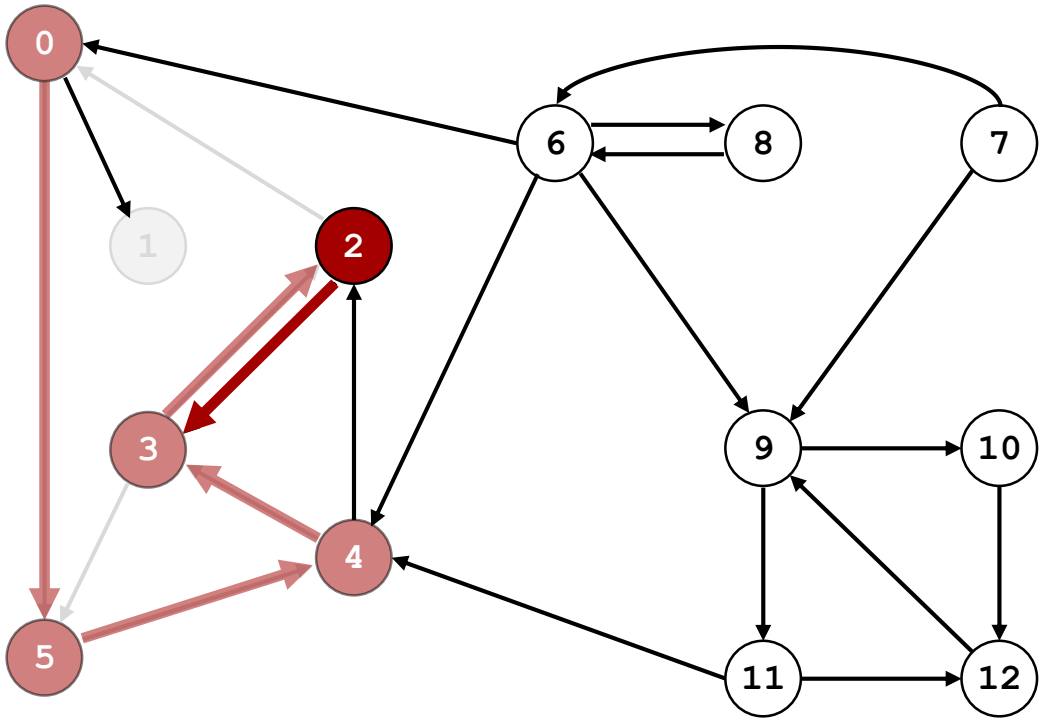
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 2: check 0, check 3

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



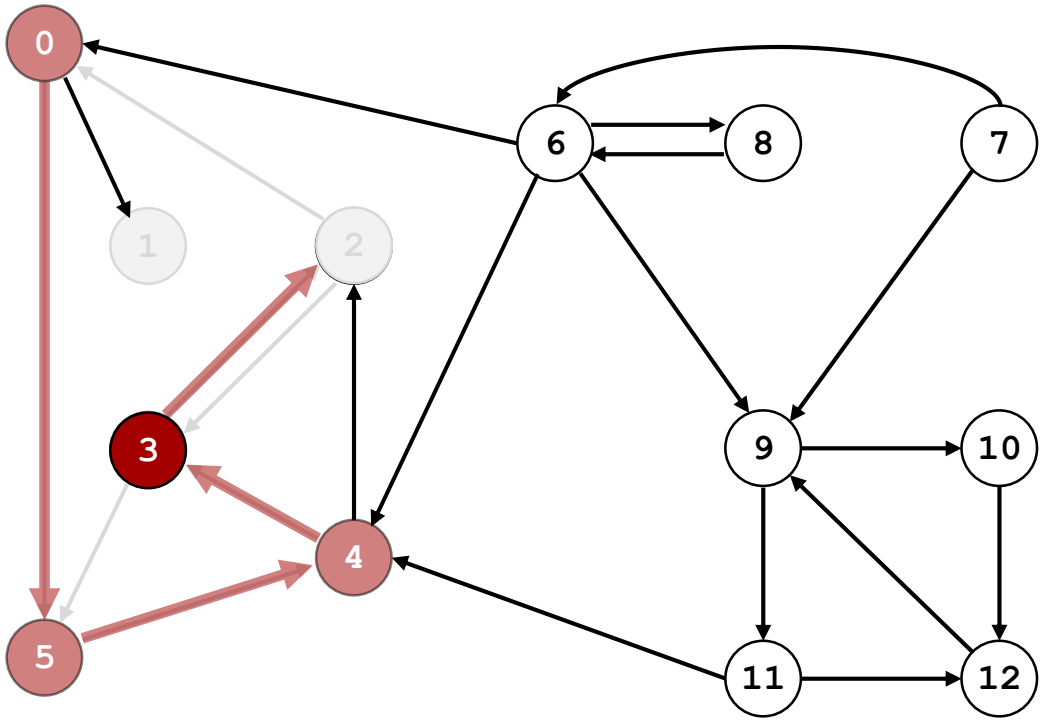
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 2: check 0, **check 3**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



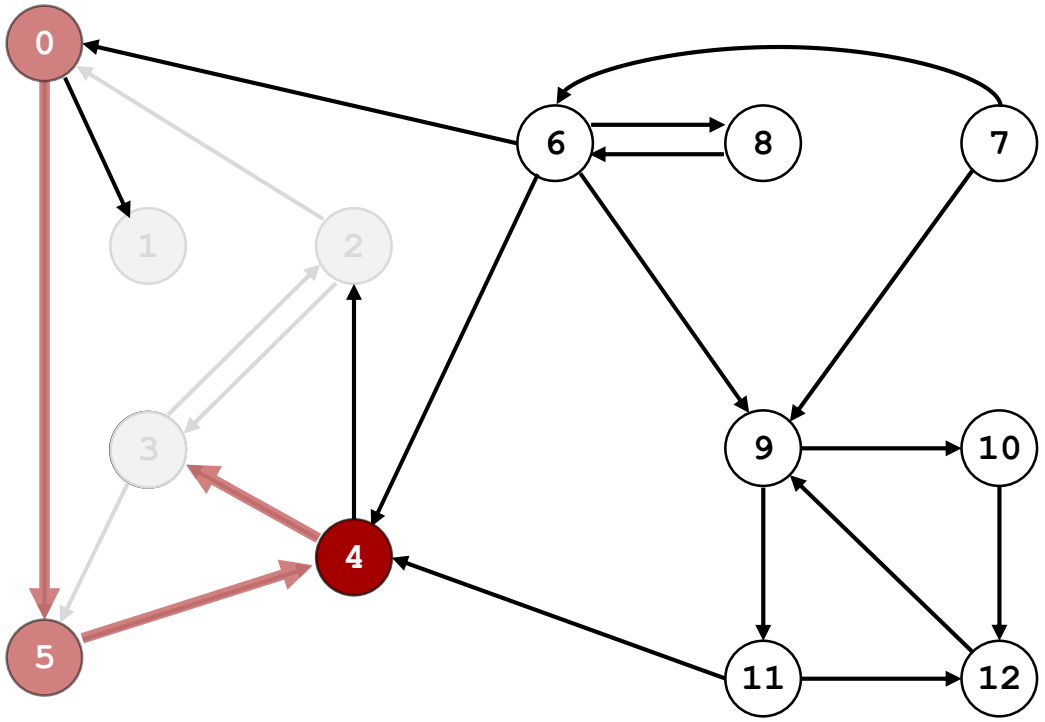
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

2 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



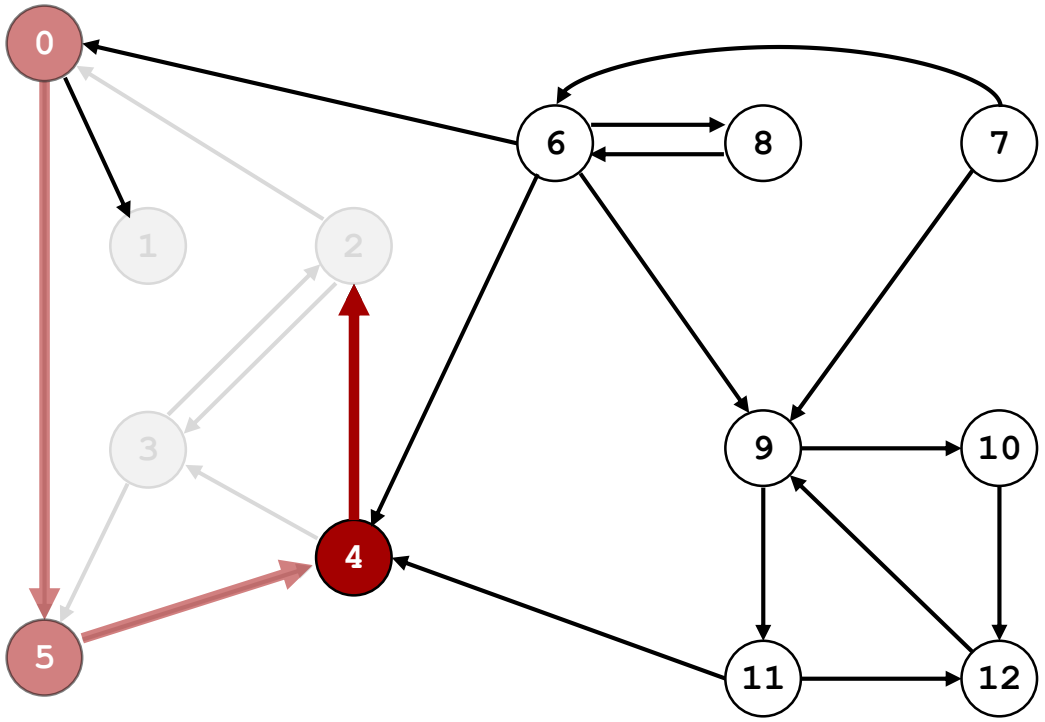
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

3 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



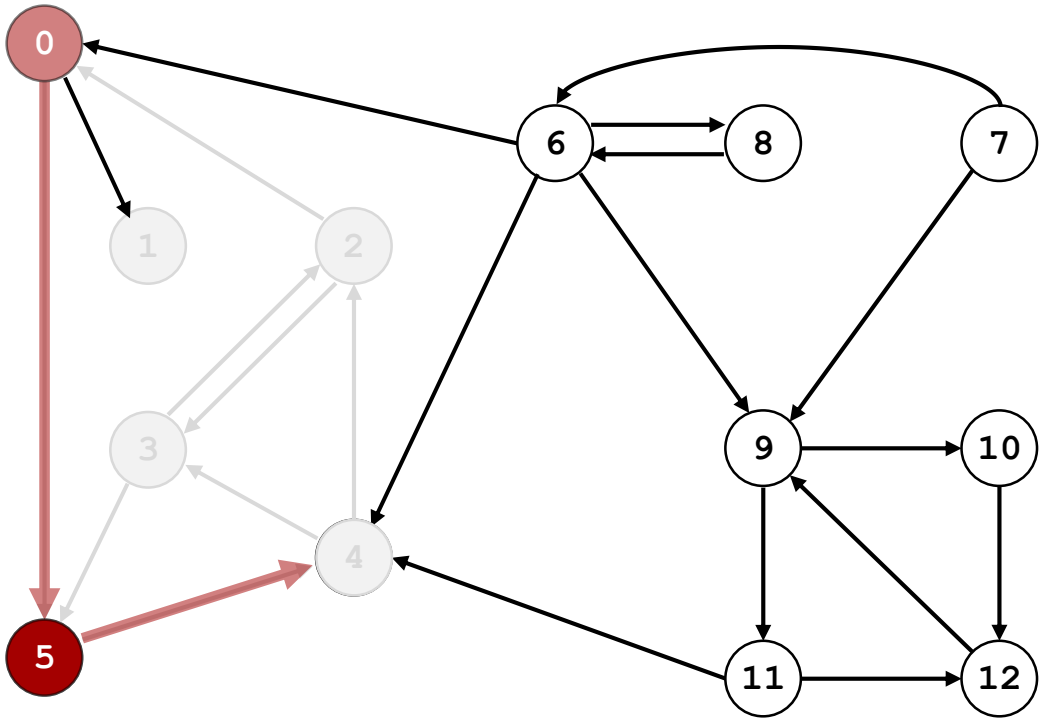
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 4: check 3, **check 2**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



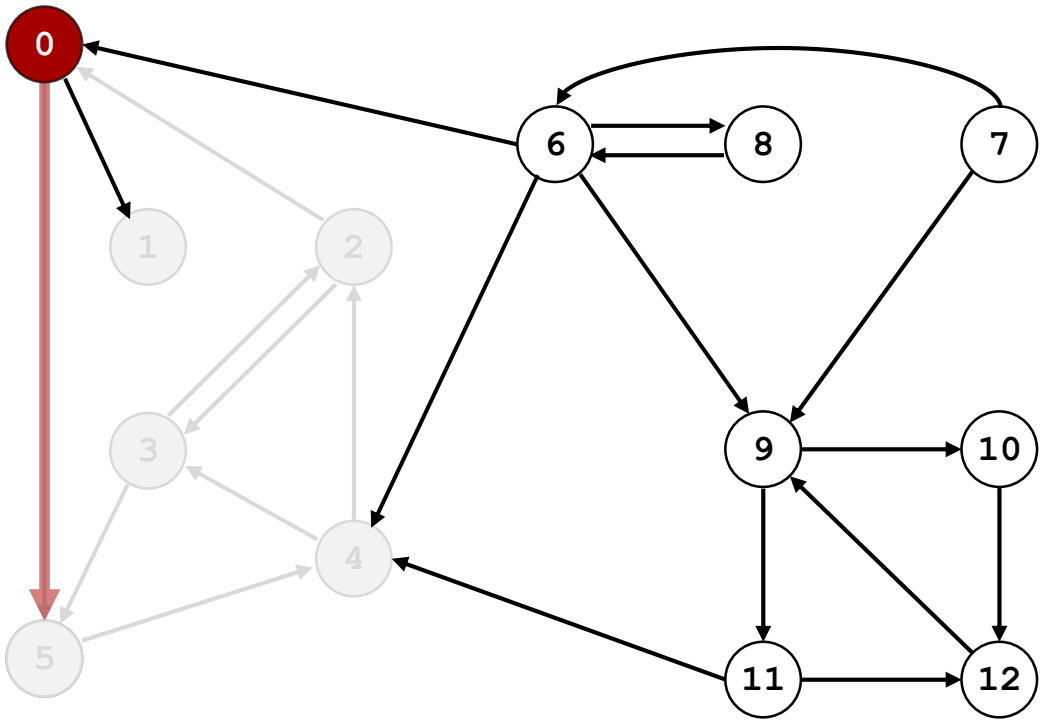
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

