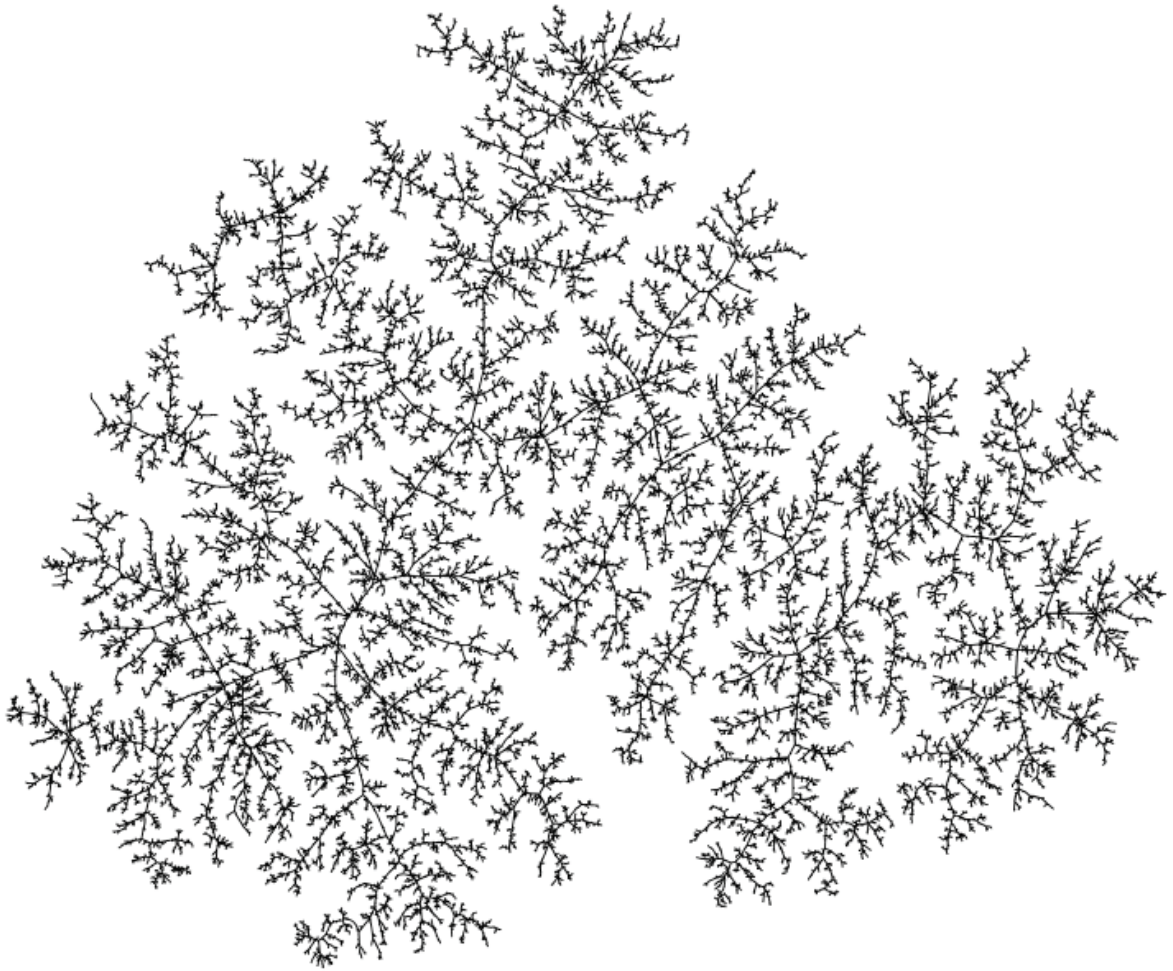
درخت پوشای کمینه

سید ناصر رضوی www.snrazavi.ir

۱۳۹۵

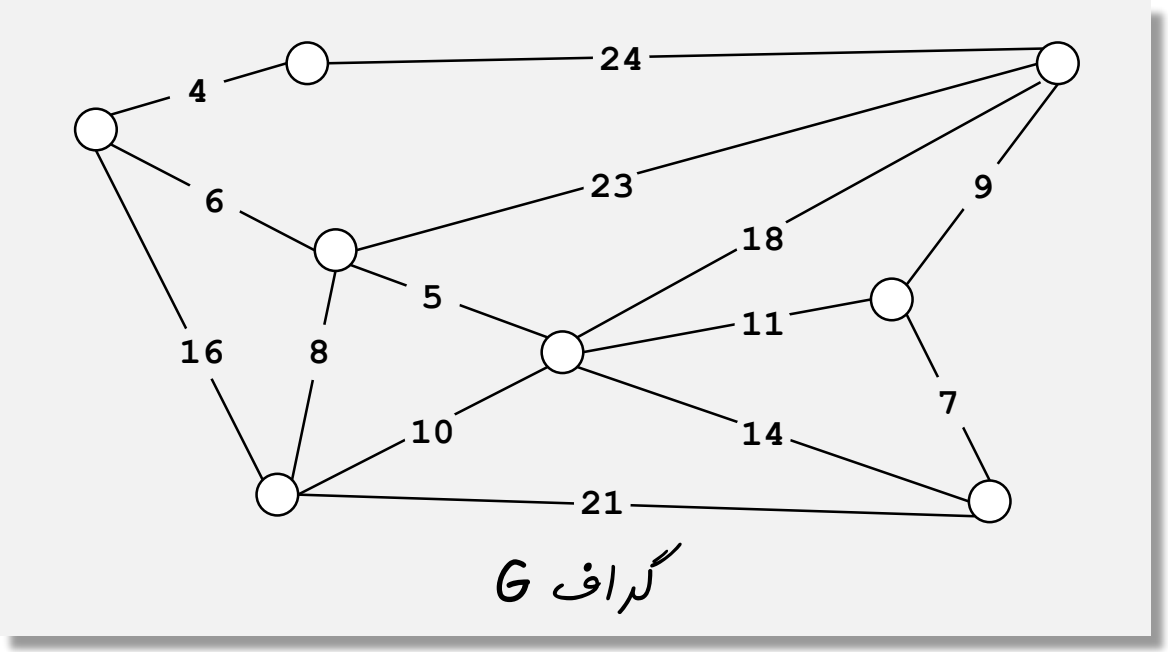
فهرست مطالب



- درخت پوشای کمینه.
  - معرفی
  - واسط گراف وزن‌دار
  - الگوریتم حریصانه
  - الگوریتم کروسکال
  - الگوریتم پریم

درخت پوشای کمینه

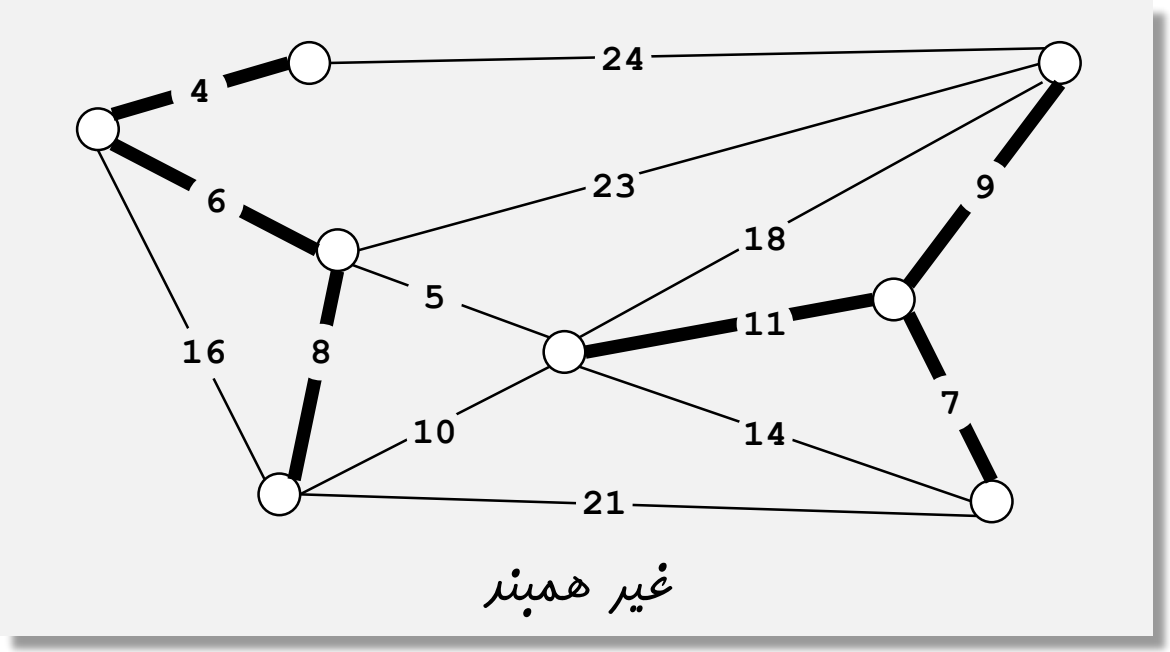
- ورودی. گراف بدون جهت و همبند G که وزن یالهای آن مثبت هستند.
- تعریف. یک درخت پوشا از گراف G یک زیرگراف همبند و بدون دور از گراف G است.
- هدف. یافتن یک درخت پوشا با کمترین وزن.



گراف G

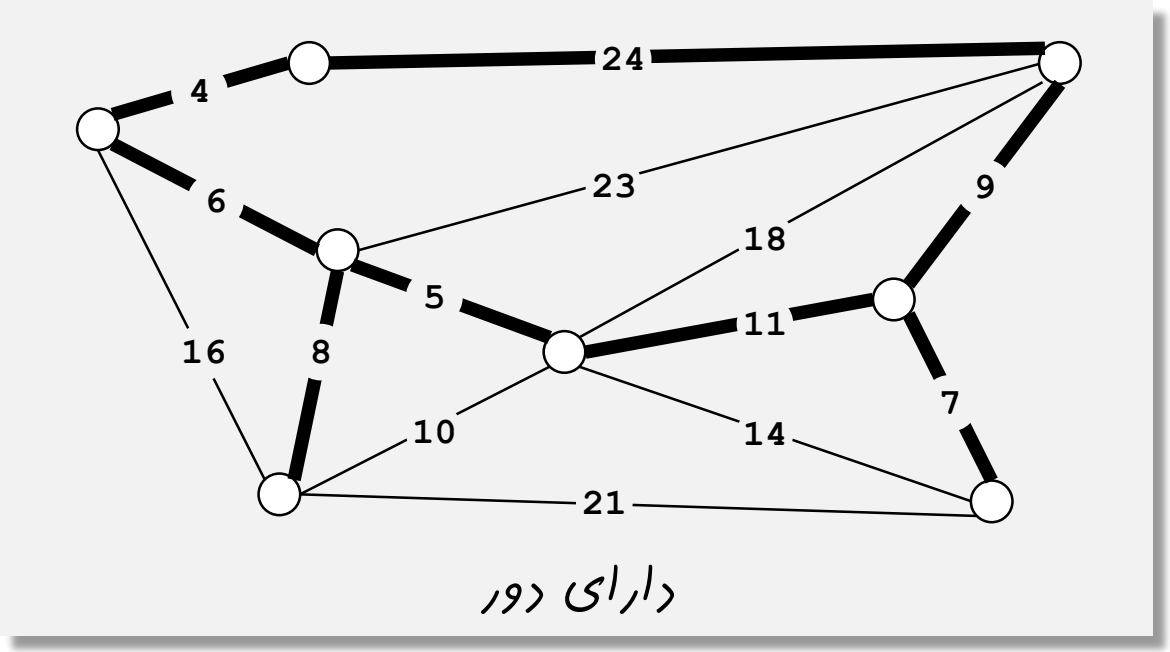
درخت پوشای کمینه

- ورودی. گراف بدون جهت و همبند G که وزن یالهای آن مثبت هستند.
- تعریف. یک درخت پوشا از گراف G یک زیرگراف همبند و بدون دور از گراف G است.
- هدف. یافتن یک درخت پوشا با کمترین وزن.



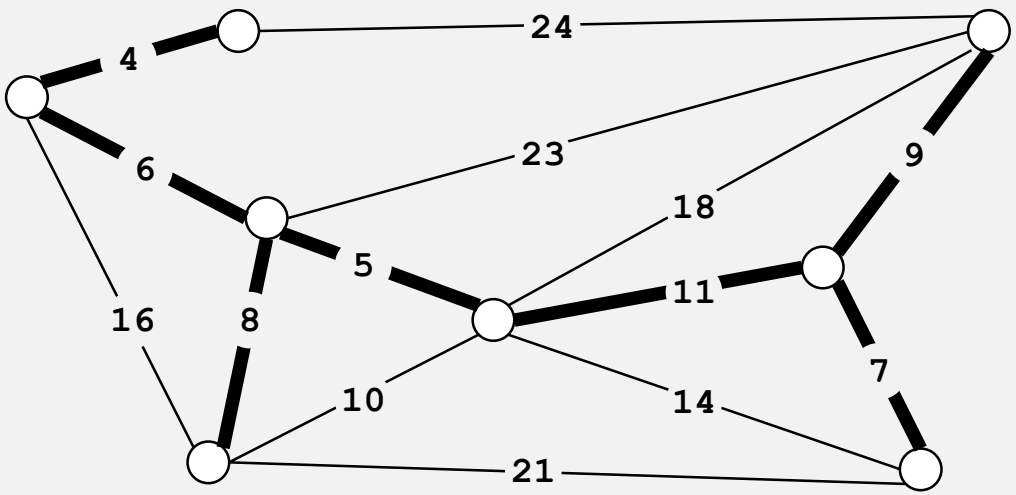
درخت پوشای کمینه

- ورودی. گراف بدون جهت و همبند G که وزن یالهای آن مثبت هستند.
- تعریف. یک درخت پوشا از گراف G یک زیرگراف همبند و بدون دور از گراف G است.
- هدف. یافتن یک درخت پوشا با کمترین وزن.



درخت پوشای کمینه

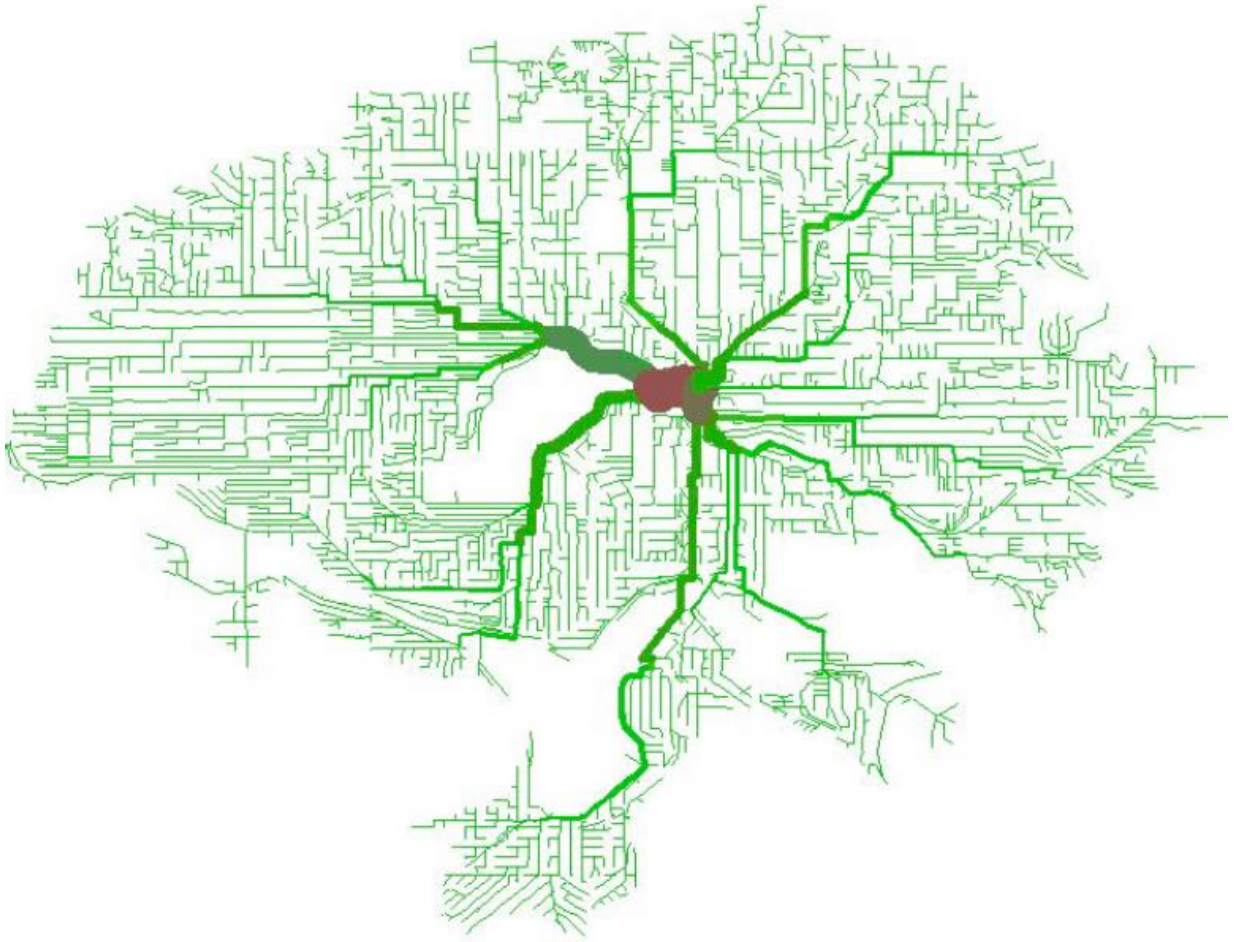
- ورودی. گراف بدون جهت و همبند G که وزن یالهای آن مثبت هستند.
- تعریف. یک درخت پوشا از گراف G یک زیرگراف همبند و بدون دور از گراف G است.
- هدف. یافتن یک درخت پوشا با کمترین وزن.



یک درخت پوشا با وزن ۵۰

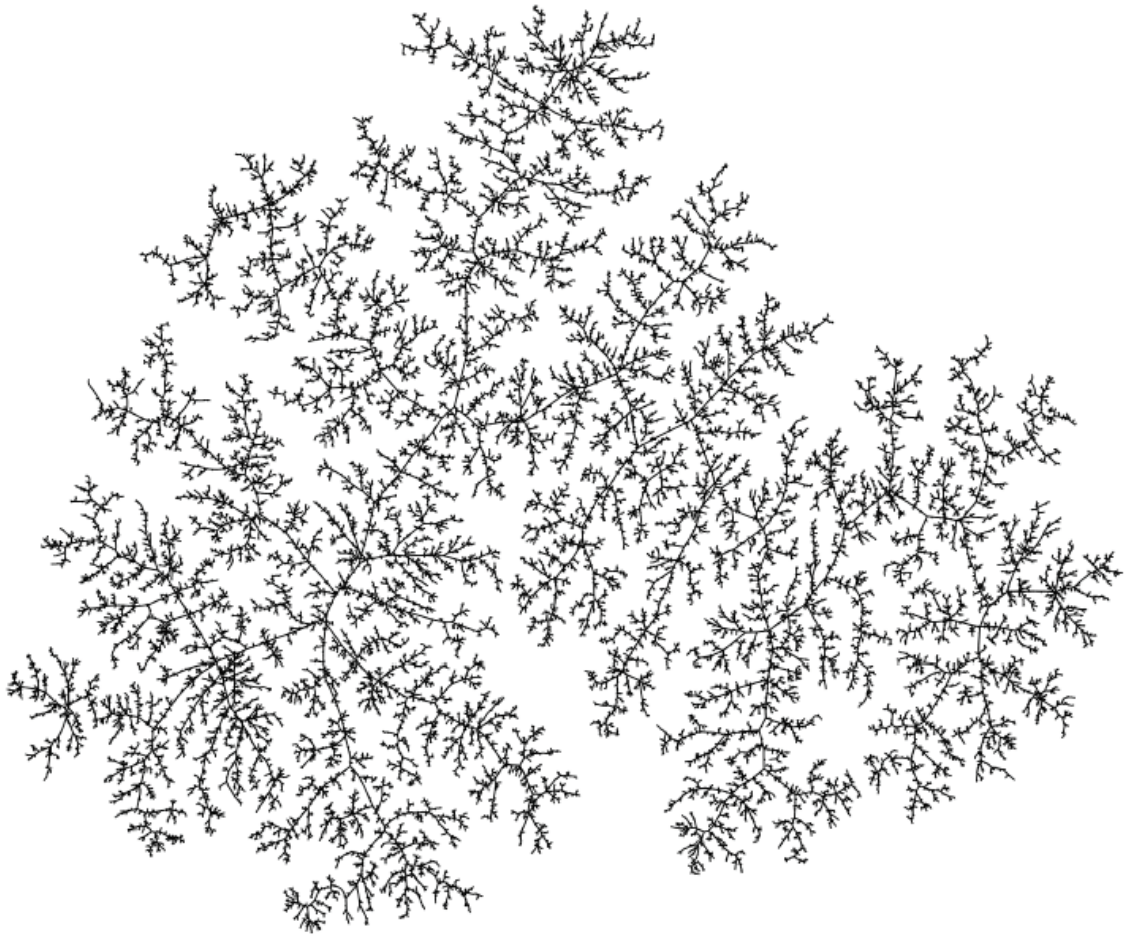
درخت پوشای کمینه

- درخت پوشای کمینه برای مسیرهای دوچرخه سواری در شهر سیاتل.



