گراف (بدون جهت)

سید ناصر رضوی **www.snrazavi.ir**
۱۳۹۵

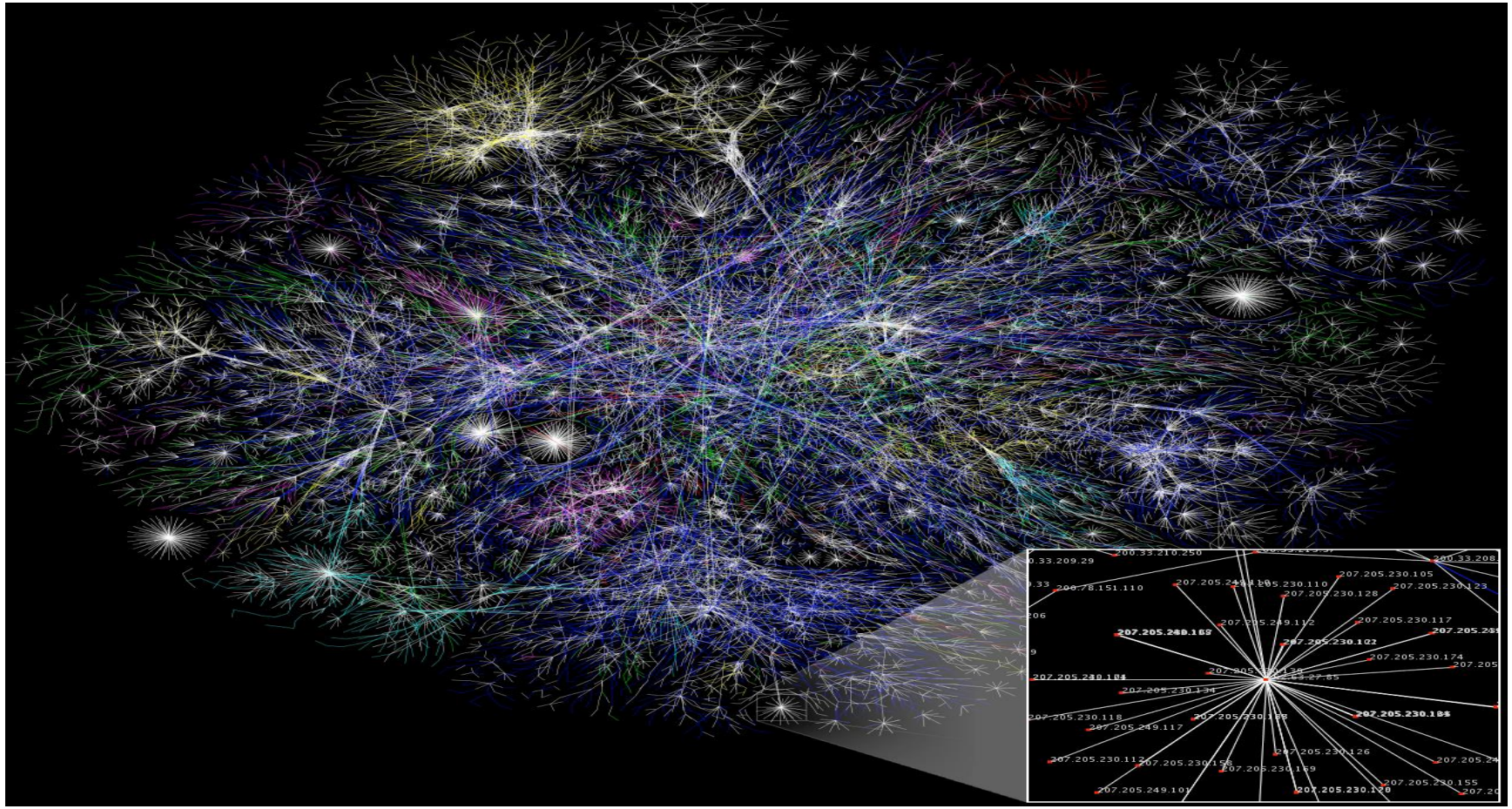
گراف بدون جهت

- گراف.

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

- اهمیت مطالعه‌ی گراف‌ها.
  - هزاران کاربرد عملی
  - صدها الگوریتم شناخته شده
  - یک شاخه‌ی جذاب و چالش برانگیز از علوم کامپیوتر

شبکه‌ی اینترنت



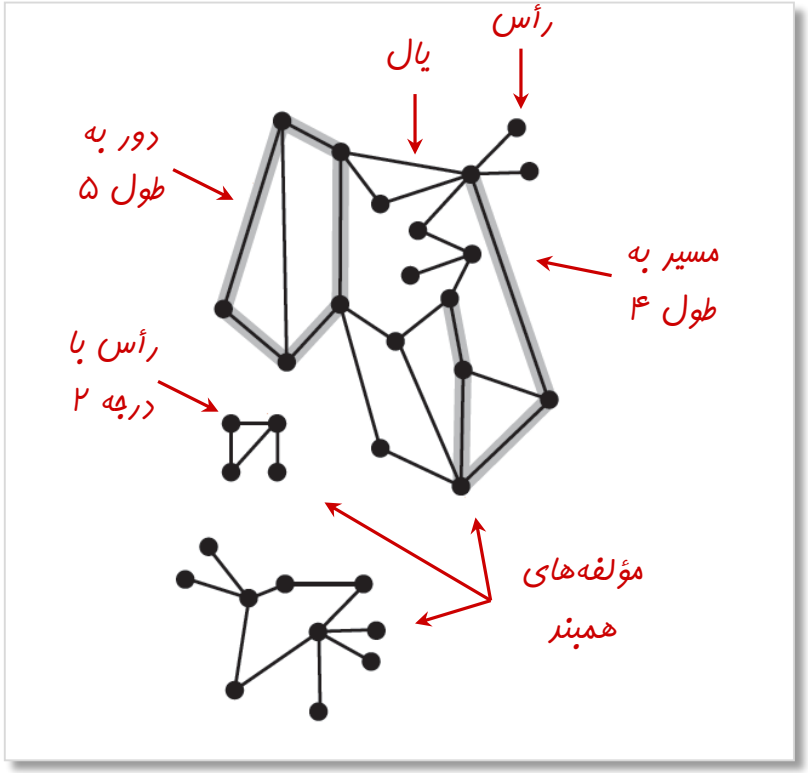
شبکه‌ی فیس‌بوک

کاربردهای گراف

گراف	رأس	یال
مخابرات	تلفن، کامپیوتر	کابل و فیبر نوری
مدار	گیت، ثبات، پردازنده	سیم
علوم مالی	سهام، ارز	تعاملات
حمل و نقل	تقاطع، فرودگاه، مترو	خیابان، خطوط هوایی و زیرزمینی
اینترنت	شبکه‌های محلی	اتصالات
بازی	وضعیت صفحه	حرکات قانونی
روابط اجتماعی	فرد	دوستی
شبکه‌های عصبی	نورون	سیناپس
شبکه‌ی پروتئین	پروتئین	تعامل بین پروتئین‌ها

اصطلاحات

- مسیر. دنباله‌ای از رئوس متصل به وسیله‌ی یال‌ها.
  - دو رأس را متصل گوییم اگر بین آن دو رأس مسیری وجود داشته باشد.
- دور (چرخه). مسیری که در آن رأس شروع و پایان یکسان باشند.



چند مسئله در رابطه با گراف‌ها

- مسیر. آیا بین دو رأس **s** و **t** مسیری وجود دارد؟
- کوتاه‌ترین مسیر. کوتاه‌ترین مسیر بین دو رأس **s** و **t** کدام است؟

- دور. آیا گراف شامل دور است؟
- تور اویلری. آیا دوری وجود دارد که از هر یال دقیقاً یک بار بگذرد؟
- تور هامیلتونی. آیا دوری وجود دارد که از هر رأس دقیقاً یک بار بگذرد؟

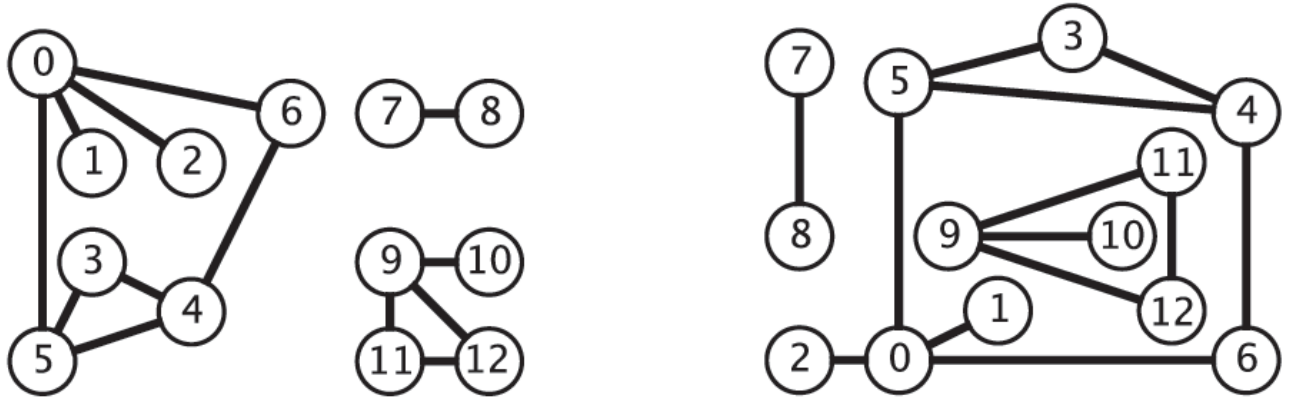
- همبندی. آیا روشی به منظور متصل کردن همه‌ی رئوس وجود دارد؟
- درخت پوشای کمینه. کم هزینه‌ترین روش به منظور متصل کردن همه‌ی رئوس کدام است؟
- دو همبندی. آیا رأسی وجود دارد که حذف آن باعث ناهمبند شدن گراف شود؟

- مسطح بودن. آیا می‌توان گراف را در صفحه طوری ترسیم نمود که هیچ دو یالی همدیگر را قطع نکنند؟
- یکریختی. آیا دو گراف داده شده بیانگر یک گراف هستند؟

واسط گراف

نمایش گراف

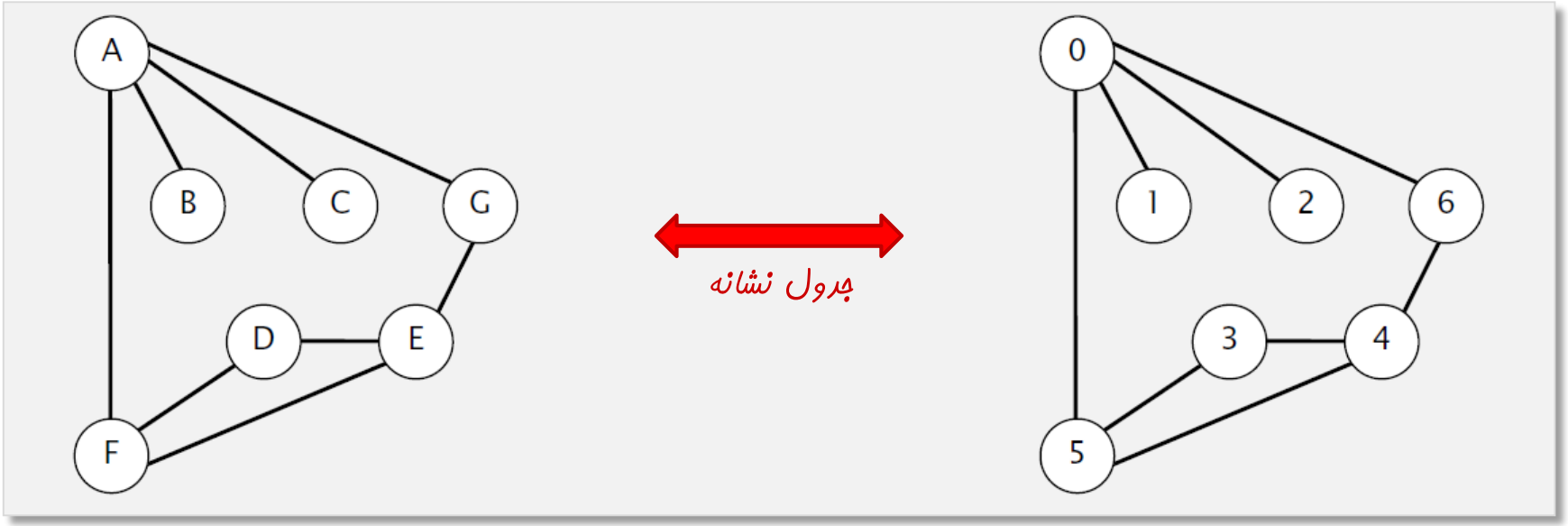
- ترسیم گراف. باعث درک ساختار آن می‌شود.
- هشدار. این درک ممکن است گمراه کننده باشد.



دو ترسیم متفاوت از یک گراف یکسان

نمایش گراف

- نمایش رئوس.
  - در این درس: با استفاده از اعداد صحیح بین ۰ و ۱ – ۷
  - در کاربردهای عملی: تبدیل اسامی رئوس به اعداد صحیح با استفاده از یک جدول نشانه



واسط کاربری گراف

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

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

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

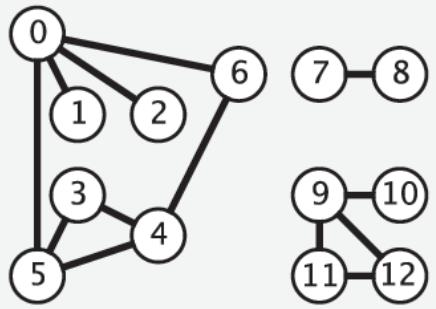
خواندن و ایجاد گراف از جریان ورودی

چاپ یالها (هر یال دو بار)

گراف

tinyG.txt

```
13
13
0 5
4 3
0 1
9 12
6 4
5 4
0 2
11 12
9 10
0 6
7 8
9 11
5 3
```




```
% java Test tinyG.txt
0-6
0-2
0-1
0-5
1-0
2-0
3-5
3-4
...
12-11
12-9
```

```
In in = new In(args[0]);
Graph G = new Graph(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 دو بار)

چند مثال از پردازش گراف


```
public static int degree(Graph G, int v)
{
   int degree = 0;
   for (int w : G.adj(v)) degree++;
   return degree;
}
```

محاسبه‌ی درجه‌ی رأس v
در گراف G

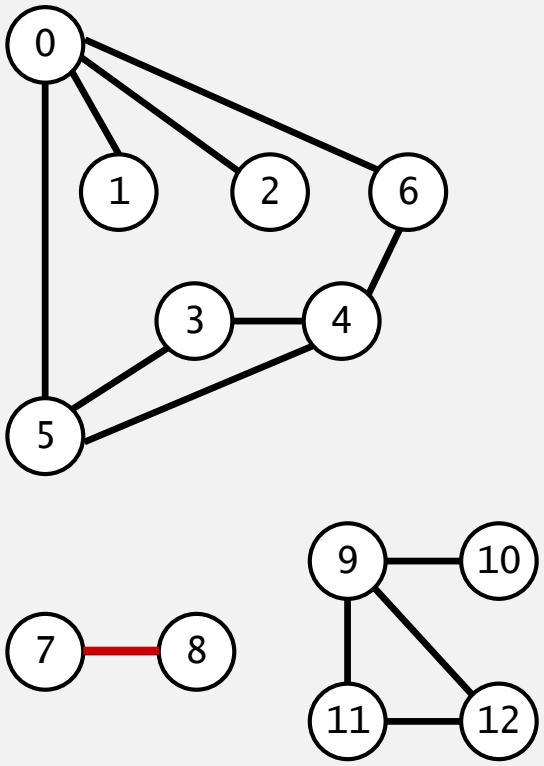
```
public static int maxDegree(Graph G)
{
   int max = 0;
   for (int v = 0; v < G.V(); v++)
      if (degree(G, v) > max)
         max = degree(G, v);
   return max;
}
```

محاسبه‌ی حداکثر درجه‌ی
رئوس در گراف G