4 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



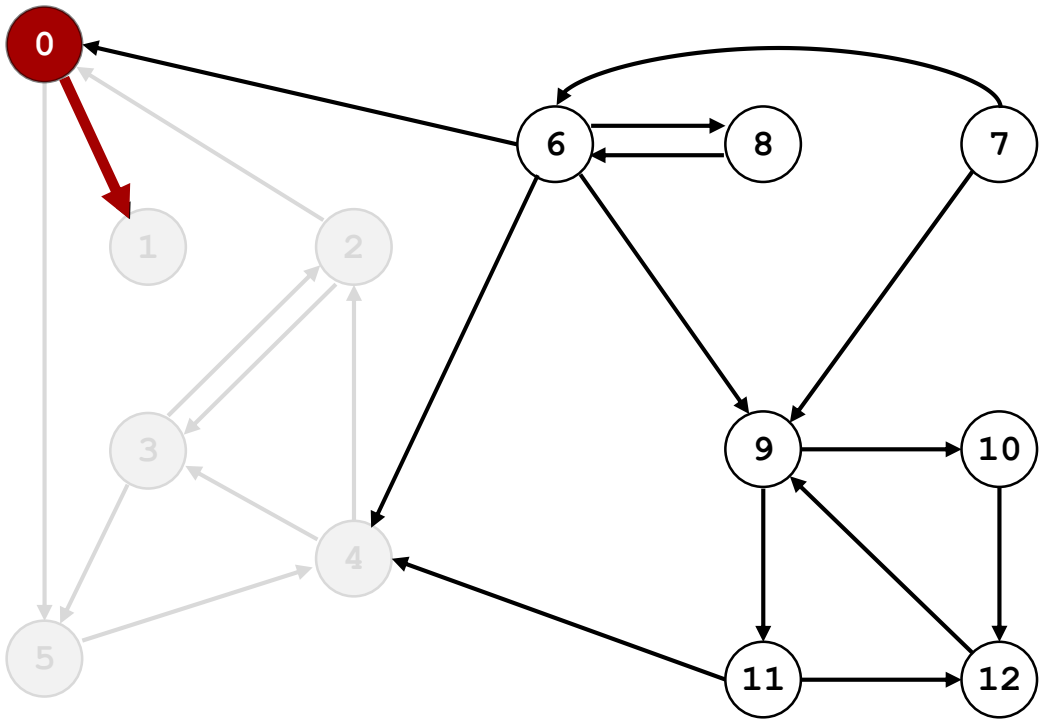
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

5 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



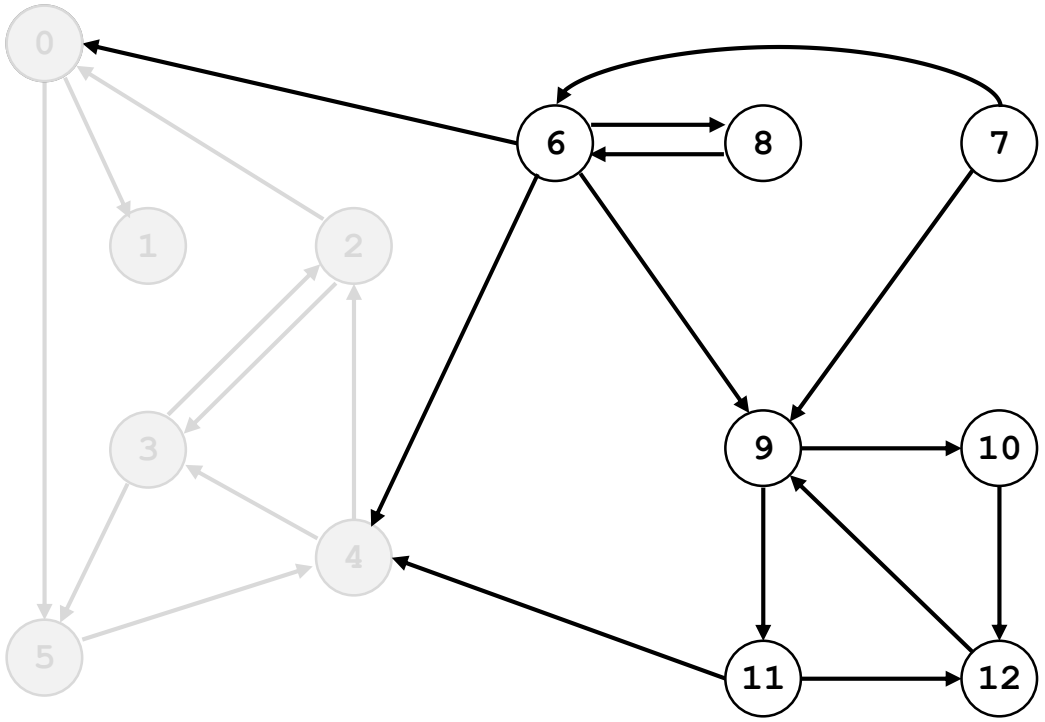
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Visit 0: check 5, **check 1**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



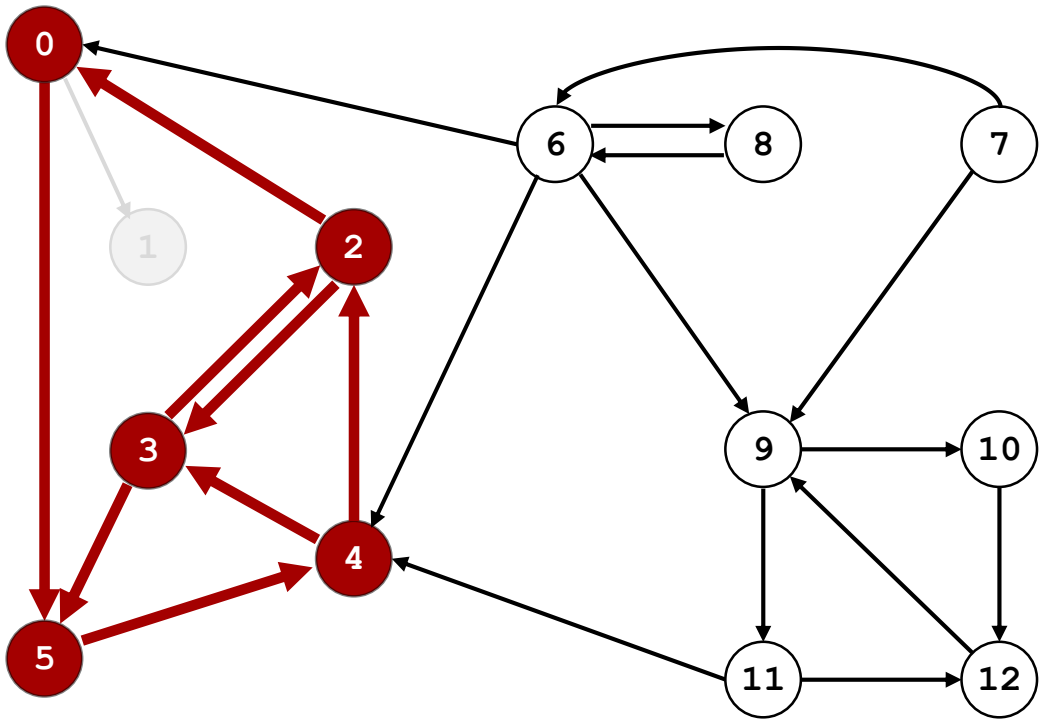
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

0 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



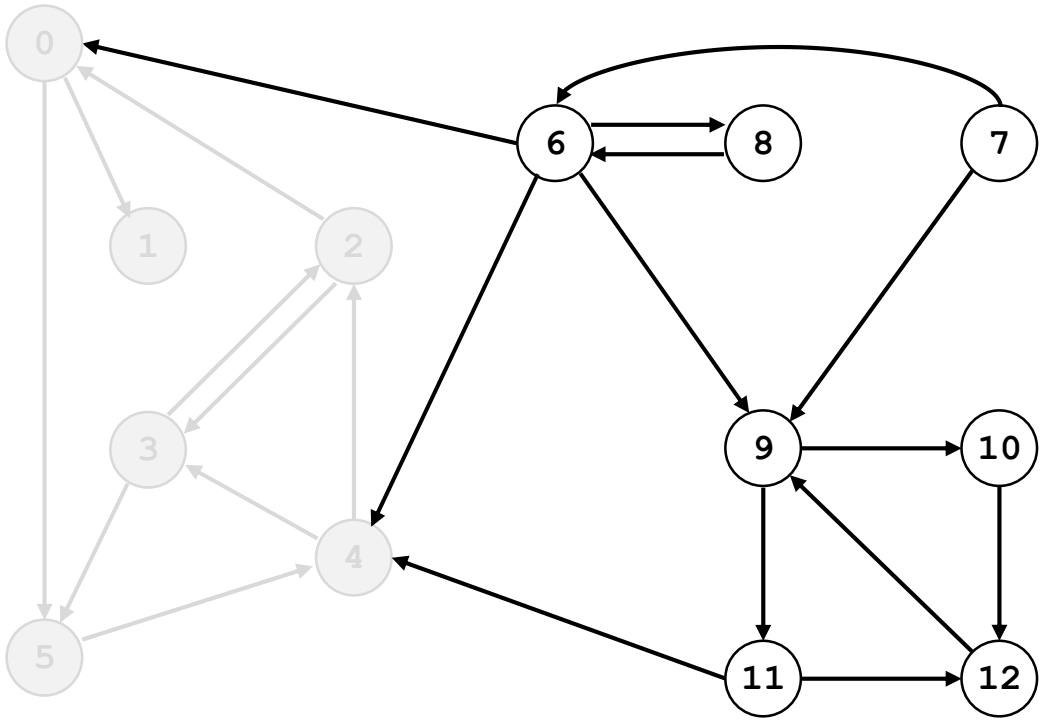
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

```
Strong component: 0 2 3 4 5
```

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



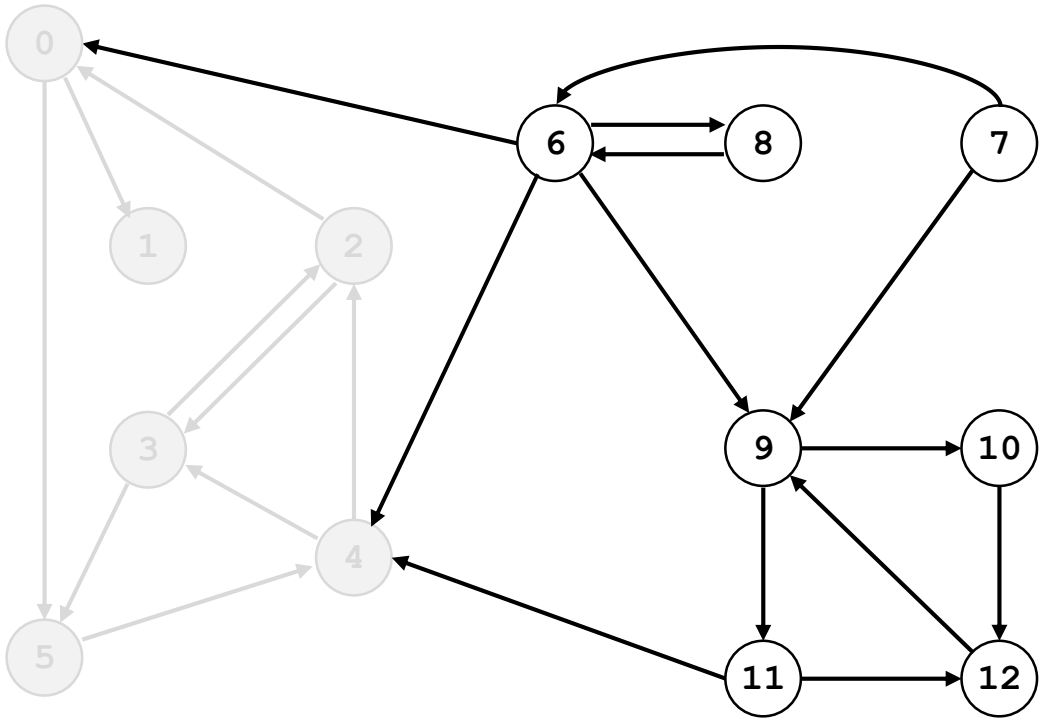
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Check 2