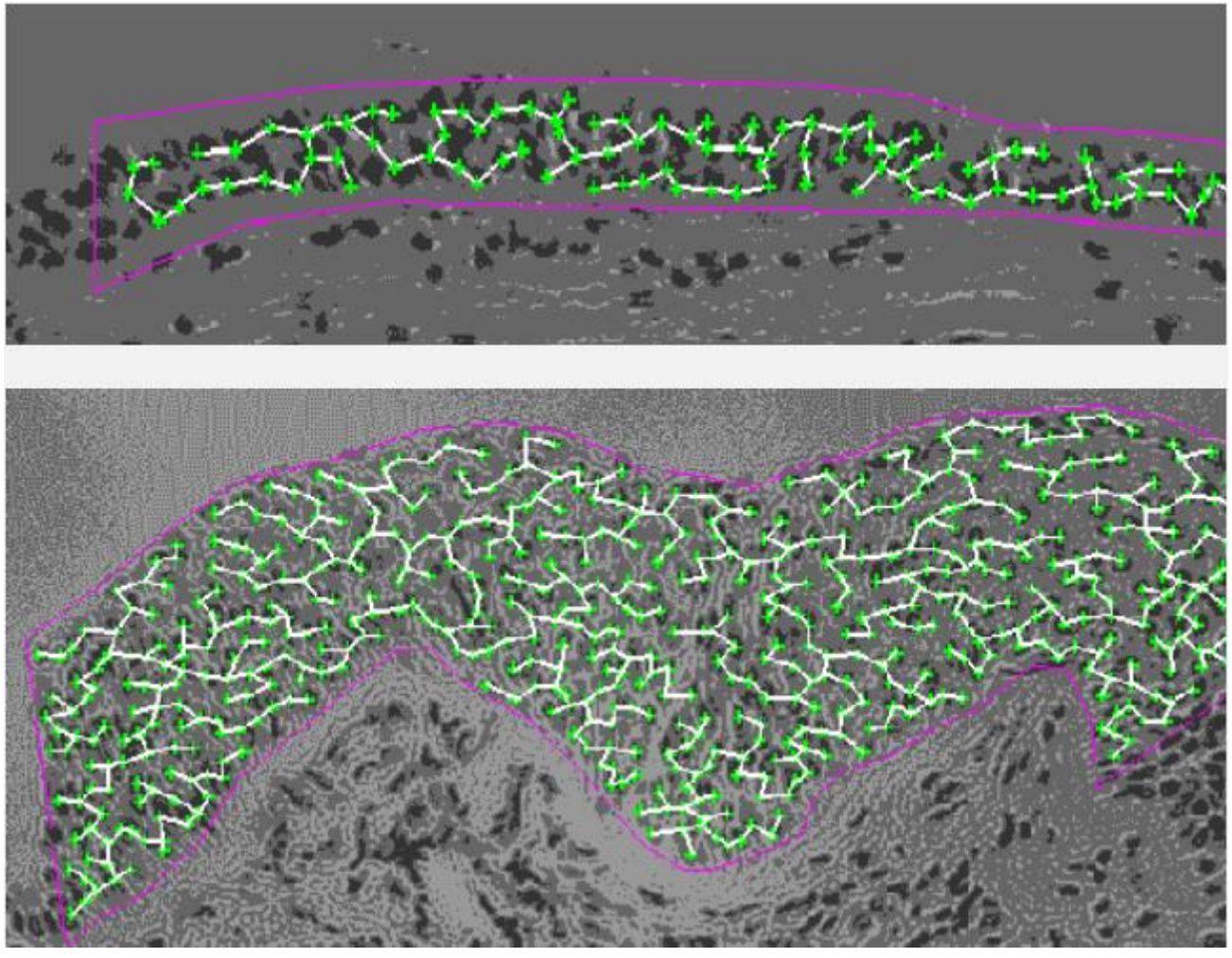
درخت پوشای کمینه

- درخت پوشای کمینه برای یک گراف تصادفی.



درخت پوشای کمینه

- درخت پوشای کمینه بیانگر آرایش هسته‌ی سلول‌ها در یک بافت پوستی برای پژوهش در مورد سرطان.



کاربردها

- برخی از کاربردهای درخت پوشای کمینه.
  - طراحی شبکه (ارتباطی، الکتریکی، کامپیوتری، کابلی، جاده‌ای)
  - الگوریتم‌های تقریبی برای مسائل ان-پی کامل (فروشنده‌ی دوره‌گرد)
  - یافتن شبکه‌ی راه‌ها در تصاویر هوایی و ماهواره‌ای
  - تشخیص چهره به صورت بلادرنگ
  - ...

واسط گراف وزن‌دار

واسط گراف وزن‌دار

- واسط کلاس یال وزن‌دار.

```
public class Edge implements Comparable<Edge>
```

`Edge(int v, int w, double weight)`	ایجاد یال وزن‌دار v–w
`int either()`	برگرداندن یکی از نقاط انتهایی
`int other(int v)`	برگرداندن نقطه‌ی انتهایی دیگر
`int compareTo(Edge that)`	مقایسه‌ی این یال با یک یال دیگر
`double weight()`	برگرداندن وزن یال
`String toString()`	نمایش یال به صورت رشته‌ای

یال وزن‌دار: پیاده‌سازی


```
public class Edge implements Comparable<Edge> {
   private final int v, w;
   private final double weight;

   public Edge(int v, int w, double weight) {
      this.v = v;
      this.w = w;
      this.weight = weight;
   }

   public int either() {  return v;  }

   public int other(int vertex) {
      if (vertex == v) return w;
      else             return v;
   }

   public int compareTo(Edge that) {
      if      (this.weight < that.weight) return -1;
      else if (this.weight > that.weight) return +1;
      else                                return  0;
   }
}
```

سازنده

یکی از رئوس

رأس دیگر

مقایسه‌ی دو یال
بر مبنای وزن آنها

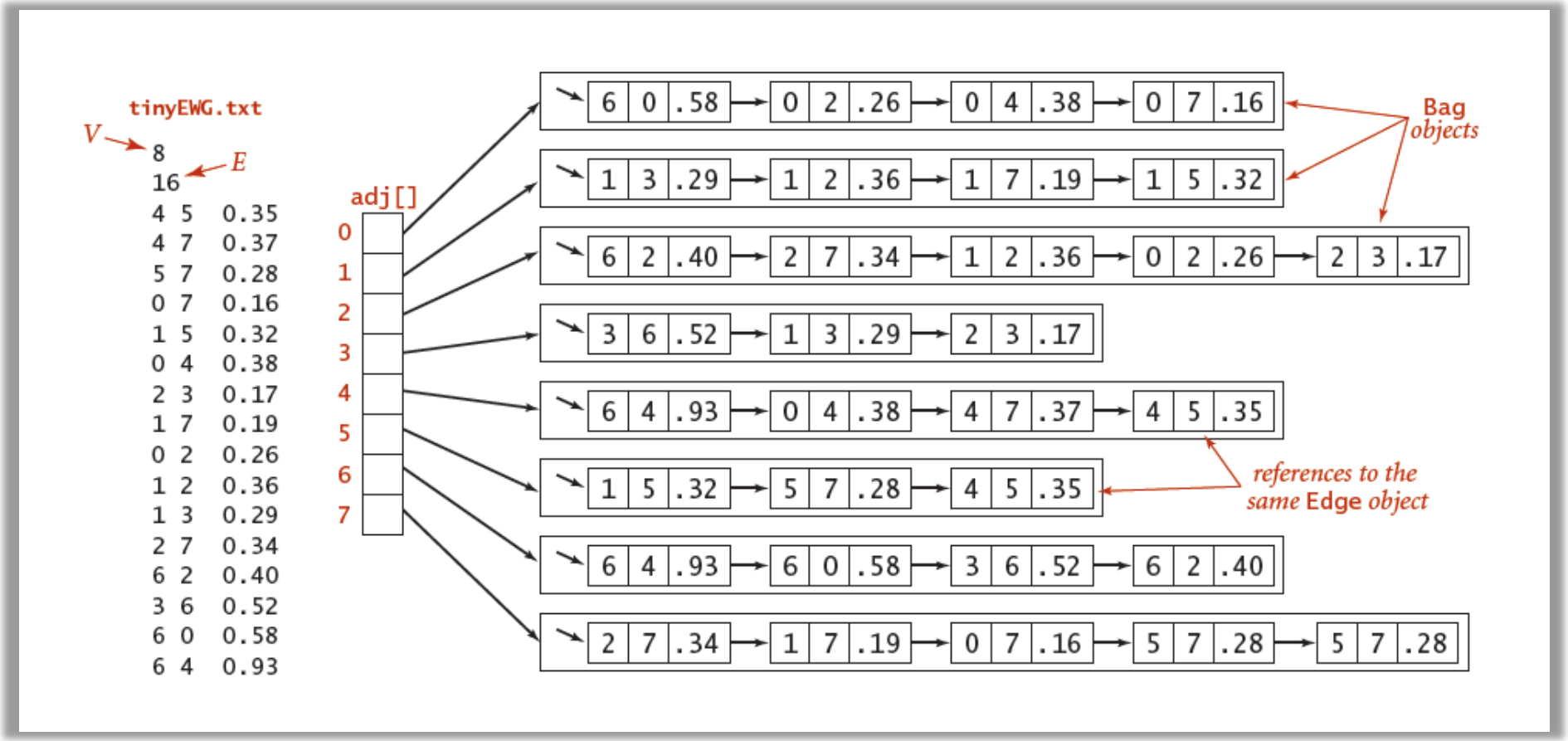
واسط گراف وزن‌دار

public class	EdgeWeightedGraph	
	EdgeWeightedGraph(int V)	ایجاد یک گراف تهی با V رأس
	EdgeWeightedGraph(In in)	ایجاد یک گراف از جریان ورودی
void	addEdge(Edge e)	افزودن یال e به گراف
Iterable<Edge>	adj(int v)	برگرداندن یالهای متلاقی با رأس v
Iterable<Edge>	edges()	برگرداندن همه یالهای گراف
int	V()	برگرداندن تعداد رئوس
int	E()	برگرداندن تعداد یالها
String	toString()	نمایش گراف وزن‌دار به صورت یک رشته

قرارداد. در این پیاده‌سازی داشتن طوقه و یالهای موازی مجاز است.

گراف وزن‌دار: پیاده‌سازی لیست همسایگی

- به ازای هر رأس، یالهای متلاقی با آن در یک لیست ذخیره می‌شوند.



گراف وزن‌دار: پیاده‌سازی جاوا


```
public class EdgeWeightedGraph {

   private final int V;
   private final Bag<Edge>[] adj;

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

   public void addEdge(Edge e) {
      int v = e.either(), w = e.other(v);
      adj[v].add(e);
      adj[w].add(e);
   }

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

}
```

سازنده

اضافه کردن یال به هر دو لیست همسایگی

واسط درخت پوشای کمینه

public class	MST	
	MST(EdgeWeightedGraph G)	سازنده
Iterable<Edge>	edges()	برگرداندن یالهای درخت پوشای کمینه
double	weight()	برگرداندن وزن درخت پوشای کمینه

tinyEWG.txt

```
8
16
4 5  0.35
4 7  0.37
5 7  0.28
0 7  0.16
1 5  0.32
0 4  0.38
2 3  0.17
1 7  0.19
0 2  0.26
1 2  0.36
1 3  0.29
2 7  0.34
6 2  0.40
3 6  0.52
6 0  0.58
6 4  0.93
```



```
% java MST tinyEWG.txt
0-7 0.16
1-7 0.19
0-2 0.26
2-3 0.17
5-7 0.28
4-5 0.35
6-2 0.40
1.81
```

واسط درخت پوشای کمینه

public class	MST	
	MST(EdgeWeightedGraph G)	سازنده
Iterable<Edge>	edges()	برگرداندن یالهای درخت پوشای کمینه
double	weight()	برگرداندن وزن درخت پوشای کمینه

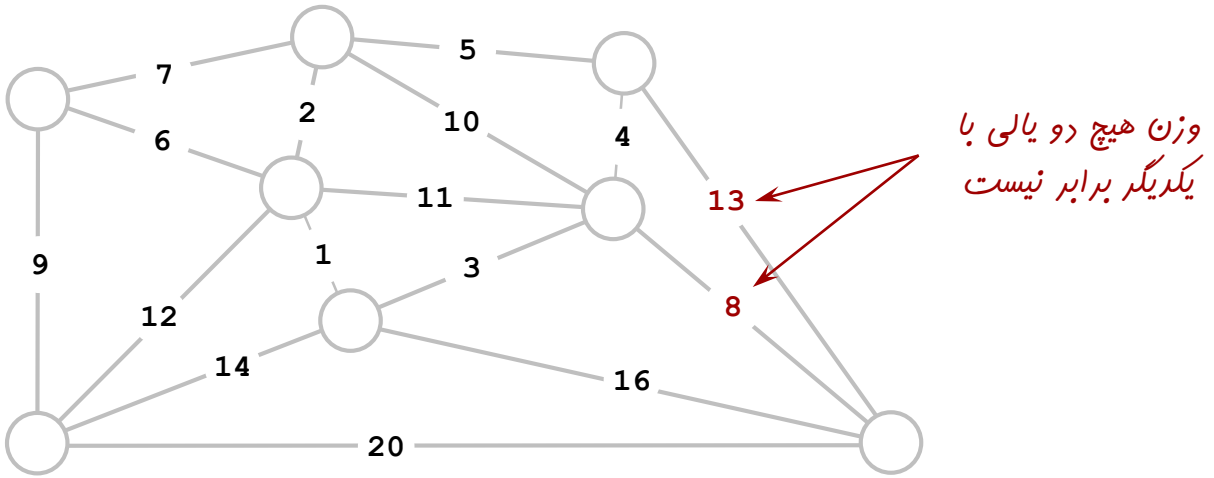
```
Public static void main(String[] args)
{
   In in = new In(args[0]);
   EdgeWeightedGraph G = new EdgeWeightedGraph(in);
   MST mst = new MST(G);
   for (Edge e : mst.edges())
      StdOut.println(e);
   StdOut.printf("%.2f\n", mst.weight());
}
```

```
% java MST tinyEWG.txt
0-7 0.16
1-7 0.19
0-2 0.26
2-3 0.17
5-7 0.28
4-5 0.35
6-2 0.40
1.81
```

الگوریتم حریصانه

ویژگی برش

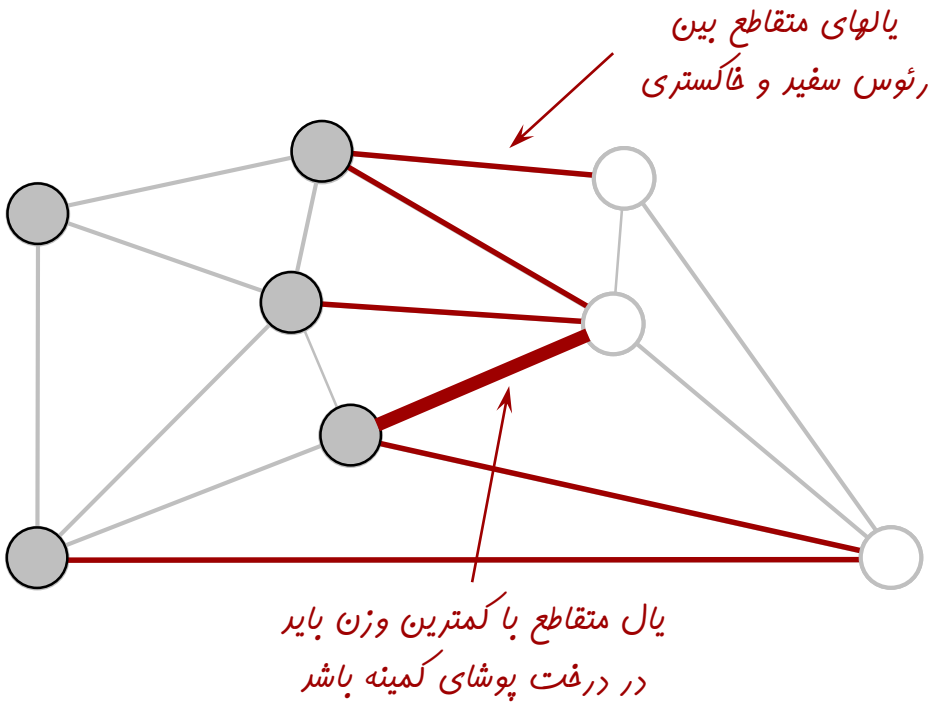
- فرضیات ساده کننده.
  - وزن یالها متفاوتند.
  - گراف همبند است.
- نتیجه. درخت پوشای کمینه وجود دارد و منحصر به فرد است.



ویژگی برش

- تعریف. یک برش از گراف رئوس آن گراف را به دو مجموعه (غیرتهی) افراز می‌کند.
- تعریف. یک یال متقاطع یک رأس از یک مجموعه را به یک رأس از مجموعه دیگر متصل می‌کند.

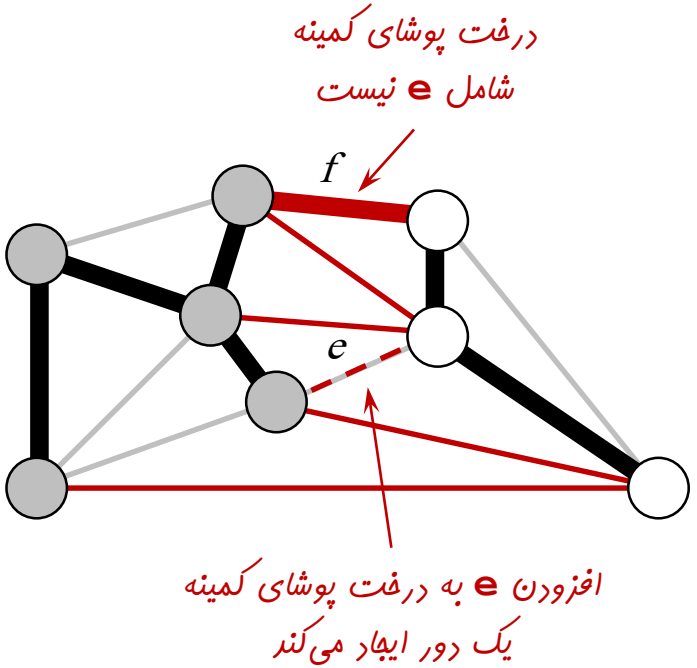
- ویژگی برش. به ازای هر برش، یال متقاطع با کمترین وزن در درخت پوشای کمینه قرار دارد.



ویژگی برش: اثبات درستی

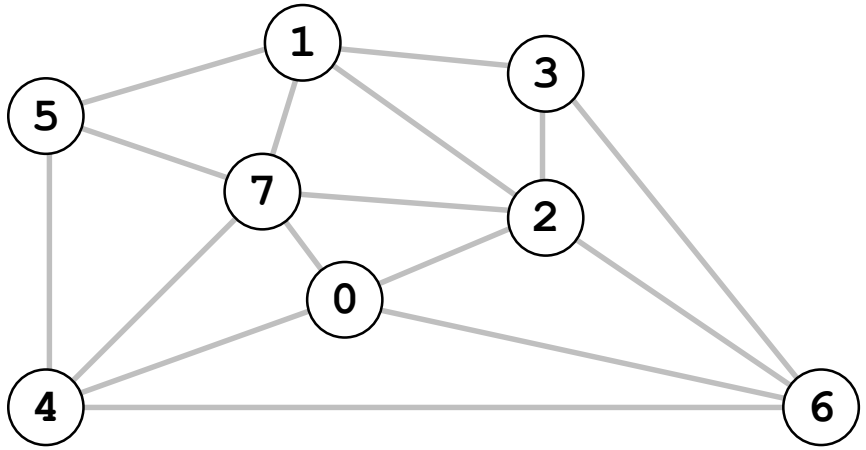
- تعریف. یک برش از گراف رئوس آن گراف را به دو مجموعه (غیرتهی) افراز می‌کند.
- تعریف. یک یال متقاطع یک رأس از یک مجموعه را به یک رأس از مجموعه دیگر متصل می‌کند.

- ویژگی برش. به ازای هر برش، یال متقاطع با کمترین وزن در درخت پوشای کمینه قرار دارد.
- اثبات. فرض کنید **e** یک یال متقاطع با کمترین وزن در یک برش باشد.
  - فرض کنید **e** در درخت پوشای کمینه نباشد.
  - افزودن **e** به درخت پوشای کمینه یک دور ایجاد می‌کند.
  - در این دور یک یال دیگر مانند **f** باید یک یال متقاطع باشد.
  - درخت حاصل از برداشتن **f** و افزودن **e** همچنان یک درخت پوشا است.
  - از آنجا که وزن **e** از وزن **f** کمتر است، این درخت پوشای جدید وزن کمتری دارد.
  - تناقض!



الگوریتم حریصانه: اجرای نمایشی

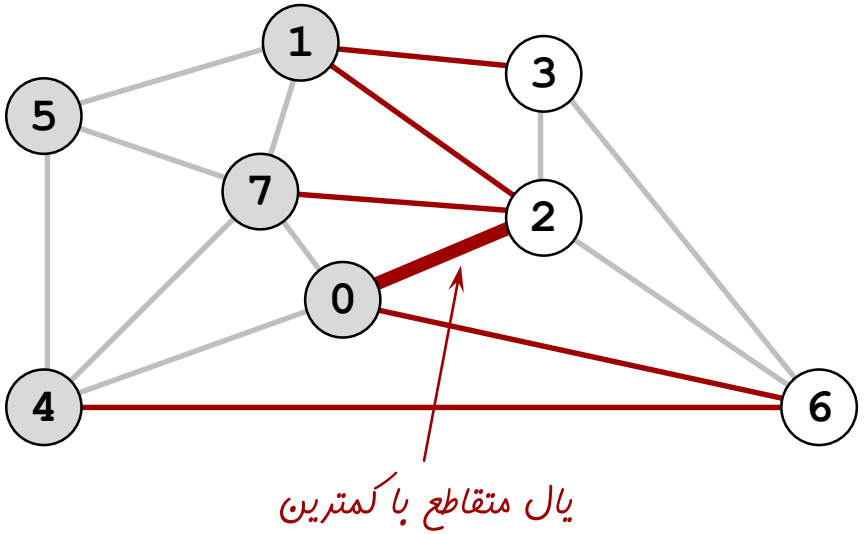
- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - 1$ یال سیاه شوند.




```
0-7   0.16
2-3   0.17
1-7   0.19
0-2   0.26
5-7   0.28
1-3   0.29
1-5   0.32
2-7   0.34
4-5   0.35
1-2   0.36
4-7   0.37
0-4   0.38
6-2   0.40
3-6   0.52
6-0   0.58
6-4   0.93
```

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا **V - 1** یال سیاه شوند.