```
public static double averageDegree(Graph G)
{
   return 2 * G.E() / G.V();
}
```

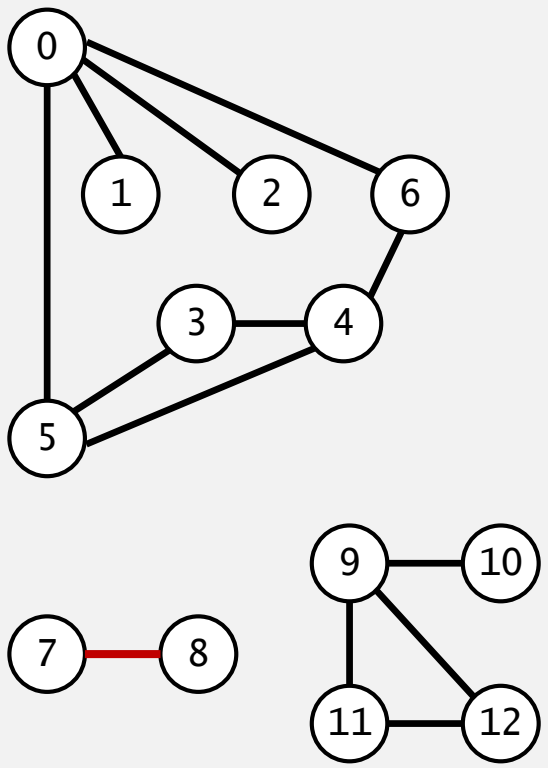
محاسبه‌ی میانگین درجه‌ی
رئوس در گراف G

نمایش گراف: لیست یالها



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

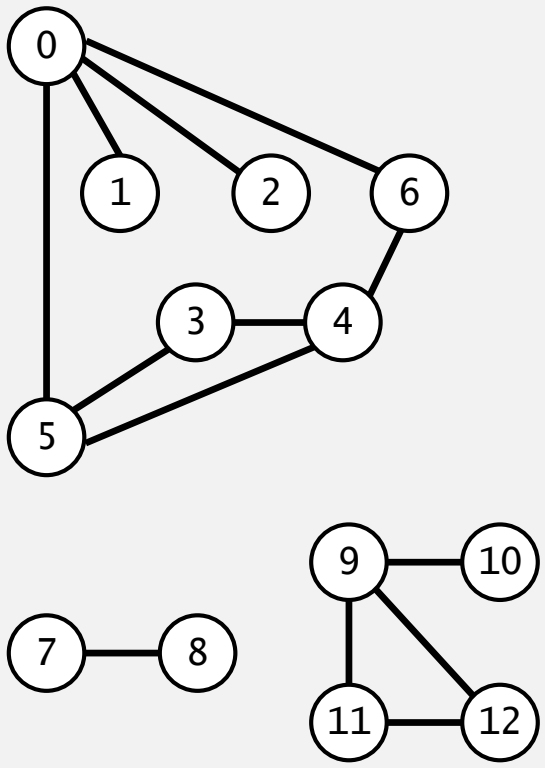
نمایش گراف: ماتریس همسایگی

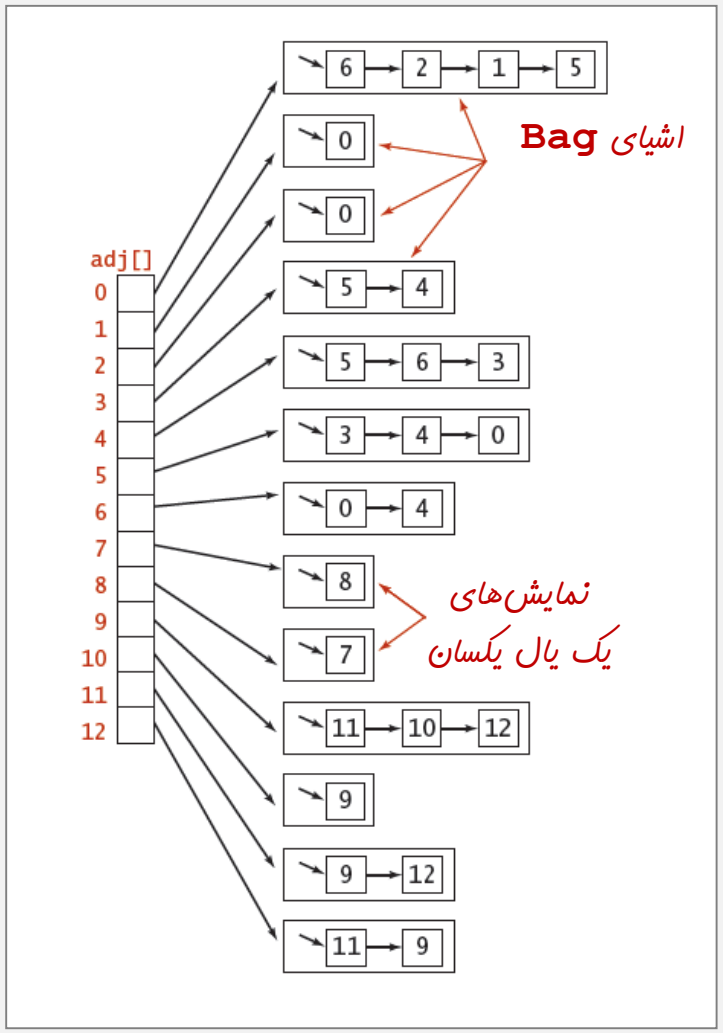


دو مدخل برای
هر یال

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

نمایش گراف: 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

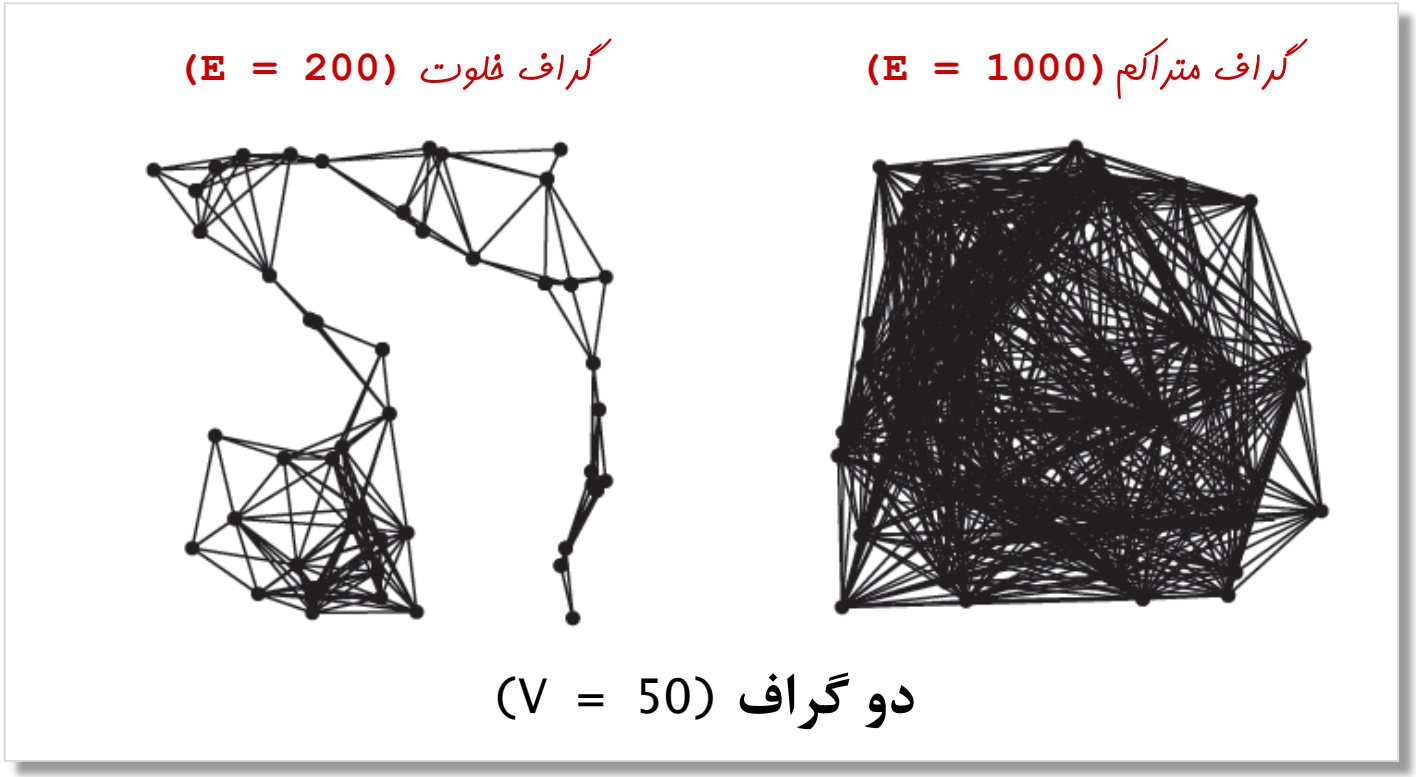
ایجاد یک گراف تهی
شامل V رأس

افزودن یال v-w
(یال‌های موازی مجاز)

تکرارگر برای رئوس
همسایه رأس v

نمایش گراف

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



دو گراف (V = 50)

نمایش گراف

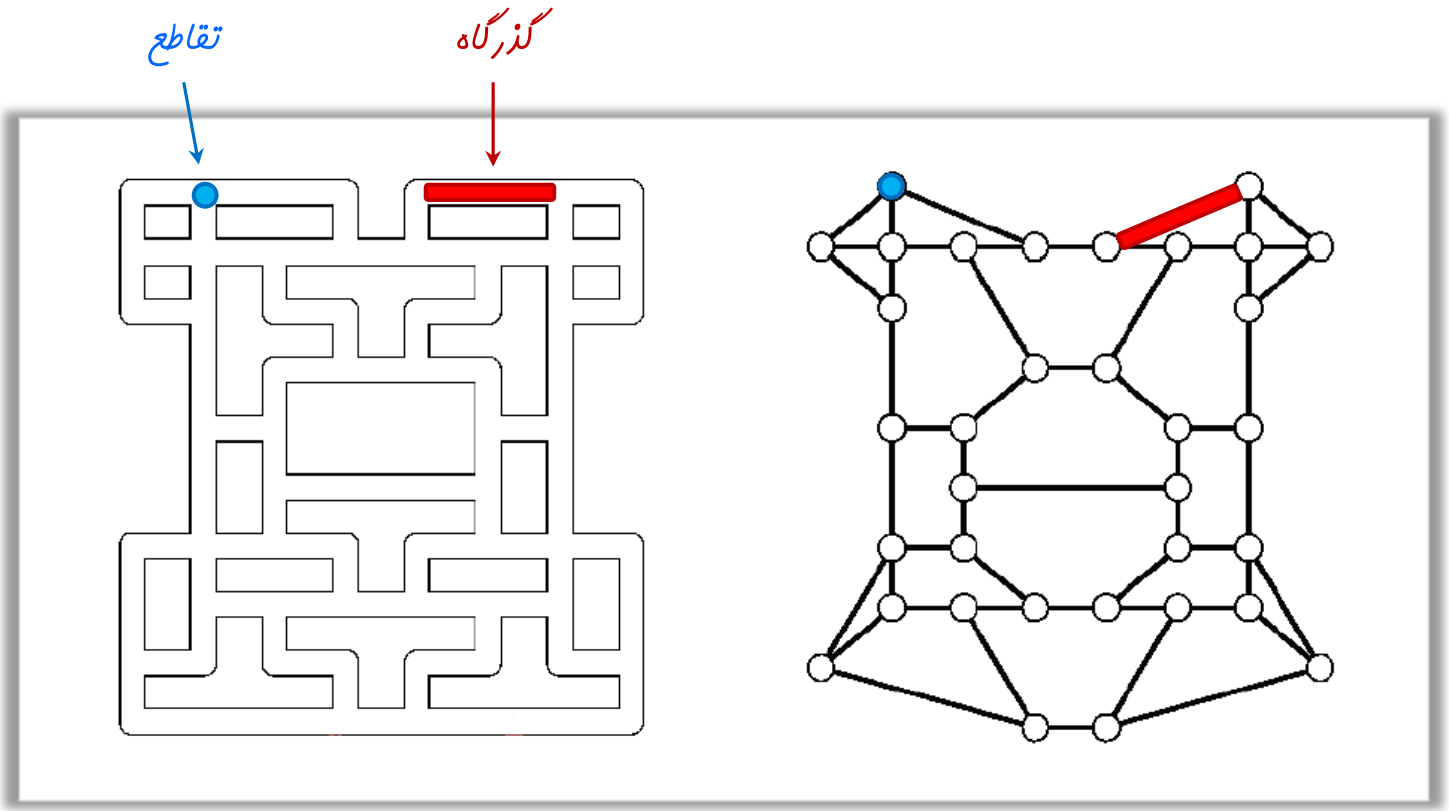
- در عمل. از لیست همسایگی استفاده کنید.
  - الگوریتم‌های گراف بر اساس حلقه زدن بر روی رئوس همسایه‌ی **v** کار می‌کنند.
  - گراف‌ها در دنیای واقعی بیشتر به سمت گراف‌های خلوت تمایل دارند.

روش نمایش گراف	حافظه	افزودن یال	بررسی وجود یال بین رئوس V و W	حلقه زدن روی رئوس مجاور V
لیست یالها	E	1	E	E
ماتریس همسایگی	V^2	1	1	V
لیست همسایگی	E + V	1	degree(v)	degree(v)

جستجوی عمقی

مسئله‌ی مسیر پر پیچ و خم

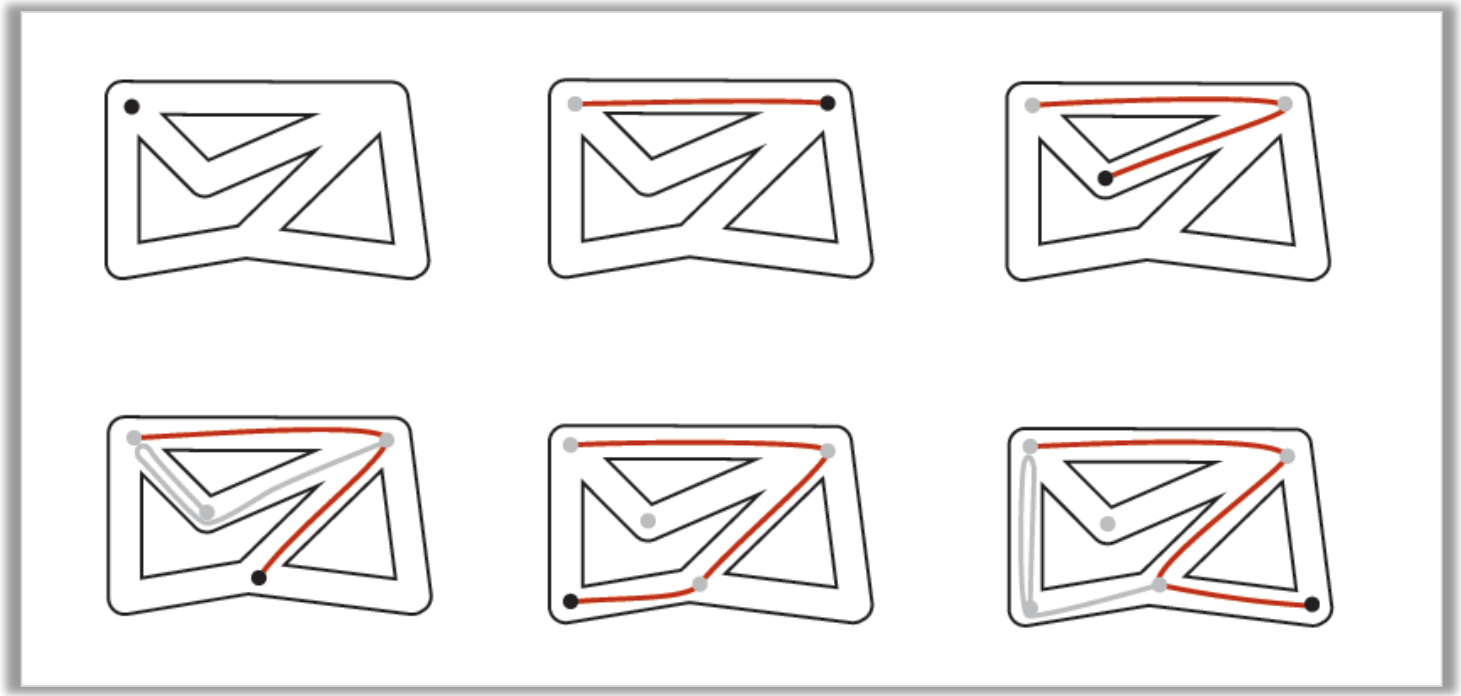
- گراف مسئله‌ی مسیر پر پیچ و خم.
  - رئوس = تقاطع‌ها
  - یال‌ها = گذرگاه‌ها



هدف: کاوش تمامی تقاطع‌ها

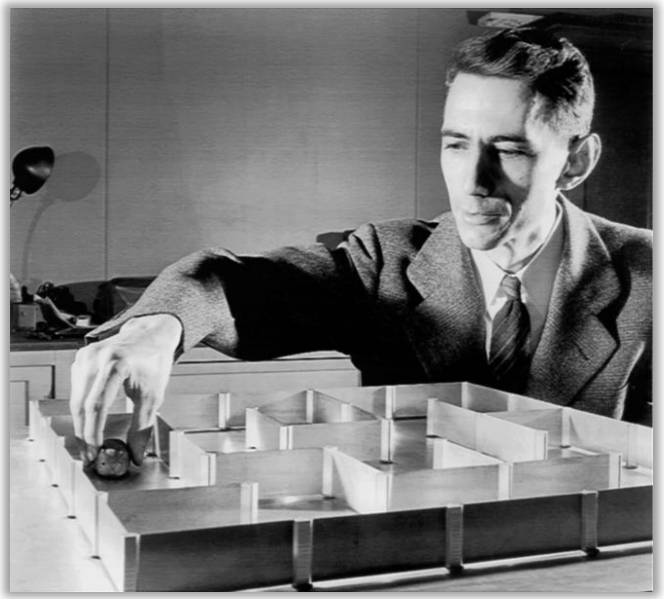
مسئله‌ی مسیر پر پیچ و خم

- الگوریتم.
  - یک قرقره‌ی نخ را به دنبال خود باز کن.
  - تقاطع‌ها و گذرگاه‌های رؤیت شده را علامت گذاری کن.
  - وقتی تقاطع یا گذرگاه رؤیت نشده‌ای وجود ندارد، به عقب برگرد.



مسئله‌ی مسیر پر پیچ و خم

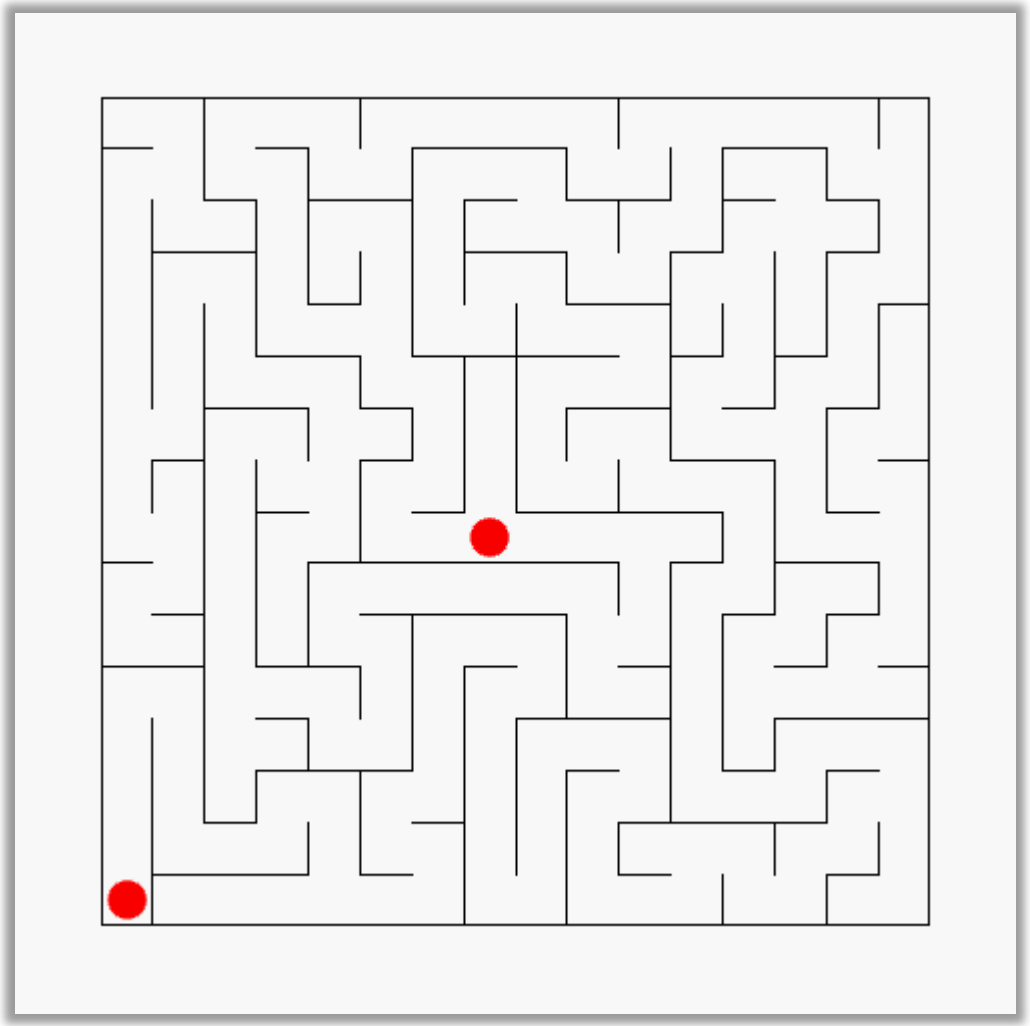
- الگوریتم.
  - یک قرقره‌ی نخ را به دنبال خود باز کن.
  - تقاطع‌ها و گذرگاه‌های رؤیت شده را علامت گذاری کن.
  - وقتی تقاطع یا گذرگاه رؤیت نشده‌ای وجود ندارد، به عقب برگرد.



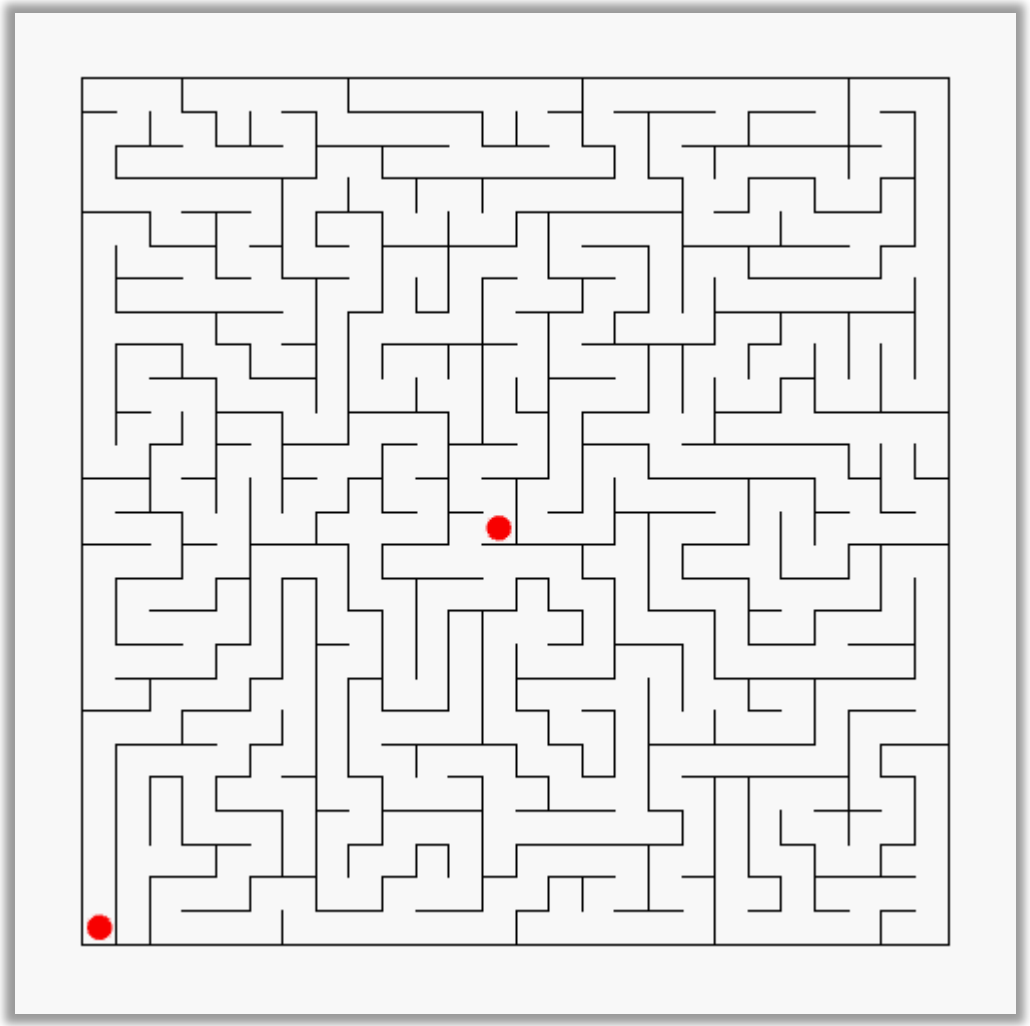
کلود شانون و موش مسیریاب

تسئوس در حال مبارزه با مینوتاوروس در هزار تو در حالی که گلوله نخ در دستان آدریانه است

مسئله‌ی مسیر پر پیچ و خم



مسئله‌ی مسیر پر پیچ و خم



جستجوی عمقی

- هدف. جستجوی یک گراف به صورت منظم
- ایده. تقلید شیوه‌ی کاوش مسیر پر پیچ و خم

DFS (to visit a vertex v)

Mark v as visited.

Recursively visit all unmarked vertices w adjacent to v.

- کاربردهای متداول.
  - یافتن تمامی رئوس متصل به یک رأس مبدأ
  - یافتن مسیر بین دو رأس (در صورت وجود)

الگوی طراحی برای پردازش گراف

- الگوی طراحی. جدا کردن نوع داده‌ای گراف از روال پردازش گراف.
  - یک شیء گراف ایجاد کن.
  - گراف ایجاد شده را به عنوان ورودی به روال پردازش گراف ارسال کن.
  - با استفاده از روال پردازش گراف، اطلاعات مورد نظر خود درباره‌ی گراف را پرس و جو کن.