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 **4** 5 3 11 9 12 10 6 7 8



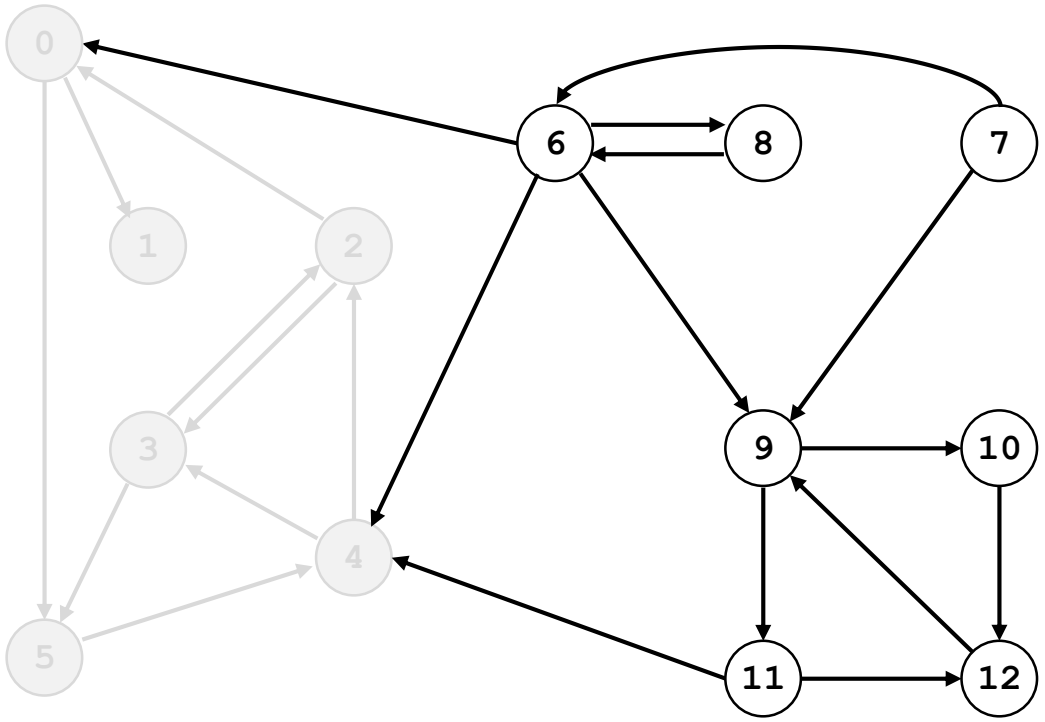
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Check 4

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



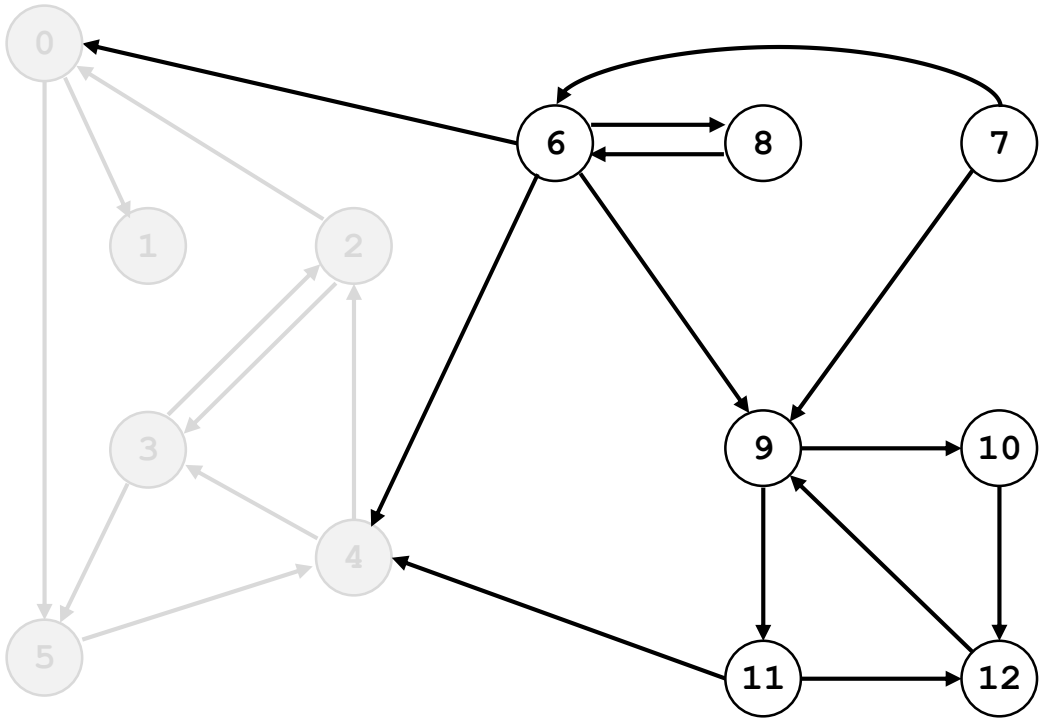
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Check 5

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



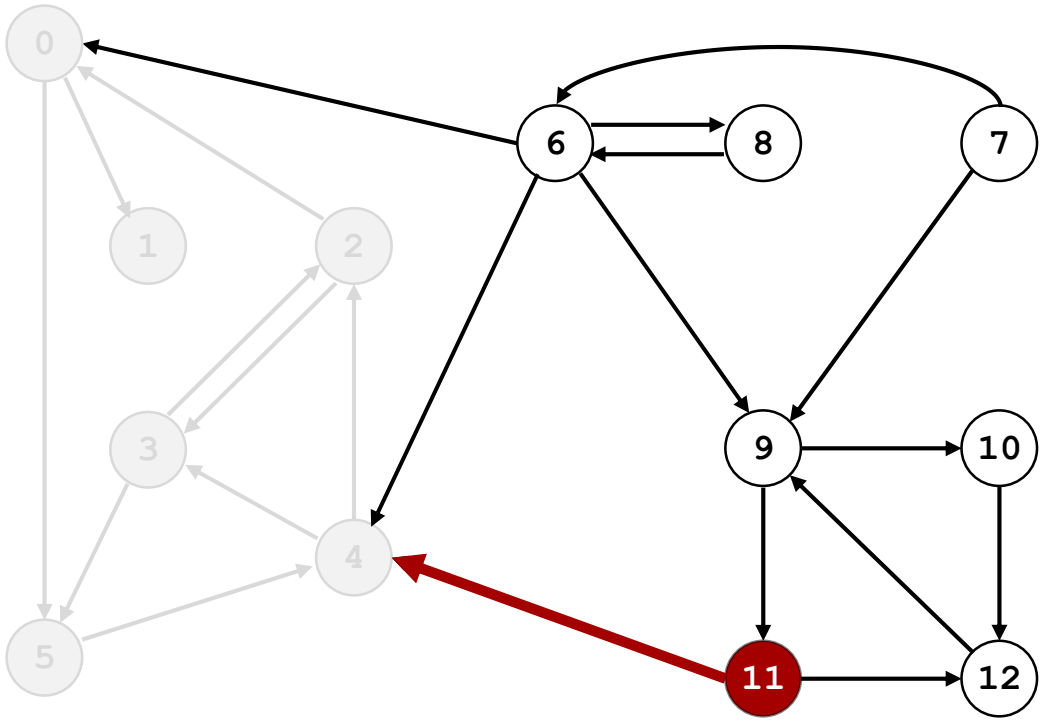
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	-
12	-

Check 3

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



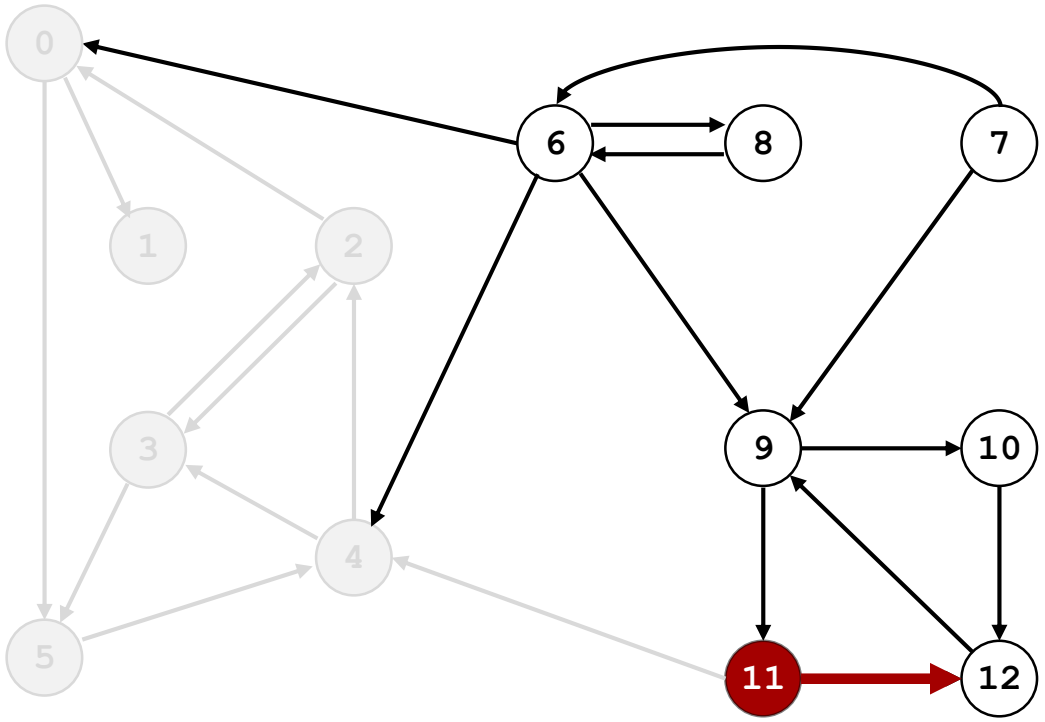
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	②
12	-

Visit 11: check 4, check 12

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



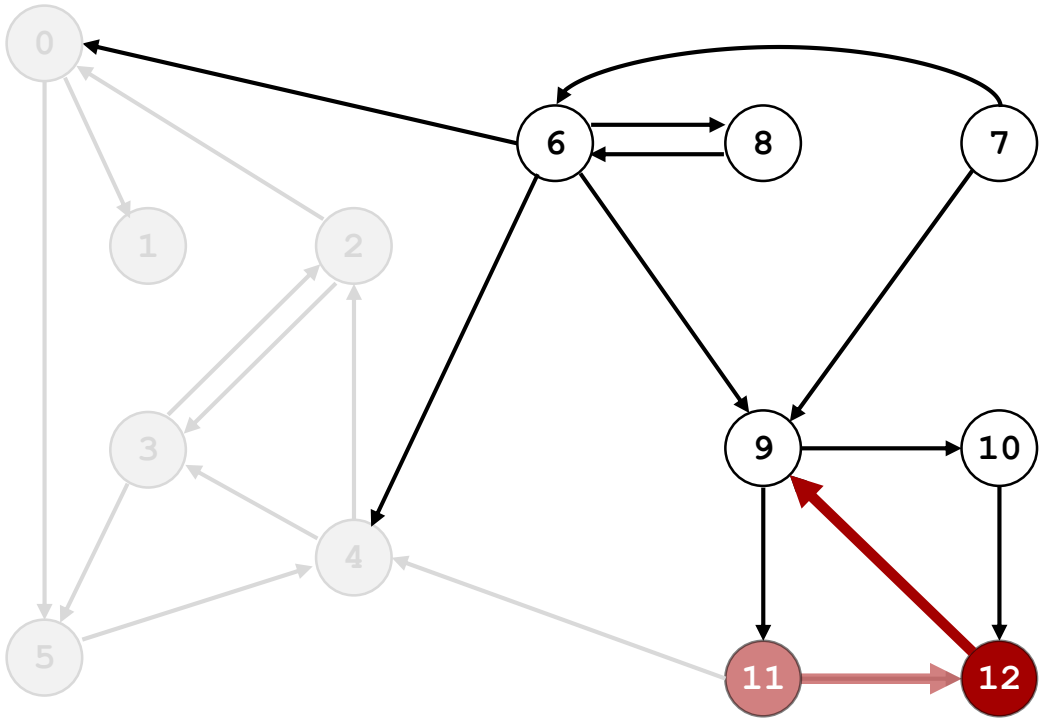
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	2
12	-

Visit 11: check 4, **check 12**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



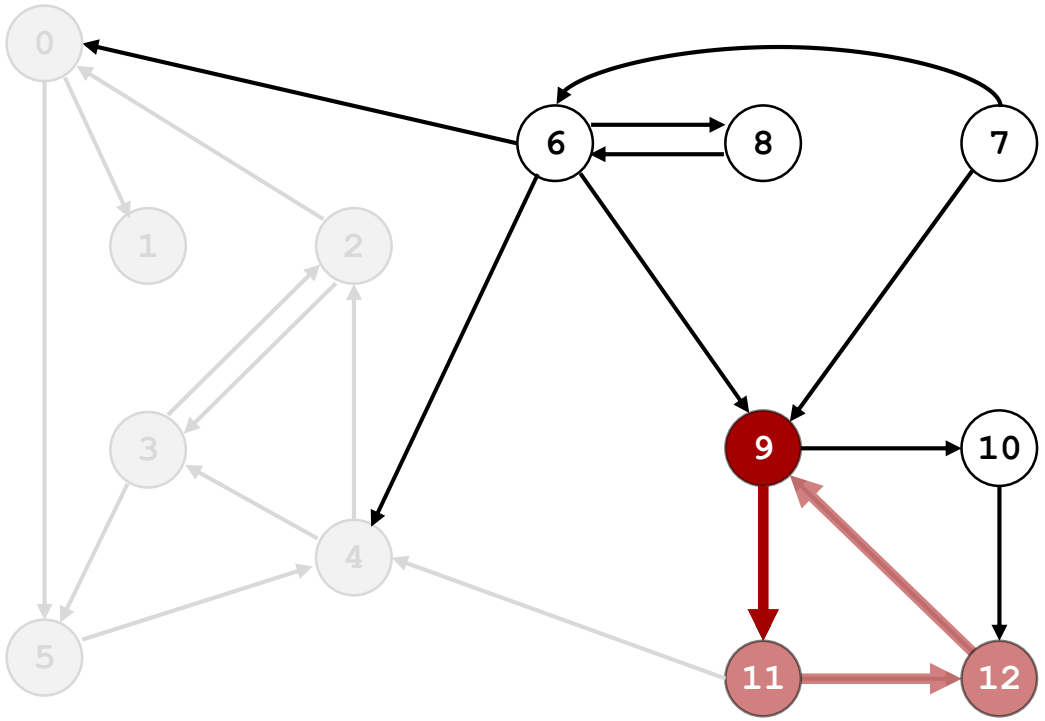
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	-
10	-
11	2
12	②