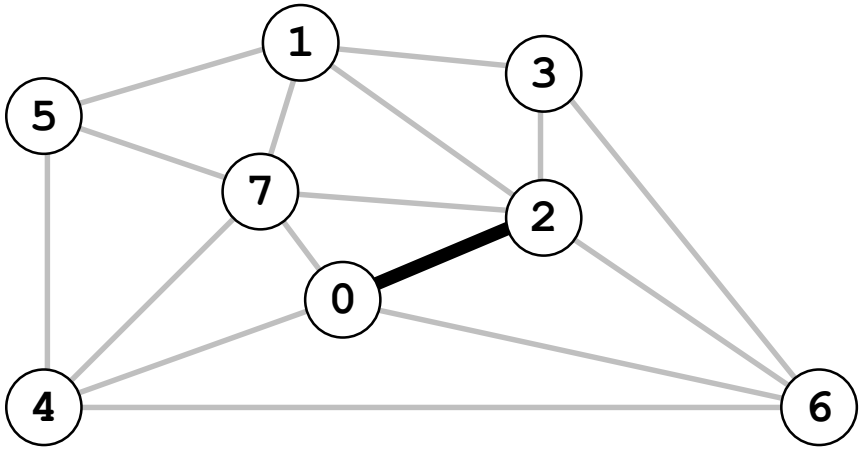
یال‌های متقاطع
(مرتب شده بر اساس وزن)

در MST →

0-2	0.26
1-3	0.29
2-7	0.34
1-2	0.36
6-0	0.58
6-4	0.93

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.

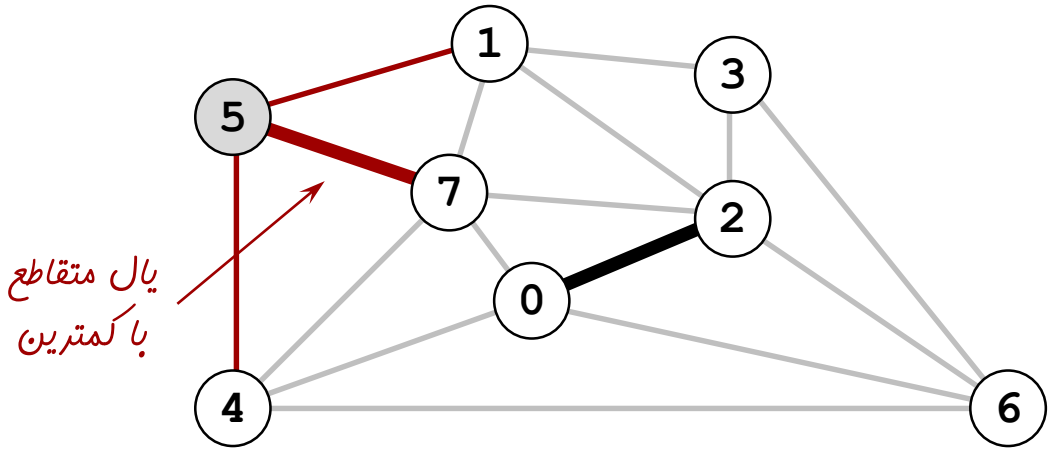


MST

0-2

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $V - 1$ یال سیاه شوند.



یال‌های متقاطع
(مرتب شده بر اساس وزن)

در MST →

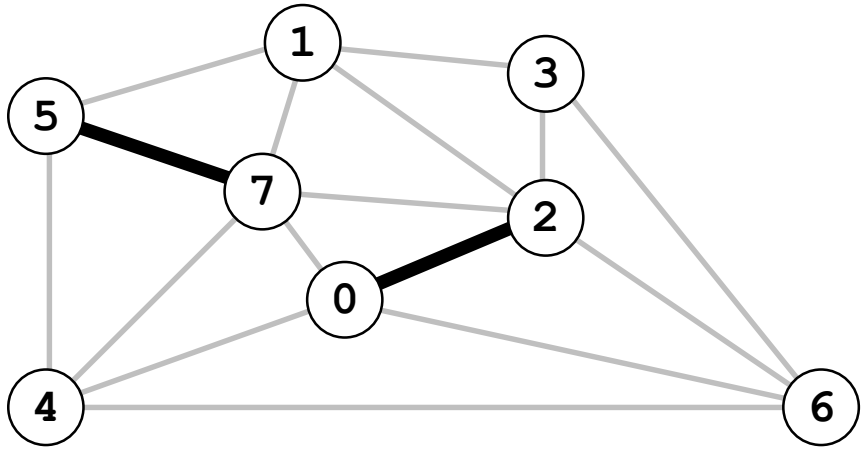
5-7	0.28
1-5	0.32
4-5	0.35

MST

0-2

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.

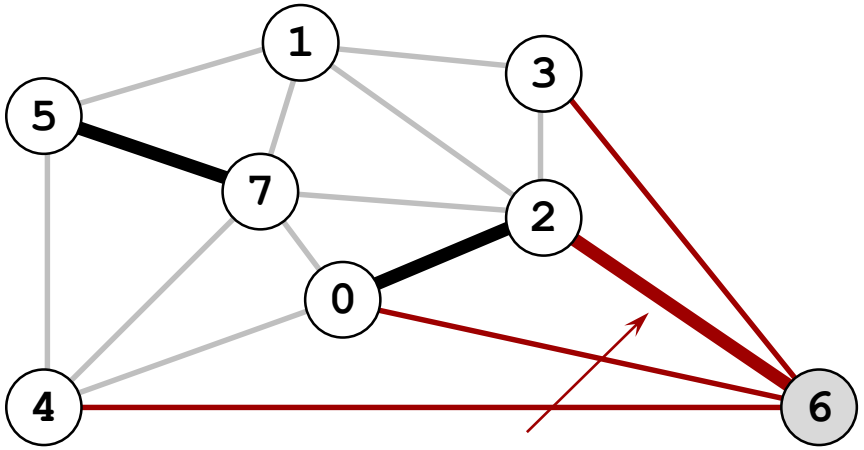


MST

0-2 5-7

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - 1$ یال سیاه شوند.



MST

0-2 5-7

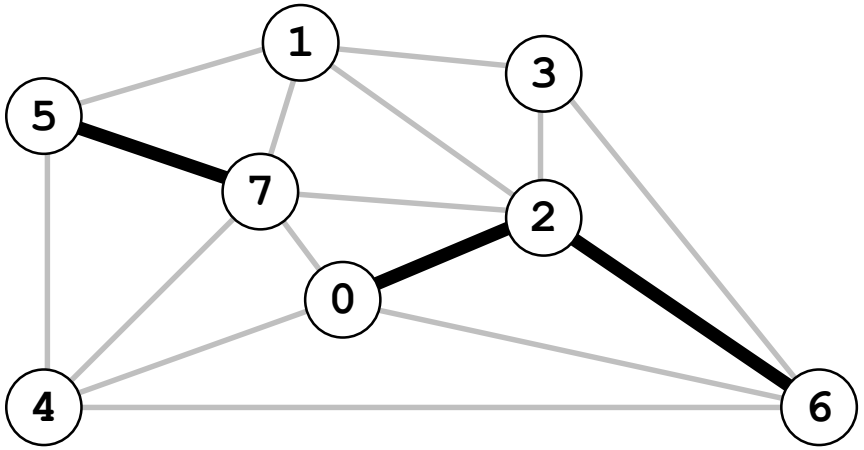
یال‌های متقاطع
(مرتب شده بر اساس وزن)

در MST ⟵

6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.

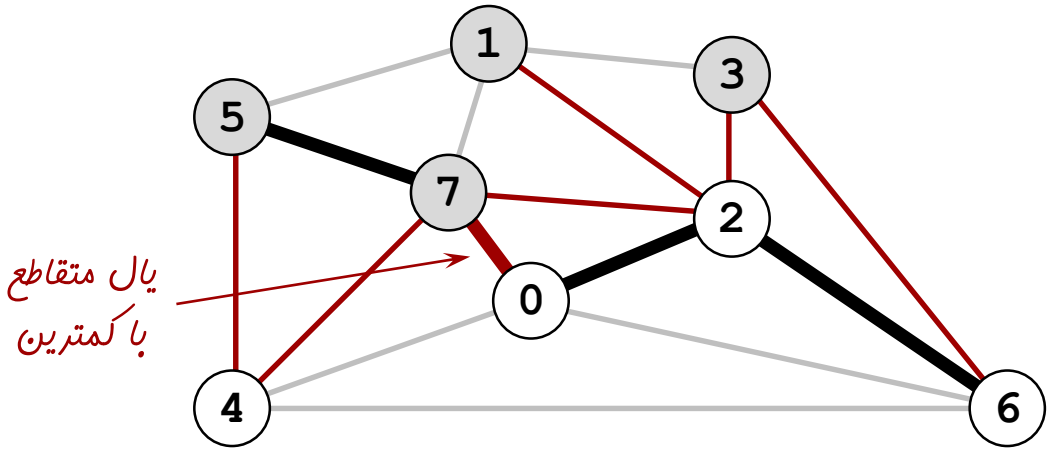


MST

0-2 5-7 6-2

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا **V - 1** یال سیاه شوند.



یال‌های متقاطع
(مرتب شده بر اساس وزن)

MST در

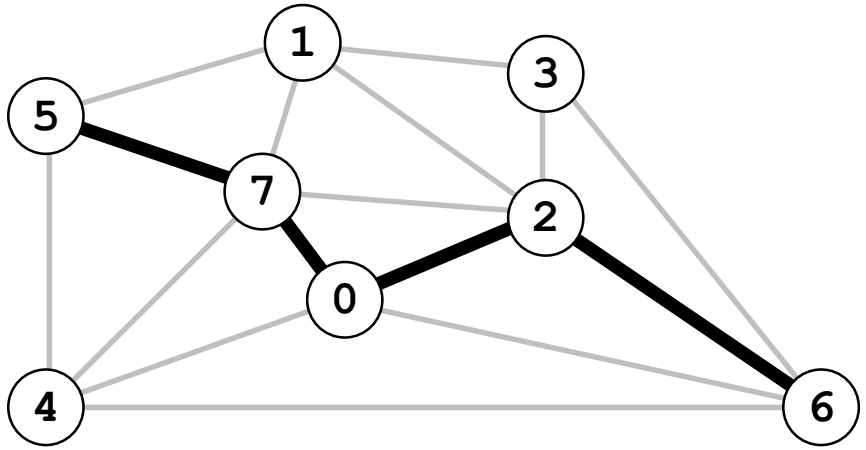
0-7	0.16
2-3	0.17
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
3-6	0.52

MST

0-2 5-7 6-2

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.

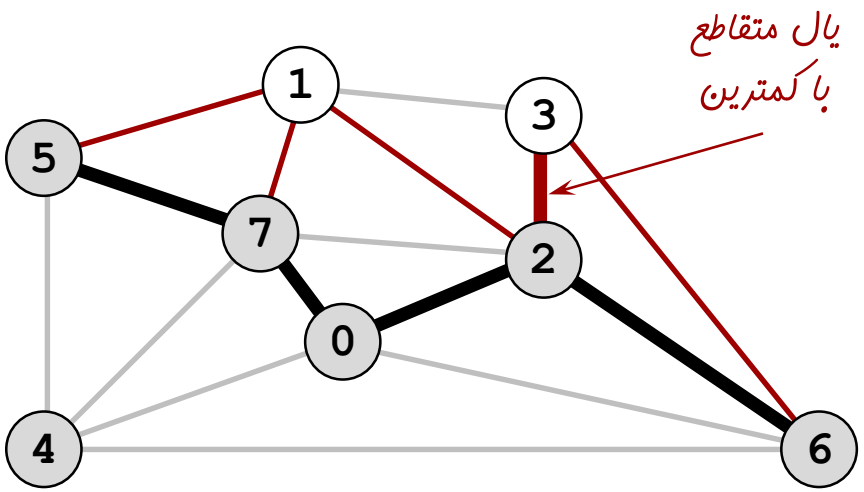


MST

0-2 5-7 6-2 0-7

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.



یال‌های متقاطع
(مرتب شده بر اساس وزن)

MST در ←

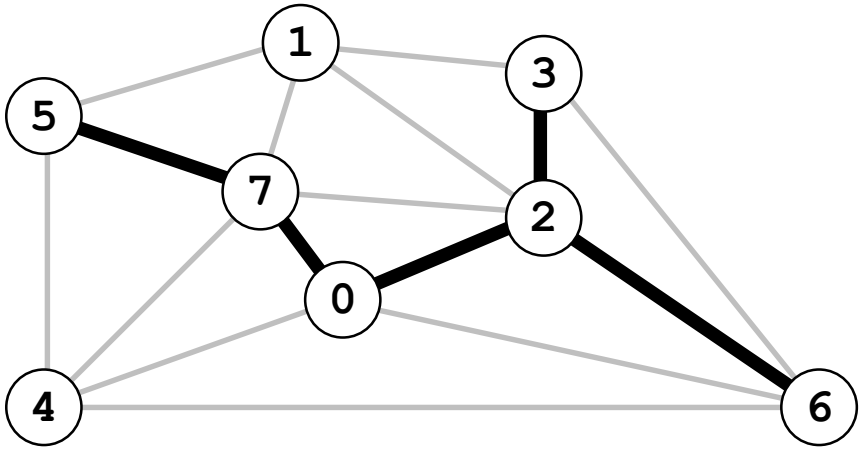
2-3	0.17
1-7	0.19
1-5	0.32
1-2	0.36
3-6	0.52

MST

0-2 5-7 6-2 0-7

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.

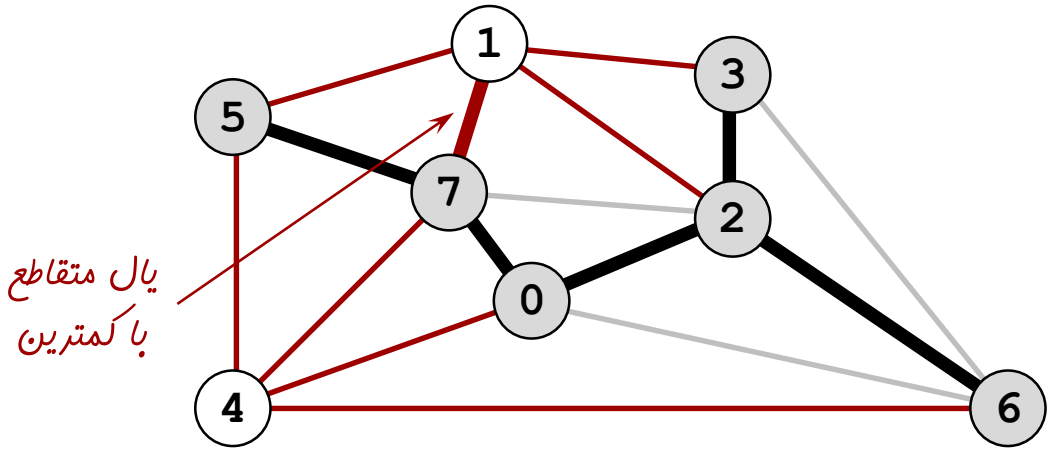


MST

0-2 5-7 6-2 0-7 2-3

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا **V - 1** یال سیاه شوند.



یال‌های متقاطع
(مرتب شده بر اساس وزن)

MST در ←

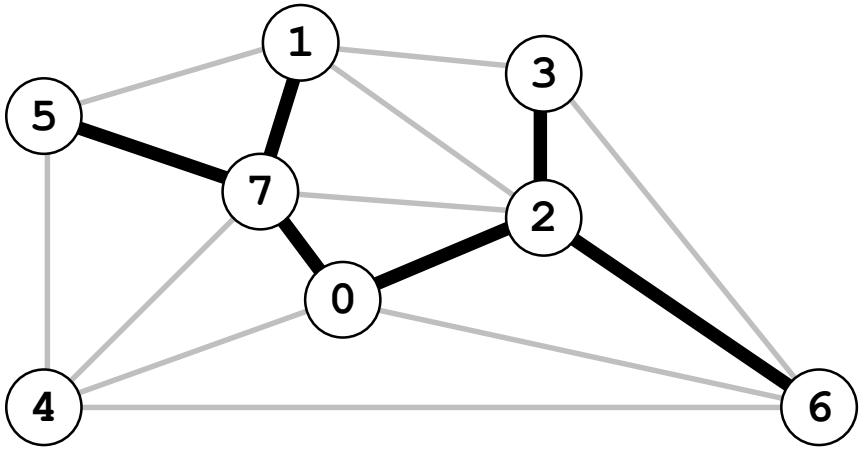
1-7	0.19
1-3	0.29
1-5	0.32
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-4	0.93

MST

0-2 5-7 6-2 0-7 2-3

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.

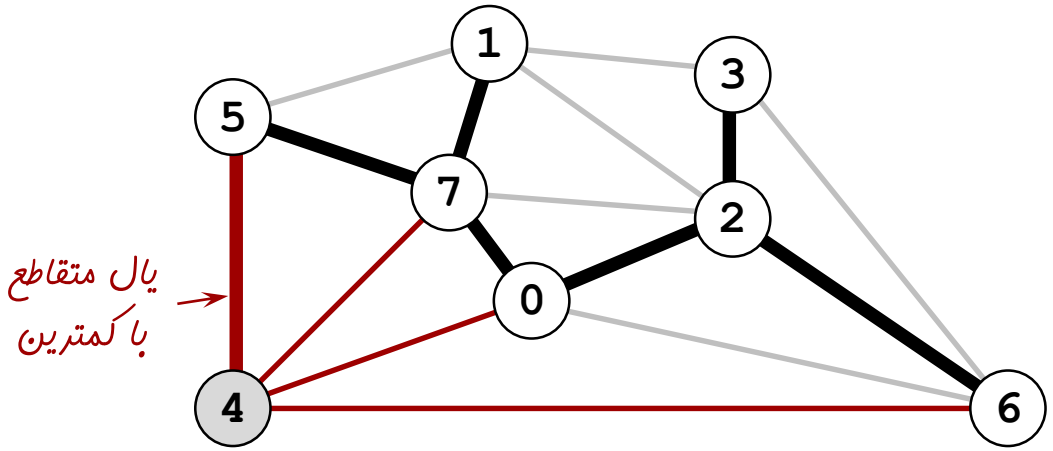


MST

0-2 5-7 6-2 0-7 2-3 1-7

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - 1$ یال سیاه شوند.



یال‌های متقاطع
(مرتب شده بر اساس وزن)

MST در ←

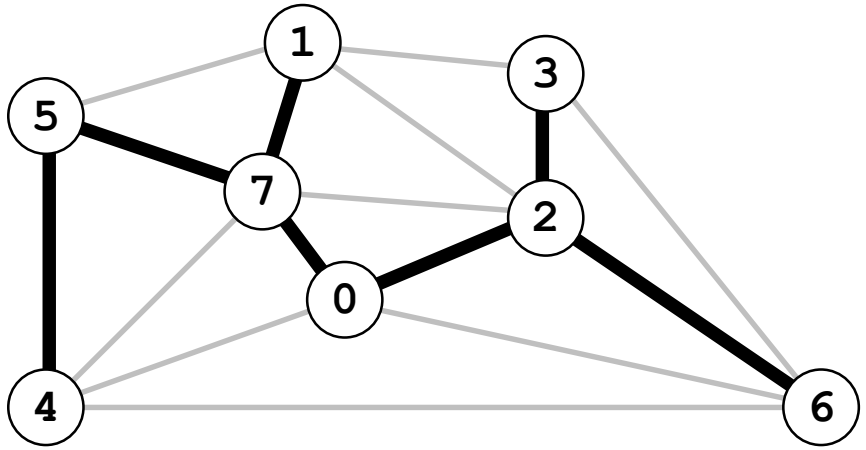
4-5	0.35
4-7	0.37
0-4	0.38
6-4	0.93

MST

0-2 5-7 6-2 0-7 2-3 1-7

الگوریتم حریصانه: اجرای نمایشی

- الگوریتم حریصانه.
  - با یال‌هایی که در ابتدا همگی خاکستری هستند شروع کن.
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن.
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.

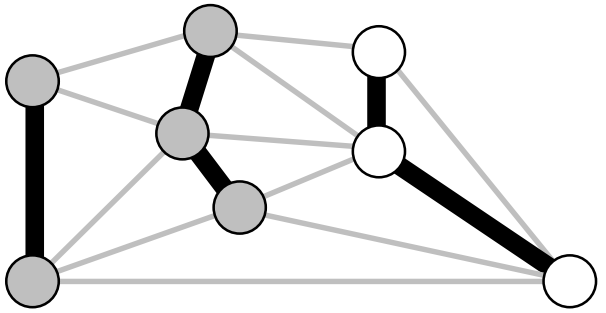


MST

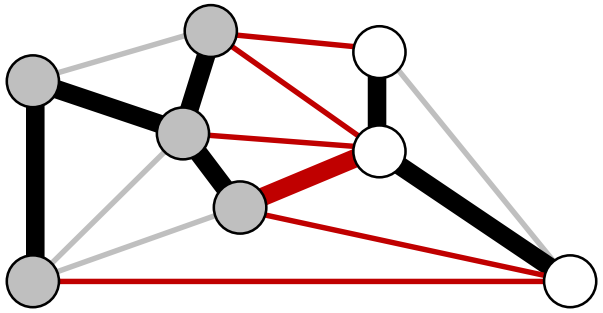
0-2 5-7 6-2 0-7 2-3 1-7 4-5

الگوریتم حریصانه: اثبات درستی

- گزاره. خروجی الگوریتم حریصانه یک درخت پوشای کمینه است.
- اثبات.
  - هر یالی که سیاه می‌شود به درخت پوشای کمینه تعلق دارد (بر اساس ویژگی برش).
  - اگر تعداد یالهای سیاه کمتر از $\mathbf{V - 1}$ باشد، یک برش بدون یال متقاطع سیاه وجود دارد. (برشی را در نظر بگیرید که رئوس آن یک مولفه‌ی همبند تشکیل می‌دهند)



تعداد یال‌های سیاه کمتر از $V - 1$



یک برش بدون یال متقاطع سیاه

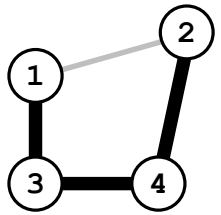
الگوریتم حریصانه: پیاده سازی کارا

- گزاره. خروجی الگوریتم زیر یک درخت پوشای کمینه است.
  - با یالهایی که در ابتدا همگی خاکستری هستند شروع کن؛
  - هر بار یک برش بدون یال متقاطع سیاه پیدا کن، و یال با کمترین وزن آن را سیاه کن؛
  - عمل بالا را آن قدر تکرار کن تا $\mathbf{V} - \mathbf{1}$ یال سیاه شوند.