```
public class Paths
                        Paths(Graph G, int s)
                boolean hasPathTo(int v)
Iterable<Integer>       pathTo(int v)
```

تمام مسیرها را در G از مبدأ s پیدا کن

آیا مسیری از s به v وجود دارد؟

مسیر از s به v را برگردان

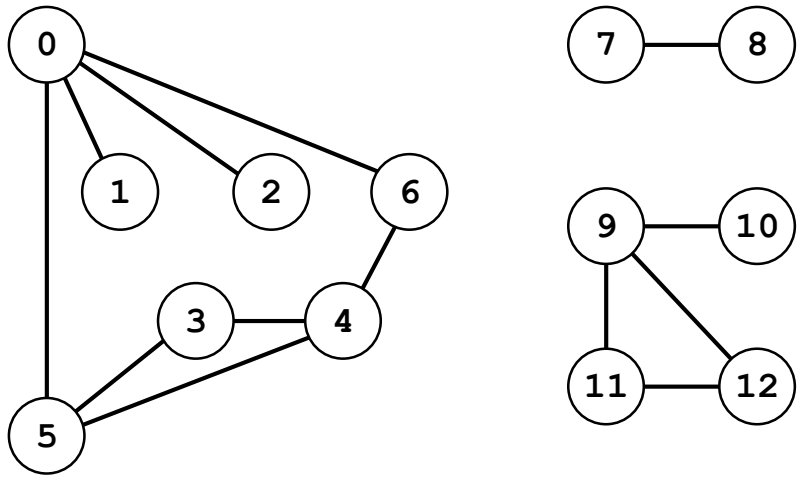
```
Paths paths = new Paths(G, s)
for (int v = 0; v < G.V(); v++)
   if (paths.hasPathTo(v))
      StdOut.println(v);
```

چاپ تمام رئوس متصل به s

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، 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	-

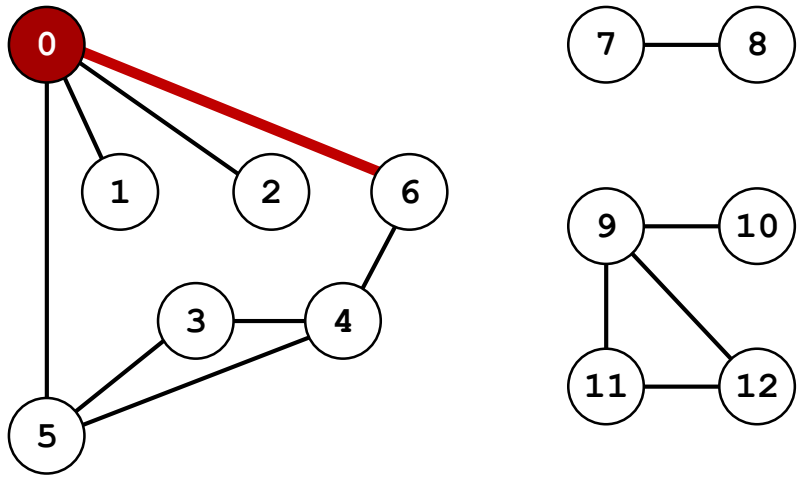


Graph G

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، 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	-

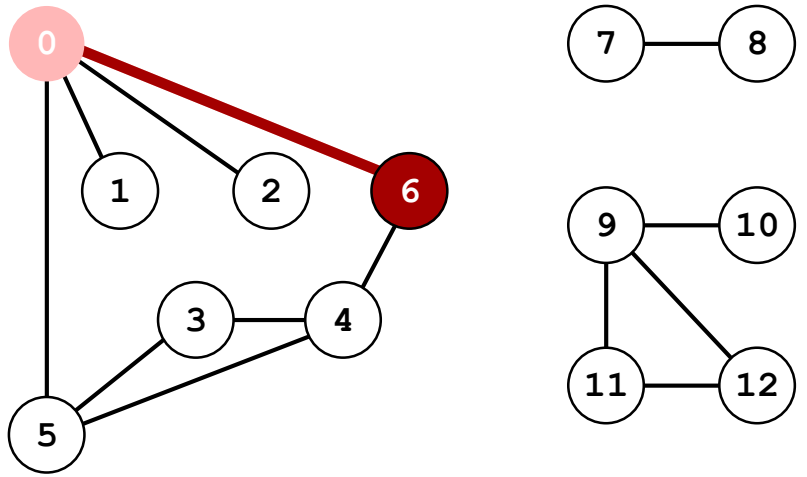



```
visit 0: check 6, check 2, check 1, check 5
```

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

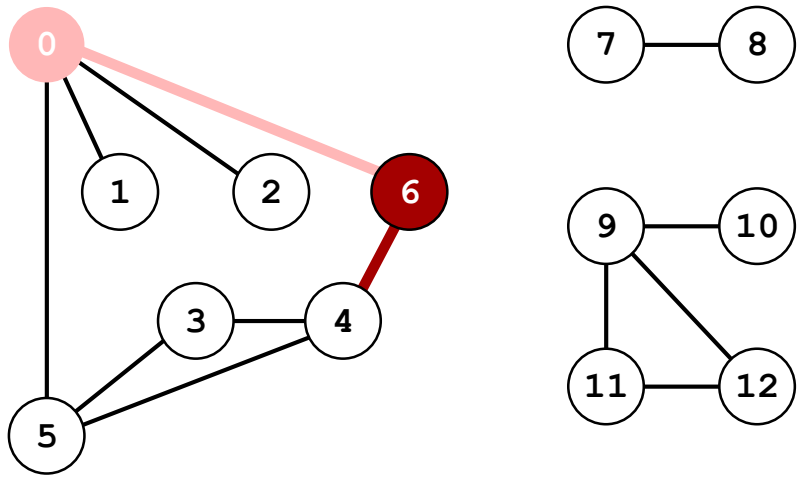


visit 6: check 0, check 4

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

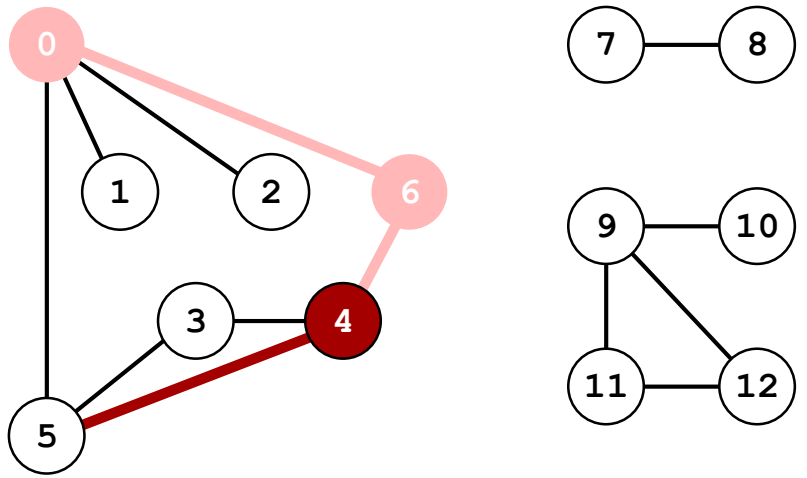


visit 6: check 0, check 4

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

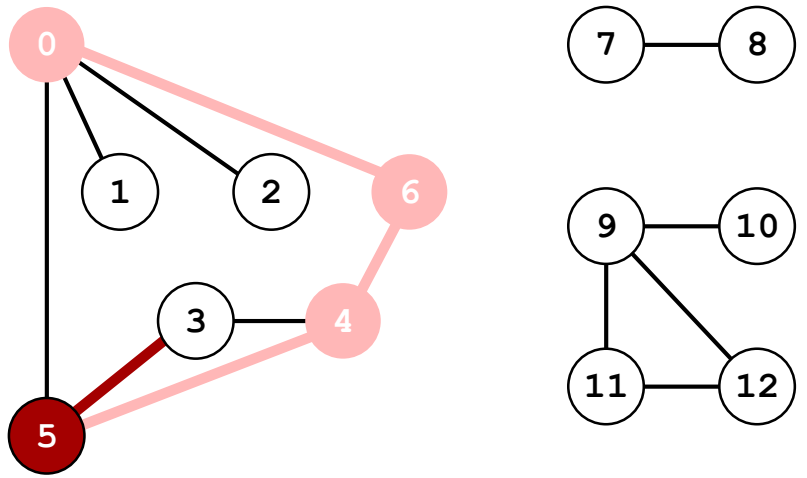


visit 4: check 5, check 3, check 6

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

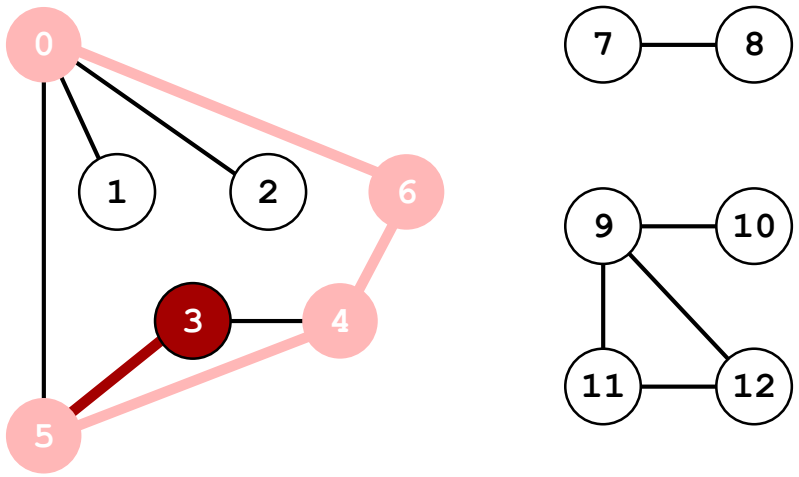


visit 5: check 3, check 4, check 0

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

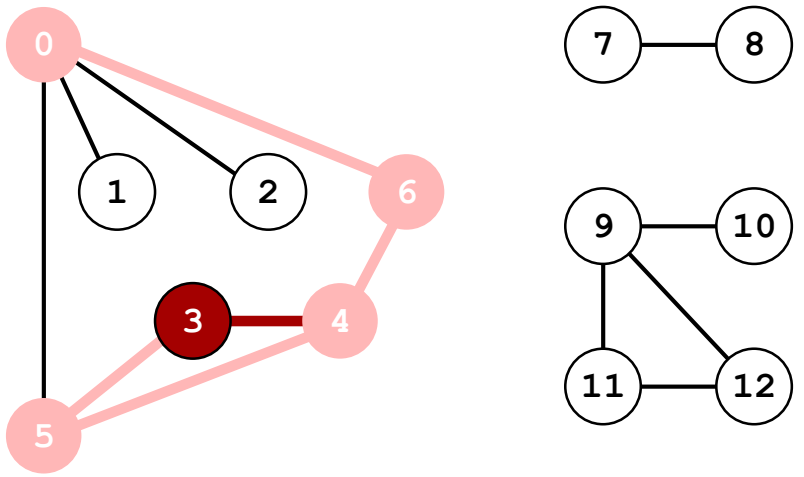


visit 3: check 5, check 4

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

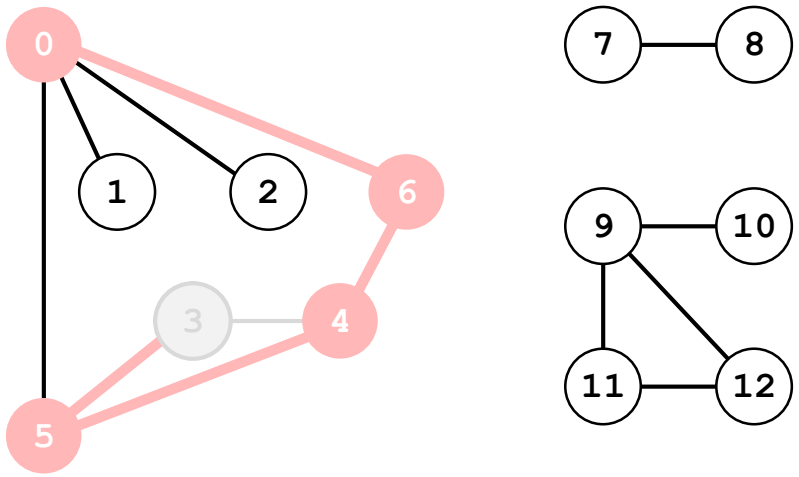


visit 3: check 5, check 4

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

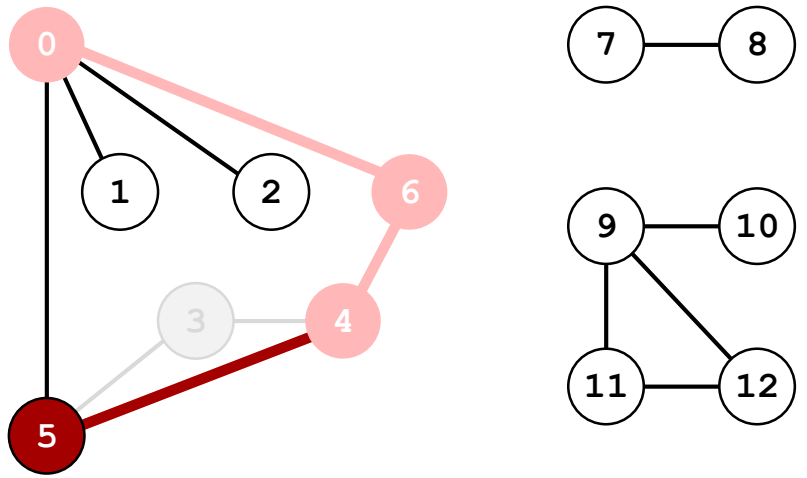


3 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

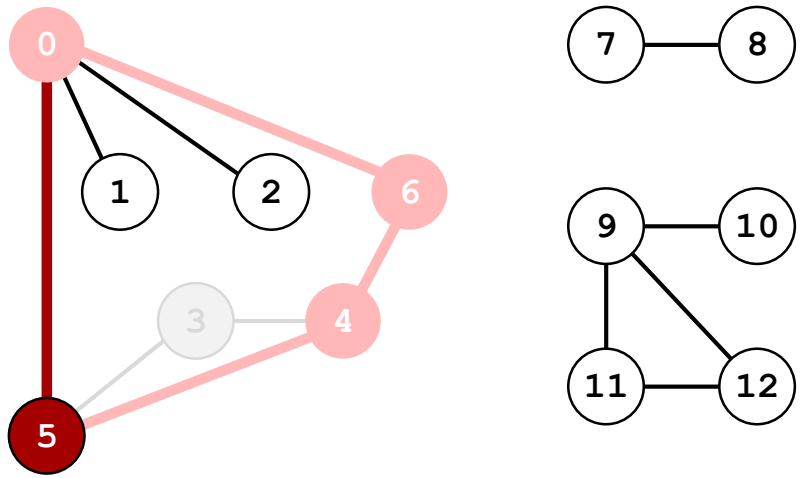


visit 5: check 3, check 4, check 0

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

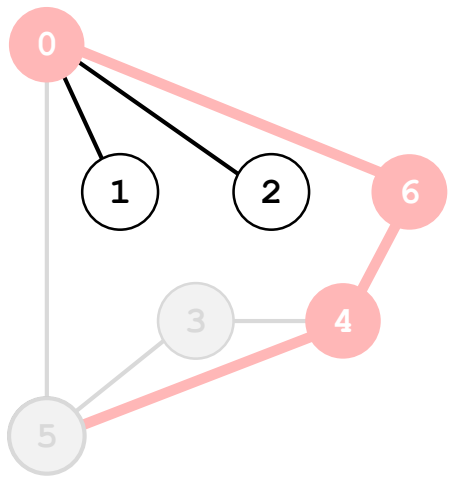


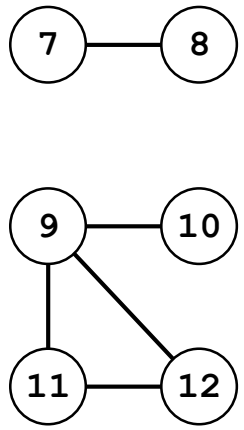
visit 5: check 3, check 4, check 0

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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



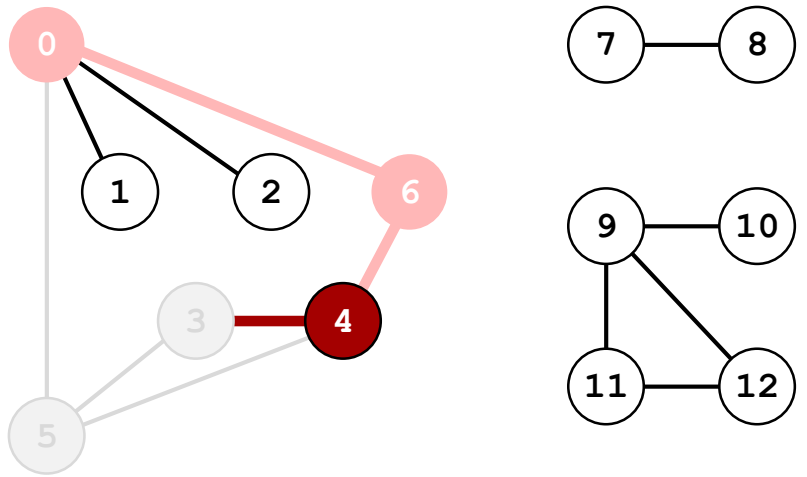


5 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

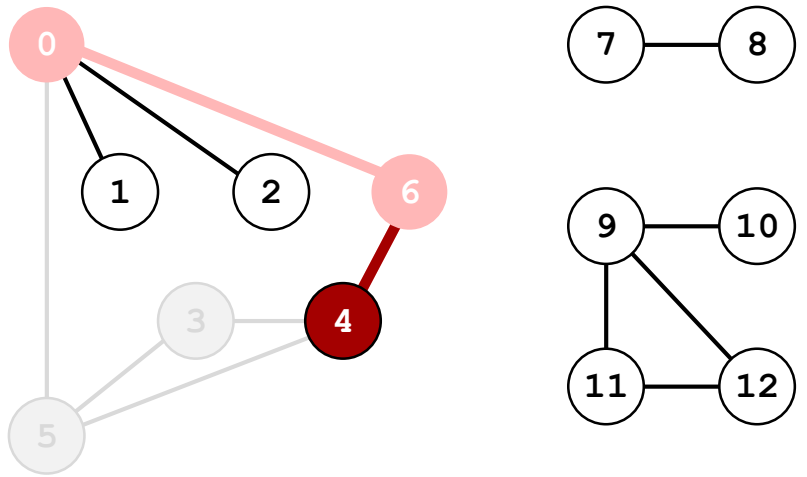


visit 4: check 5, check 3, check 6

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

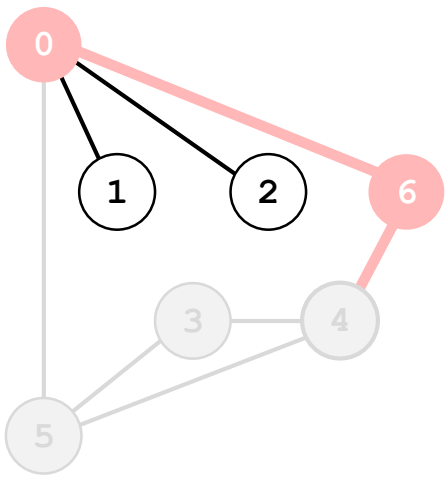


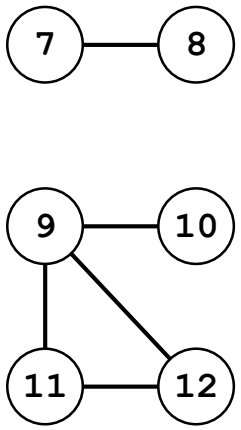
visit 4: check 5, check 3, check 6

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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



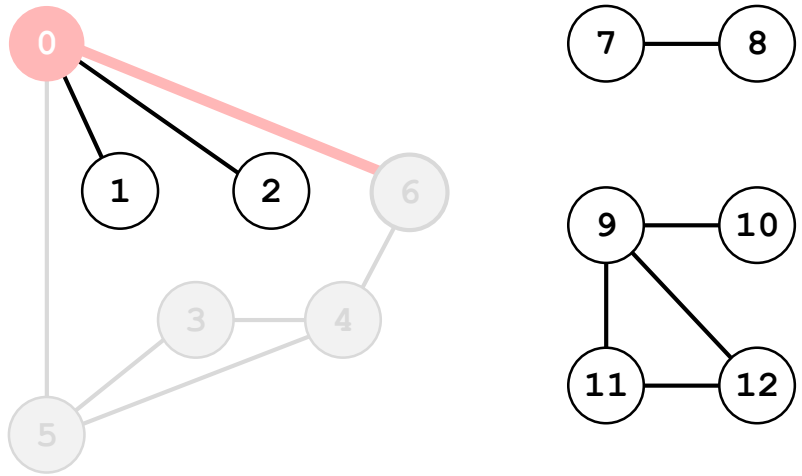


4 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

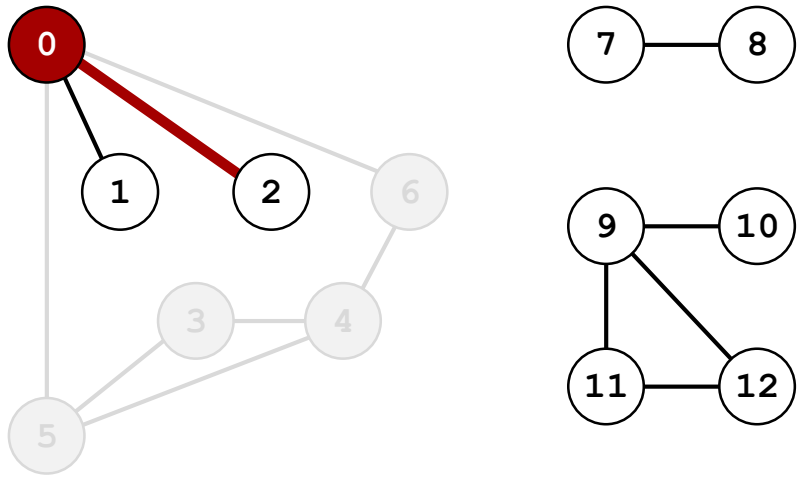


6 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

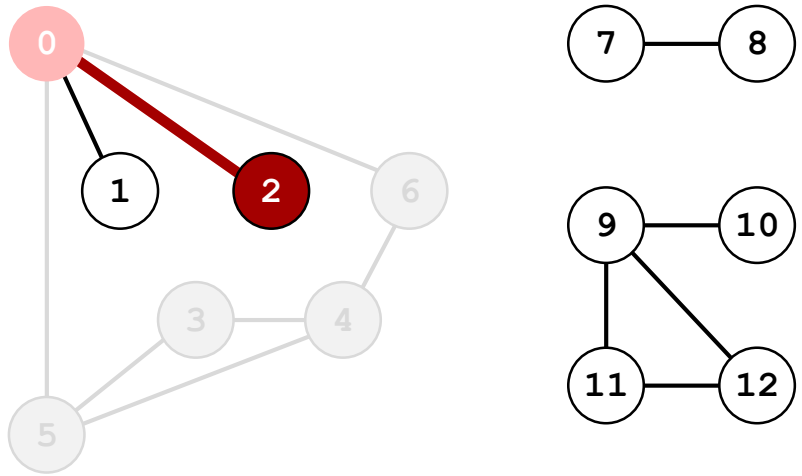