Visit 12: check 9

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



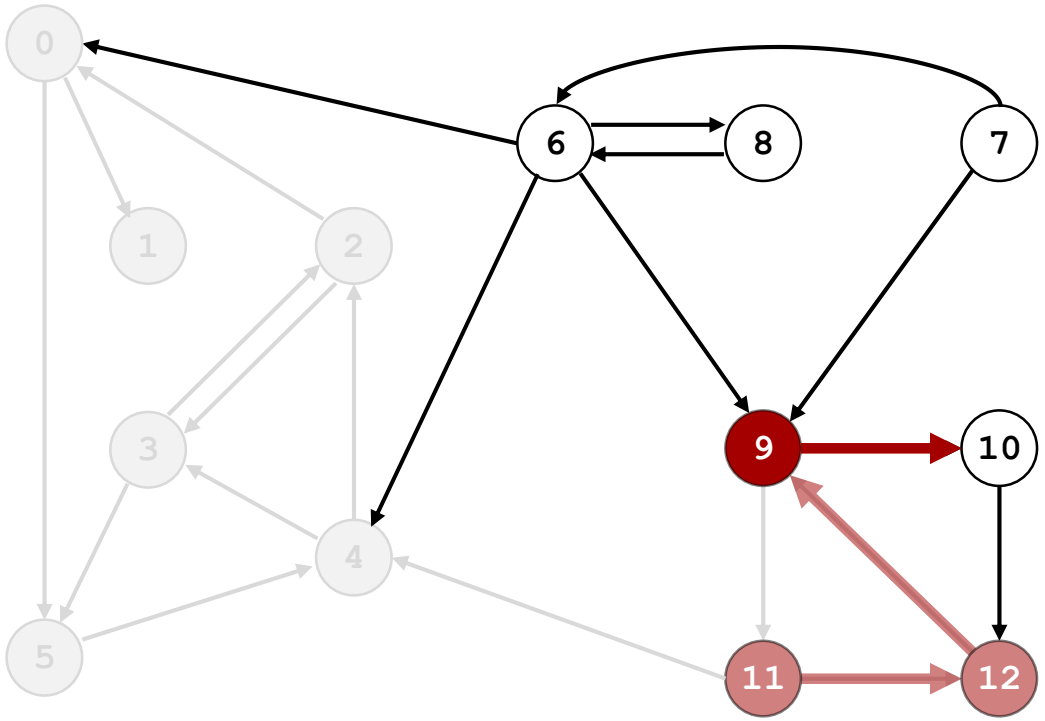
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	-
11	2
12	2

Visit 9: check 11, check 10

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



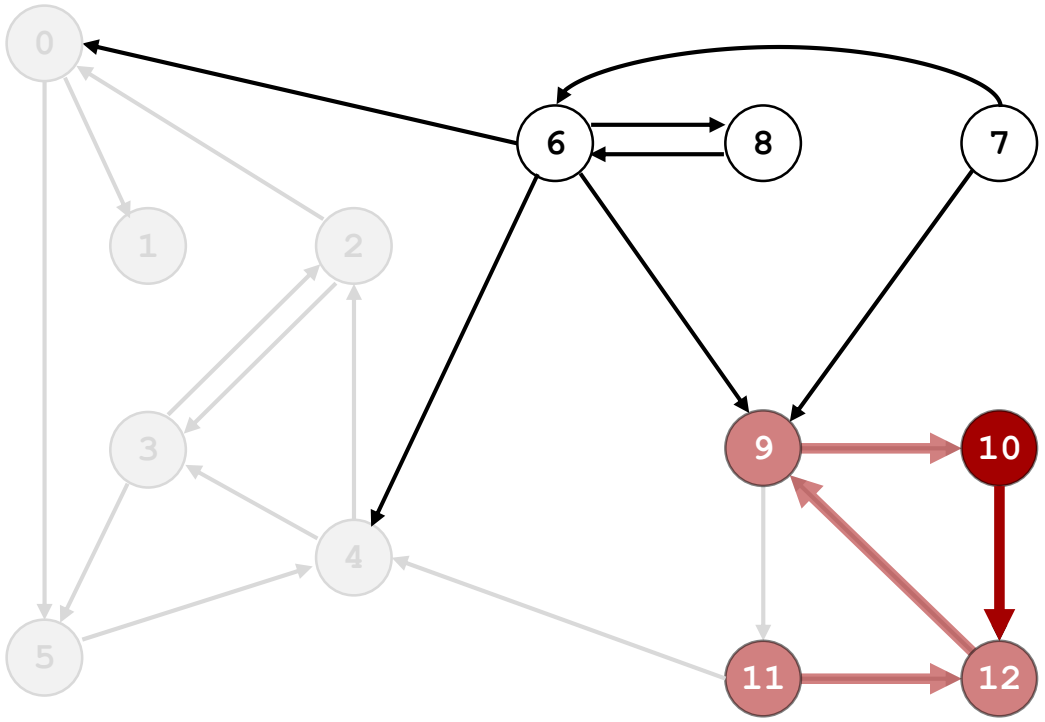
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	-
11	2
12	2

Visit 9: check 11, **check 10**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



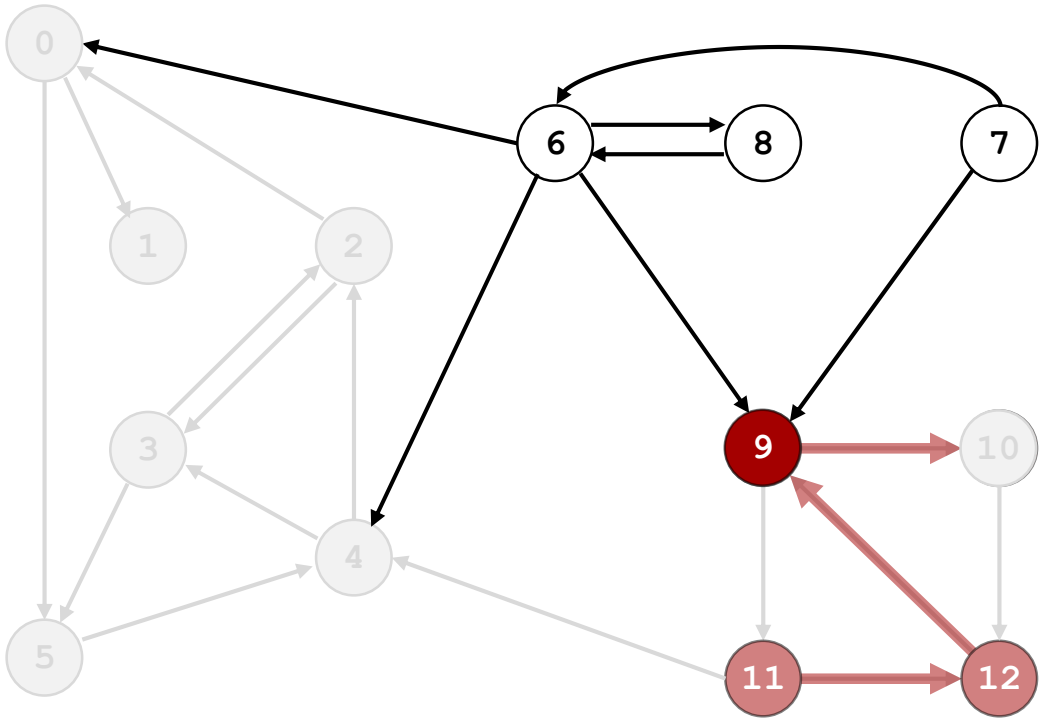
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

Visit 10: check 12

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



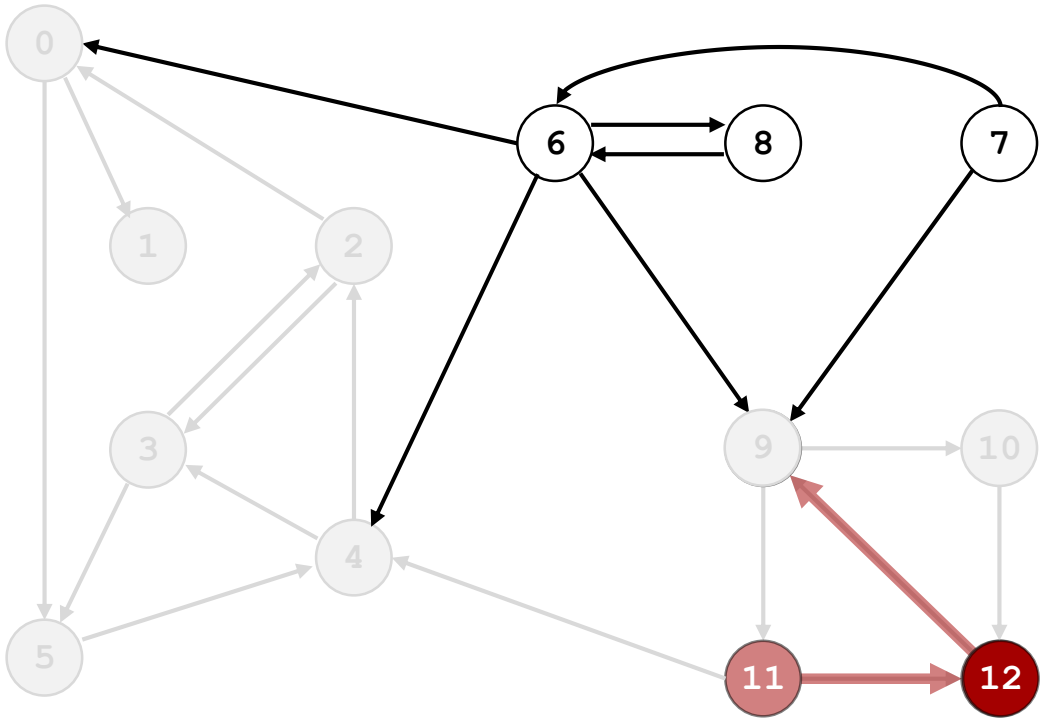
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

10 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



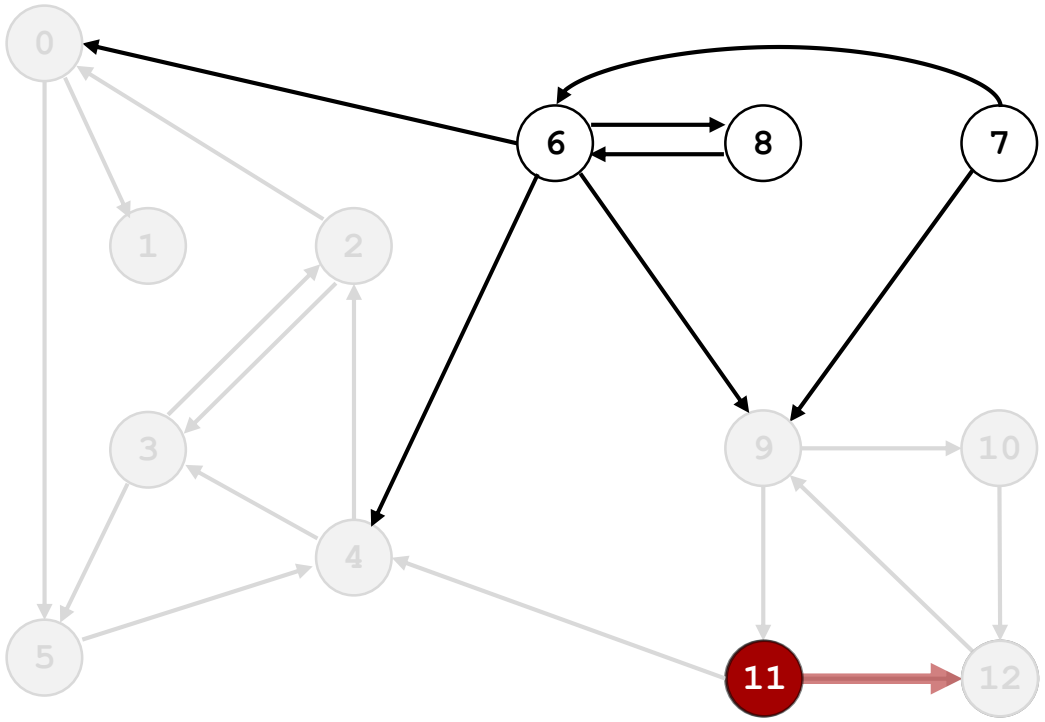
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