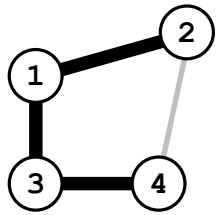
- پیاده‌سازی کارا. انتخاب برش چگونه باشد؟ یال با وزن کمینه چگونه پیدا شود؟
  - مثال ۱. الگوریتم کروسکال [با ما باشید]
  - مثال ۲. الگوریتم پریم [با ما باشید]
  - مثال ۳. الگوریتم برووکا

برداشتن فرضیات ساده کننده

- س. اگر وزن همه یال‌ها متفاوت نباشد چه می‌شود؟
- ج. الگوریتم حریصانه با وجود یال‌هایی که دارای وزن برابر هستند، هنوز به درستی کار می‌کند! (اثبات ارائه شده نیاز به اصلاح دارد)

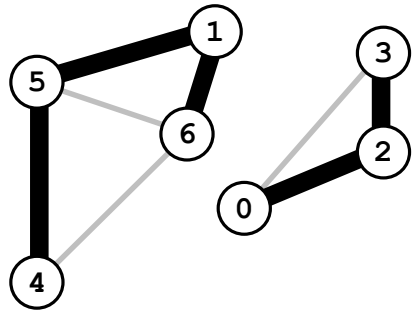



```
1  2  1.00
1  3  0.50
2  4  1.00
3  4  0.50
```




```
1  2  1.00
1  3  0.50
2  4  1.00
3  4  0.50
```

- س. اگر گراف همبند نباشد چه می‌شود؟
- ج. خروجی یک جنگل پوشای کمینه است = درخت پوشای کمینه برای هر مولفه.

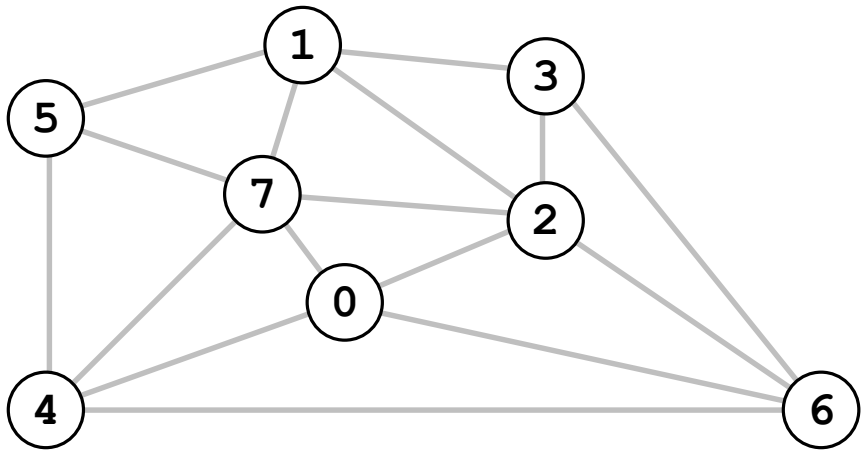



```
4  5  0.61
4  6  0.62
5  6  0.88
1  5  0.11
2  3  0.35
0  3  0.60
1  6  0.10
0  2  0.22
```

الگوریتم کروسکال

الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.



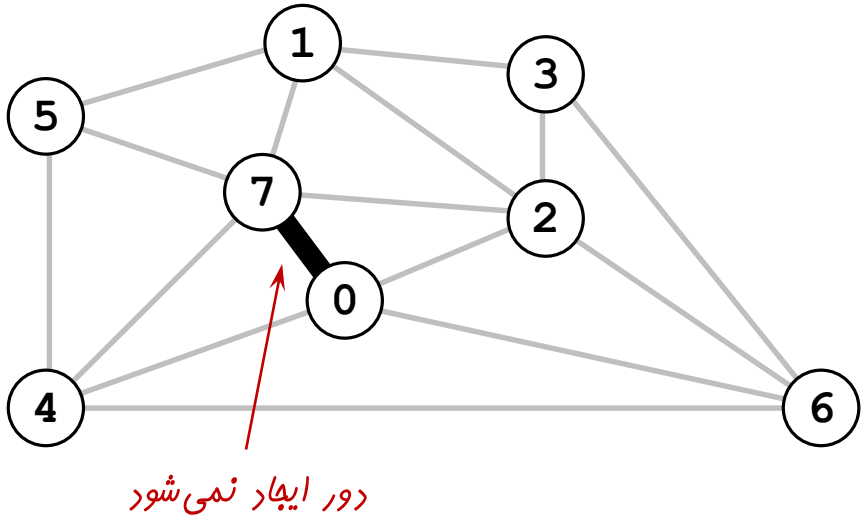
یالهای گراف که بر اساس وزن مرتب شده‌اند

```
0-7  0.16
2-3  0.17
1-7  0.19
0-2  0.26
5-7  0.28
1-3  0.29
1-5  0.32
2-7  0.34
4-5  0.35
1-2  0.36
4-7  0.37
0-4  0.38
6-2  0.40
3-6  0.52
6-0  0.58
6-4  0.93
```

الگوریتم کروسکال: اجرای نمایشی

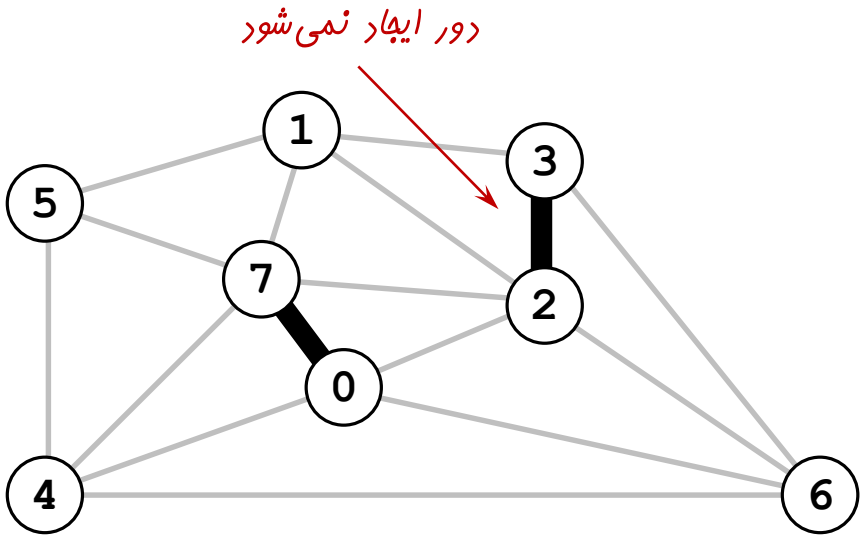
- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

MST در ⟵ 0-7 0.16



الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

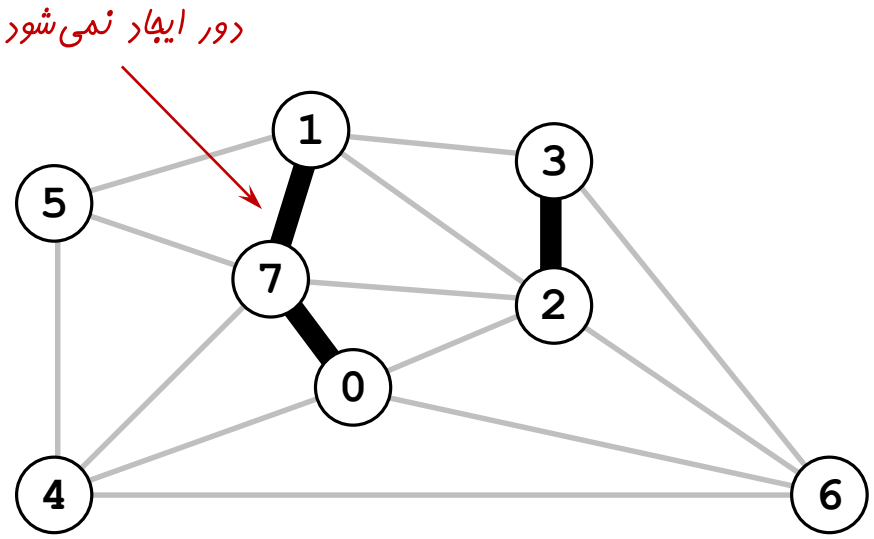


در **MST** ⟵

```
0-7   0.16
2-3   0.17
```

الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.



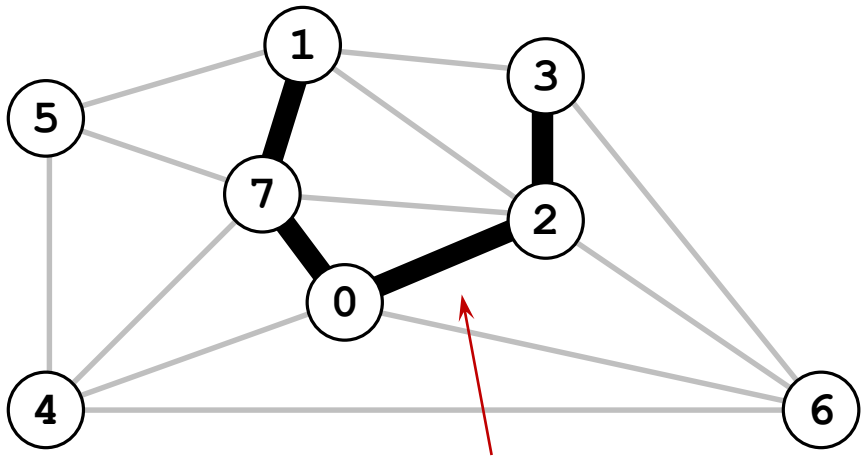
در **MST** ⟵

```
0-7  0.16
2-3  0.17
1-7  0.19
```

الگوریتم کروسکال: اجرای نمایشی

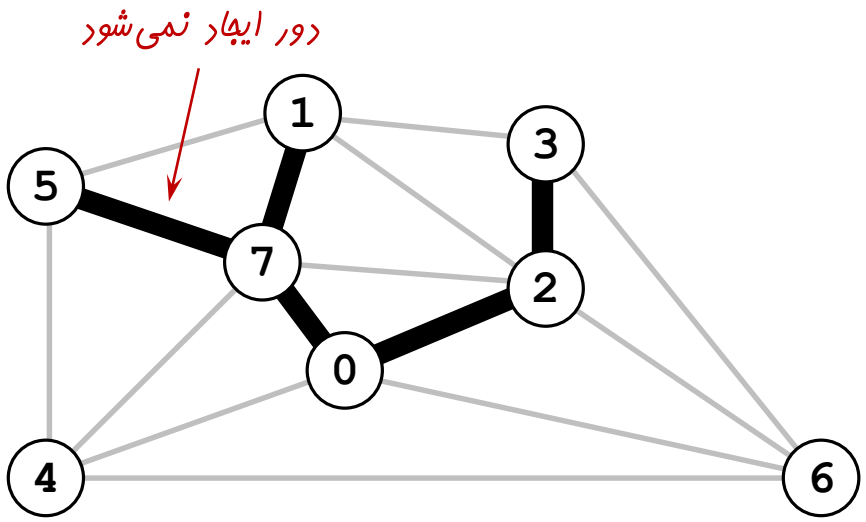
- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

```
0-7  0.16
2-3  0.17
1-7  0.19
0-2  0.26   ← در MST
```



الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

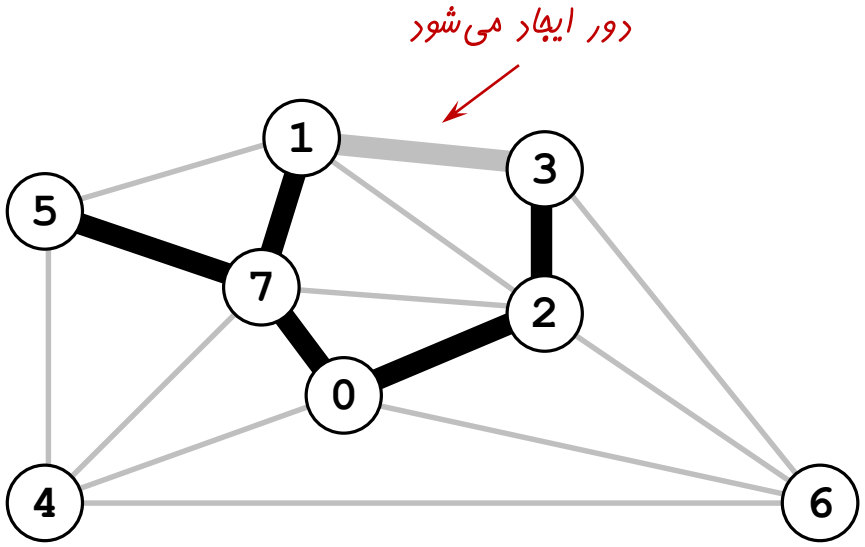



```
0-7   0.16
2-3   0.17
1-7   0.19
0-2   0.26
5-7   0.28
```

در MST ←

الگوریتم کروسکال: اجرای نمایشی

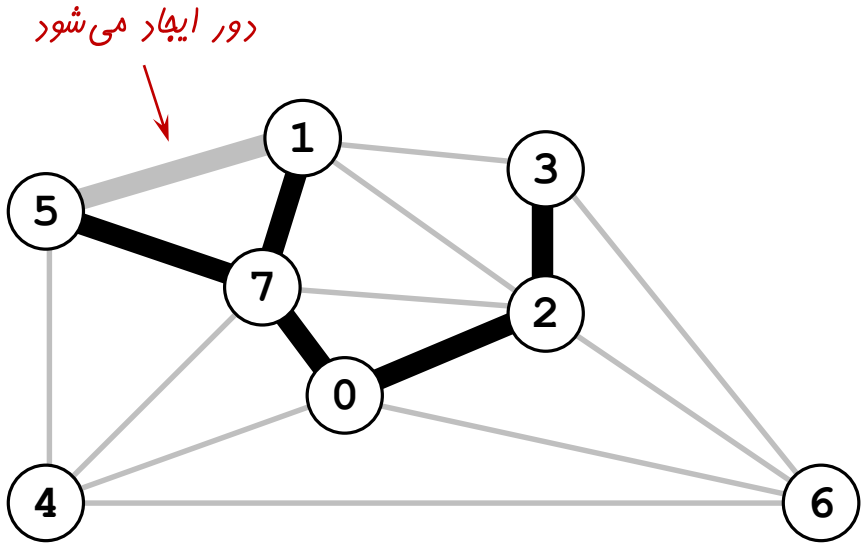
- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.




```
0-7  0.16
2-3  0.17
1-7  0.19
0-2  0.26
5-7  0.28
1-3  0.29   → در MST نیست
```

الگوریتم کروسکال: اجرای نمایشی

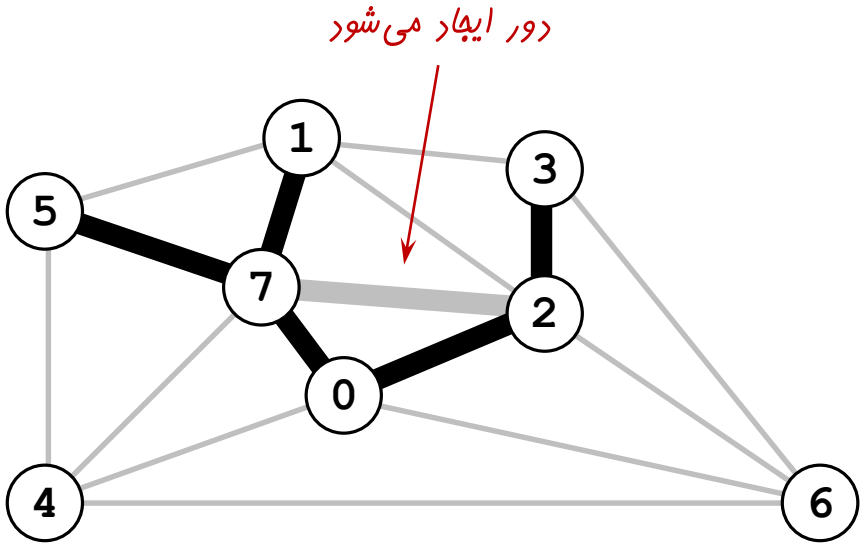
- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.




```
0-7  0.16
2-3  0.17
1-7  0.19
0-2  0.26
5-7  0.28
1-3  0.29
1-5  0.32   → در MST نیست
```

الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

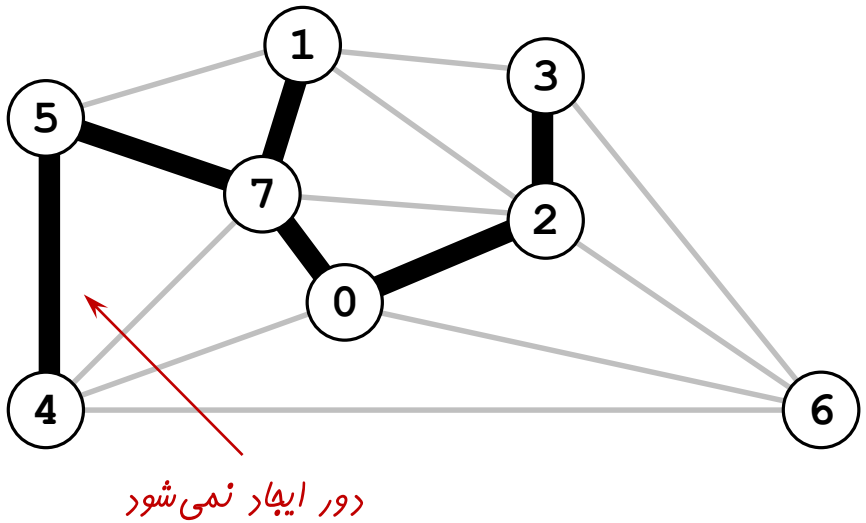



```
0-7  0.16
2-3  0.17
1-7  0.19
0-2  0.26
5-7  0.28
1-3  0.29
1-5  0.32
2-7  0.34
```

→ در **MST** نیست

الگوریتم کروسکال: اجرای نمایشی

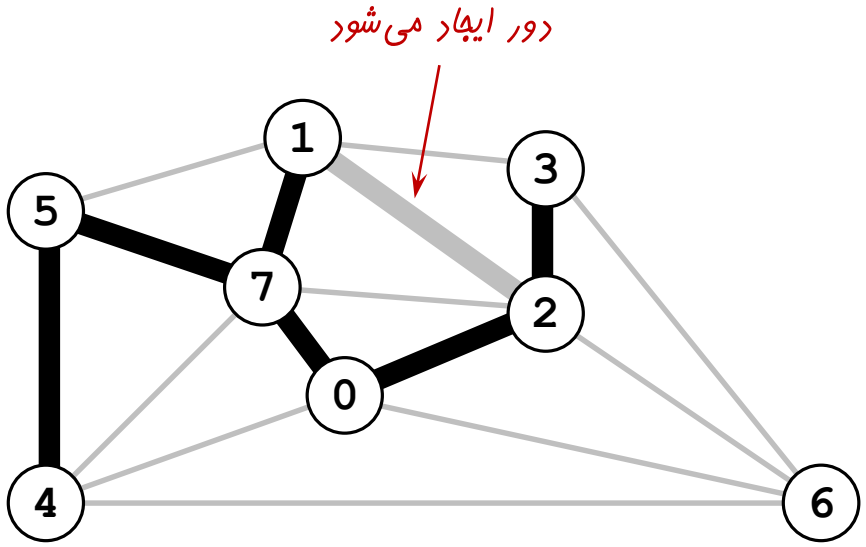
- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.