```
visit 0: check 6, check 2, check 1, check 5
```

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

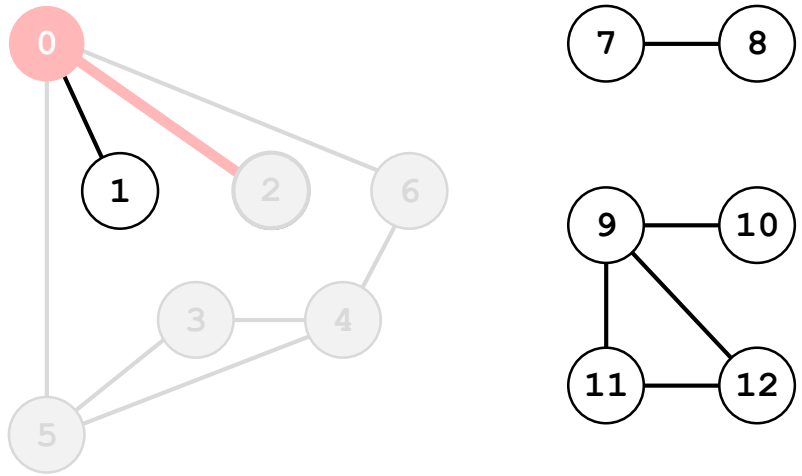


visit 2: check 0

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

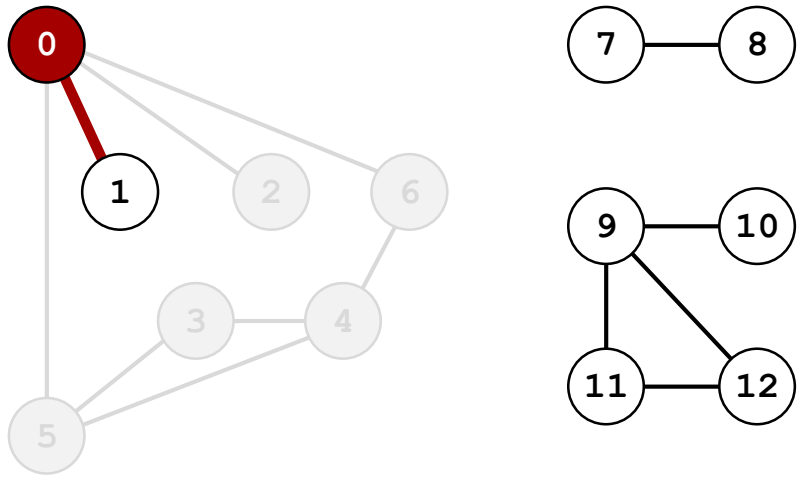


2 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

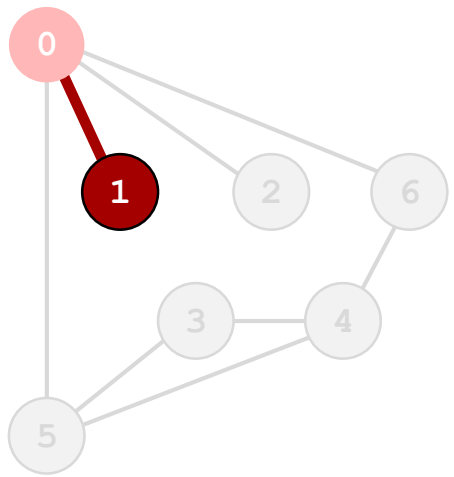


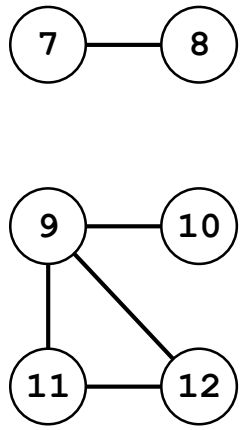

```
visit 0: check 6, check 2, check 1, check 5
```

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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



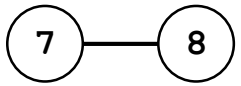


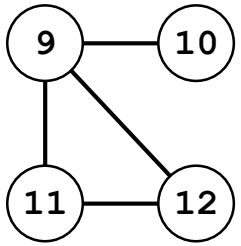
visit 1: check 0

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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



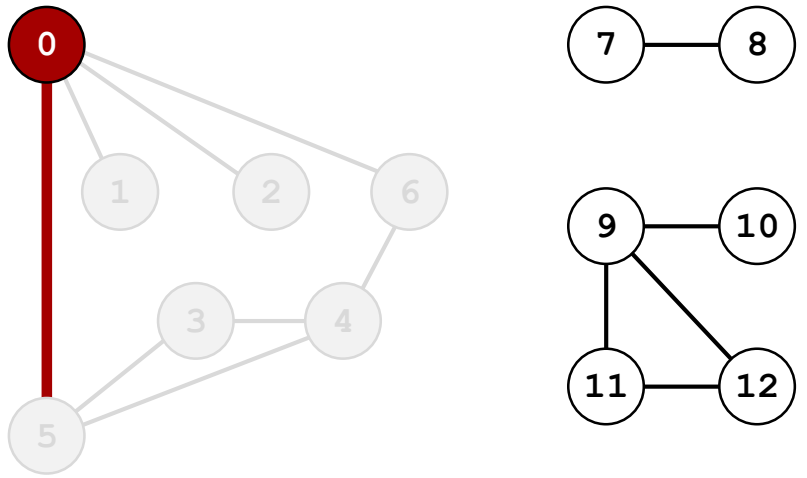


1 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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

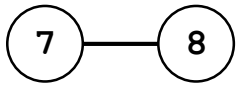


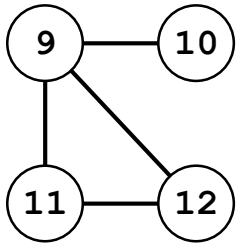

```
visit 0: check 6, check 2, check 1, check 5
```

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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



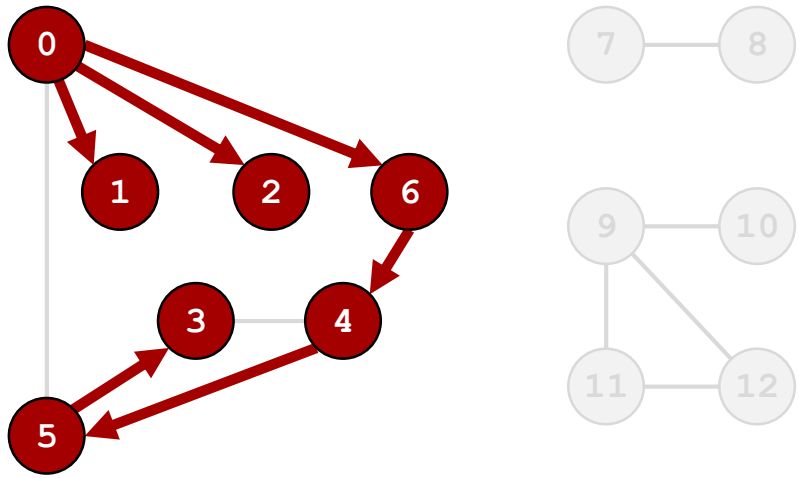


0 done

جستجوی عمقی: اجرای نمایشی

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن.

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



رئوس قابل دسترسی از رأس ۰

جستجوی عمقی: پیاده‌سازی در جاوا


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

   public DepthFirstPaths(Graph G, int s) {
      ...
      dfs(G, s);
   }

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

اگر **v** به **s** متصل باشد
marked[v] = true

مقداردهی اولیه

یافتن رئوس متصل به **s**

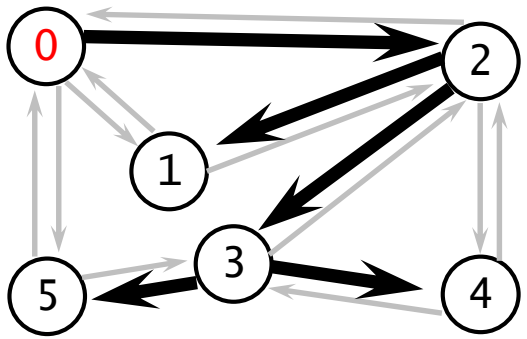
جستجوی بازگشتی **DFS**

جستجوی عمقی

- گزاره. پس از اجرای DFS، می‌توان رئوس متصل به s را در زمان ثابت و مسیر از رأس شروع s به رأس دلخواه v را در زمانی متناسب با طول مسیر محاسبه نمود.

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

```
public Iterable<Integer> pathTo(int v)
{
   if (!hasPathTo(v)) return null;
   Stack<Integer> path = new Stack<Integer>();
   for (int x = v; x != s; x = prev[x])
      path.push(x);
   path.push(s);
   return path;
}
```



prev[]

0	
1	2
2	0
3	2
4	3
5	3

جستجوی سطحی

جستجوی سطحی

- جستجوی عمقی. رئوس رؤیت نشده بر بالای پشته قرار داده می‌شوند.
- جستجوی سطحی. رئوس رؤیت نشده در انتهای صف قرار داده می‌شوند.
- کوتاه‌ترین مسیر. یافتن مسیر از **s** به **t** با حداقل تعداد یالها.

BFS (from source vertex s)

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

Repeat until the queue is empty:

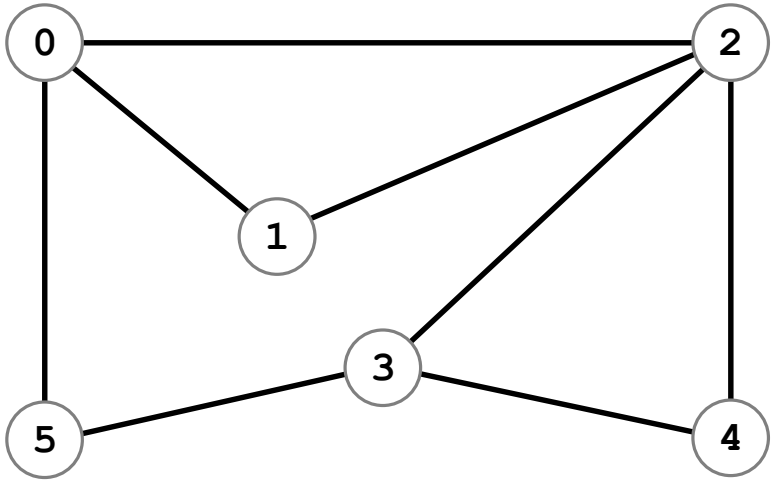
- remove the least recently added vertex v

- add each of v's unvisited neighbors to the queue,

and mark them 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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.



tinyCG.txt

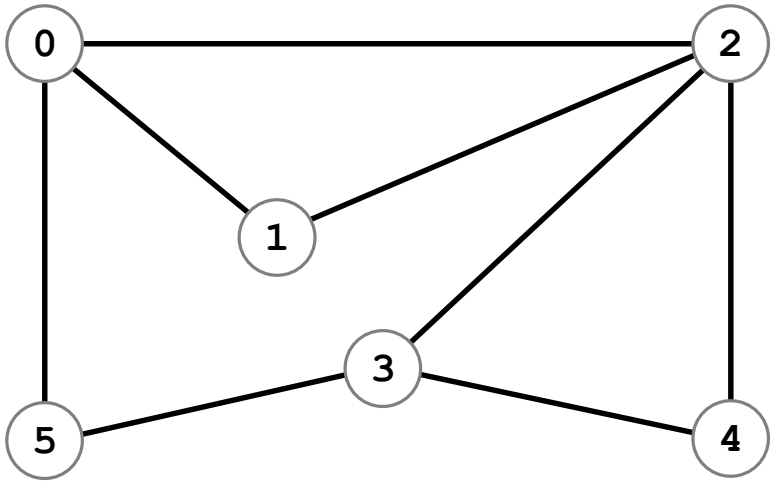
V

E

```
6
8
0 5
2 4
2 3
1 2
0 1
3 4
3 5
0 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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

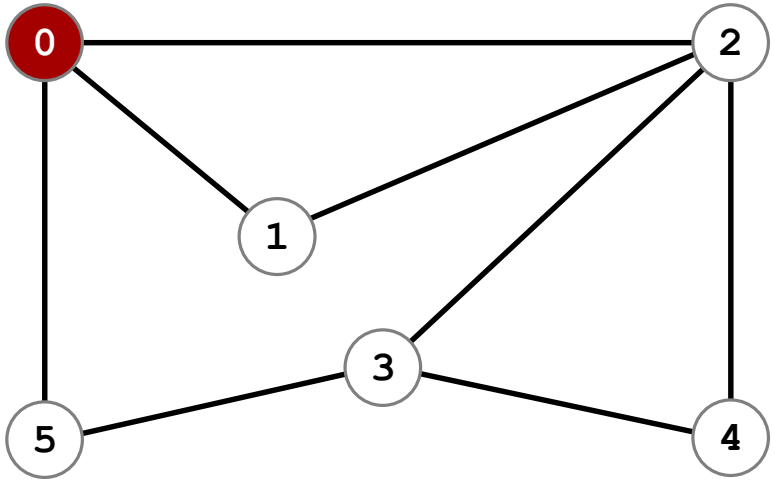


enqueue (0)

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

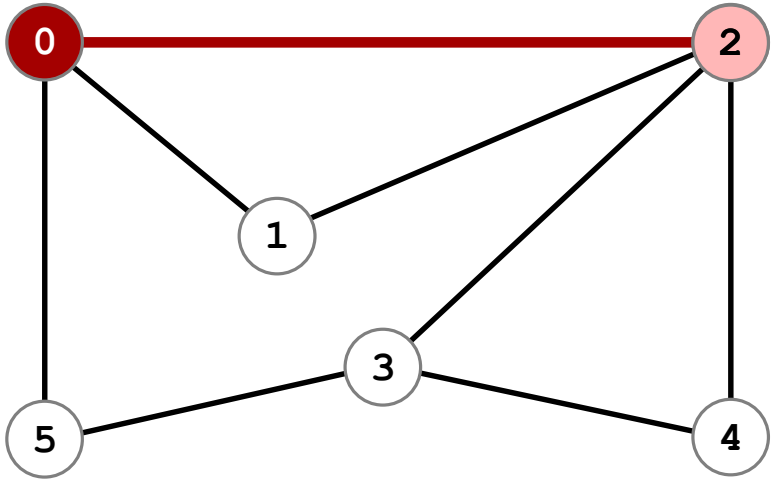


dequeue 0

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

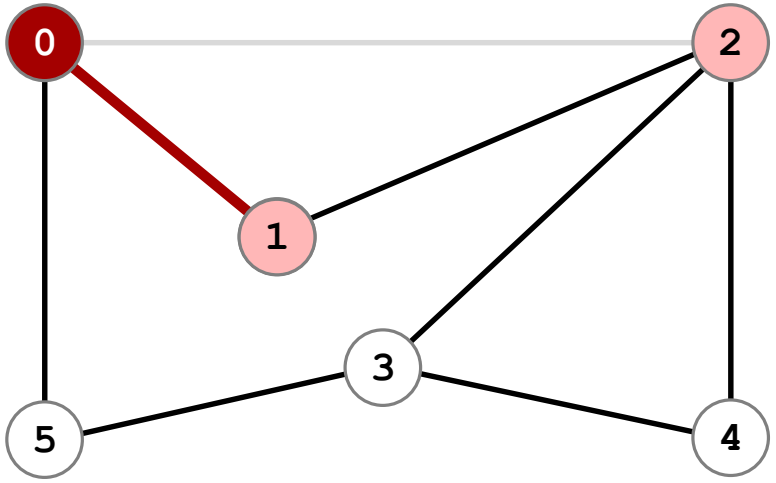


dequeue 0: check 2, check 1, check 5

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

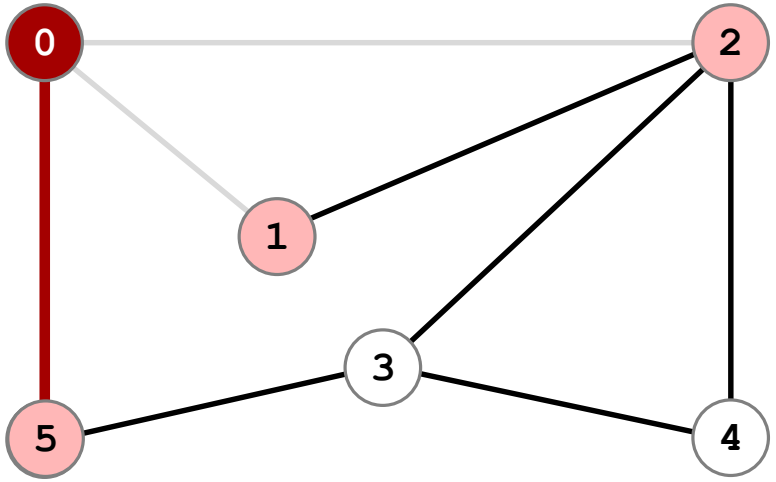


dequeue 0: check 2, **check 1**, check 5

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	3	-	-
	4	-	-
	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