9 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



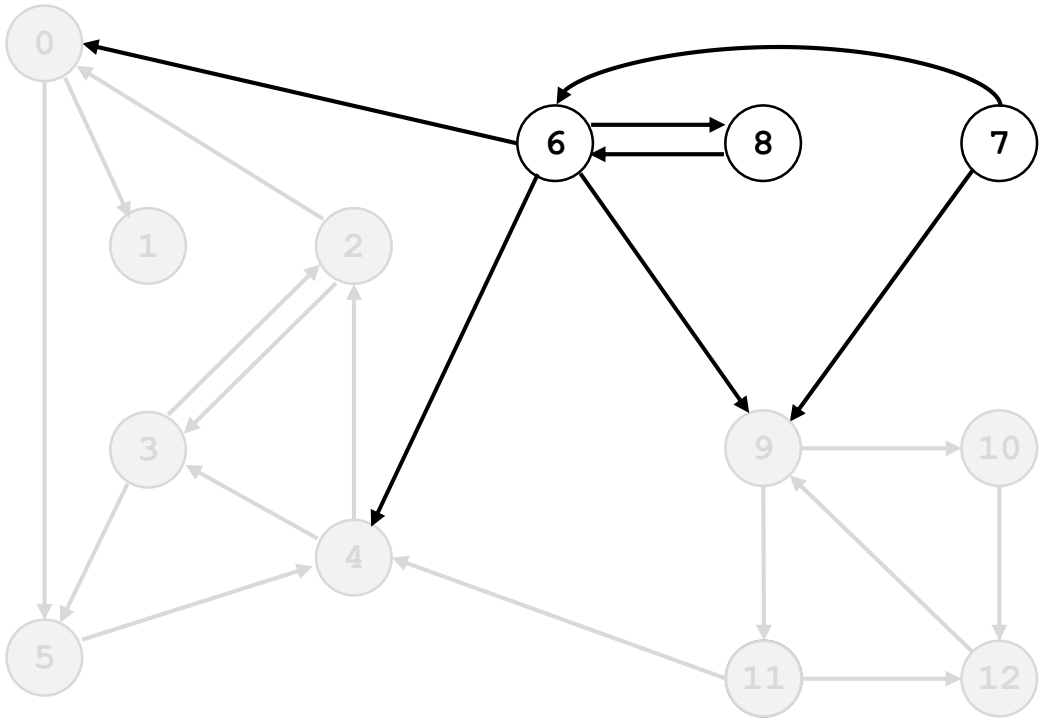
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

12 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



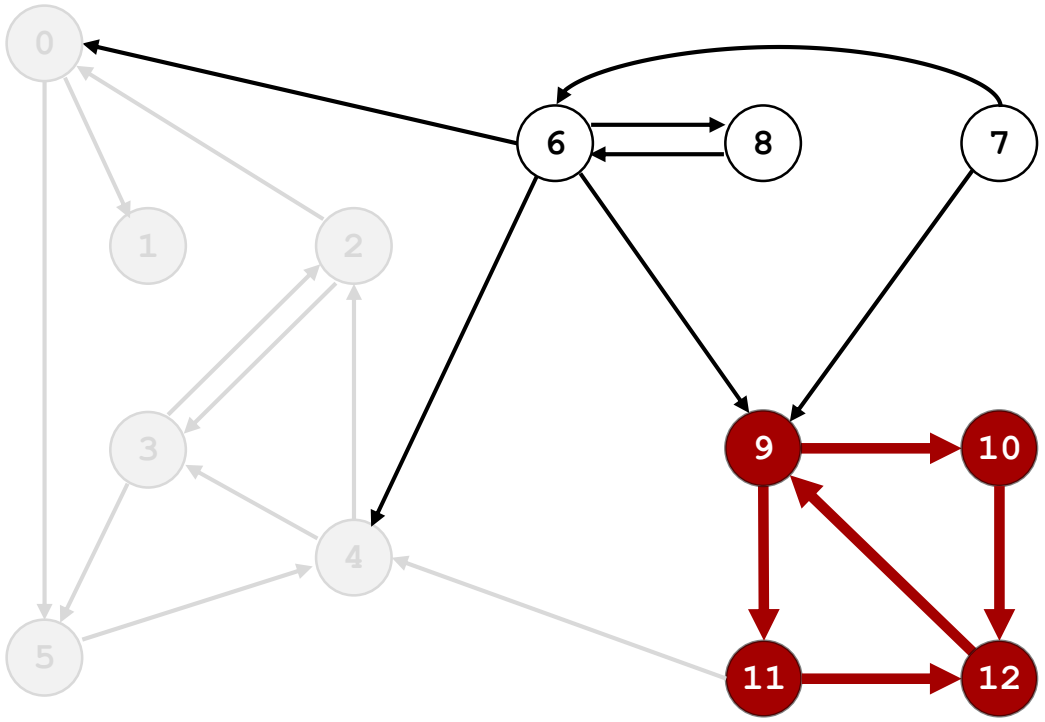
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

11 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 **11** 9 12 10 6 7 8



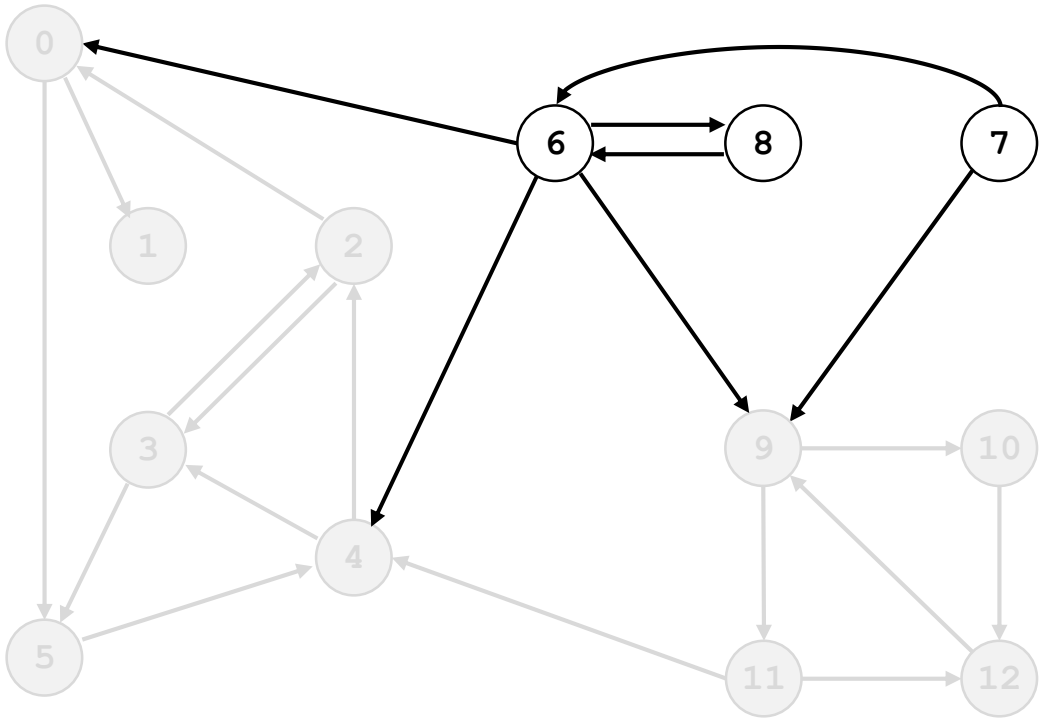
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

Strong component: 9 10 11 12

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 **9** 12 10 6 7 8



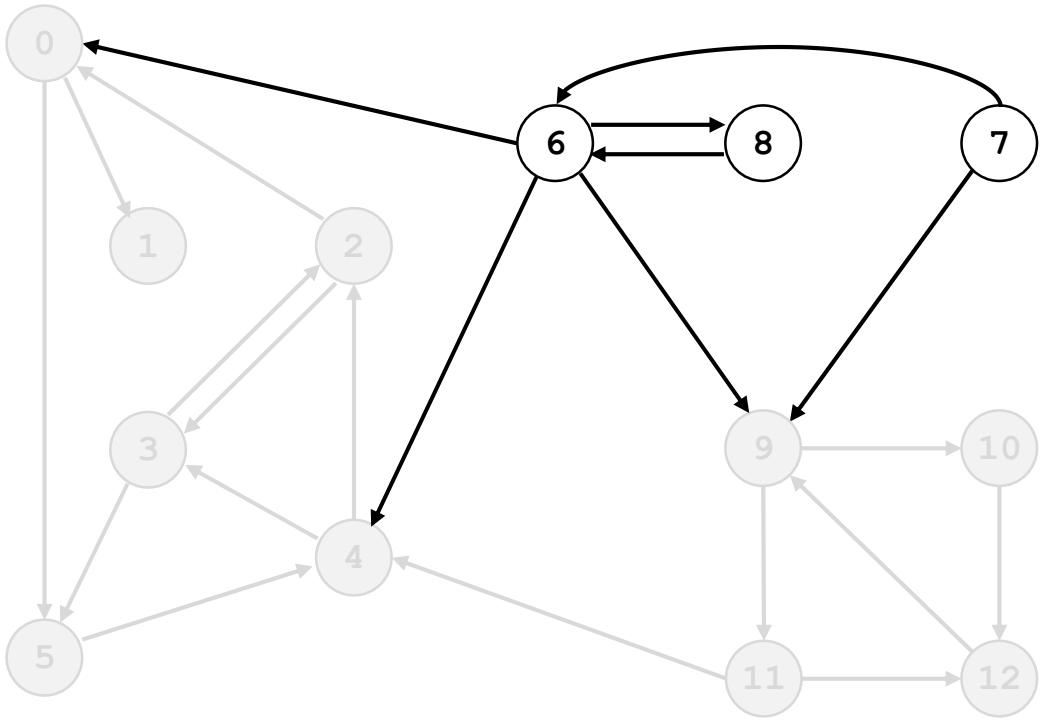
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

check 9

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



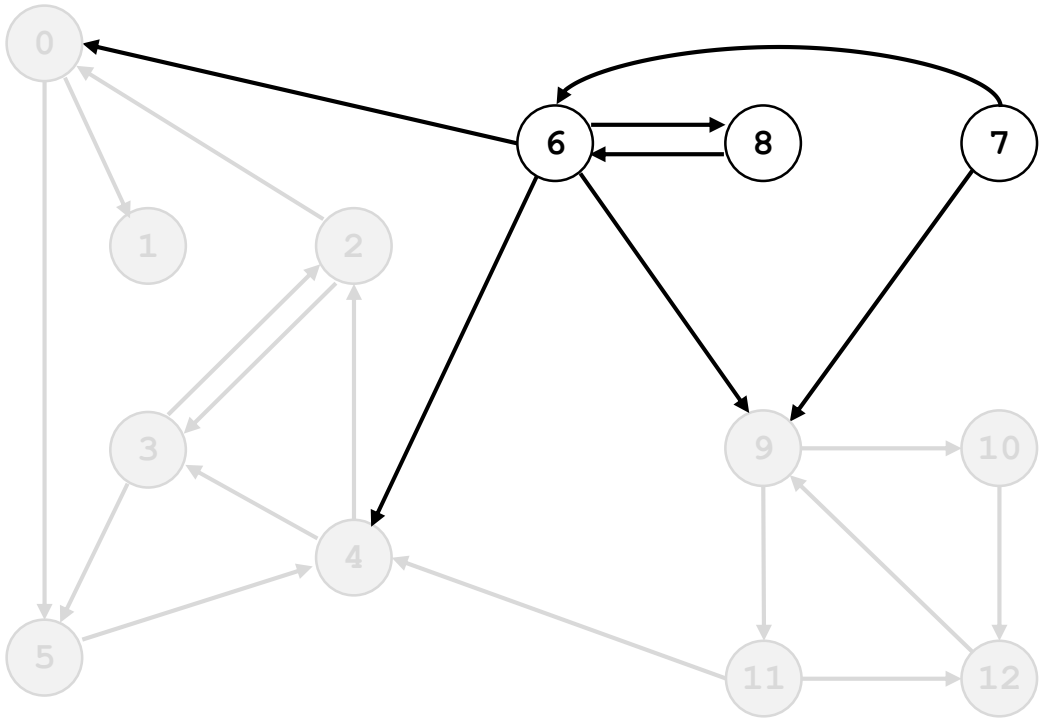
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

check 12

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 **10** 6 7 8



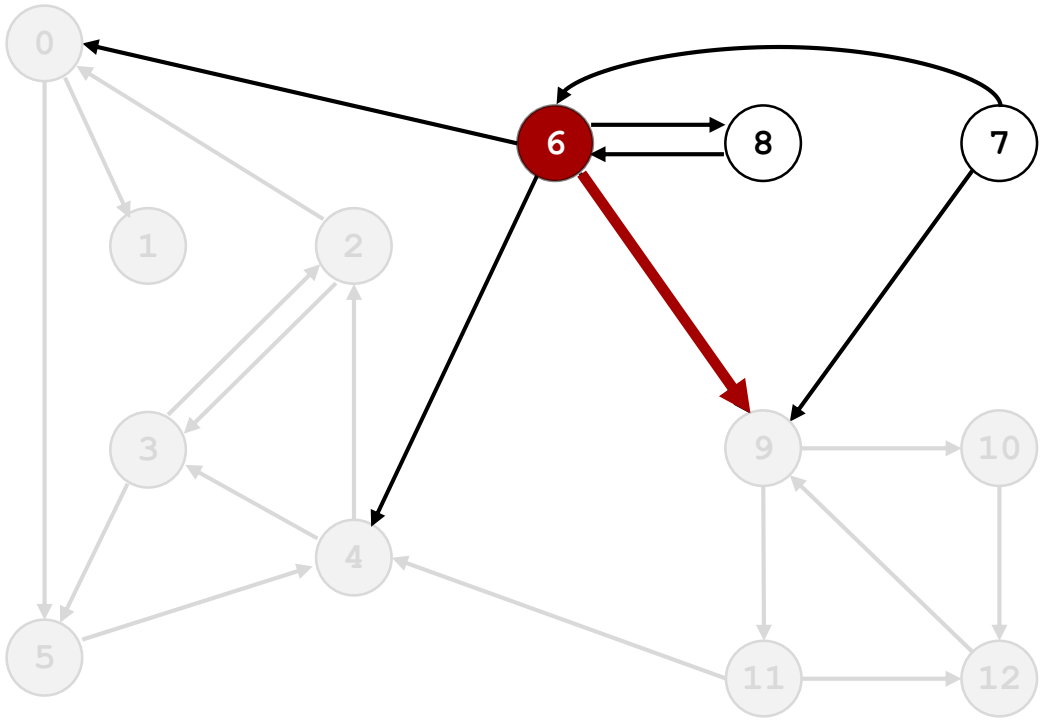
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	-
7	-
8	-
9	2
10	2
11	2
12	2