	0-7	0.16
	2-3	0.17
	1-7	0.19
	0-2	0.26
	5-7	0.28
	1-3	0.29
	1-5	0.32
	2-7	0.34
در MST ←	4-5	0.35

الگوریتم کروسکال: اجرای نمایشی

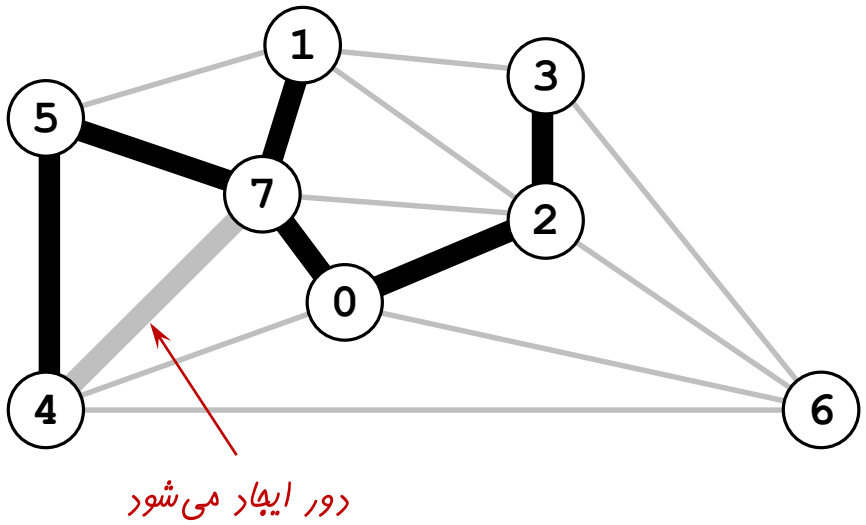
- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.




```
0-7   0.16
2-3   0.17
1-7   0.19
0-2   0.26
5-7   0.28
1-3   0.29
1-5   0.32
2-7   0.34
4-5   0.35
1-2   0.36   → در MST نیست
```

الگوریتم کروسکال: اجرای نمایشی

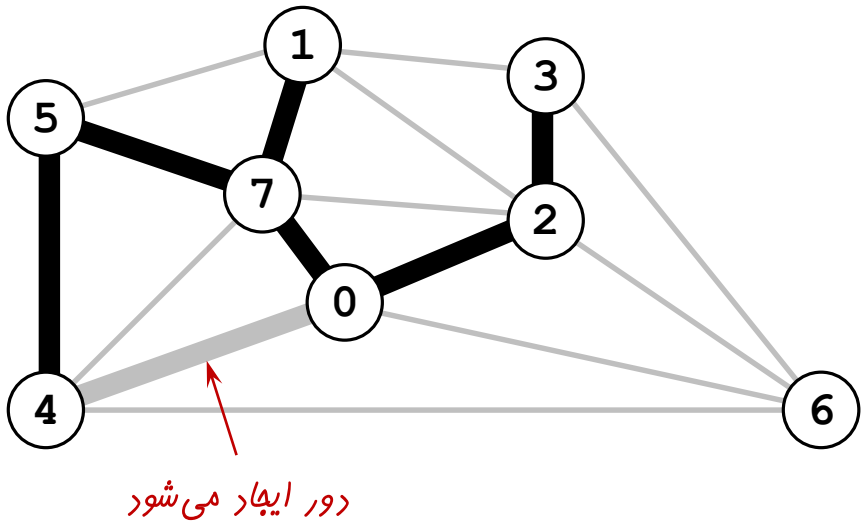
- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.




```
0-7   0.16
2-3   0.17
1-7   0.19
0-2   0.26
5-7   0.28
1-3   0.29
1-5   0.32
2-7   0.34
4-5   0.35
1-2   0.36
4-7   0.37   → در MST نیست
```

الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

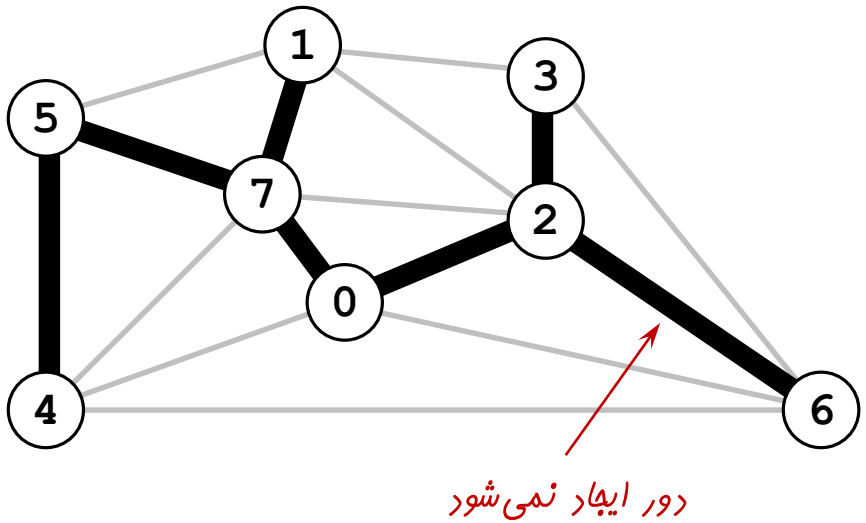


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38

→ در MST نیست

الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

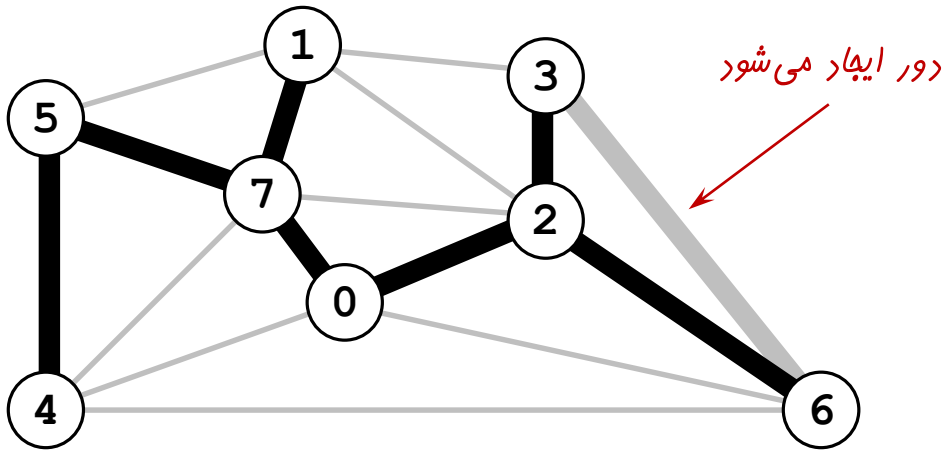


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40

در MST ←

الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

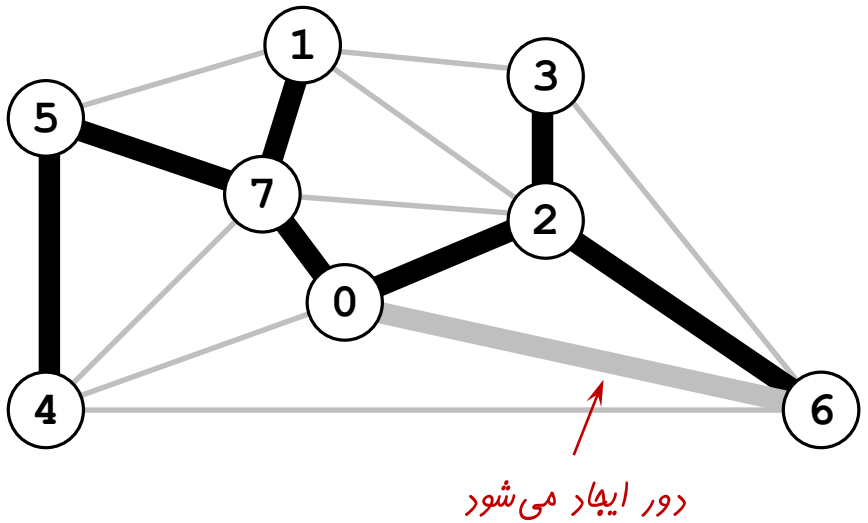


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52

→ در MST نیست

الگوریتم کروسکال: اجرای نمایشی

- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.

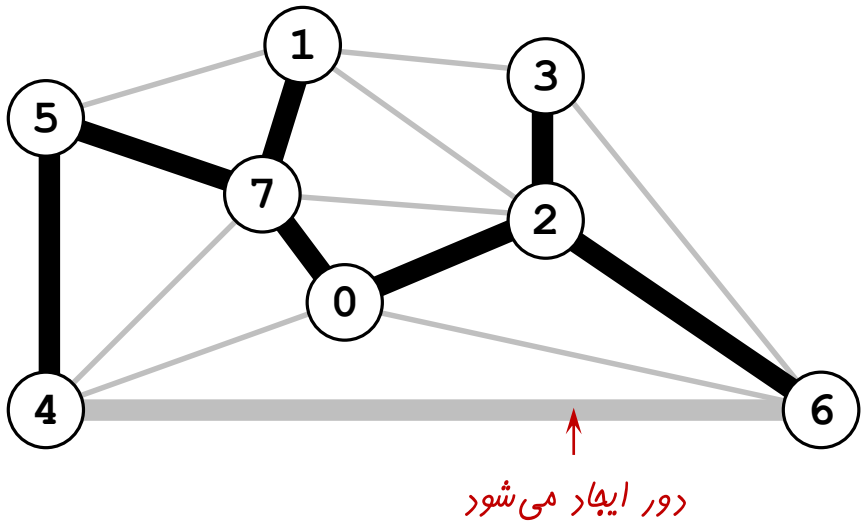


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

در **MST** نیست ←

الگوریتم کروسکال: اجرای نمایشی

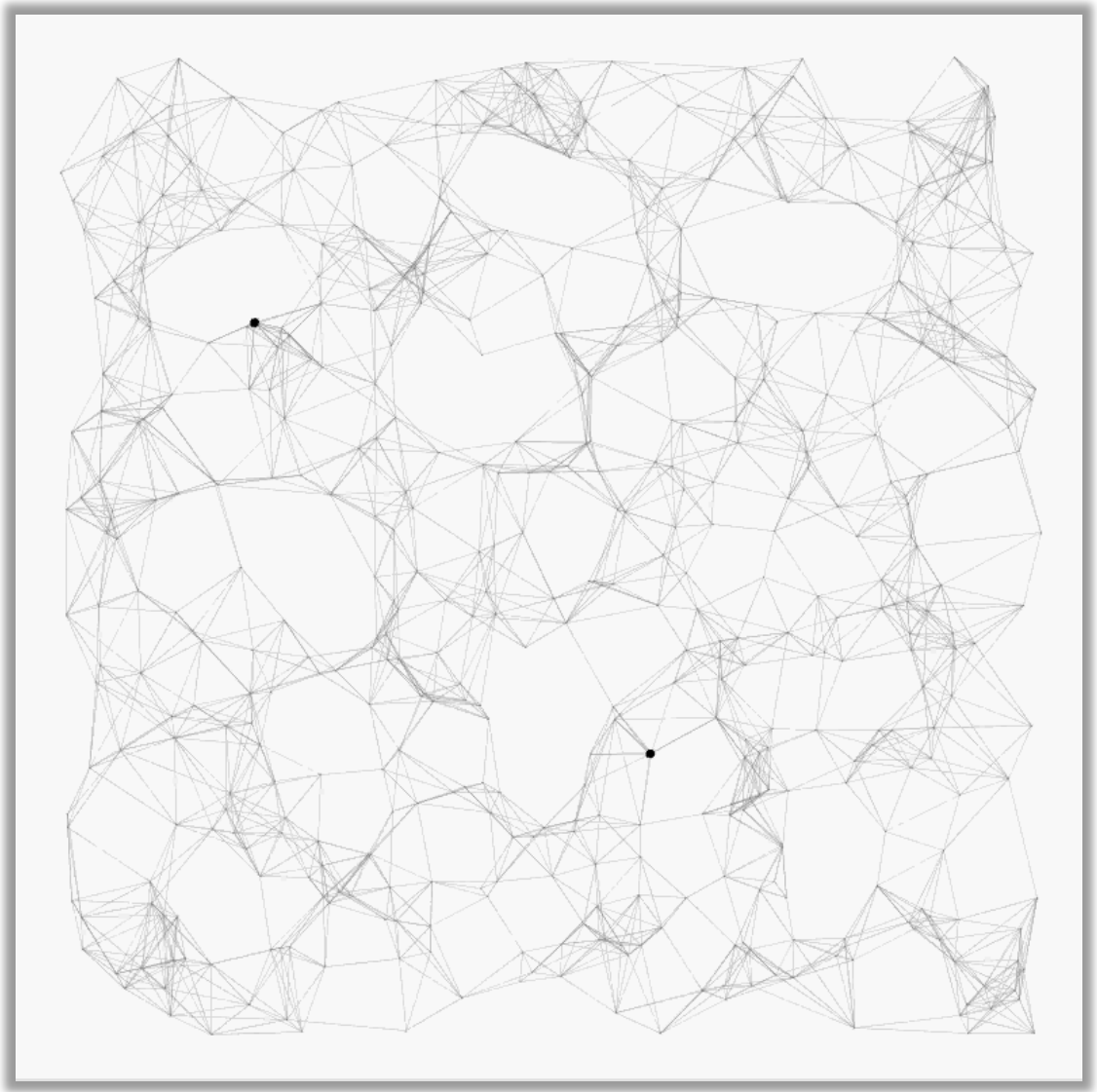
- الگوریتم کروسکال. [کروسکال، ۱۹۵۶]
  - یالها را به ترتیب صعودی وزن در نظر بگیر؛
  - کم وزن‌ترین یال باقیمانده را به درخت **T** اضافه کن مگر آن که انجام این کار باعث ایجاد دور شود.



0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

← در MST نیست

الگوریتم کروسکال: اجرای نمایشی



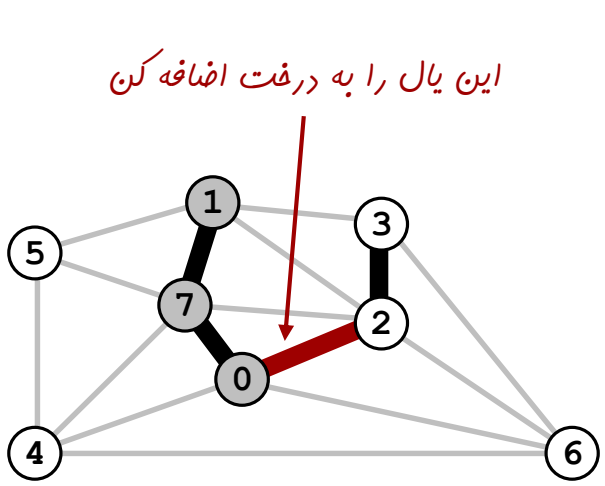
الگوریتم کروسکال: چالش پیاده‌سازی

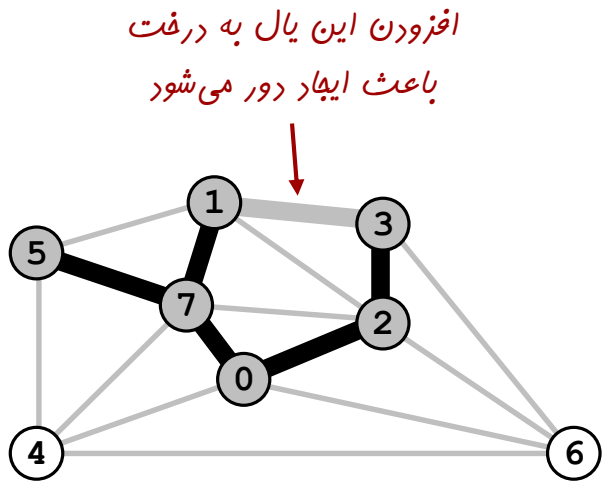
- چالش. آیا افزودن یال v-w به درخت T باعث ایجاد دور می‌شود؟
- درجه سختی؟

- E + V
- V
- log V
- log* V
- 1

با شروع از رأس v الگوریتم DFS را اجرا کن،
بررسی کن آیا رأس w قابل دسترسی است

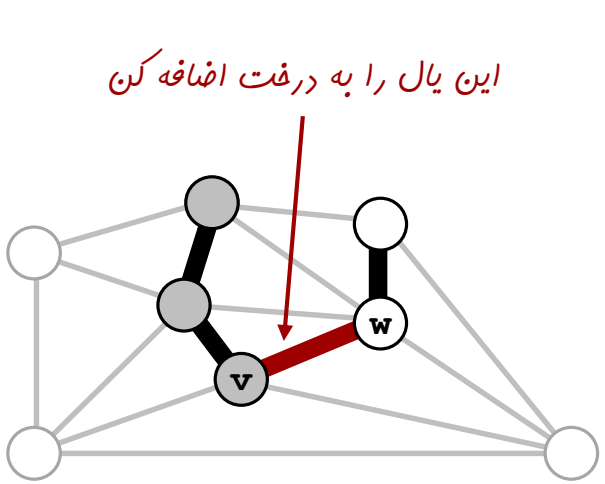
با استفاده از ساختمان داده مجموعه های مجزا !

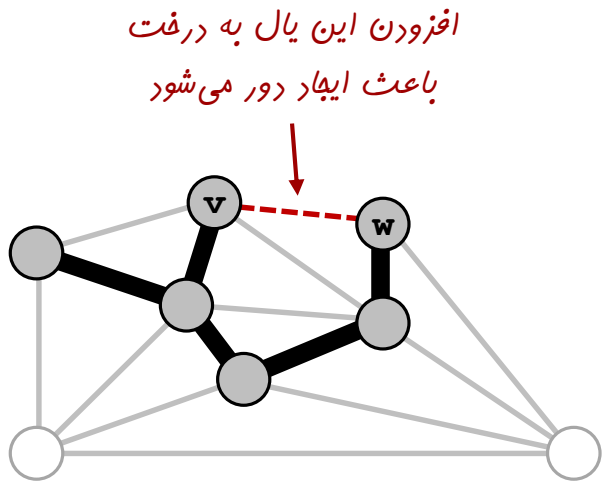




الگوریتم کروسکال: چالش پیاده‌سازی

- چالش. آیا افزودن یال **v-w** به درخت **T** باعث ایجاد دور می‌شود؟
- را ه حل کارا. استفاده از ساختمان داده‌ی مجموعه‌های مجزا.
  - هر مولفه‌ی همبند در **T** را در یک مجموعه‌ی مجزا نگهداری کن؛
  - اگر دو رأس **v** و **w** در یک مجموعه باشند، آنگاه افزودن یال **v-w** باعث ایجاد دور می‌شود؛
  - برای افزودن یال **v-w** به **T**، مجموعه های شامل دو رأس **v** و **w** را با یکدیگر ادغام کن.





الگوریتم کروسکال: پیاده‌سازی در جاوا


```
public class KruskalMST {
    private Queue<Edge> mst = new Queue<Edge>();

    public KruskalMST (EdgeWeightedGraph G) {
        MinPQ<Edge> pq = new MinPQ<Edge>();
        for (Edge e : G.edges())
            pq.insert(e);

        UF uf = new UF(G.V());
        while (!pq.isEmpty() && mst.size() < G.V() - 1) {
            Edge e = pq.delMin();
            int v = e.either(), w = e.other(v);
            if (!uf.connected(v, w))
            {
                uf.union(v, w);
                mst.enqueue(e);
            }
        }
    }

    public Iterable<Edge> edges() { return mst; }
}
```

الگوریتم کروسکال: زمان اجرا

- گزاره. الگوریتم کروسکال در بدترین حالت درخت پوشای کمینه را در زمانی متناسب با E log E محاسبه می‌کند.
- اثبات.

عمل	فراوانی	زمان هر اجرا
ایجاد صف اولویت	1	E
حذف کوچک‌ترین	E	log E
اجتماع	V	log* V
بررسی متصل بودن	E	log* V

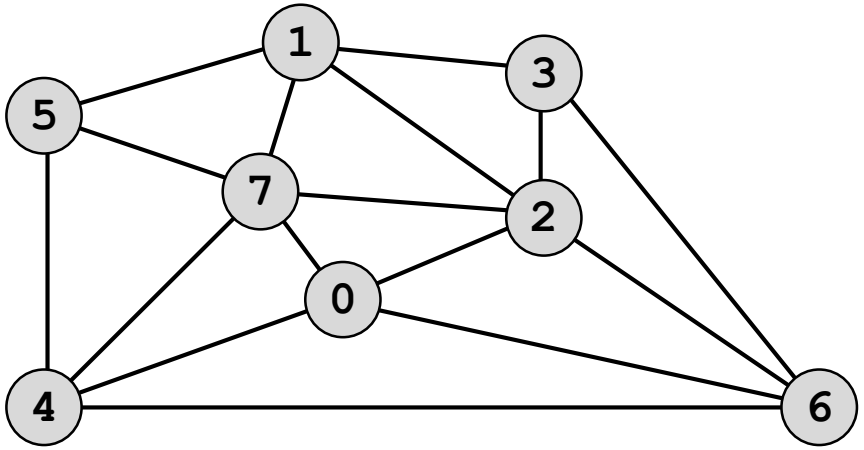
یادآوری: در دنیای واقعی log* V <= 5

- توجه. اگر یالها از قبل مرتب باشند، نرخ رشد برابر با E log* V خواهد بود.

الگوریتم پریم

الگوریتم پریم: اجرای نمایشی

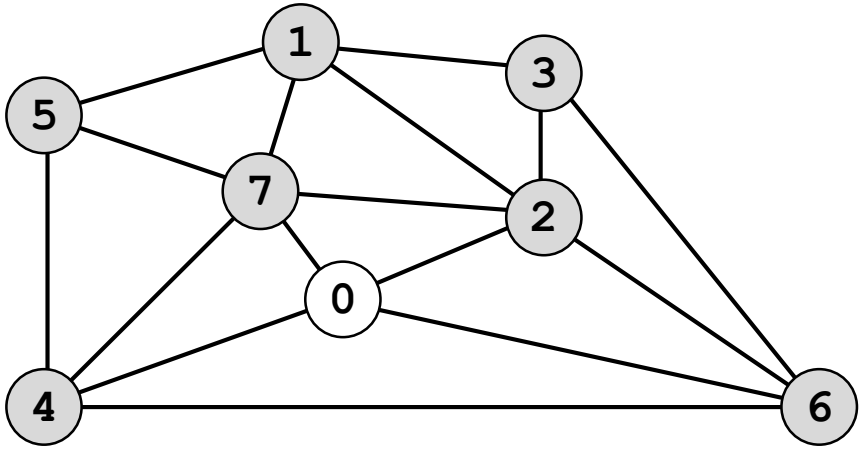
- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.




```
0-7   0.16
2-3   0.17
1-7   0.19
0-2   0.26
5-7   0.28
1-3   0.29
1-5   0.32
2-7   0.34
4-5   0.35
1-2   0.36
4-7   0.37
0-4   0.38
6-2   0.40
3-6   0.52
6-0   0.58
6-4   0.93
```

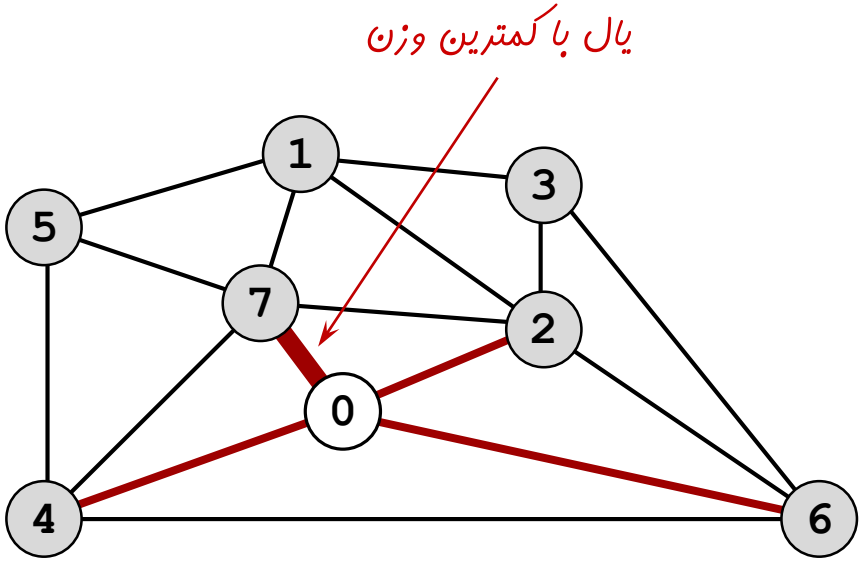
الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.



الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.



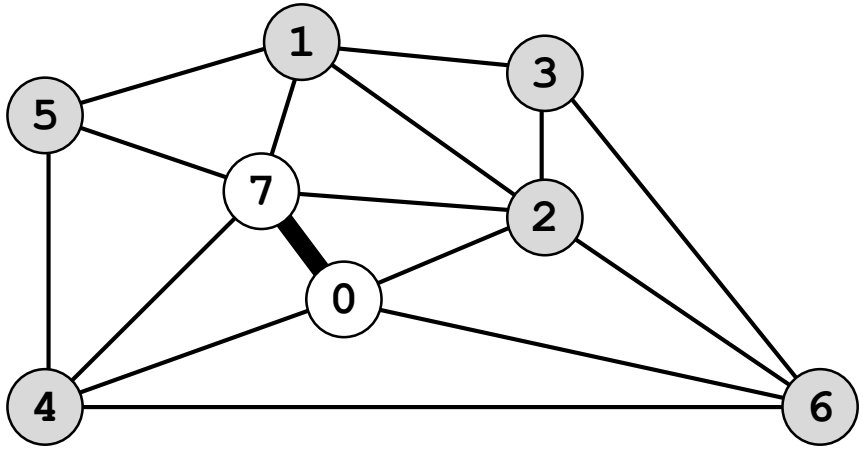
یالهایی که دقیقا یک رأس‌شان در T است

MST در ⟵

0-7	0.16
0-2	0.26
0-4	0.38
6-0	0.58

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.