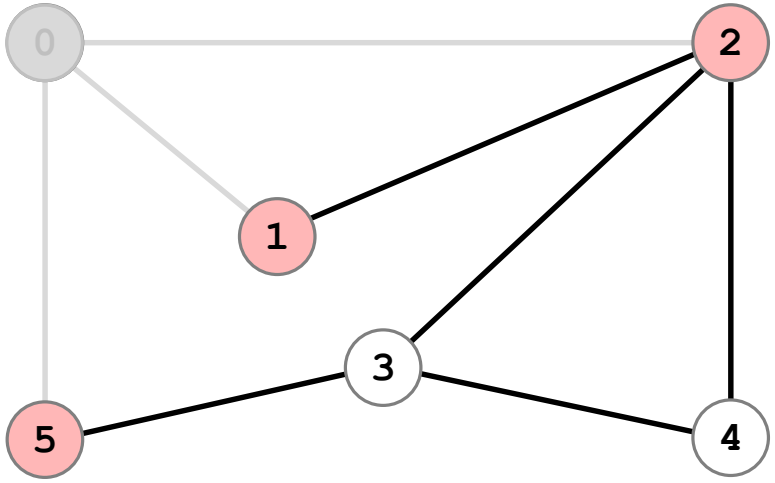


dequeue 0: check 2, check 1, **check 5**

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

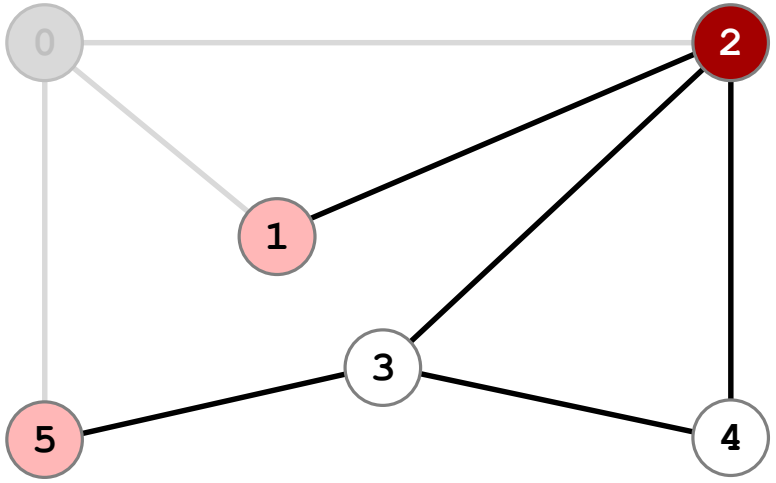


0 done

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
5	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

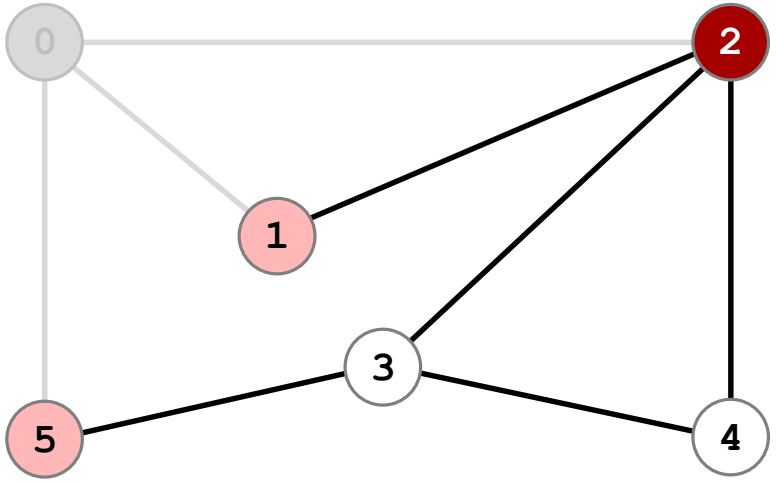


dequeue 2

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
5	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

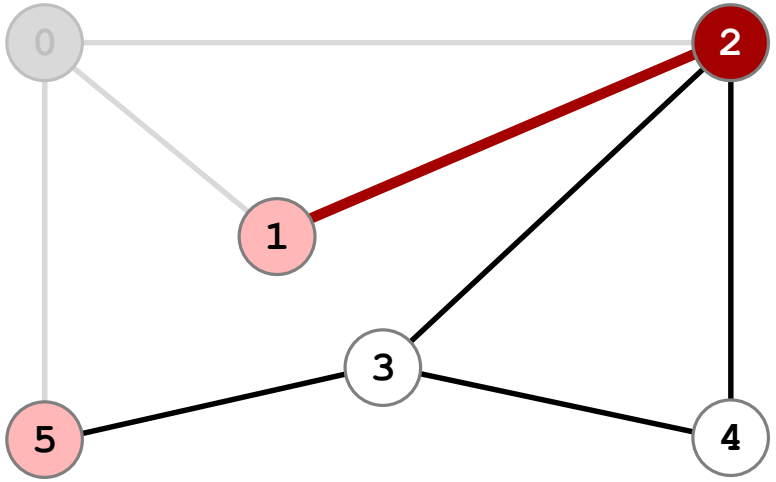


queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
5	3	-	-
	4	-	-
1	5	-	-

dequeue 2: check 0, check 1, check 3, 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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

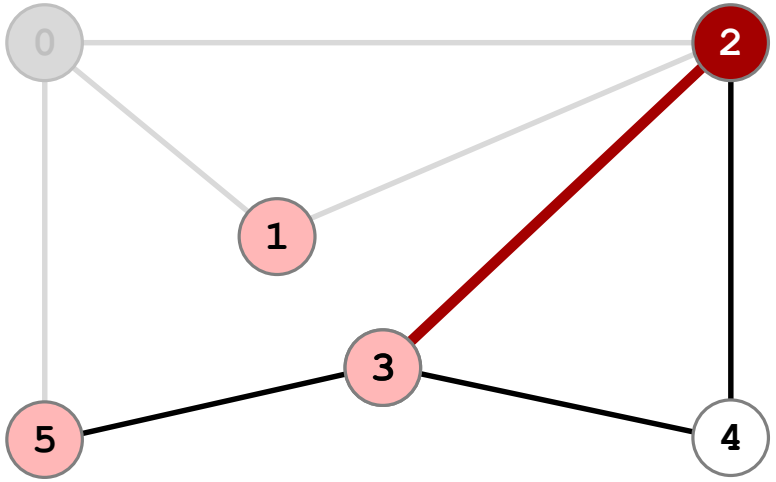


dequeue 2: check 0, **check 1**, check 3, check 4

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	3	-	-
	4	-	-
5	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

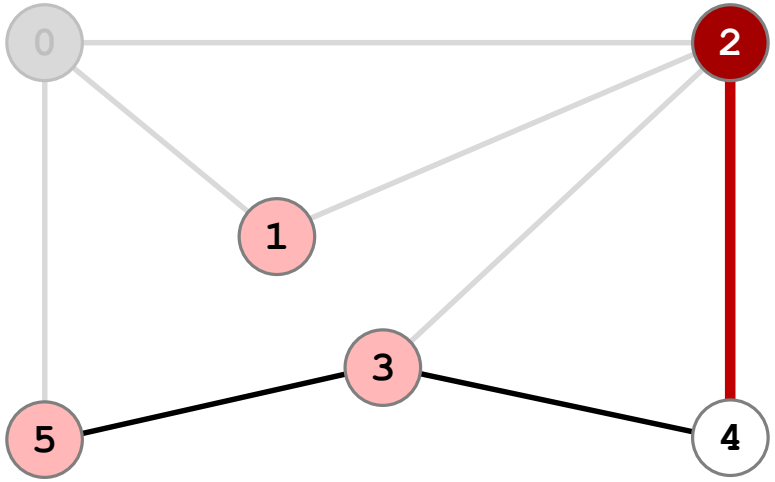


dequeue 2: check 0, check 1, **check 3**, check 4

queue	v	edegeTo[]	distTo[]
	0	-	0
	1	-	-
	2	-	-
	3	-	-
	4	-	-
5	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 علامت گذاری نشده‌اند به صف اضافه کن و آنها را علامت گذاری کن.

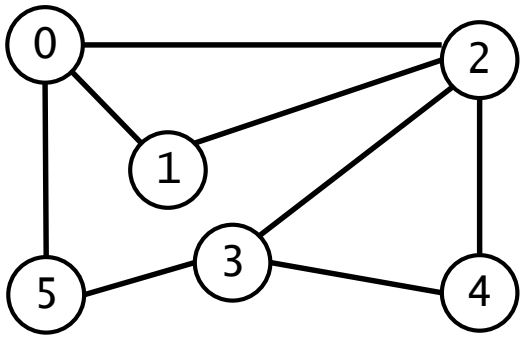


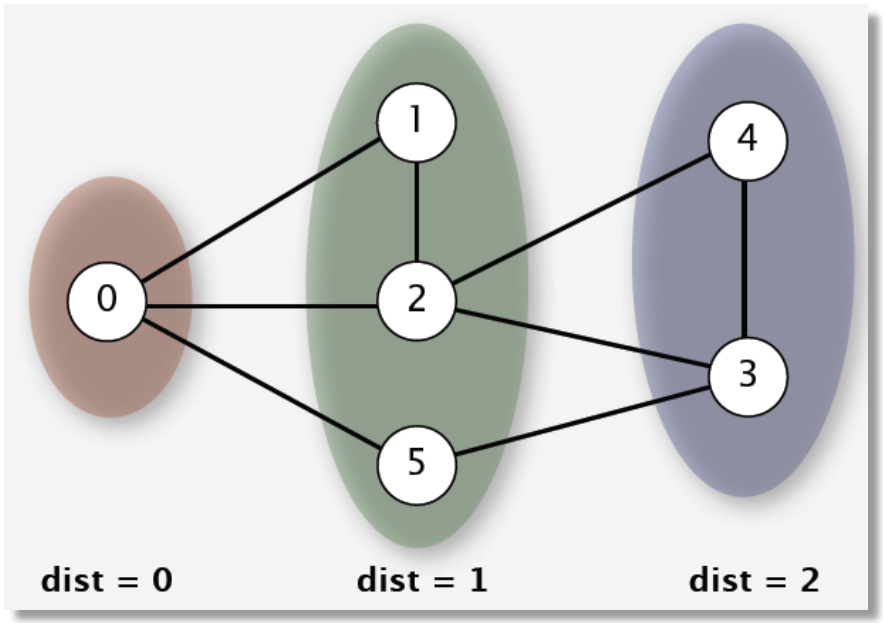
dequeue 2: check 0, check 1, check 3, **check 4**

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

جستجوی سطحی

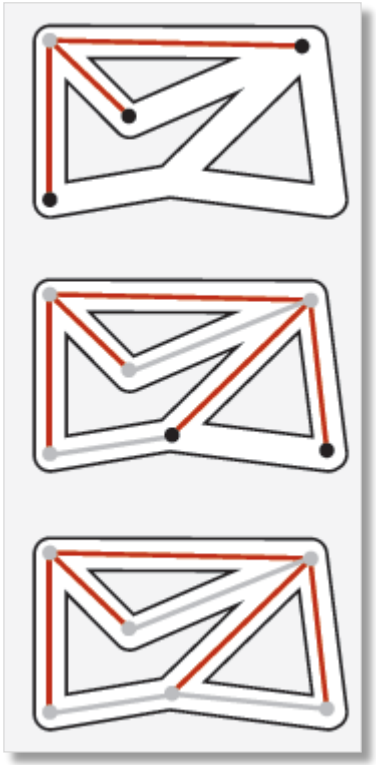
- ادعا. BFS در یک گراف همبند کوتاه‌ترین مسیرها (بر حسب تعداد یالها) از رأس مبدأ s تا سایر رئوس را در زمانی متناسب با V + E محاسبه می‌کند.