check 10

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



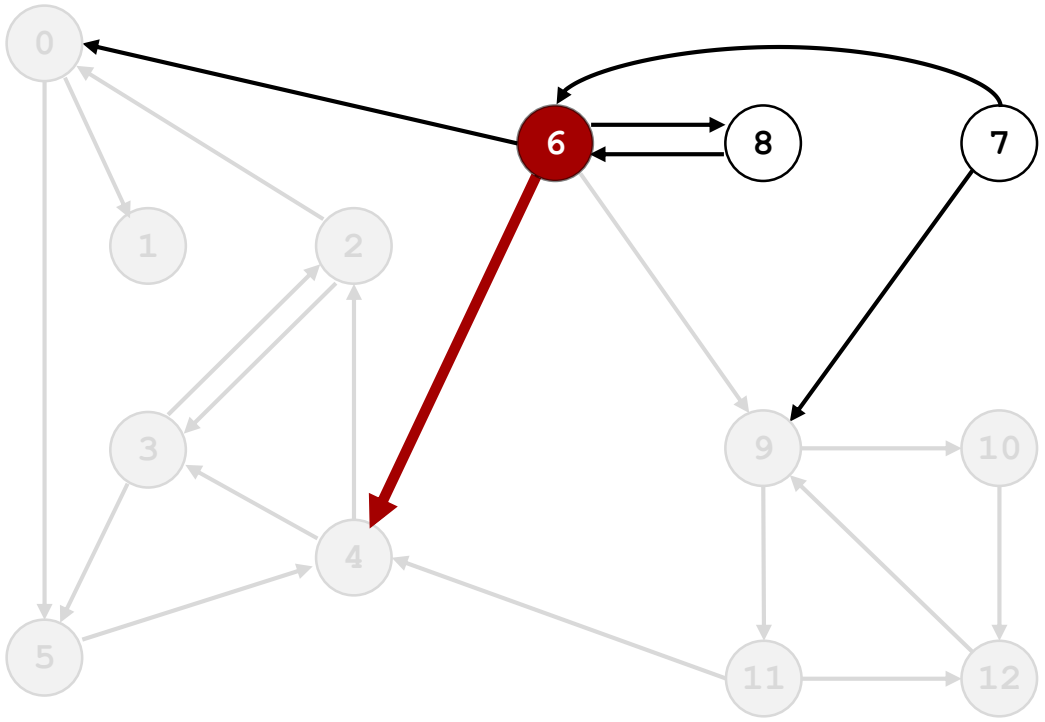
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	-
8	-
9	2
10	2
11	2
12	2

Visit 6: check 9, check 4, check 8, check 0

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



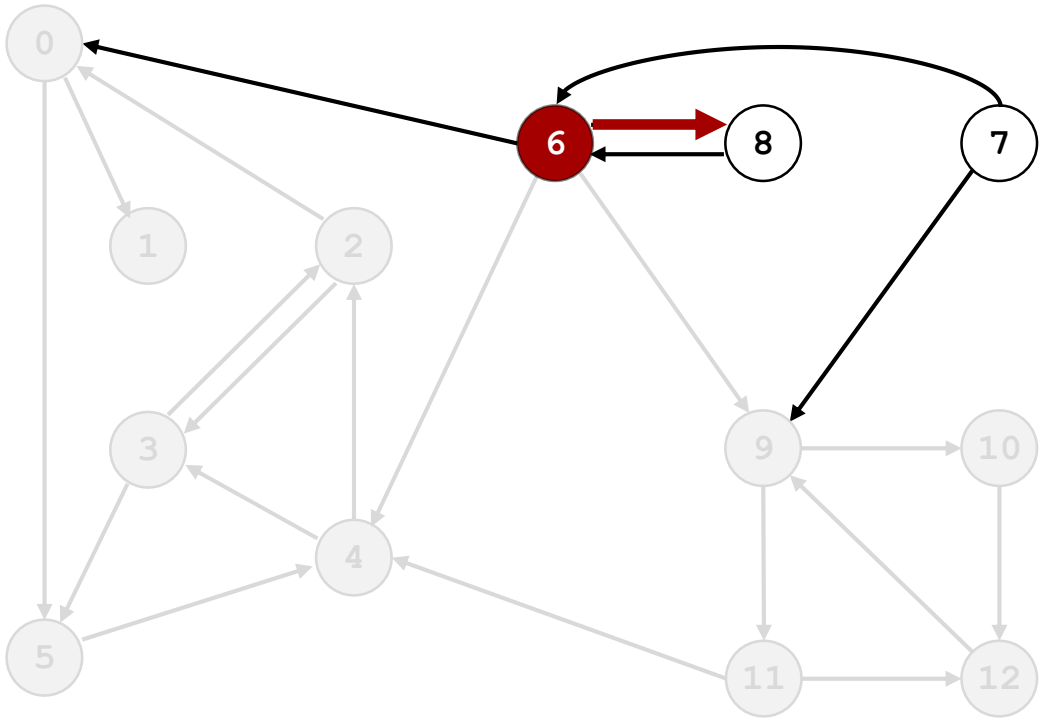
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	-
8	-
9	2
10	2
11	2
12	2

Visit 6: check 9, **check 4**, check 8, check 0

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



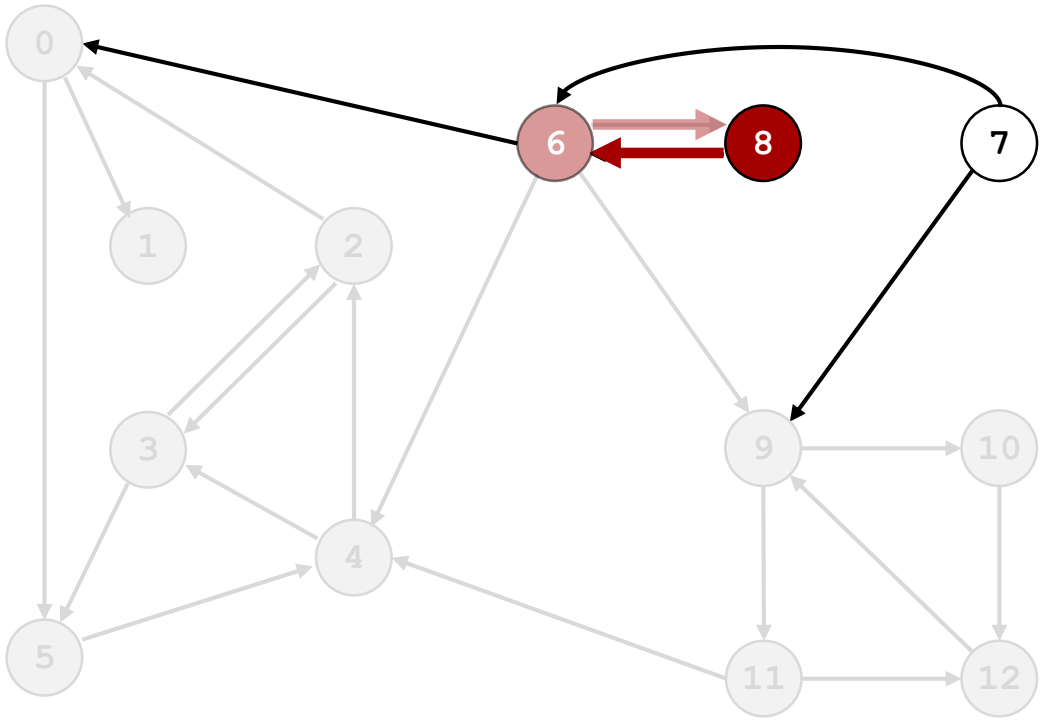
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	-
8	-
9	2
10	2
11	2
12	2

Visit 6: check 9, check 4, **check 8**, check 0

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



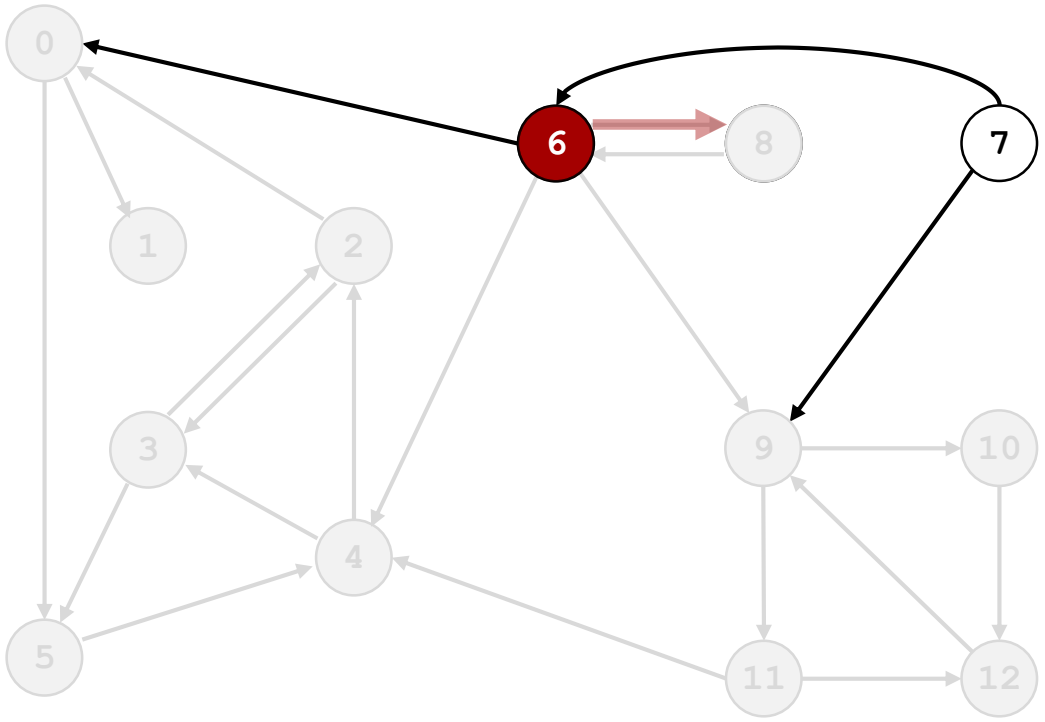
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	-
8	3
9	2
10	2
11	2
12	2

Visit 8: check 6

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



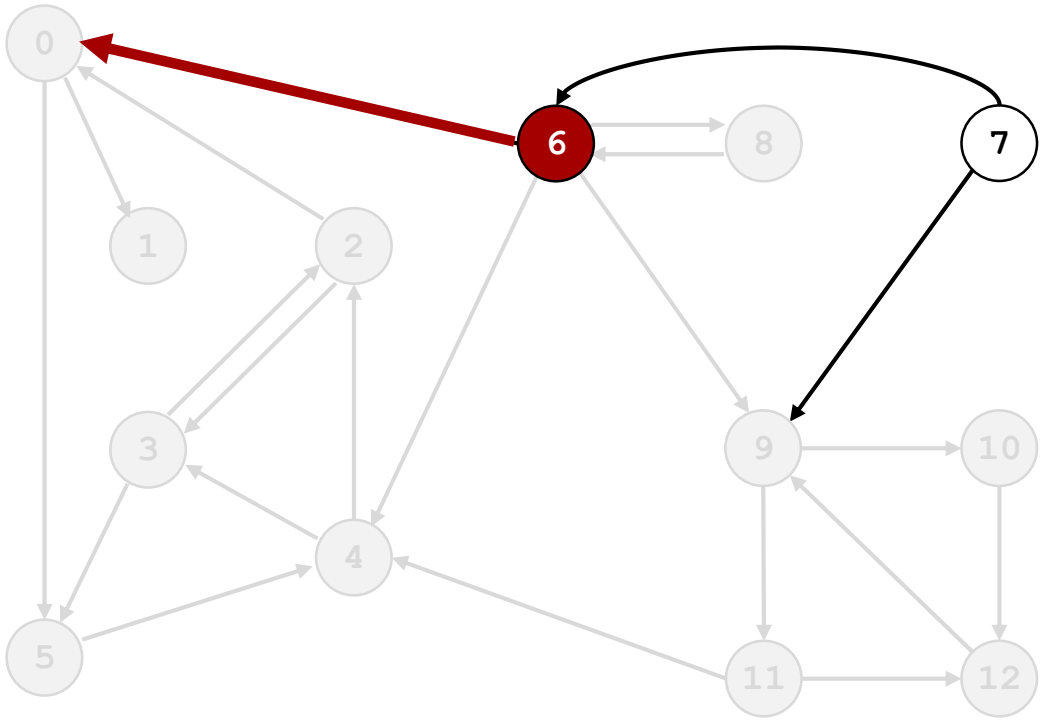
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	-
8	3
9	2
10	2
11	2
12	2