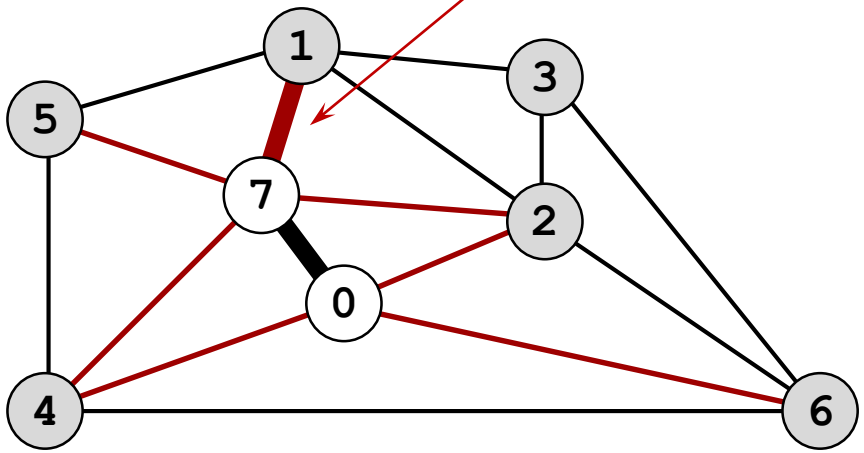


یال‌های MST **0-7**

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.

یال با کمترین وزن



یال‌های MST 0-7

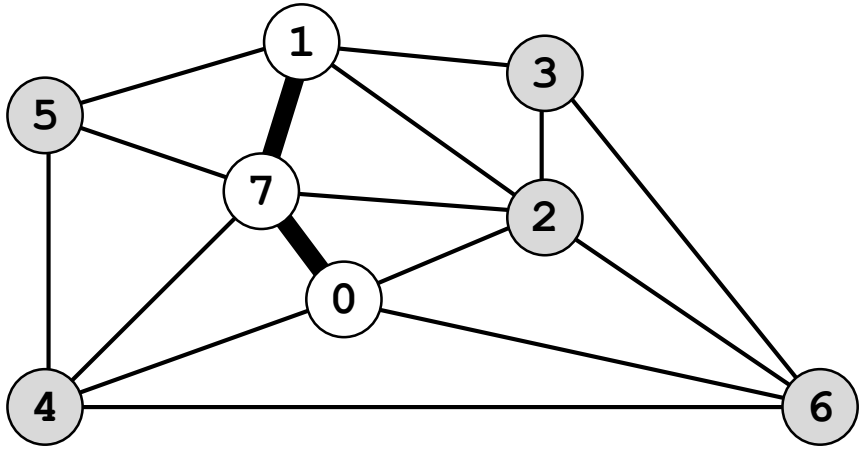
یال‌هایی که دقیقا یک رأس‌شان در T است

MST در ⟵

1-7	0.19
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.

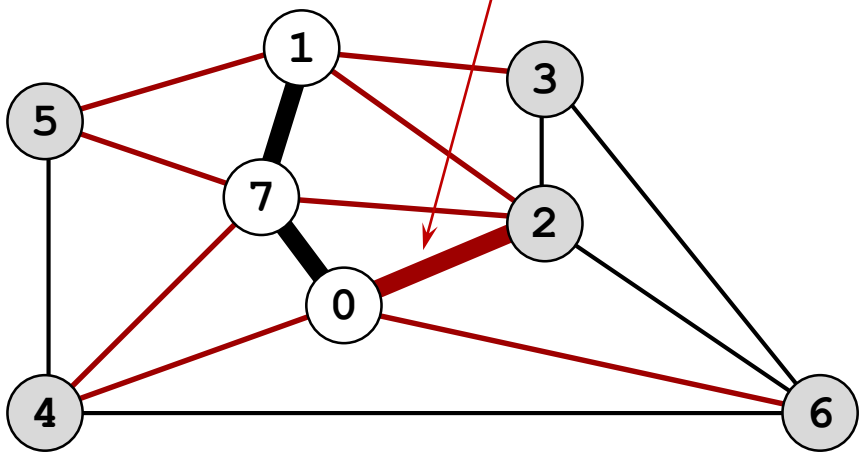


یال‌های MST **0-7 1-7**

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.

یال با کمترین وزن



یال‌های MST 0-7 1-7

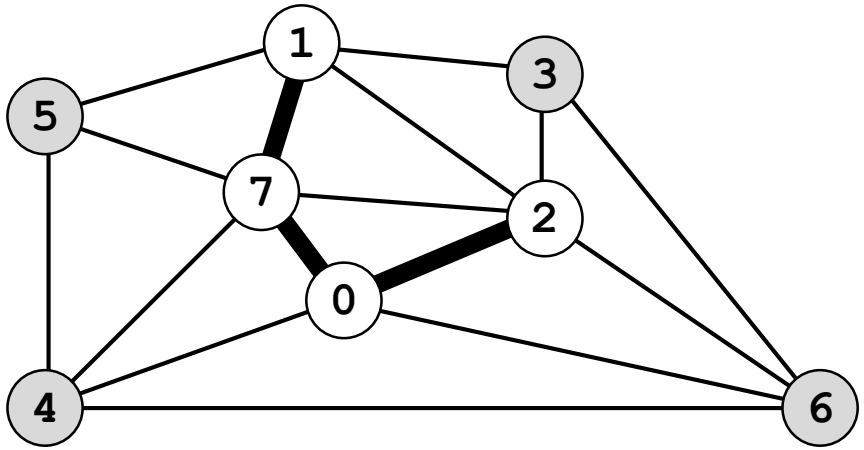
یال‌هایی که دقیقا یک رأس‌شان در T است

MST در ⟵

0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-0	0.58

الگوریتم پریم: اجرای نمایشی

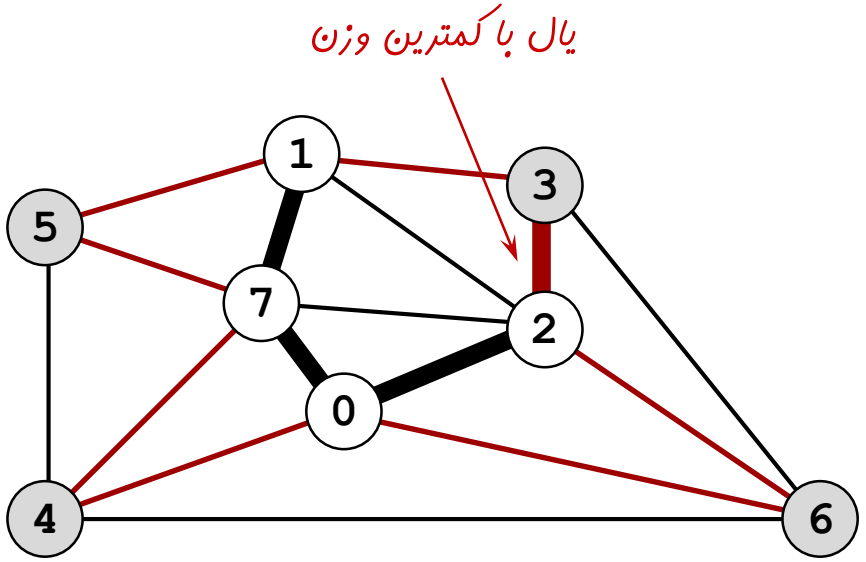
- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.



یال‌های MST **0-7 1-7 0-2**

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.



یال‌های MST 0-7 1-7 0-2

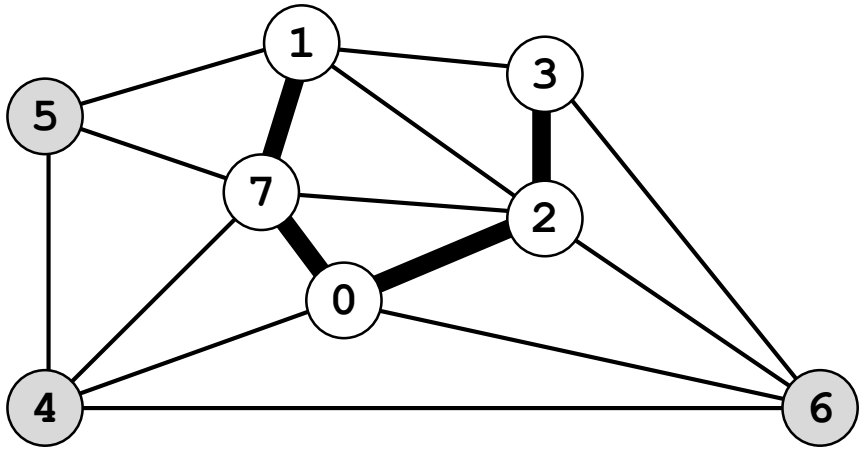
یالهایی که دقیقا یک رأس‌شان در T است

MST در ⟵

2-3	0.17
5-7	0.28
1-3	0.29
1-5	0.32
4-7	0.37
0-4	0.38
6-2	0.40
6-0	0.58

الگوریتم پریم: اجرای نمایشی

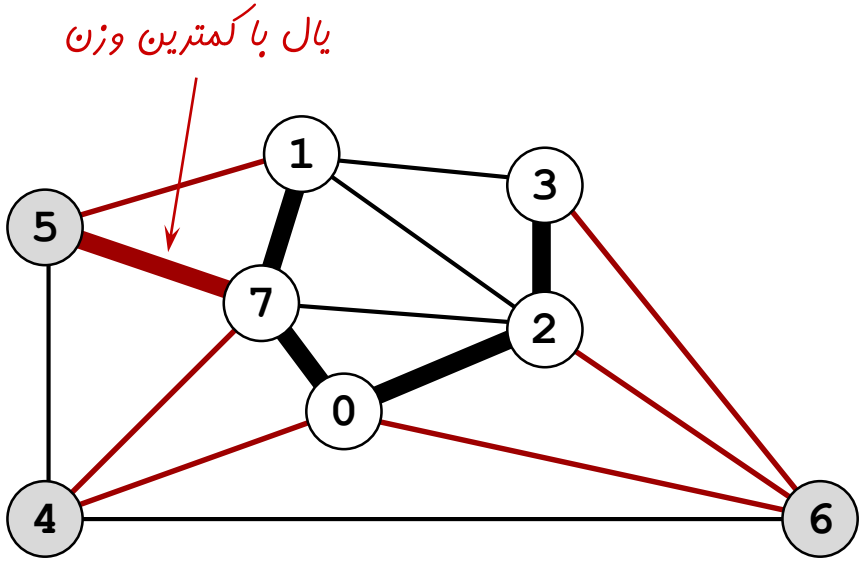
- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.



یال‌های MST 0-7 1-7 0-2 2-3

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.



یال‌های MST 0-7 1-7 0-2 2-3

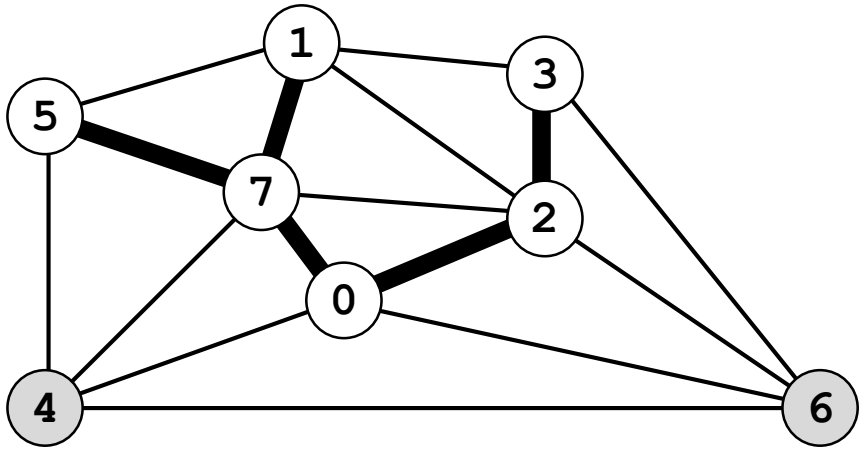
یالهایی که دقیقا یک رأس‌شان در T است

MST در ⟵

5-7	0.28
1-5	0.32
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.

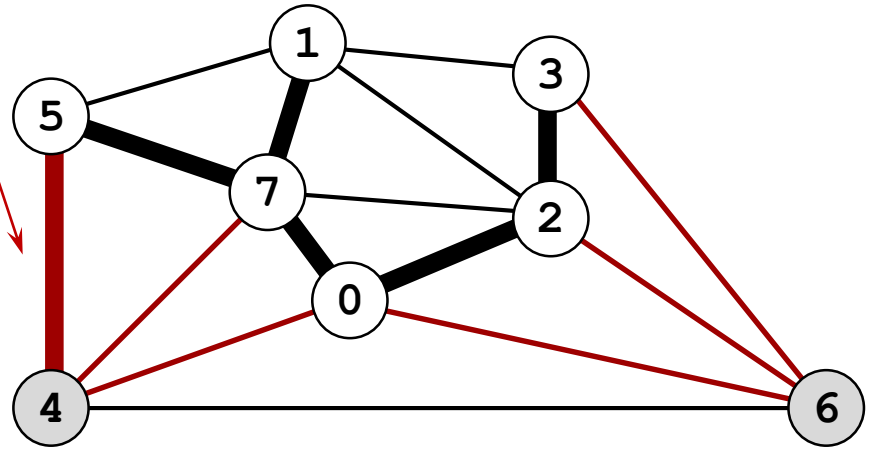


یال‌های MST 0-7 1-7 0-2 2-3 5-7

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.

یال با کمترین وزن



یال‌های MST 0-7 1-7 0-2 2-3 5-7

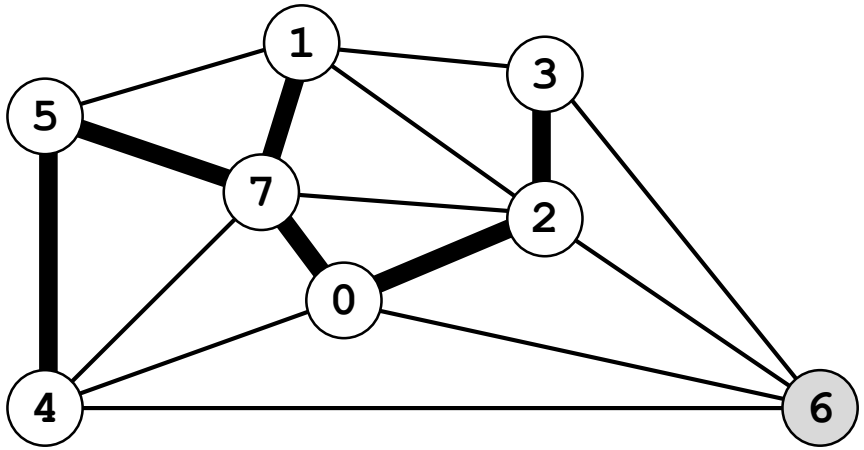
یالهایی که دقیقا یک رأس‌شان در T است

در MST →

4-5	0.35
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.

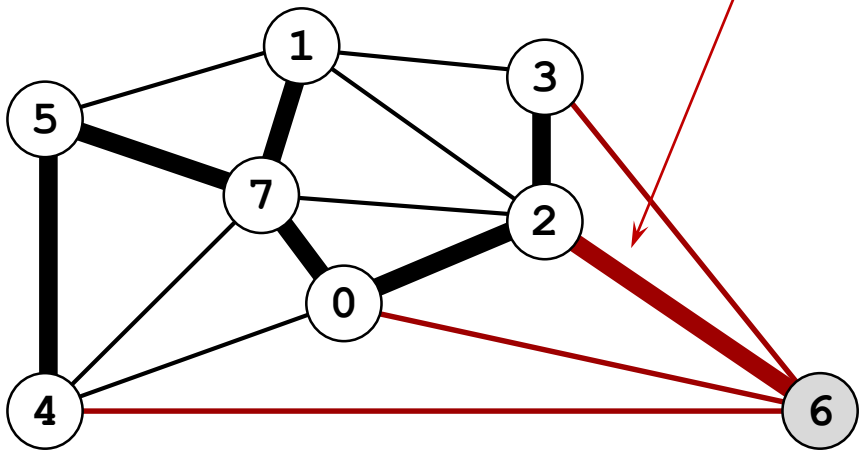


یال‌های MST 0-7 1-7 0-2 2-3 5-7 4-5

الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.

یال با کمترین وزن



یال‌هایی که دقیقا یک رأسشان در T است

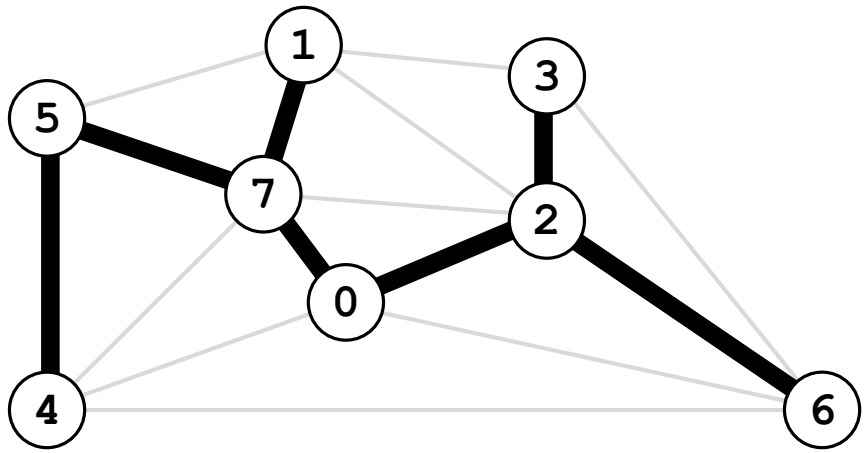
MST در ⟵

6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

یال‌های MST 0-7 1-7 0-2 2-3 5-7 4-5

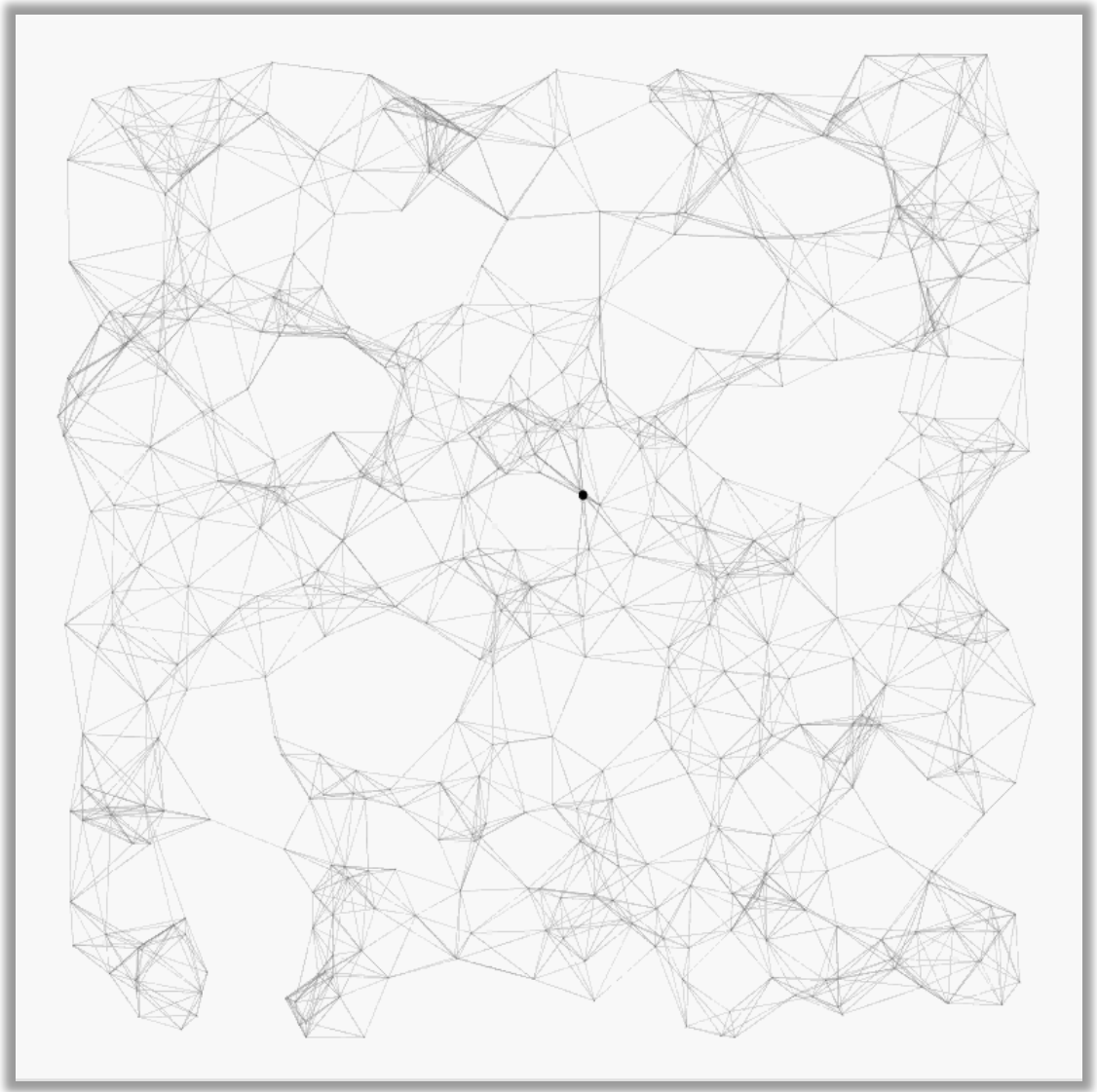
الگوریتم پریم: اجرای نمایشی

- الگوریتم پریم. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن.