جستجوی سطحی : پیاده‌سازی در 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]) {
               q.enqueue(w);
               marked[w] = true;
               prev[w] = v;
            }
      }
   }
}
```



مؤلفه‌های همبند

مؤلفه‌های همبند

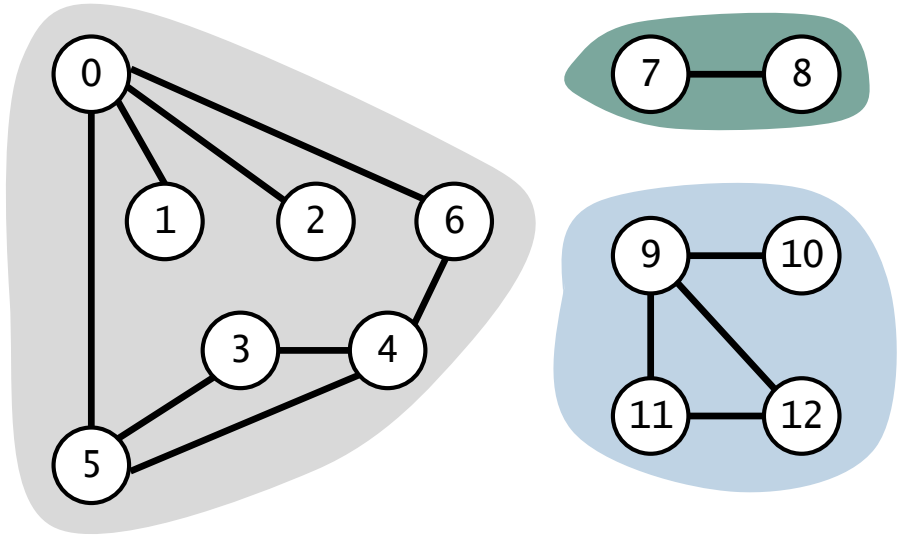
- تعریف. رئوس **v** و **w** به هم متصل هستند اگر مسیری بین آنها وجود داشته باشد.
- هدف. پردازش گراف برای پاسخ دادن به این که «آیا **v** به **w** متصل است؟» در زمان ثابت.

public class	CC	
	CC(Graph G)	یافتن مؤلفه‌های همبند در G
boolean	connected(int v, int w)	آیا v به w متصل هستند؟
int	count()	تعداد مؤلفه های همبند
int	id(int v)	شناسه مؤلفه شامل v

مؤلفه‌های همبند

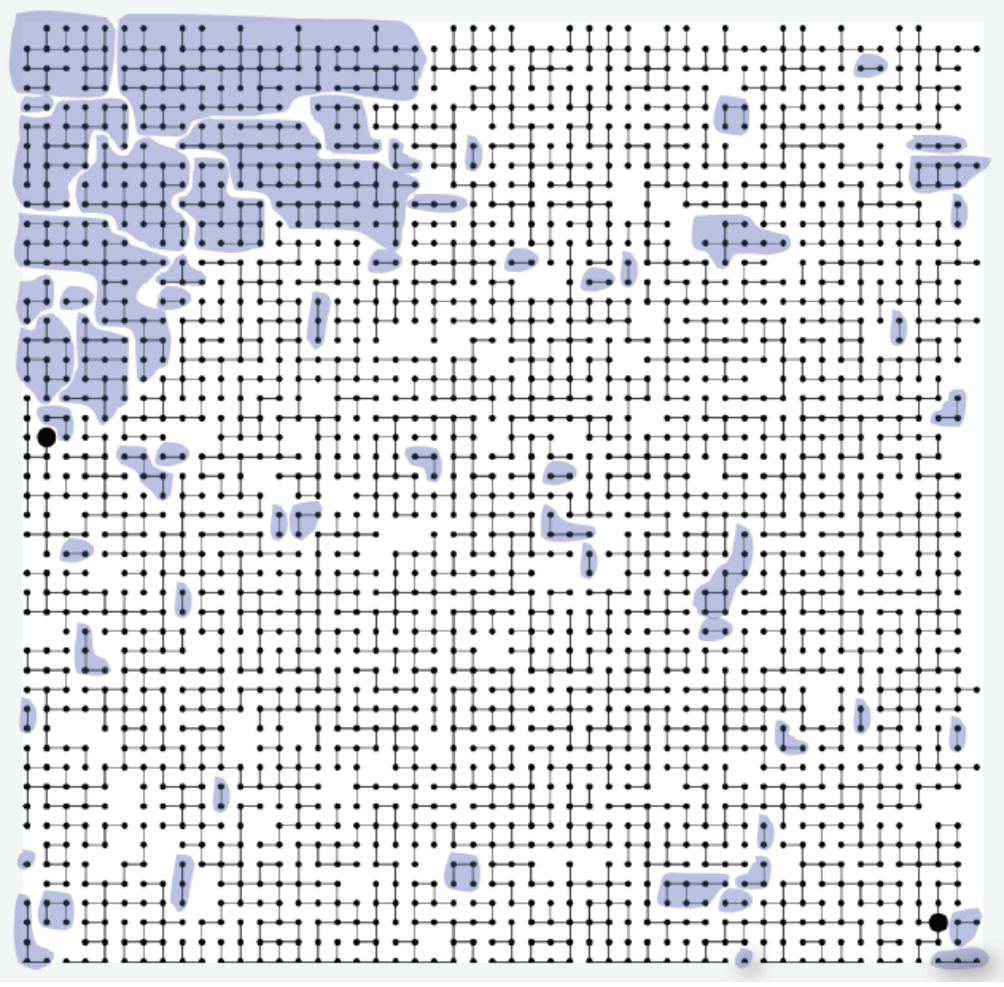
- رابطه‌ی همبندی. یک رابطه هم ارزی است:
  - بازتابی: هر رأس به خودش متصل است.
  - تقارنی: اگر v به w متصل باشد، آنگاه w به v متصل است.
  - تعدی: اگر v به w و w به x متصل باشد، آنگاه v به x متصل است.
- تعریف. یک مؤلفه‌ی همبند یک مجموعه‌ی بیشینه از رئوس متصل است.

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



مؤلفه‌های همبند

- تعریف. یک مؤلفه‌ی همبند یک مجموعه‌ی بیشینه از رئوس متصل است.



۶۳ مؤلفه‌ی همبند

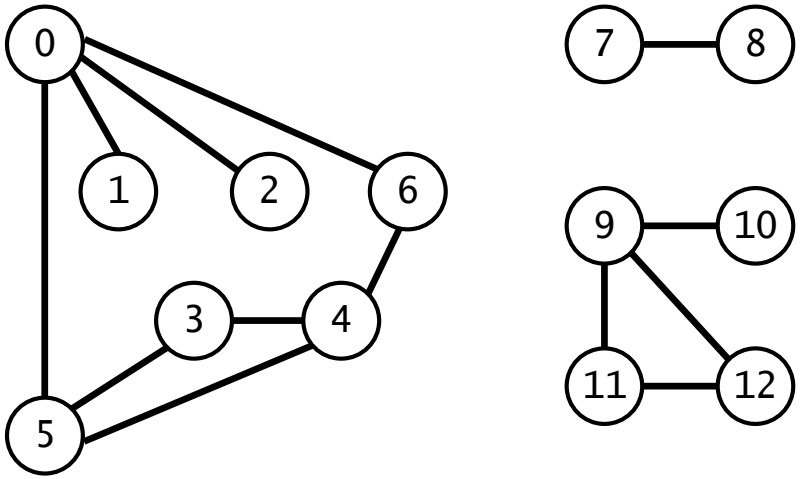
مؤلفه‌های همبند

- هدف. افراز رئوس به مؤلفه‌های همبند.

Connected components

Initialize all vertex v as unmarked.

For each unmarked vertex v, run DFS to identify all vertices discovered as part of the same component.



V → 13
E → 13

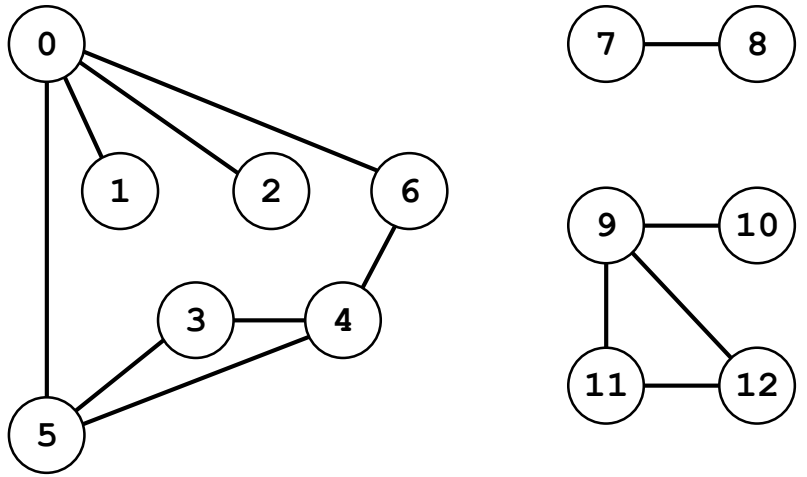
0	5
4	3
0	1
9	12
6	4
5	4
0	2
11	12
9	10
0	6
7	8
9	11
5	3

tinyG.txt

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

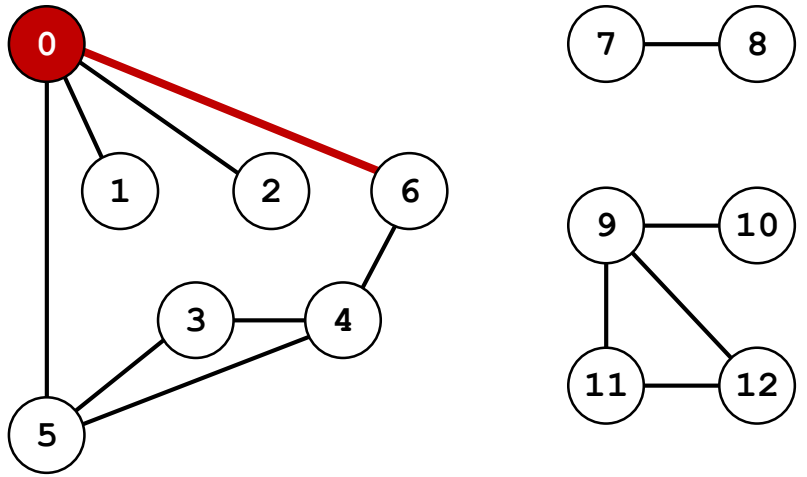
v	marked[]	id[]
0	F	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-



مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

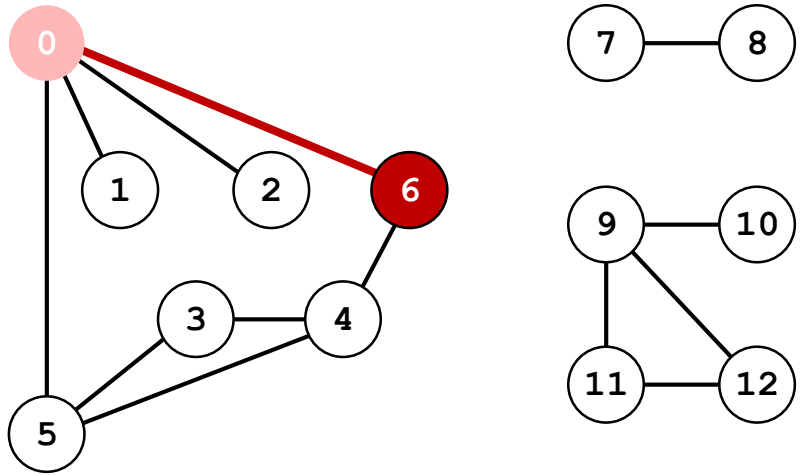


visit 0: check 6, check 2, check 1, check 5

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

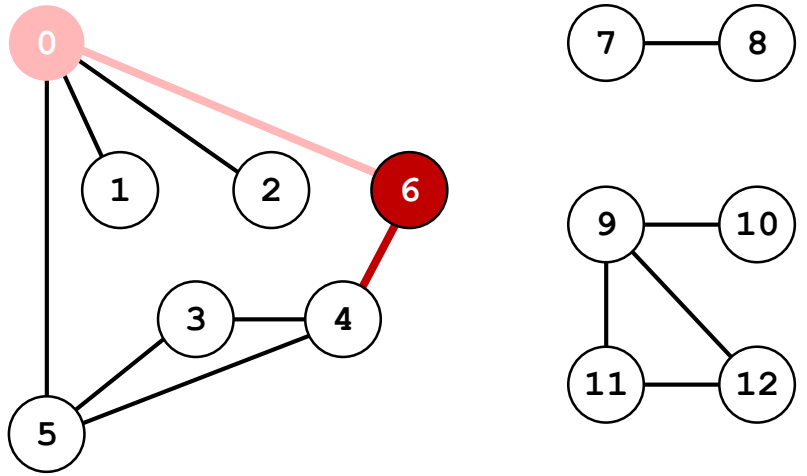


visit 6: check 0, check 4

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

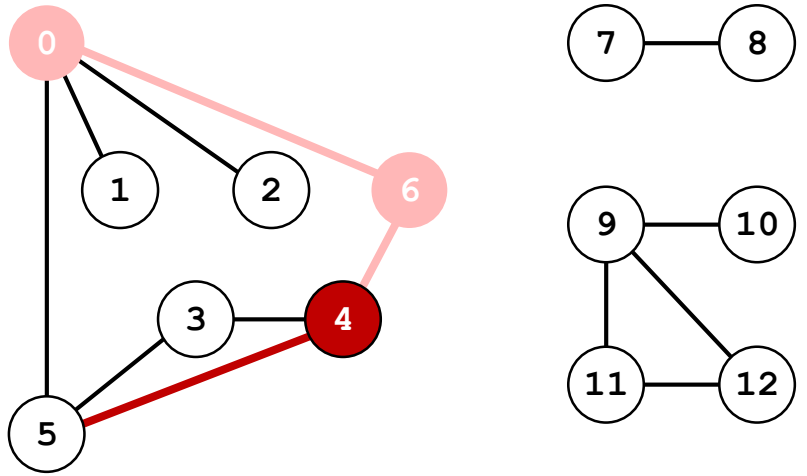


visit 6: check 0, check 4

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

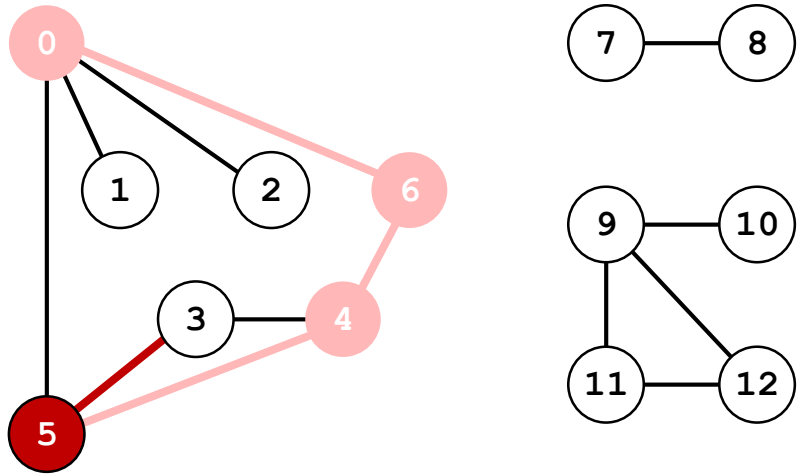


visit 4: check 5, check 6, check 3

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

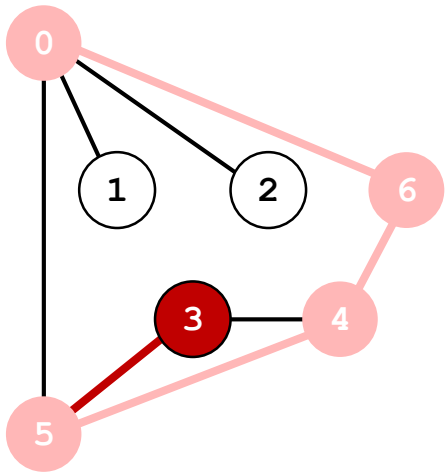


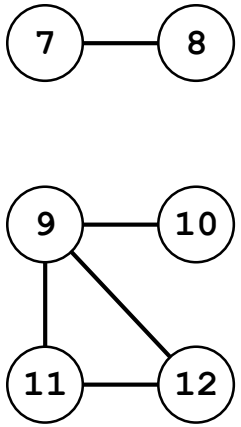
visit 5: check 3, check 4, check 0

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



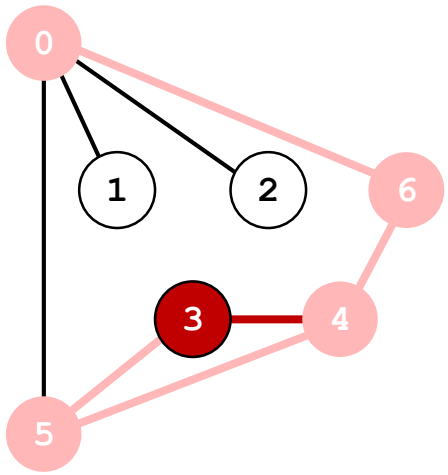


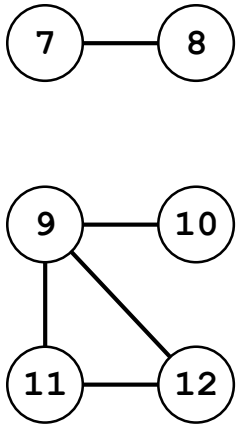
visit 3: check 5, check 4

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



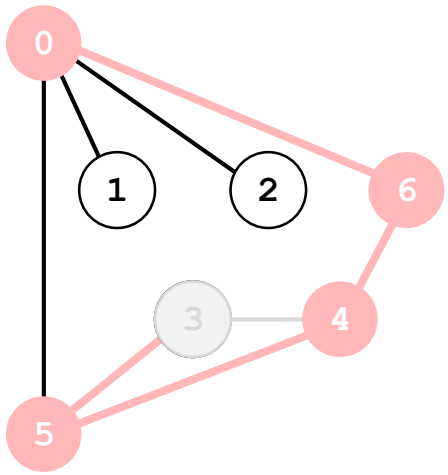


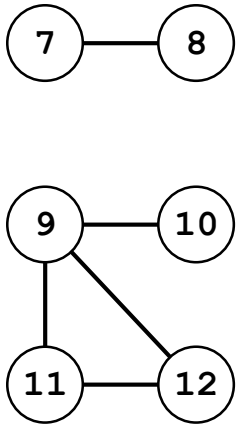
visit 3: check 5, check 4

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



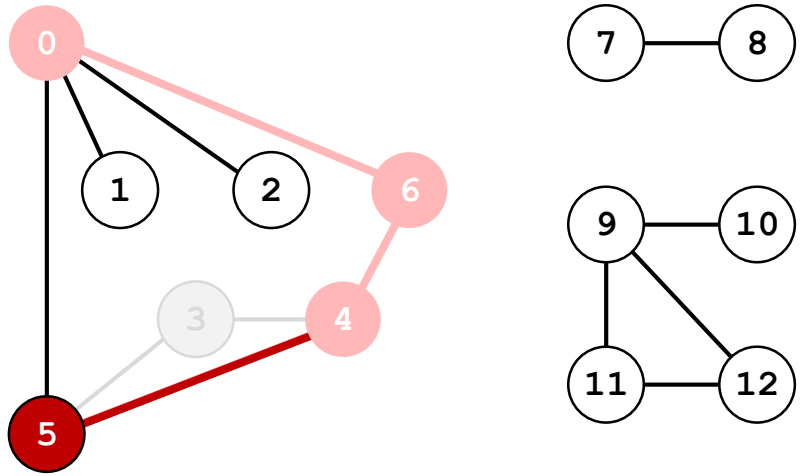


3 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

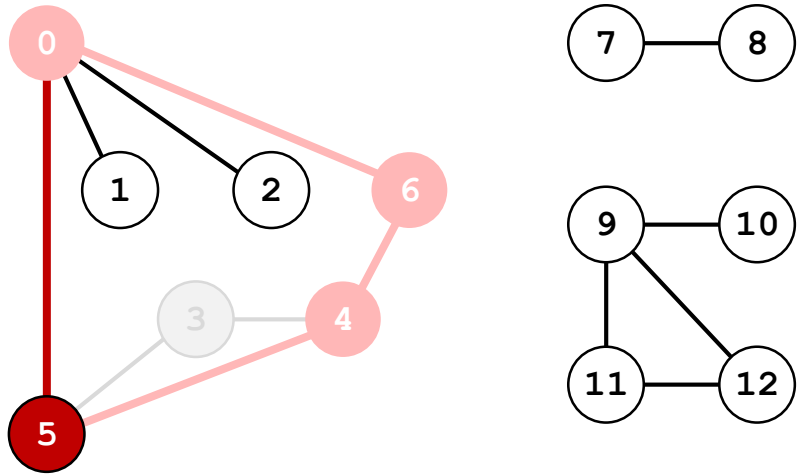


visit 5: check 3, check 4, check 0

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

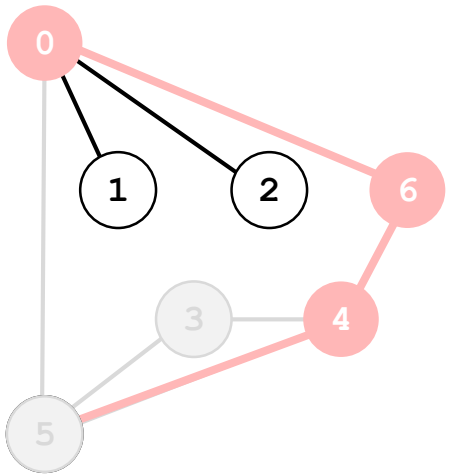


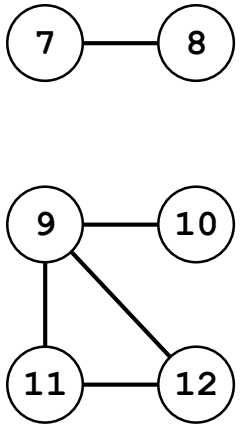
visit 5: check 3, check 4, check 0

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



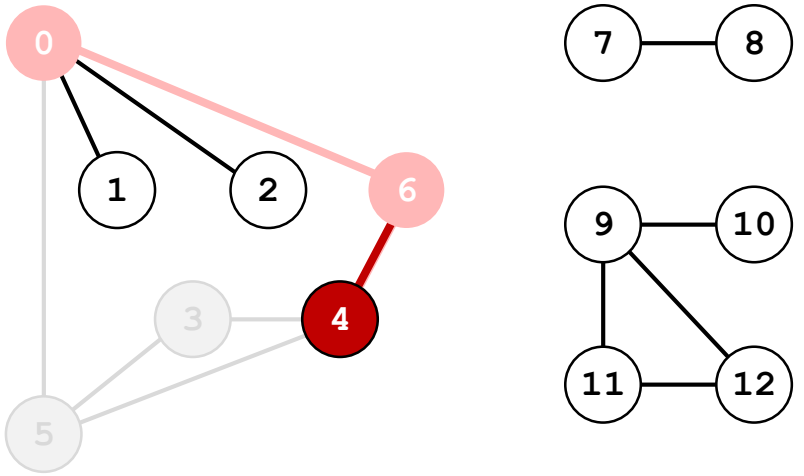


5 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

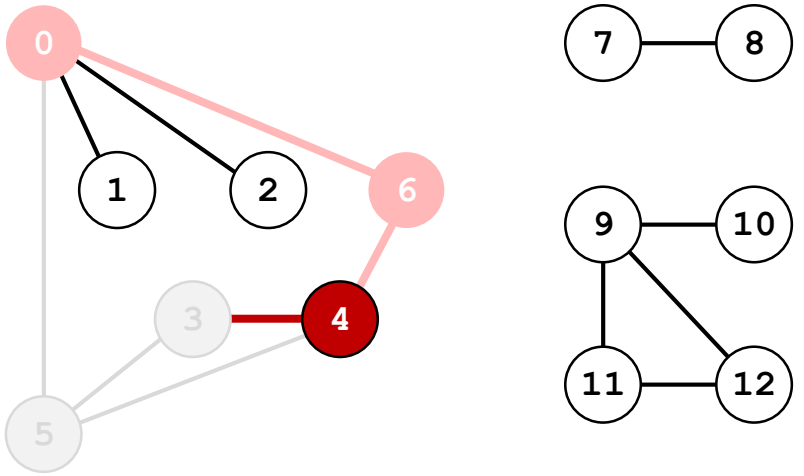


visit 4: check 5, check 6, check 3

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

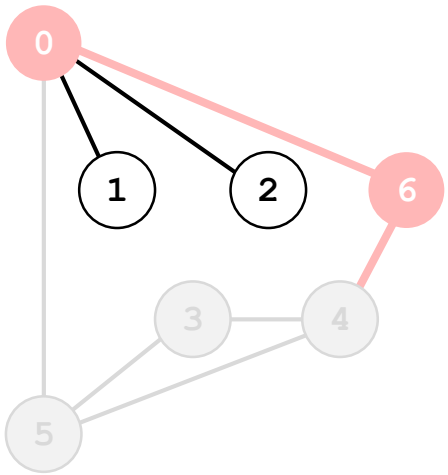


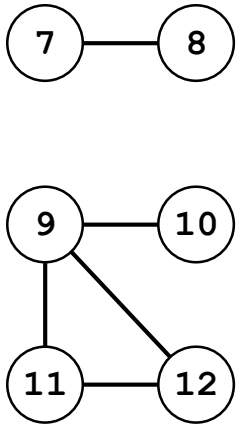
visit 4: check 5, check 6, check 3

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



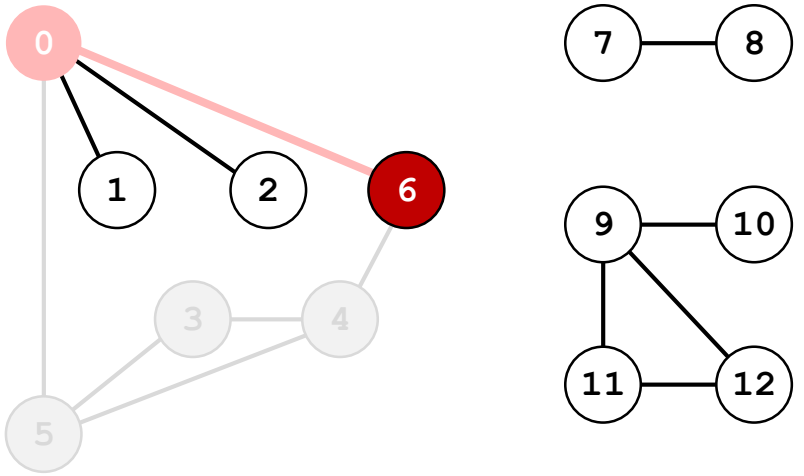


4 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

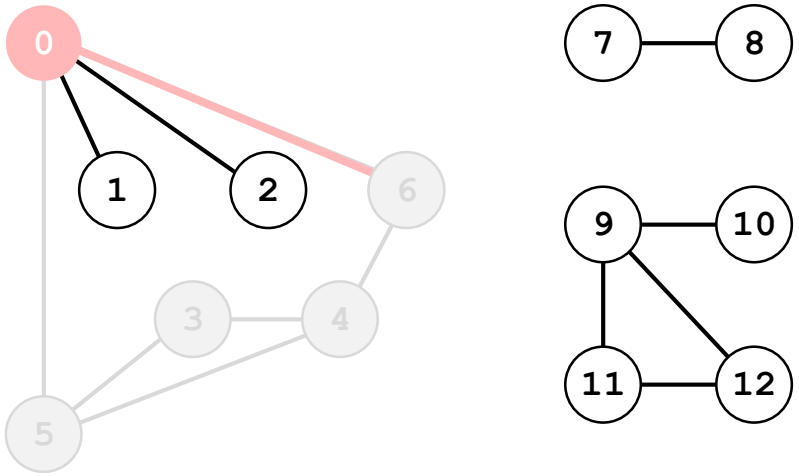