8 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



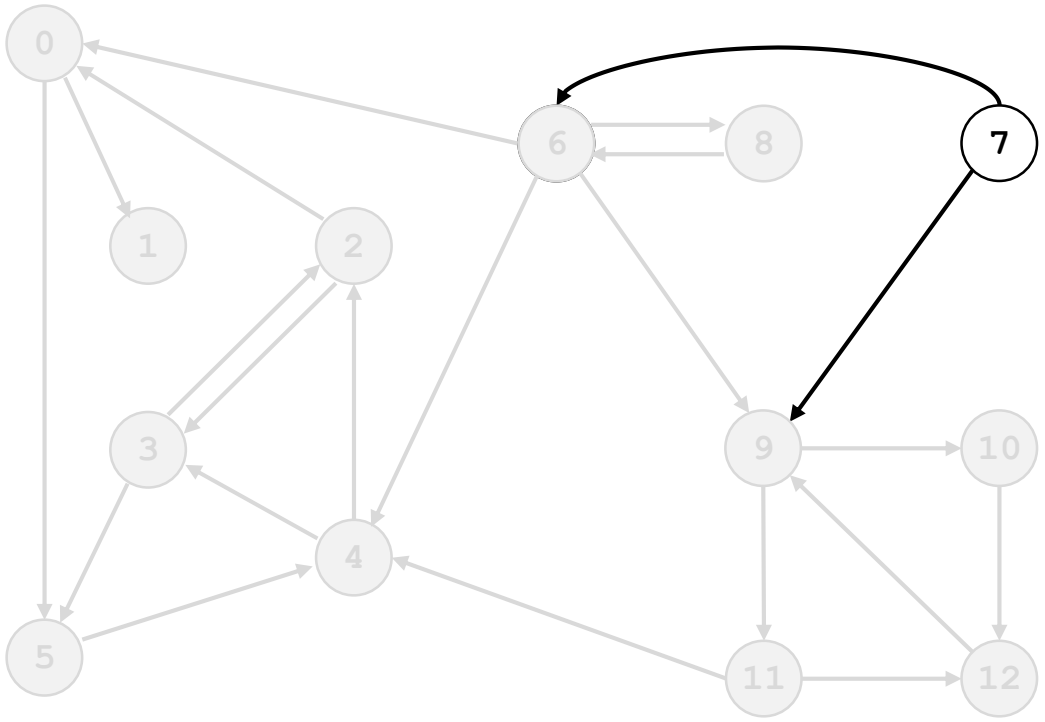
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	-
8	3
9	2
10	2
11	2
12	2

Visit 6: check 9, check 4, check 8, **check 0**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



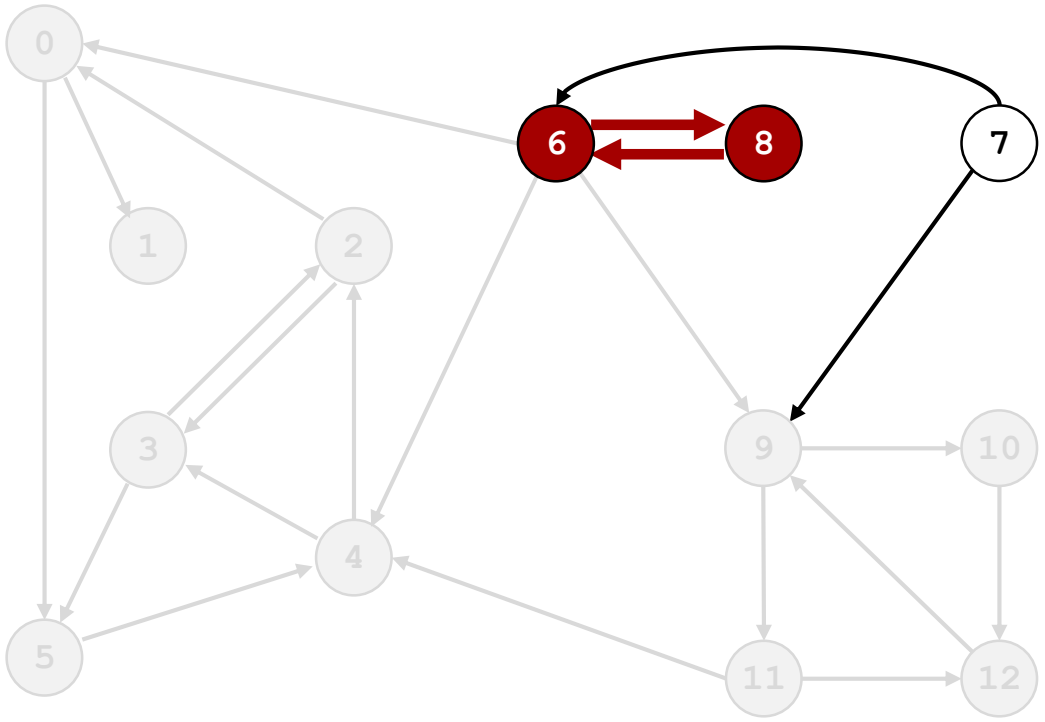
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	-
8	3
9	2
10	2
11	2
12	2

6 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



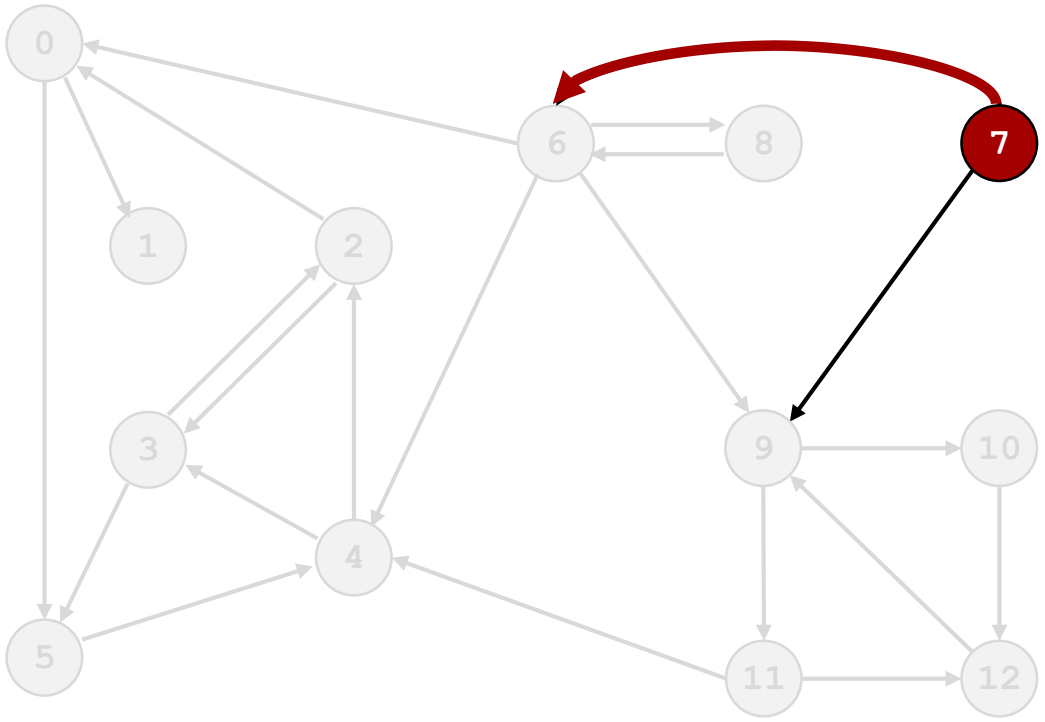
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	-
8	3
9	2
10	2
11	2
12	2

```
Strong component: 6 8
```

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 **7** 8



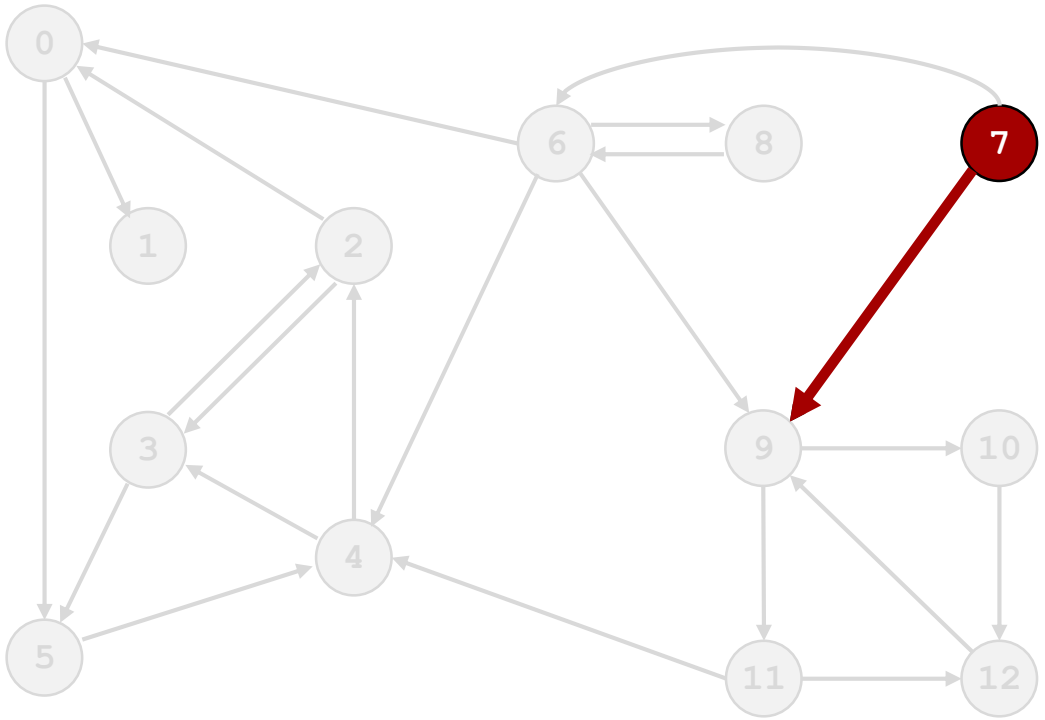
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	4
8	3
9	2
10	2
11	2
12	2

Visit 7: check 6, check 9

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8



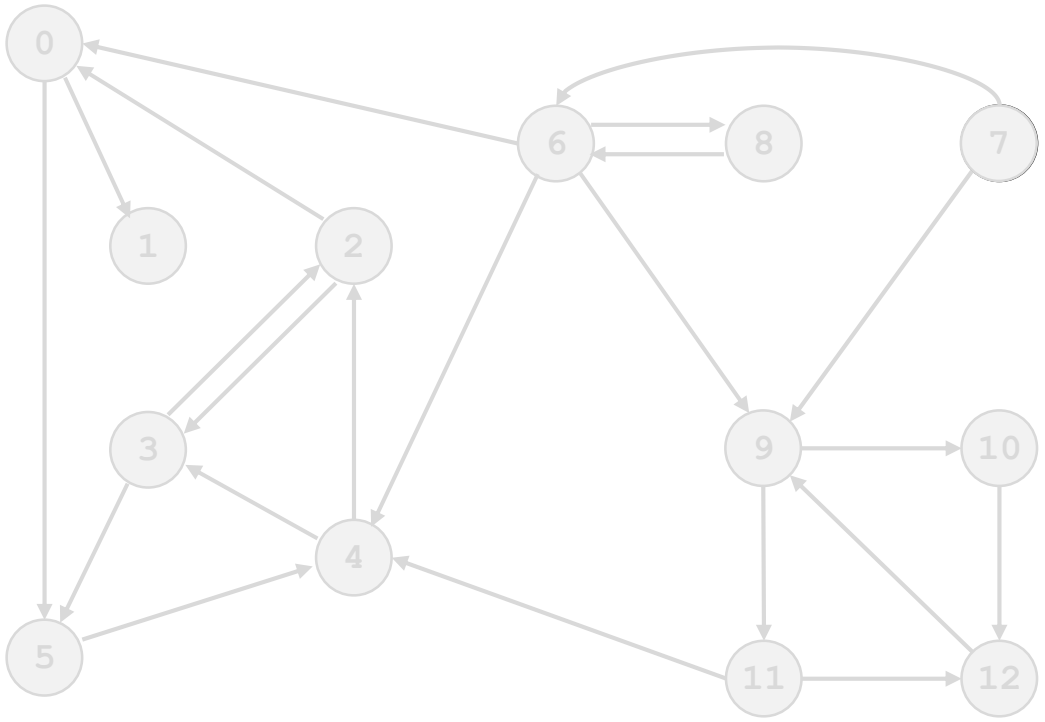
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	4
8	3
9	2
10	2
11	2
12	2

Visit 7: check 6, **check 9**

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 **7** 8



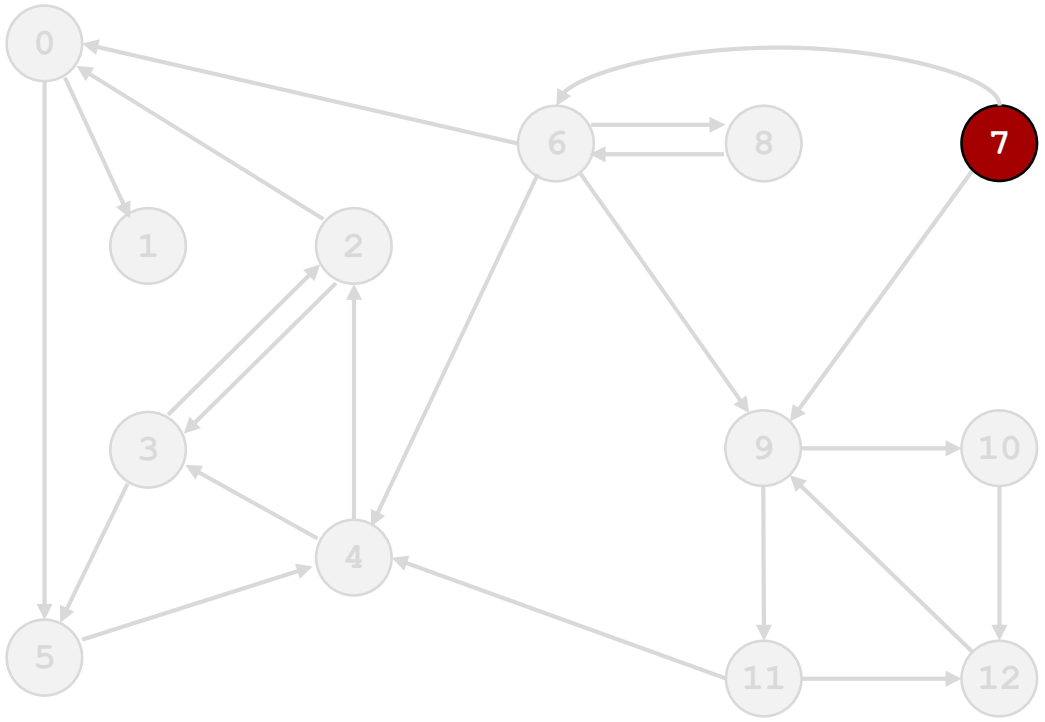
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	**4**
8	3
9	2
10	2
11	2
12	2