یال‌های MST 0-7 1-7 0-2 2-3 5-7 4-5 6-2

الگوریتم پریم: اجرا

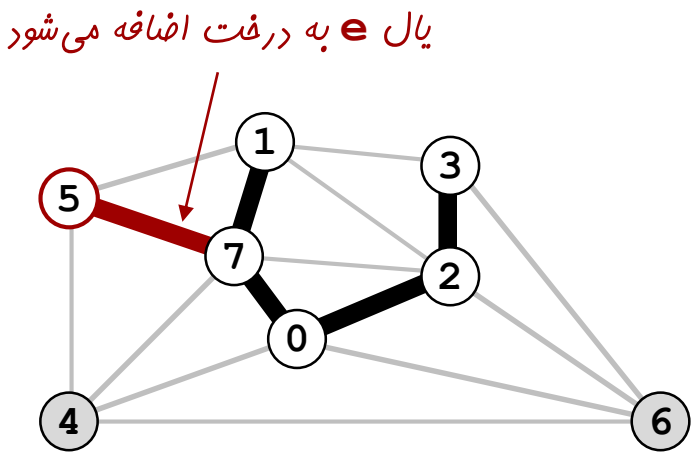


الگوریتم پریم: اثبات درستی

- گزاره. [یارنیک ۱۹۳۰، دیکسترا ۱۹۵۷، پریم ۱۹۵۹]

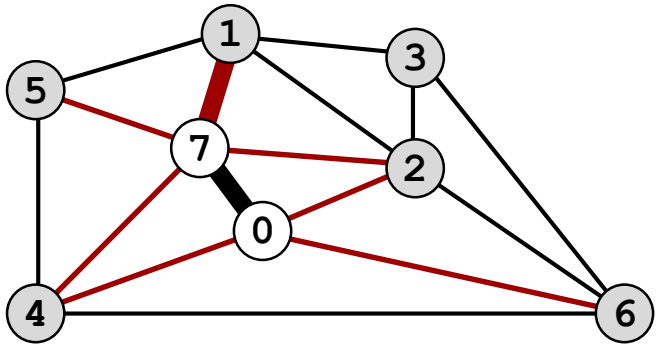
  خروجی الگوریتم پریم یک درخت پوشای کمینه است.

- اثبات. الگوریتم پریم یک حالت خاص از الگوریتم حریصانه‌ی محاسبه MST است.
  - فرض کنید e یالی باشد که یک رأس از درخت را به یک رأس دیگر که به درخت تعلق ندارد متصل می‌کند.
  - برش = مجموعه رئوس درخت.
  - هیچ یال متقاطعی سیاه نیست.
  - وزن هیچ یال متقاطعی از e کمتر نیست.



الگوریتم پریم: چالش های پیاده سازی

- چالش. یافتن یال با کمترین وزن که دقیقاً یک سرش در T باشد.
- درجه سختی.
  - E ← بررسی تمام یالها
  - V
  - log E ← استفاده از یک صف اولویت
  - log* E
  - 1

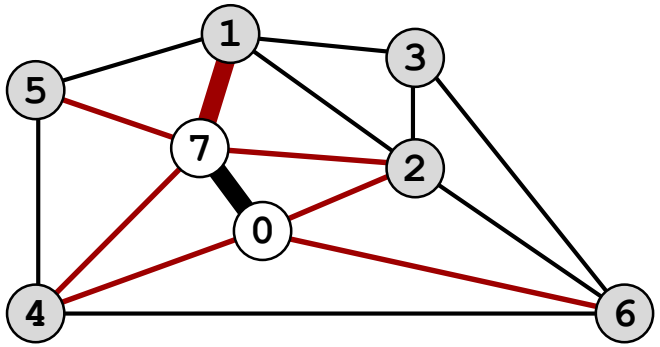



```
1-7   0.19
0-2   0.26
5-7   0.28
2-7   0.34
4-7   0.37
0-4   0.38
6-0   0.58
```

یال 1-7 یال با کمترین وزن است
که دقیقا یک سر آن در T است

الگوریتم پریم: پیاده سازی تنبل

- چالش. یافتن یال با کمترین وزن که دقیقاً یک سر آن در T باشد.
- راه حل تنبل. یالهایی را که (حداقل) یک سرشان در T است، در یک صف اولویت نگهداری کن.
  - کلید = یال؛ اولویت = وزن یال.
  - به منظور تعیین یال بعدی که باید به T اضافه شود، یال با کمترین وزن را از صف اولویت حذف کن.
  - اگر هر دو سر یال حذف شده در T قرار دارد، آن را نادیده بگیر.
  - در غیر این صورت، اگر **w** رأسی باشد که در T نیست:
    - یال‌های متلاقی با **w** را به صف اولویت اضافه کن (با این فرض که رأس دیگرشان در T نباشد)
    - رأس **w** را به T اضافه کن

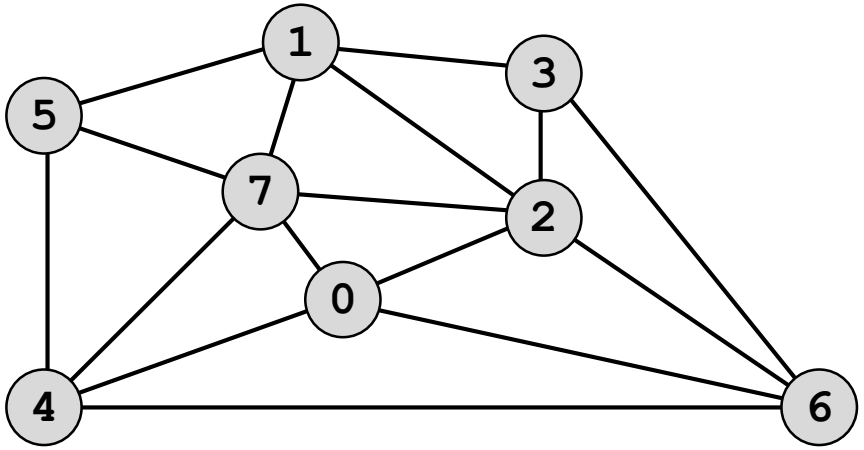



```
1-7   0.19
0-2   0.26
5-7   0.28
2-7   0.34
4-7   0.37
0-4   0.38
6-0   0.58
```

یال 1-7 یال با کمترین وزن است
که دقیقا یک سر آن در T است

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

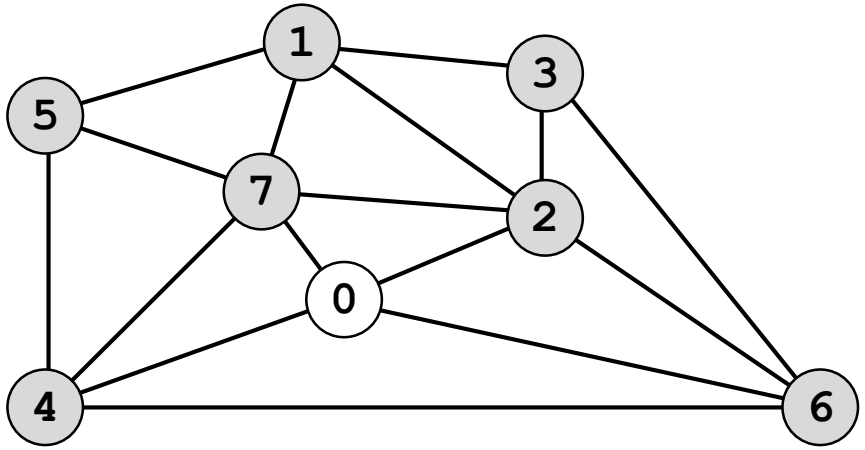
- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.




```
0-7  0.16
2-3  0.17
1-7  0.19
0-2  0.26
5-7  0.28
1-3  0.29
1-5  0.32
2-7  0.34
4-5  0.35
1-2  0.36
4-7  0.37
0-4  0.38
6-2  0.40
3-6  0.52
6-0  0.58
6-4  0.93
```

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

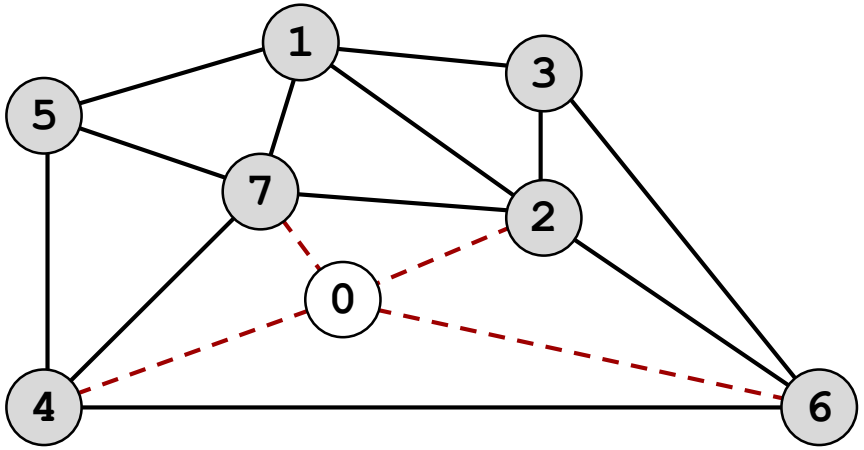
- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن $V - 1$ یال تکرار کن.



الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

تمام یال‌های متلاقی با رأس 0 را به صف اولویت اضافه کن

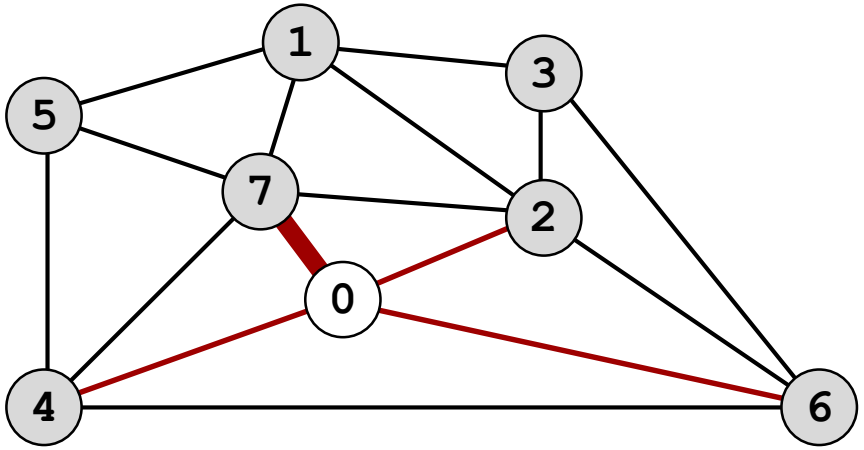


یال‌های موجود در صف اولویت	
* 0-7	0.16
* 0-2	0.26
* 0-4	0.38
* 6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

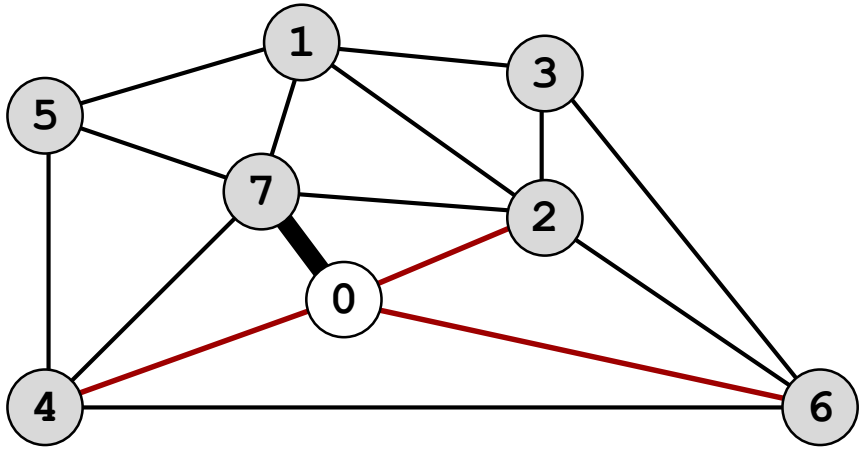
یال **0–7** *را از صف اولویت حذف و به درخت اضافه کن*



یال‌های موجود در صف اولویت	
0–7	0.16
0–2	0.26
0–4	0.38
6–0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



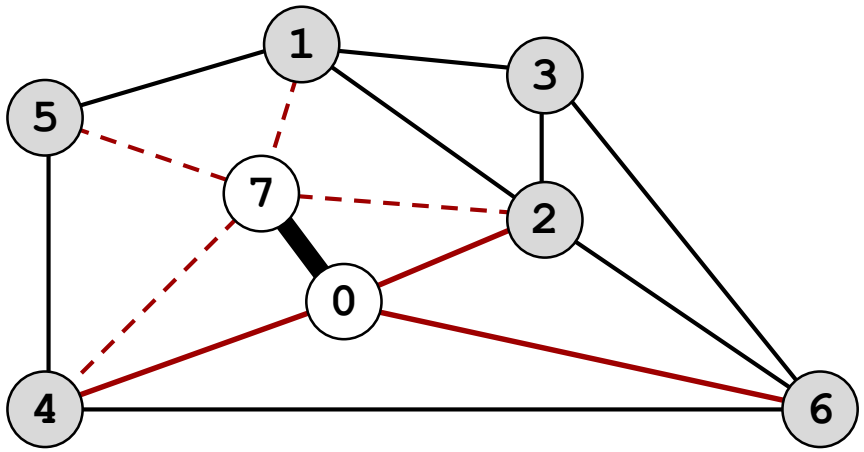
یال‌های موجود در صف اولویت	
0-2	0.26
0-4	0.38
6-0	0.58

یالهای MST 0-7

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

تمام یال‌های متلاقی با رأس **7** *را به صف اولویت اضافه کن*



یال‌های موجود در صف اولویت	
* 1-7	0.19
0-2	0.26
* 5-7	0.28
* 2-7	0.34
* 4-7	0.37
0-4	0.38
6-0	0.58

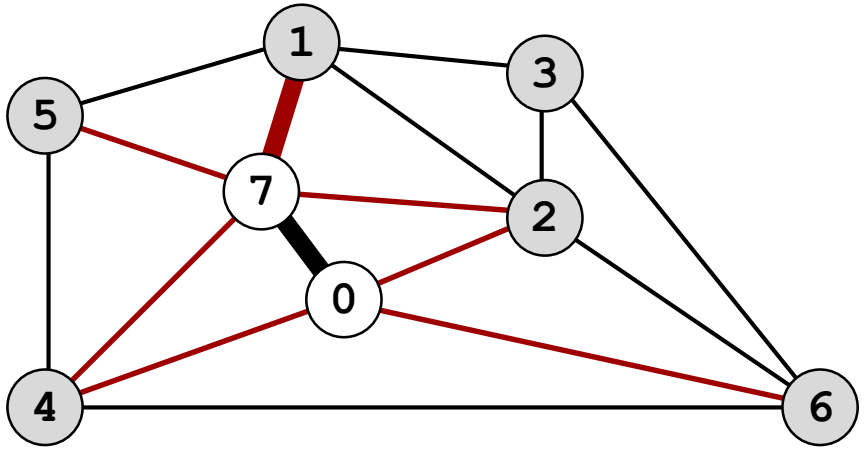
یالهای MST **0-7**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **1-7** *را از صف اولویت حذف و به درخت اضافه کن*

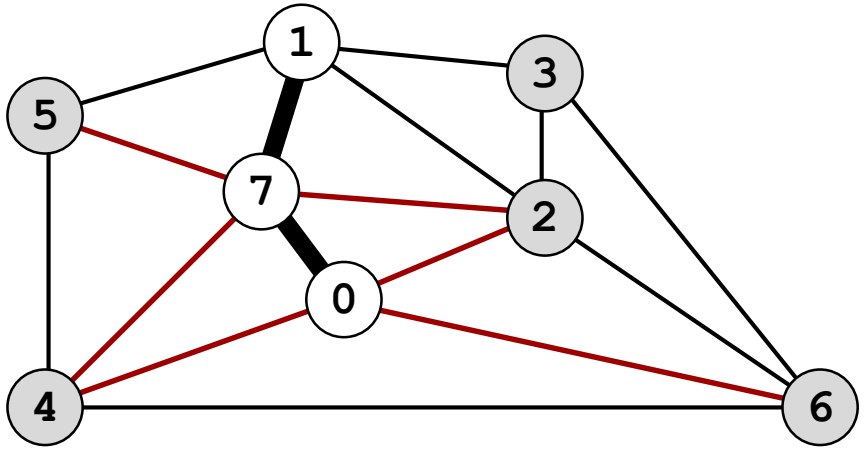
یال‌های موجود در صف اولویت	
1-7	0.19
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58



یالهای MST **0-7**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



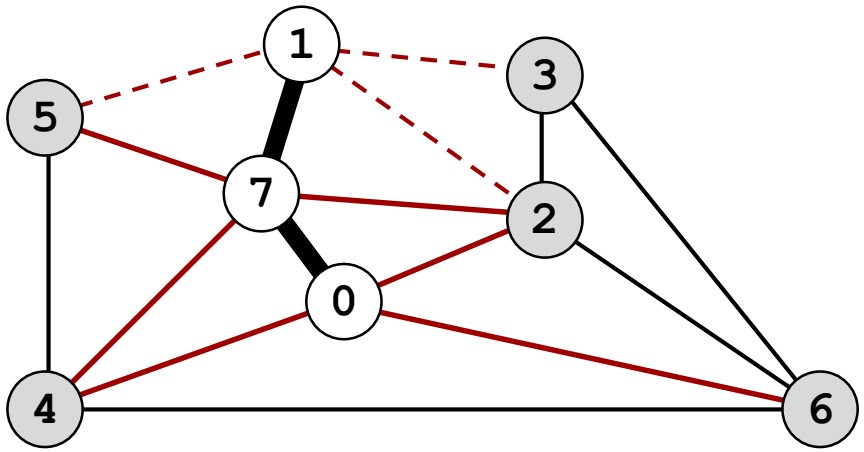
یال‌های موجود در صف اولویت	
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58

یالهای MST **0-7 1-7**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

تمام یال‌های متلاقی با رأس 1 را به صف اولویت اضافه کن



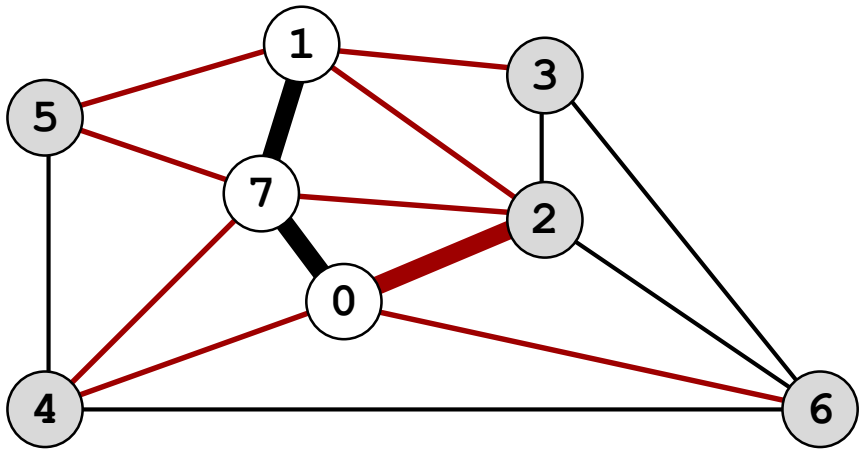
یال‌های موجود در صف اولویت	
0-2	0.26
5-7	0.28
* 1-3	0.29
* 1-5	0.32
2-7	0.34
* 1-2	0.36
4-7	0.37
0-4	0.38
6-0	0.58

یالهای MST **0-7 1-7**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **0-2** *را از صف اولویت حذف و به درخت اضافه کن*

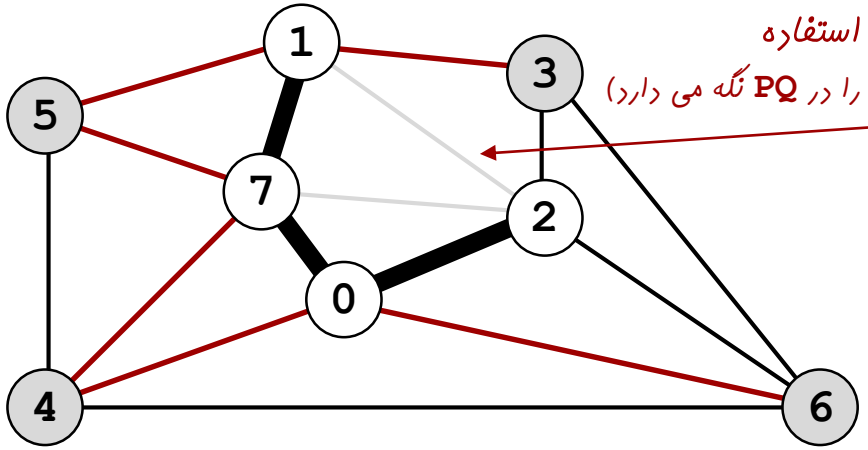


یال‌های موجود در صف اولویت	
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-0	0.58

یالهای MST **0-7 1-7**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



یال‌های بلا استفاده

(پیاده سازی تنبل این یال‌ها را در PQ نگه می دارد)

یال‌های موجود در صف اولویت

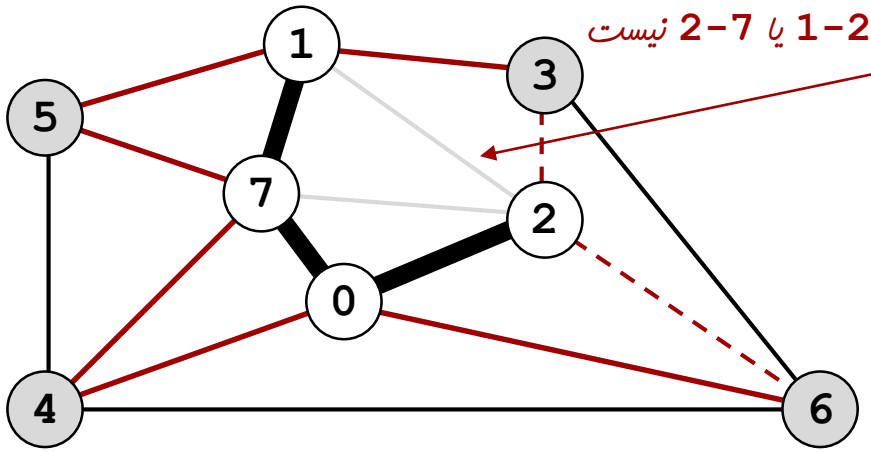
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-0	0.58