```
6 done
```

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

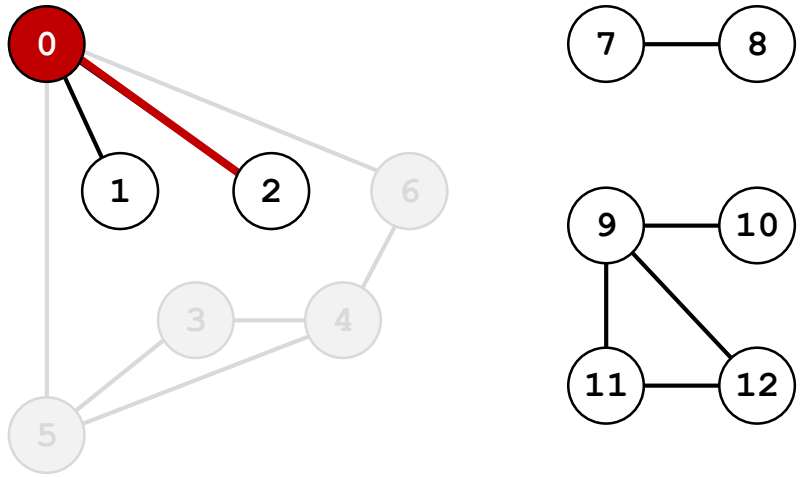


6 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

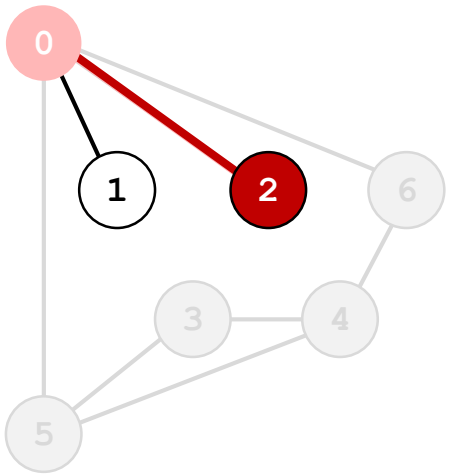


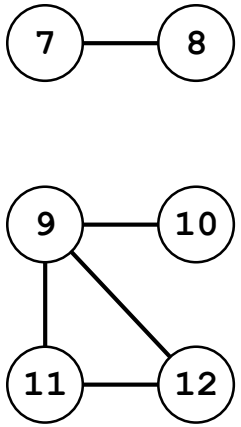
visit 0: check 6, check 2, check 1, check 5

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



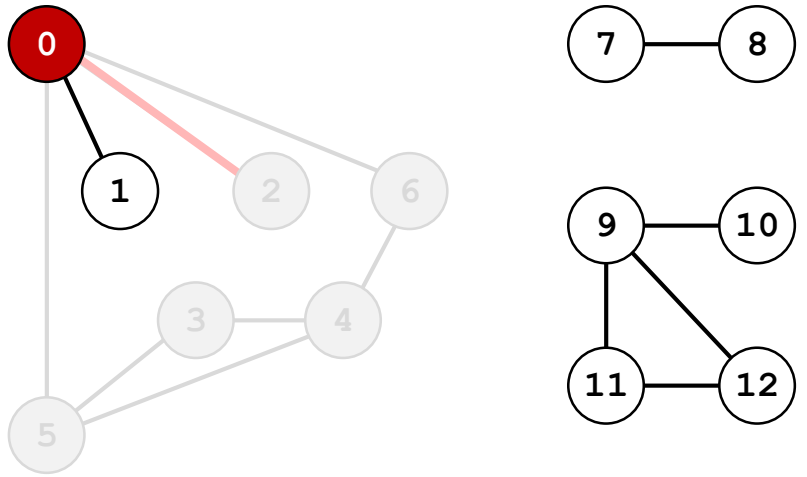


visit 2: check 0

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

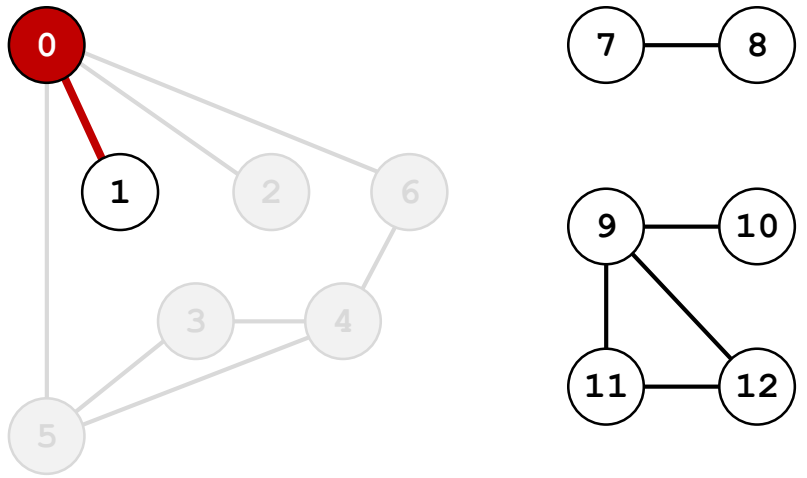


2 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

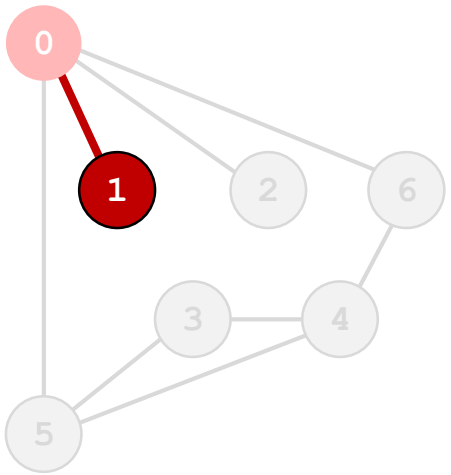


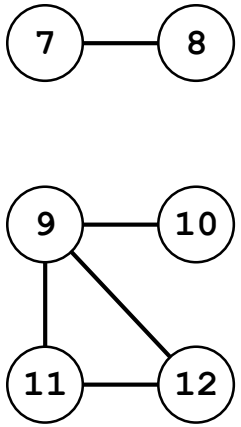
visit 0: check 6, check 2, **check 1**, check 5

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



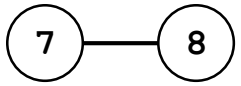


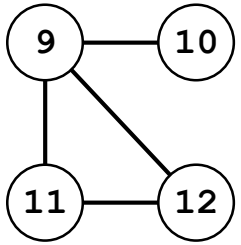
visit 1: check 0

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



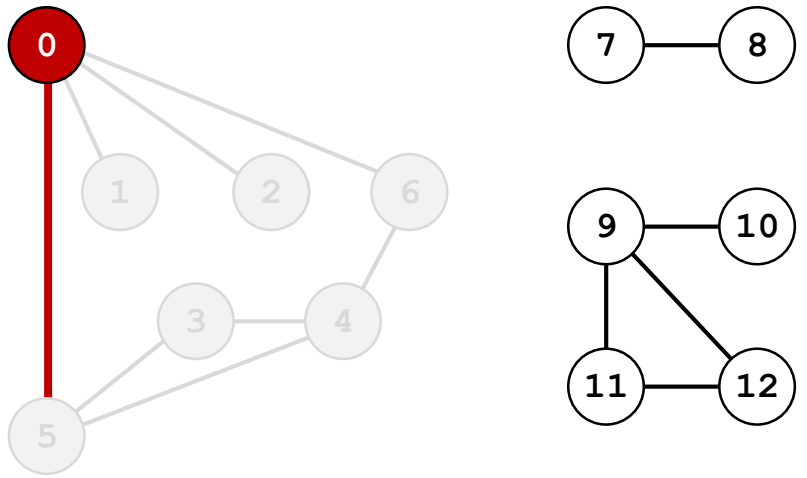


1 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

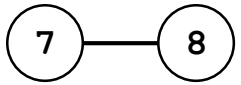


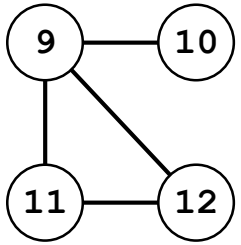

```
visit 0: check 6, check 2, check 1, check 5
```

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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





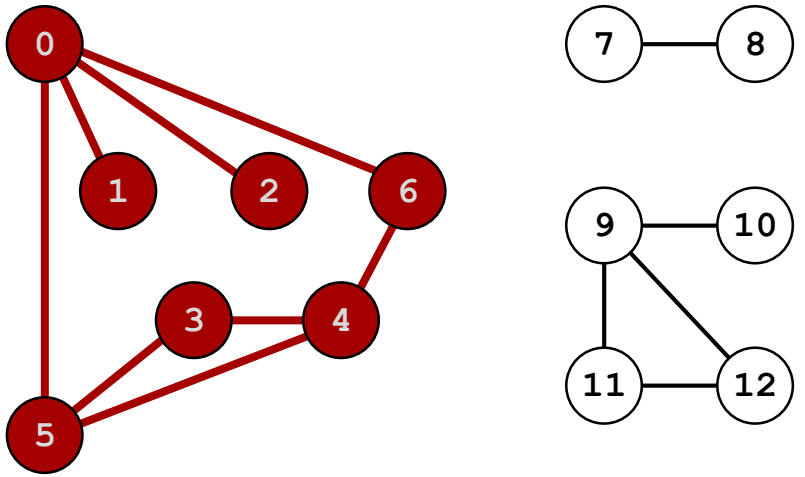
0 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

مولفه‌ی همبند

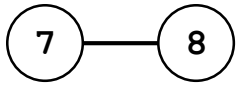


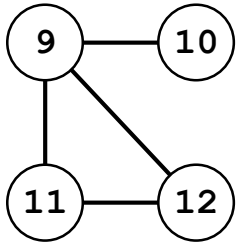
connected component: 0 1 2 3 4 5 6

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



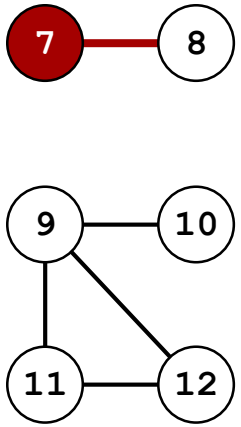


check 1 2 3 4 5 6

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

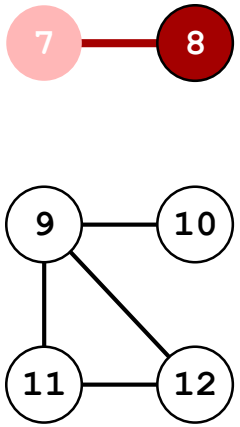


visit 7: check 8

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

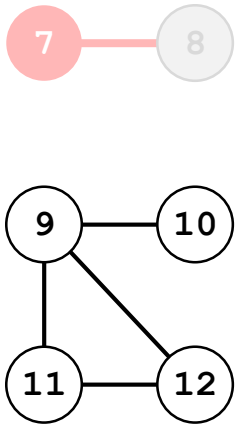


visit 8: check 7

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

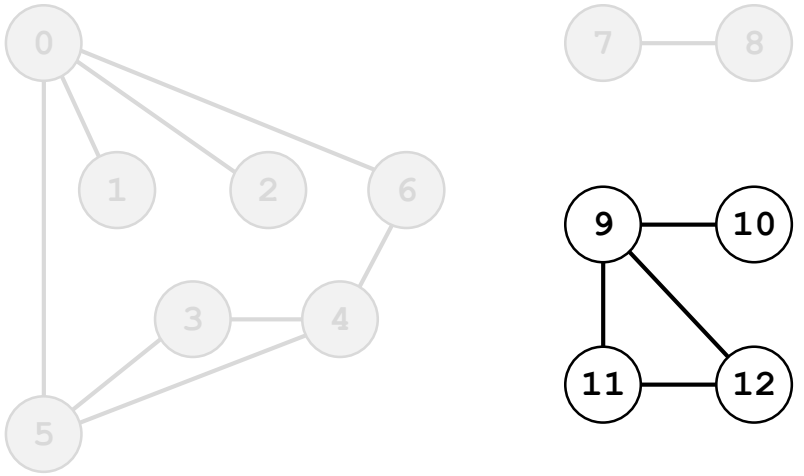


8 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



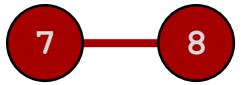
7 done

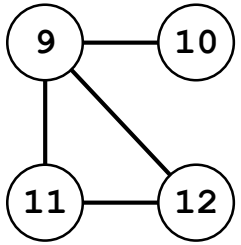
مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

مولفه‌ی همبند



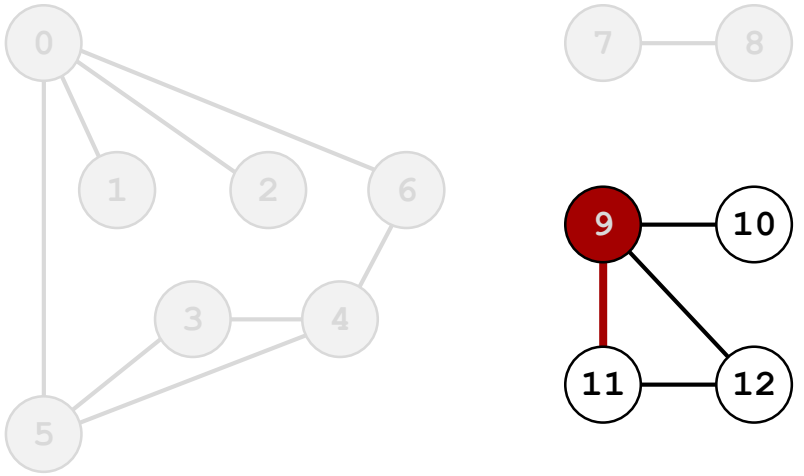


connected component: 7 8

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

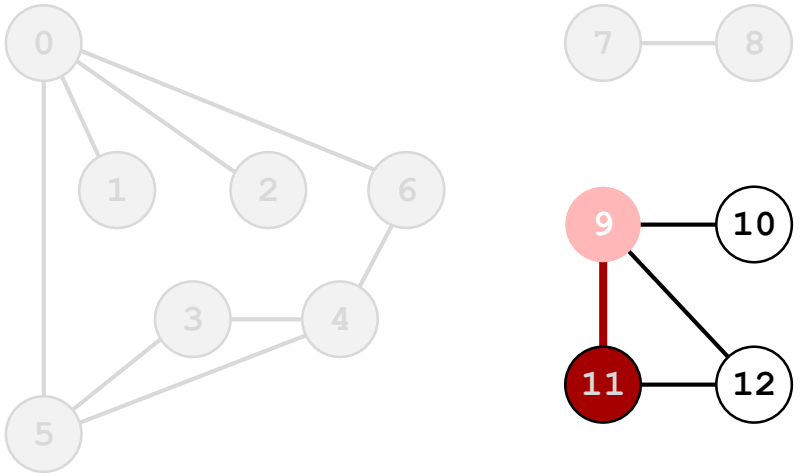


visit 9: check 11, check 10, check 12

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

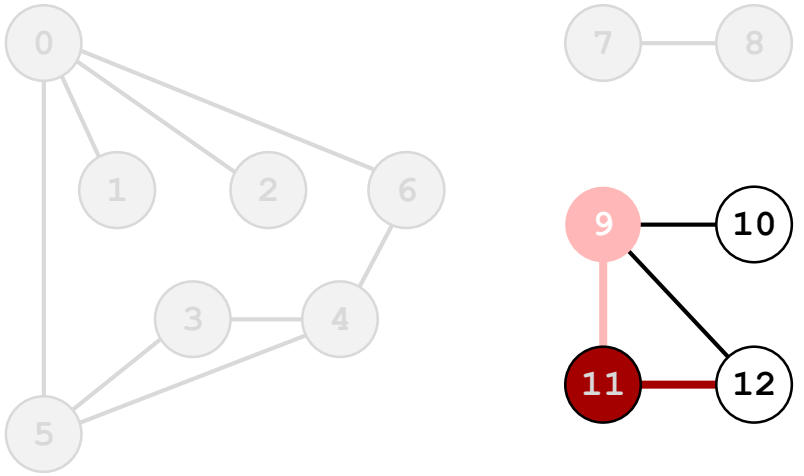


visit 11: check 9, check 12

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

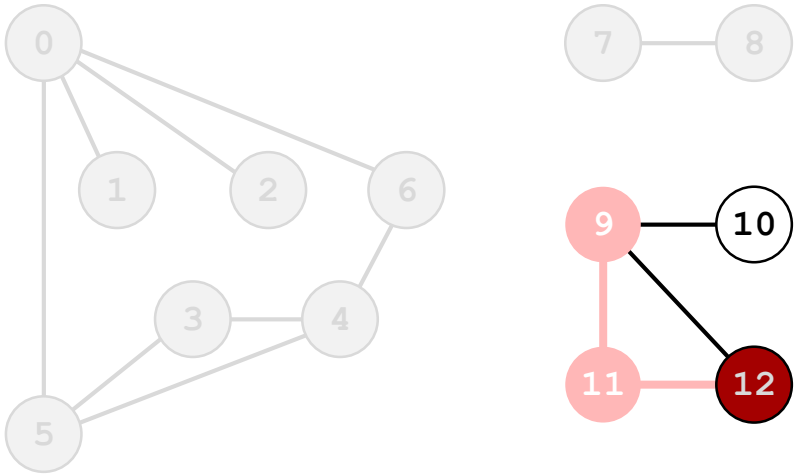


visit 11: check 9, check 12

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

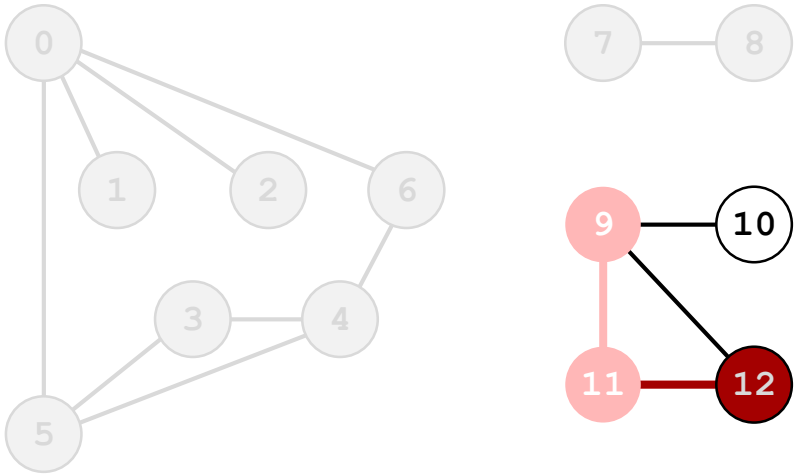


visit 12: check 11, check 9

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

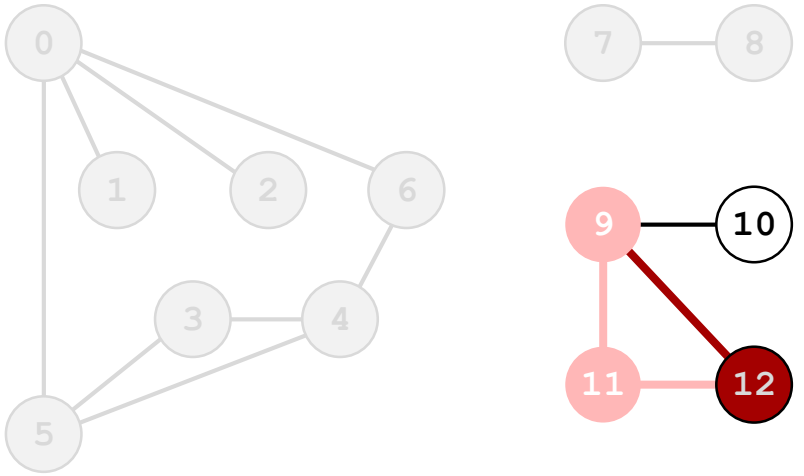


visit 12: check 11, check 9

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

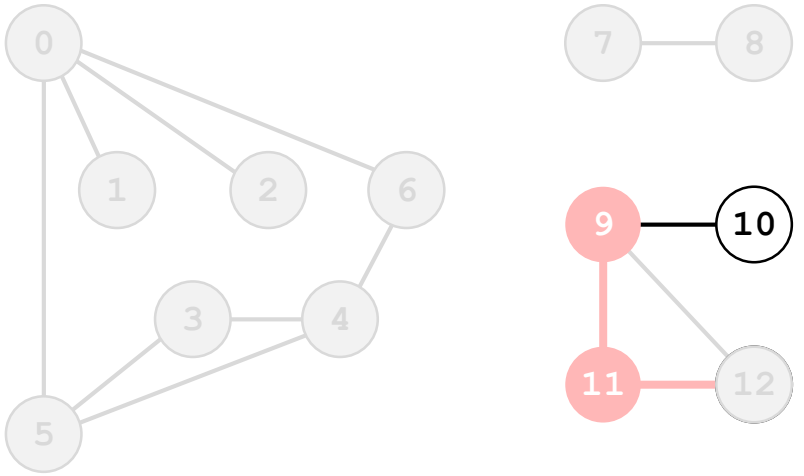


visit 12: check 11, check 9

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

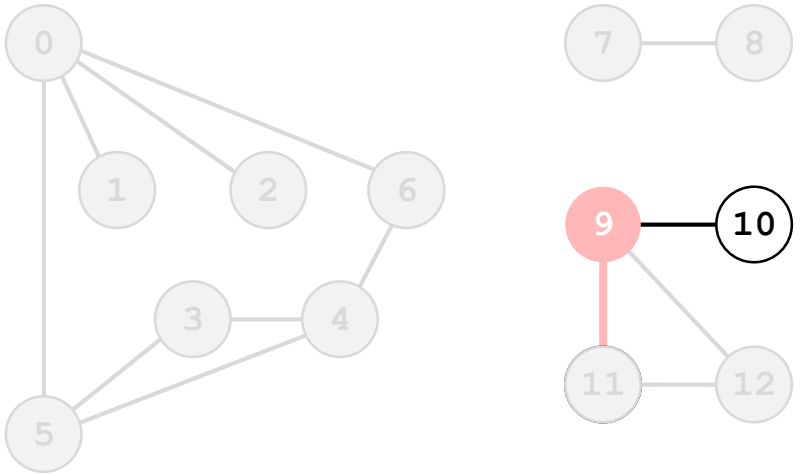