7 done

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 **7** 8



v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	**4**
8	3
9	2
10	2
11	2
12	2

```
Strong component: 7
```

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 **8**

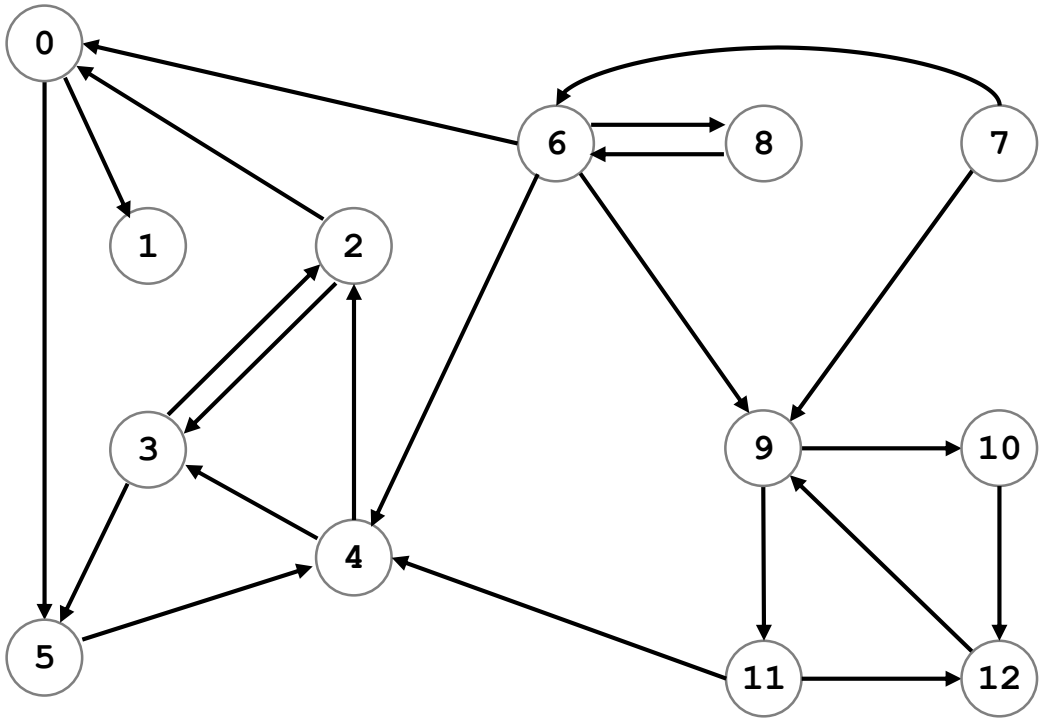
v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	4
8	3
9	2
10	2
11	2
12	2

Check 8

الگوریتم کاساراجو: اجرای نمایشی

- فاز ۲. اجرای جستجوی عمقی بر روی G به ترتیب عکس پس‌ترتیب در G^R

1 0 2 4 5 3 11 9 12 10 6 7 8

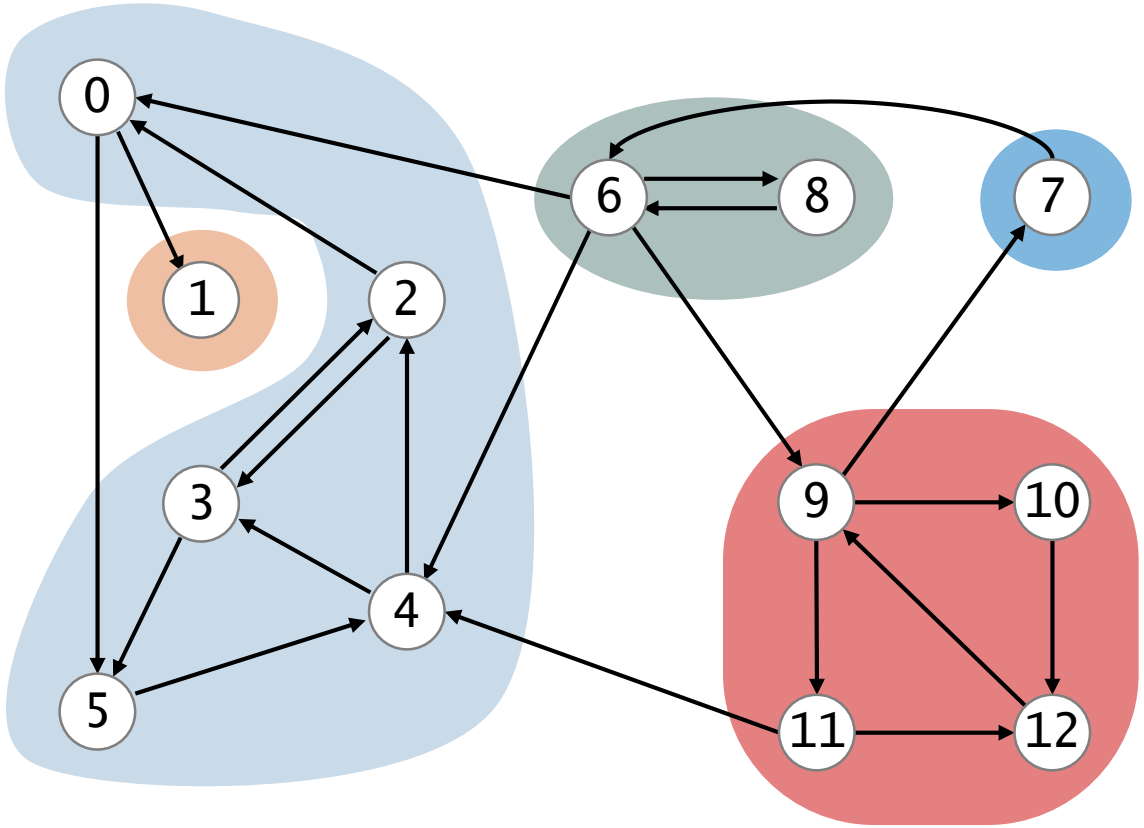


v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	4
8	3
9	2
10	2
11	2
12	2

done

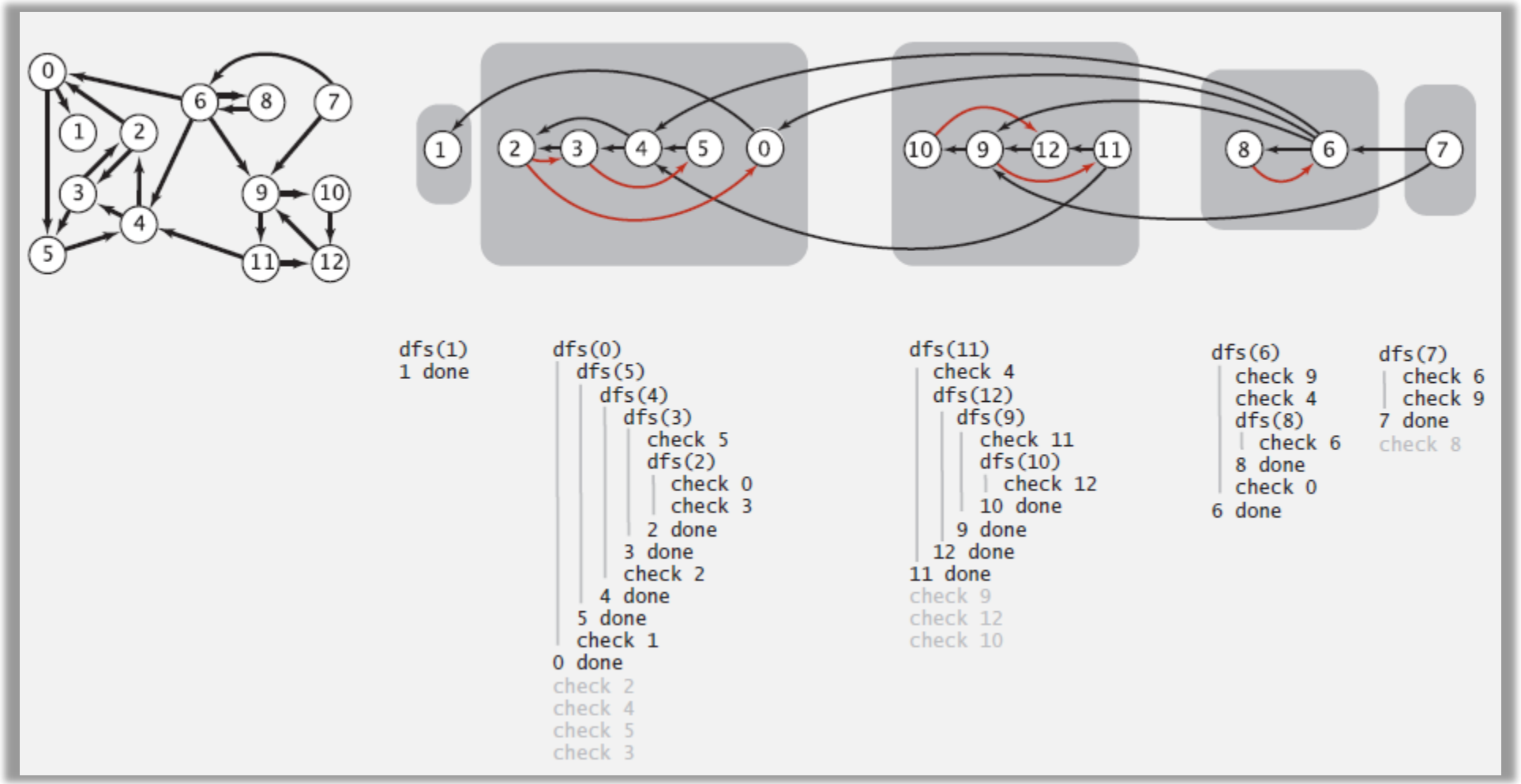
الگوریتم کاساراجو: اجرای نمایشی

- مؤلفه‌های همبند قوی.



v	scc[]
0	1
1	0
2	1
3	1
4	1
5	1
6	3
7	4
8	3
9	2
10	2
11	2
12	2

الگوریتم کاساراجو



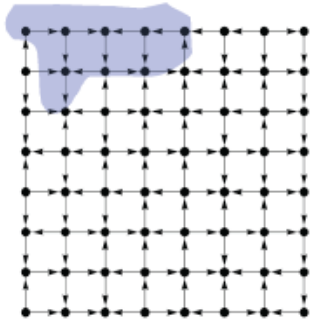
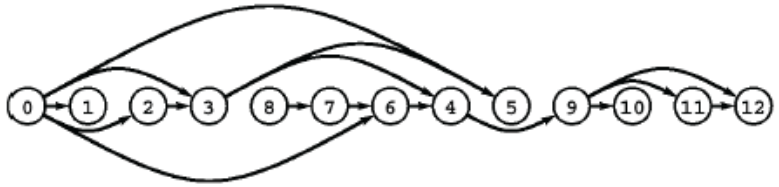
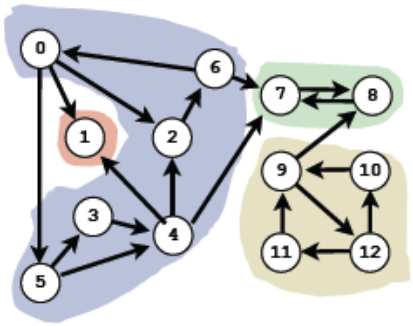
مؤلفه‌های همبند در گراف بدون جهت


```
public class CC
{
   private boolean[] marked;
   private int[] id;
   private int count;

   public CC(Graph G) {
      marked = new boolean[G.V()];
      id     = new int[G.V()];

      for (int v = 0; v < G.V(); v++) {
         if (!marked[v]) {
            dfs(G, v);
            count++;
         }
      }
   }

   private void dfs(Graph G, int v) {
      marked[v] = true;
      id[v] = count;
      for (int w : G.adj(v))
         if (!marked[v]) dfs(G, w);
   }

   public boolean connected(int v, int w)
   {  return id[v] == id[w];  }
}
```

مؤلفه‌های همبند قوی در گراف جهت‌دار


```
public class KosarajuSCC
{
   private boolean[] marked;
   private int[] id;
   private int count;

   public KosarajuSCC(Digraph G)
   {
      marked = new boolean[G.V()];
      id     = new int[G.V()];
      DepthFirstOrder dfs = new DepthFirstOrder(G.reverse());
      for (int v : dfs.reversePost())
      {
         if (!marked[v])
         {
            dfs(G, v);
            count++;
         }
      }
   }

   private void dfs(Digraph G, int v)
   {
      marked[v] = true;
      id[v] = count;
      for (int w : G.adj(v))
         if (!marked[v]) dfs(G, w);
   }

   public boolean stronglyConnected(int v, int w)
   {  return id[v] == id[w];  }
}
```

الگوریتم‌های پردازش گراف جهت‌دار: خلاصه

دسترس پذیری تک مبدأ		جستجوی عمقی
مرتب‌سازی توپولوژیکی		جستجوی عمقی
مؤلفه‌های همبند قوی		جستجوی عمقی (دو بار)