```
12 done
```

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

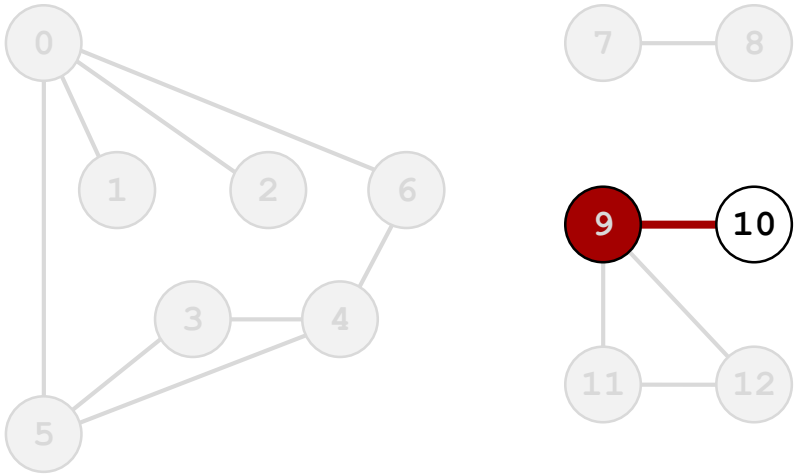


11 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

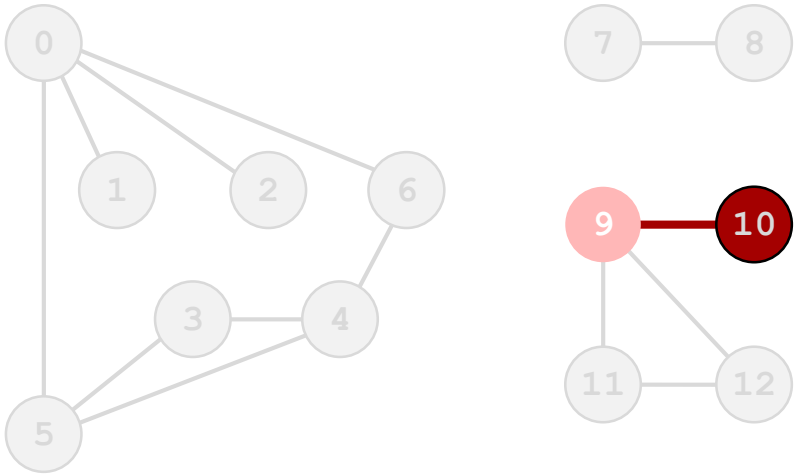


visit 9: check 11, check 10, check 12

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

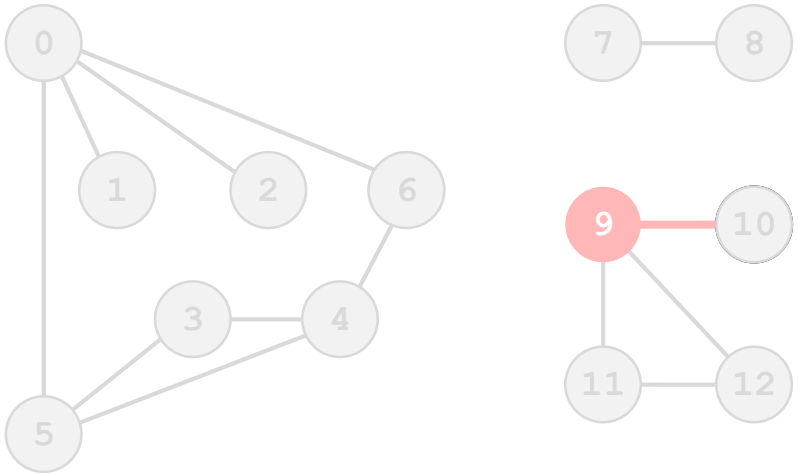


visit 10: check 9

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

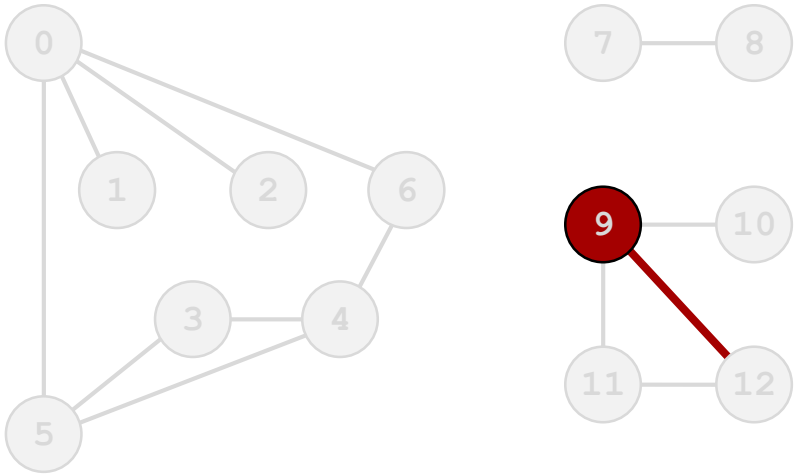


10 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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



visit 9: check 11, check 10, check 12

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

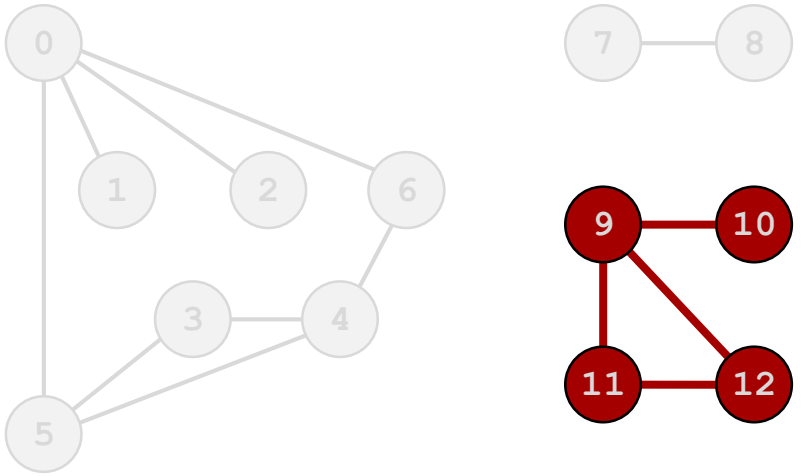
9 done

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

مولفه‌ی همبند




```
connected component: 9 10 11 12
```

مؤلفه‌های همبند (اجرای نمایشی)

- برای ملاقات رأس v:
  - رأس v را به عنوان یک رأس ملاقات شده علامت گذاری کن.
  - به صورت بازگشتی تمام رئوس همسایه‌ی v را که علامت گذاری نشده‌اند، ملاقات کن .

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

done

مؤلفه‌های همبند: پیاده‌سازی در جاوا


```
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++;
          }
       }
    }

    public int count()                  { }
    public int id(int v)                { }
    private void dfs(Graph G, int v)  { }
}
```

شناسه‌ی مؤلفه شامل v

تعداد مؤلفه‌ها

اجرای DFS با شروع از یک رأس در هر مؤلفه

اسلاید بعدی

مؤلفه‌های همبند: پیاده‌سازی در جاوا


```
public int count()
{  return count;  }
```

تعداد مؤلفه‌ها

```
public int id()
{  return id[v];  }
```

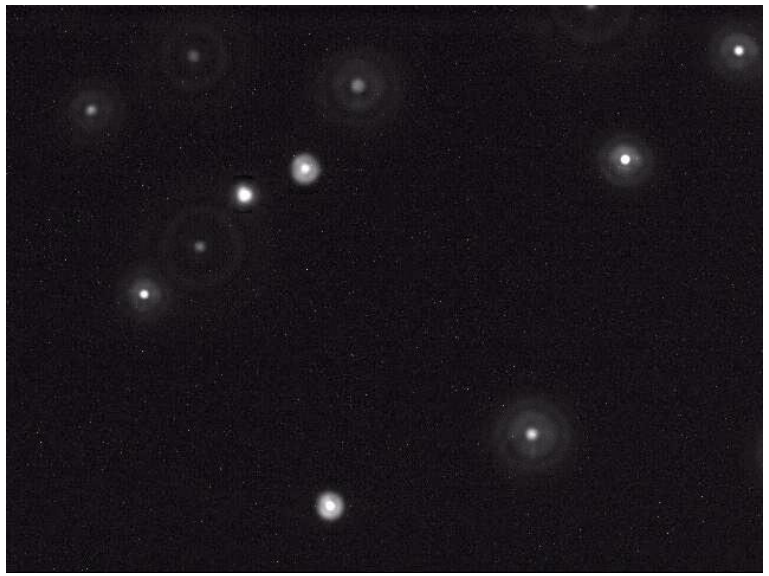
شناسه مؤلفه شامل v

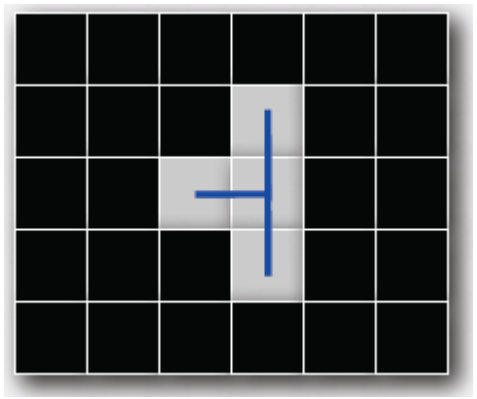
```
private void dfs(Graph G, int v)
{
   marked[v] = true;
   id[v]     = count;

   for (int w : G.adj(v))
      if (!marked[w])
         dfs(G, w);
}
```

مؤلفه‌های همبند: کاربرد

- تشخیص ذرات. تشخیص ذرات در یک تصویر داده شده در مقیاس خاکستری.
  - رئوس: پیکسل‌ها؛ یالها: بین هر دو پیکسل همسایه با مقدار خاکستری بزرگ‌تر یا مساوی ۷۰.
  - ذرات: مؤلفه‌های همبند شامل ۲۰ الی ۳۰ پیکسل.



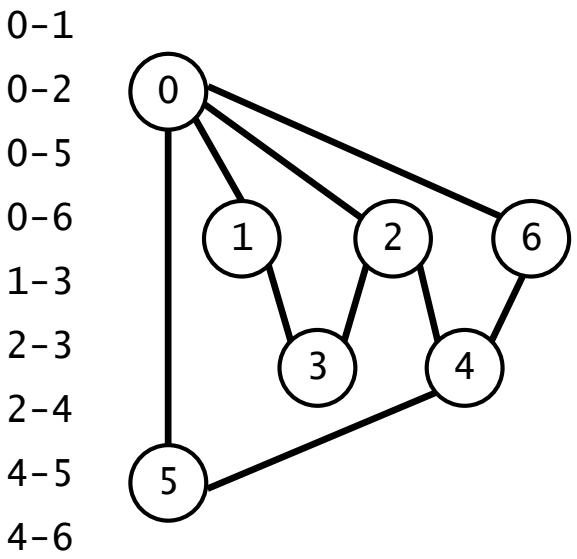


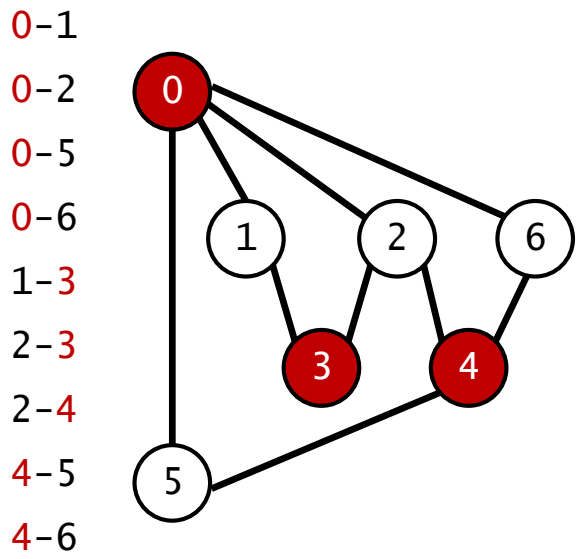
- دنبال کردن ذرات. دنبال کردن حرکت ذرات در طول زمان.

چالش‌ها

چالش ۱: بررسی دوبخشی بودن گراف

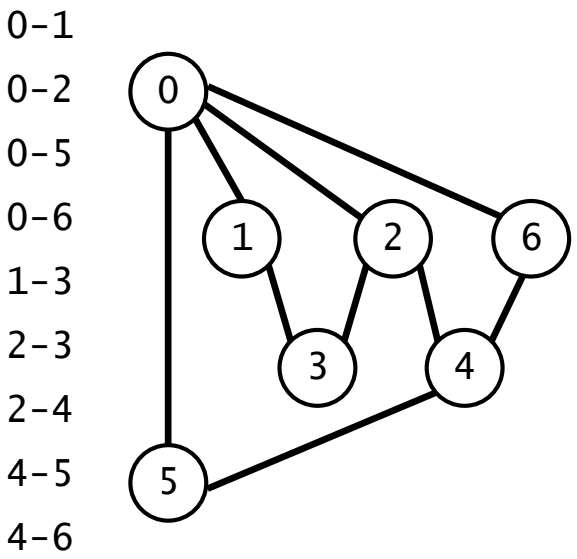
- مسئله. آیا یک گراف داده شده دوبخشی است؟
  - یک راه‌حل ساده بر اساس جستجوی عمقی!

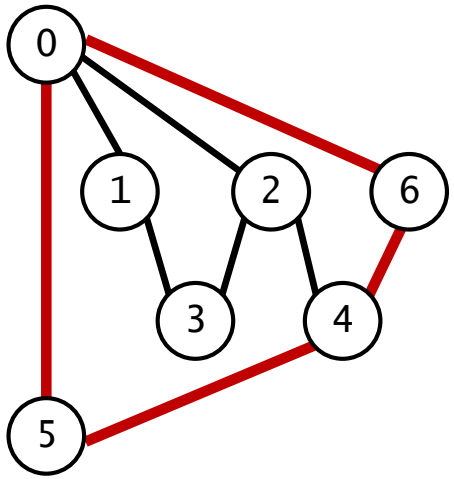




چالش ۲: بررسی وجود دور

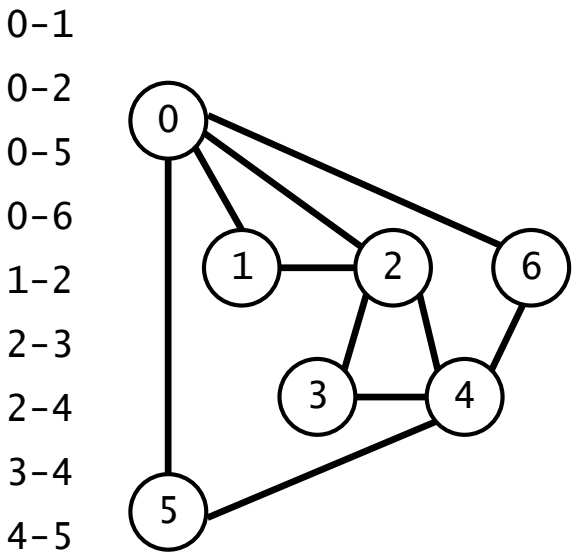
- مسئله. آیا یک گراف داده شده شامل دور است؟
  - یک راه‌حل ساده بر اساس جستجوی عمقی!

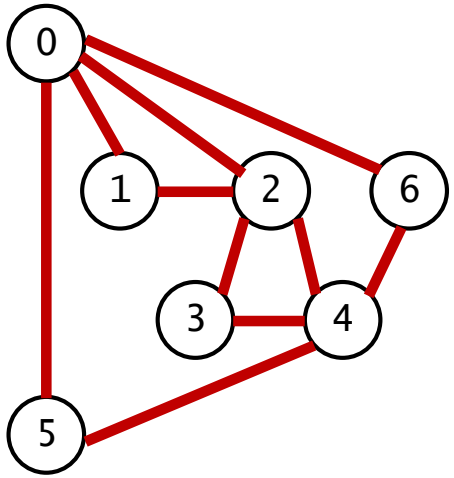




چالش ۳: گراف اویلری

- مسئله. یافتن دوری که از هر یال دقیقاً یک بار بگذرد.



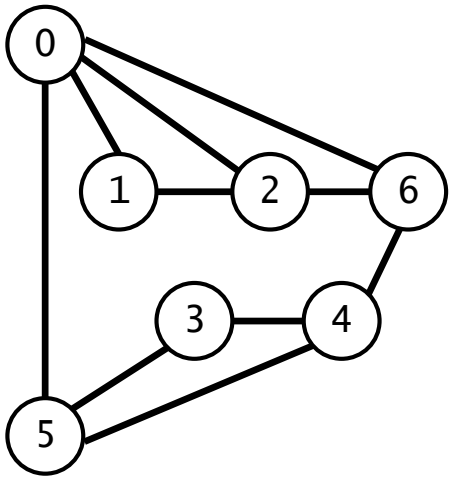


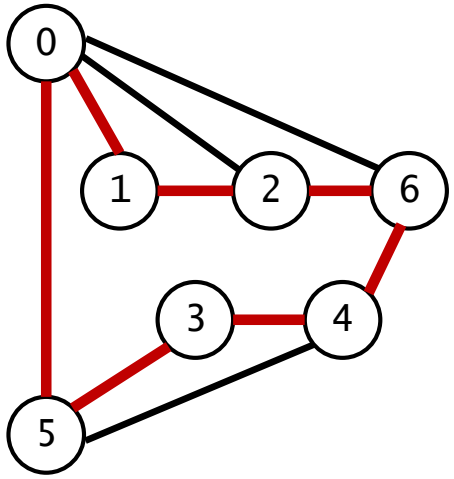
0-1-2-3-4-2-0-6-4-5-0

چالش ۴: گراف هامیلتونی

- مسئله. یافتن دوری که از هر رأس دقیقاً یک بار بگذرد.

0-1
0-2
0-5
0-6
1-2
2-6
3-4
3-5
4-5
4-6

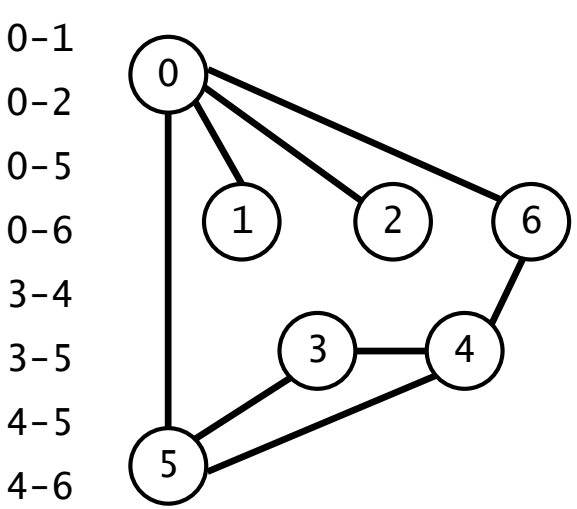


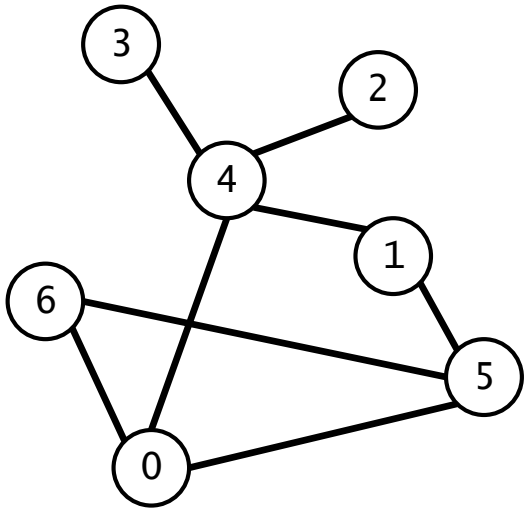


0-5-3-4-6-2-1-0

چالش ۵: بررسی یکریختی

- مسئله. آیا دو گراف داده شده یکریخت هستند؟





4↔0, 1↔3, 2↔2, 3↔6, 4↔5, 5↔0, 6↔1

چالش ۶: بررسی مسطح بودن گراف

- مسئله. آیا می‌توان یک گراف داده شده را در صفحه طوری ترسیم نمود که هیچ دو یالی همدیگر را قطع نکنند؟