یالهای MST 0-7 1-7 0-2

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

تمام یال‌های متلاقی با رأس 2 را به صف اولویت اضافه کن



نیازی به اضافه کردن یال 2-1 یا 7-2 نیست

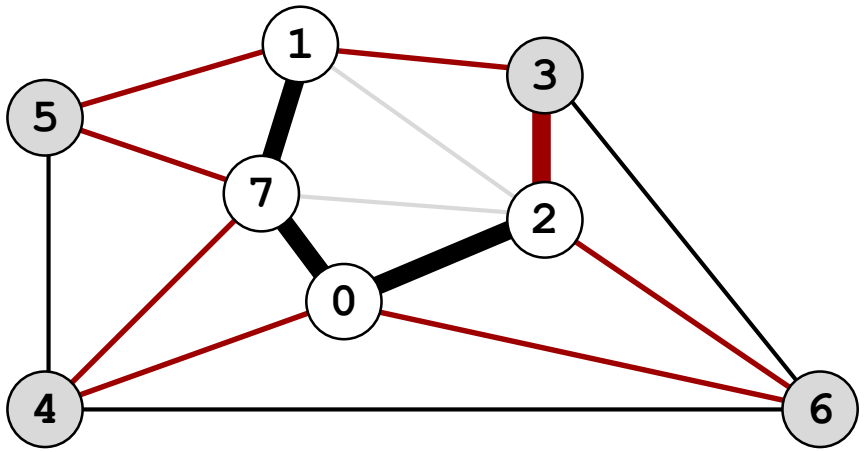
یال‌های موجود در صف اولویت	
* 2-3	0.17
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
* 6-2	0.40
6-0	0.58

یال‌های **MST** 0-7 1-7 0-2

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **2-3** *را از صف اولویت حذف و به درخت اضافه کن*

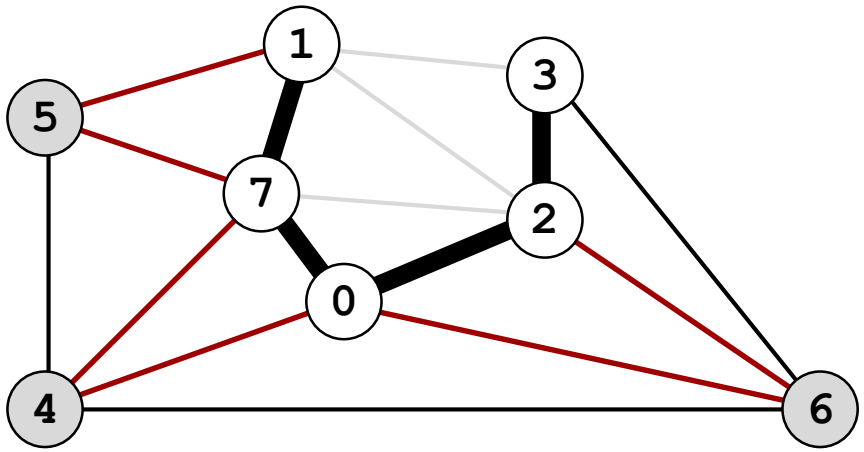


یالهای MST **0-7 1-7 0-2**

یال‌های موجود در صف اولویت	
* 2-3	0.17
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
* 6-2	0.40
6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



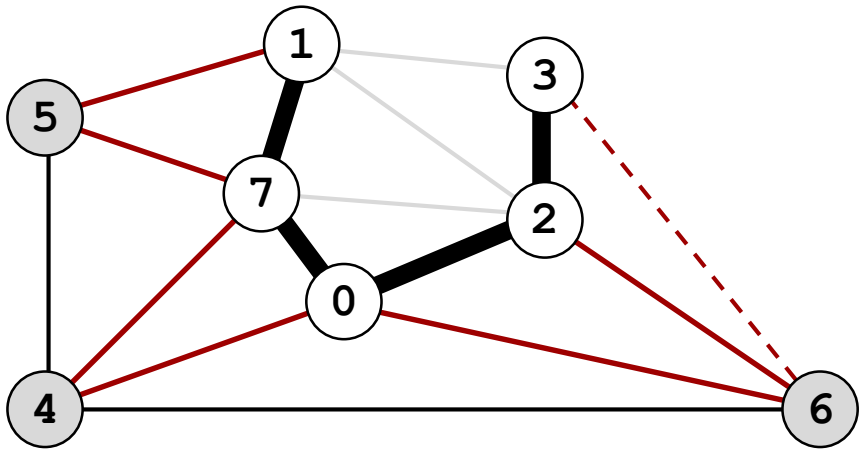
یالهای MST 0-7 1-7 0-2 2-3

یال‌های موجود در صف اولویت	
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

تمام یال‌های متلاقی با رأس 3 را به صف اولویت اضافه کن



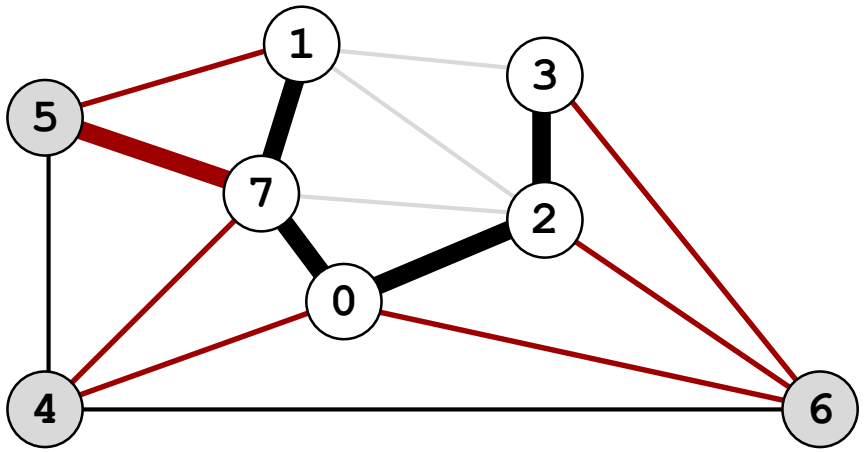
یالهای MST **0-7 1-7 0-2 2-3**

یال‌های موجود در صف اولویت	
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
* 3-6	0.52
6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **5-7** *را از صف اولویت حذف و به درخت اضافه کن*

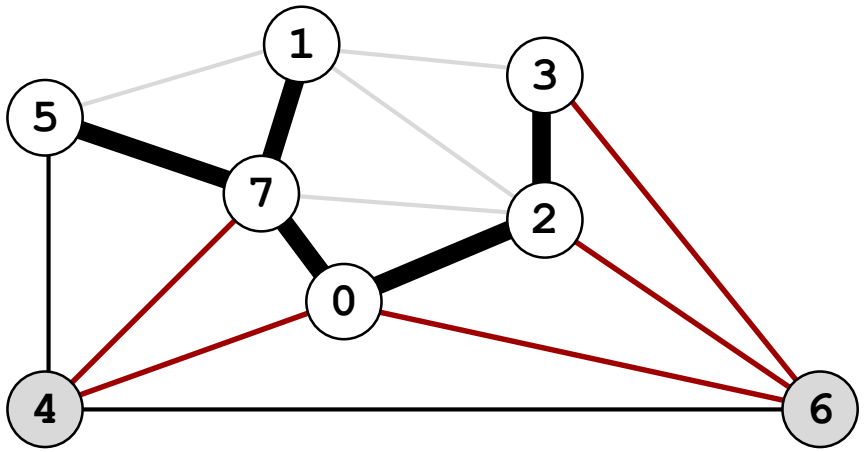


یالهای MST **0-7 1-7 0-2 2-3**

یال‌های موجود در صف اولویت	
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



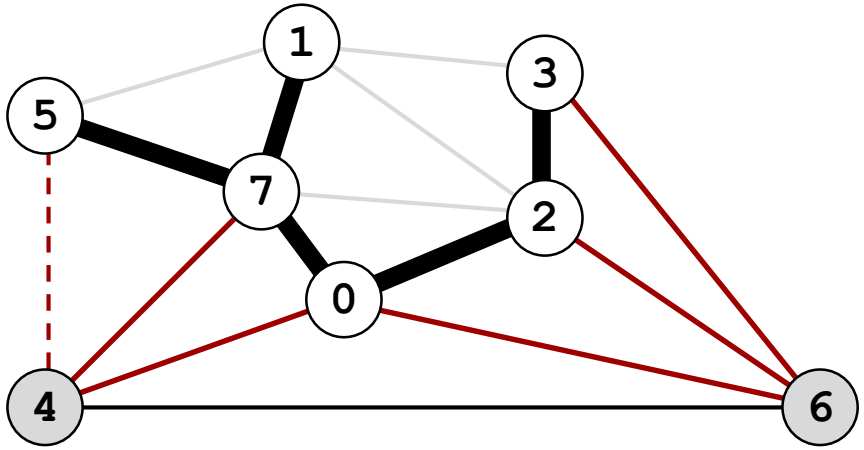
یالهای MST 0-7 1-7 0-2 2-3 5-7

یال‌های موجود در صف اولویت	
1-3	0.29
1-5	0.32
2-7	0.34
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

تمام یال‌های متلاقی با رأس 5 را به صف اولویت اضافه کن



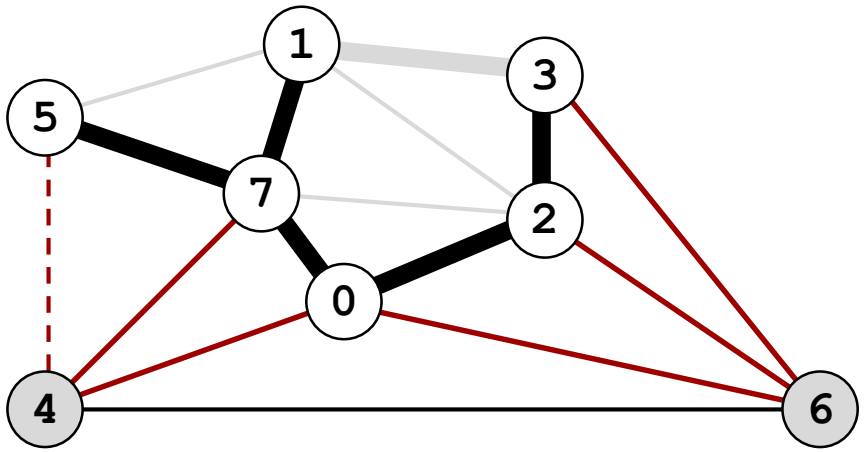
یال‌های MST 0-7 1-7 0-2 2-3 5-7

یال‌های موجود در صف اولویت	
1-3	0.29
1-5	0.32
2-7	0.34
* 4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **1-3** *را از صف اولویت حذف کن*



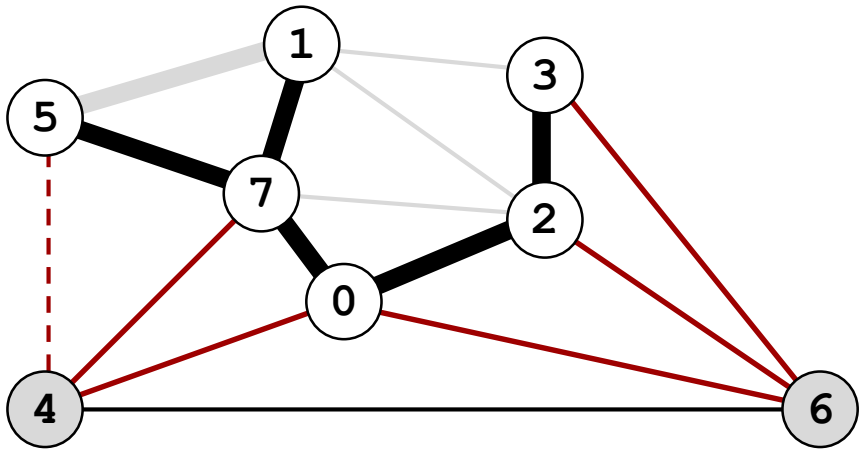
یالهای MST **0-7 1-7 0-2 2-3 5-7**

یال‌های موجود در صف اولویت	
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **1-5** *را از صف اولویت حذف کن*



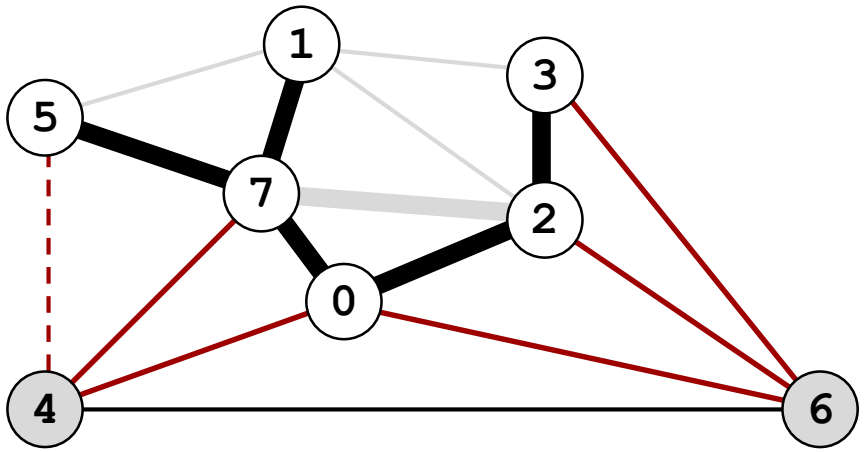
یالهای MST **0-7 1-7 0-2 2-3 5-7**

یال‌های موجود در صف اولویت	
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **2-7** *را از صف اولویت حذف کن*



یال‌های موجود در صف اولویت	
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

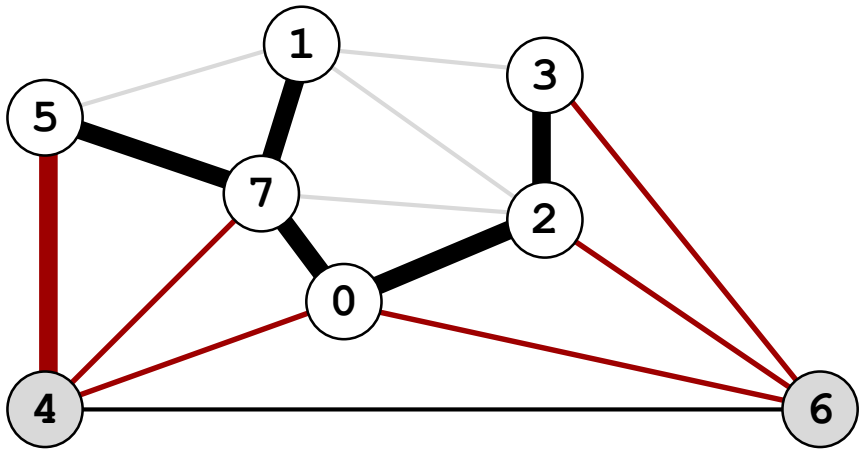
یالهای MST **0-7 1-7 0-2 2-3 5-7**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **4-5** *را از صف اولویت حذف و به درخت اضافه کن*

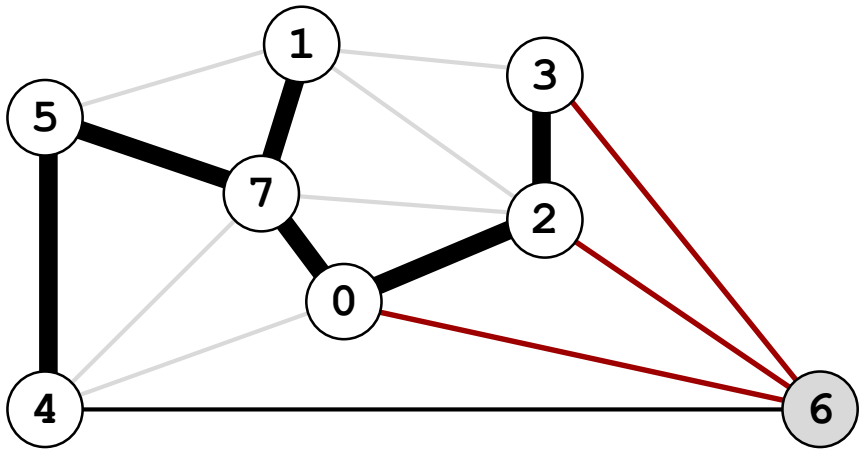
یال‌های موجود در صف اولویت	
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58



یالهای MST **0-7 1-7 0-2 2-3 5-7**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



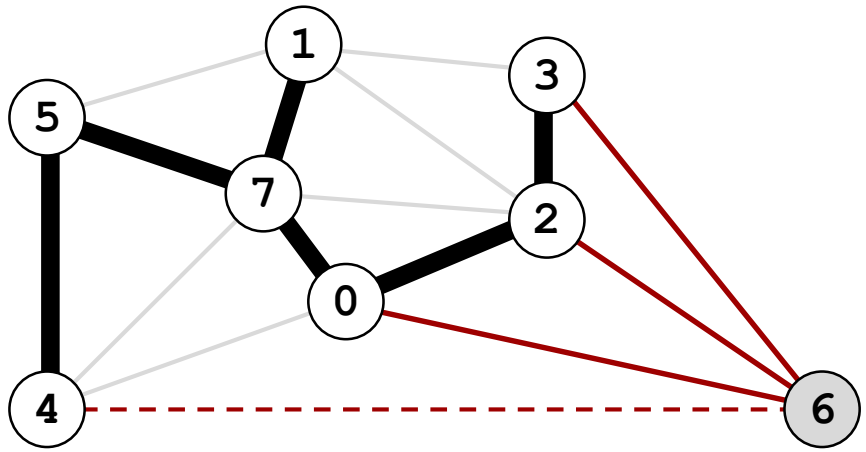
یال‌های موجود در صف اولویت	
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

یالهای MST 0-7 1-7 0-2 2-3 5-7 4-5

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

تمام یال‌های متلاقی با رأس **4** *را به صف اولویت اضافه کن*



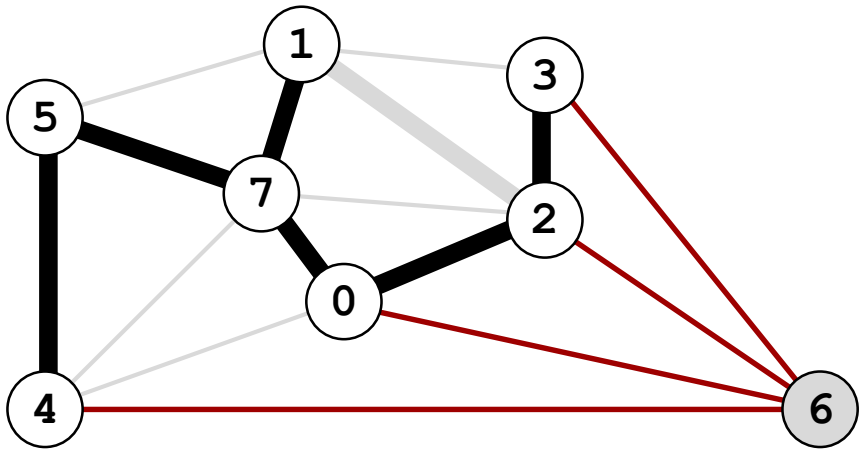
یال‌های موجود در صف اولویت	
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
* 6-4	0.93

یالهای MST 0-7 1-7 0-2 2-3 5-7 4-5

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **1-2** *را از صف اولویت حذف کن*



یال‌های موجود در صف اولویت	
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

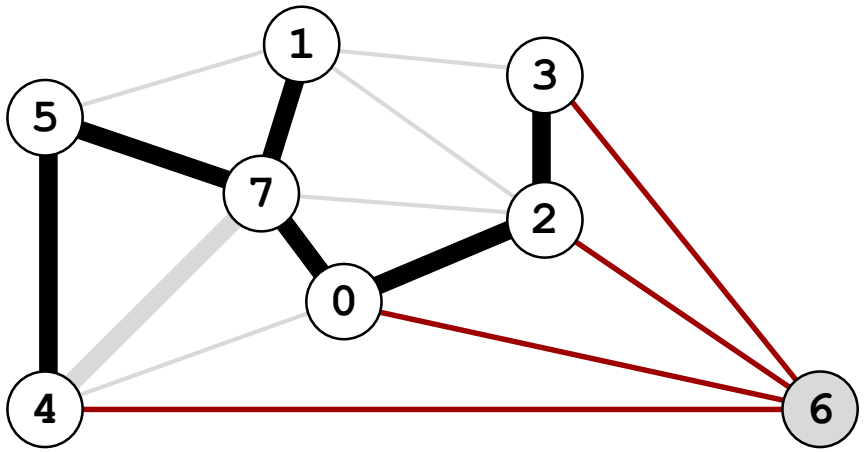
یالهای MST **0-7 1-7 0-2 2-3 5-7 4-5**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **4-7** *را از صف اولویت حذف کن*

یال‌های موجود در صف اولویت	
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93



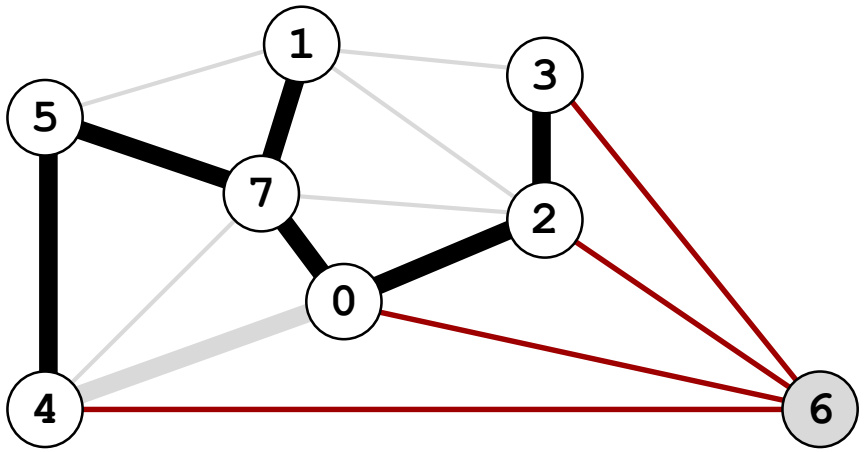
یالهای MST **0-7 1-7 0-2 2-3 5-7 4-5**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **0–4** *را از صف اولویت حذف کن*

یال‌های موجود در صف اولویت	
0–4	0.38
6–2	0.40
3–6	0.52
6–0	0.58
6–4	0.93



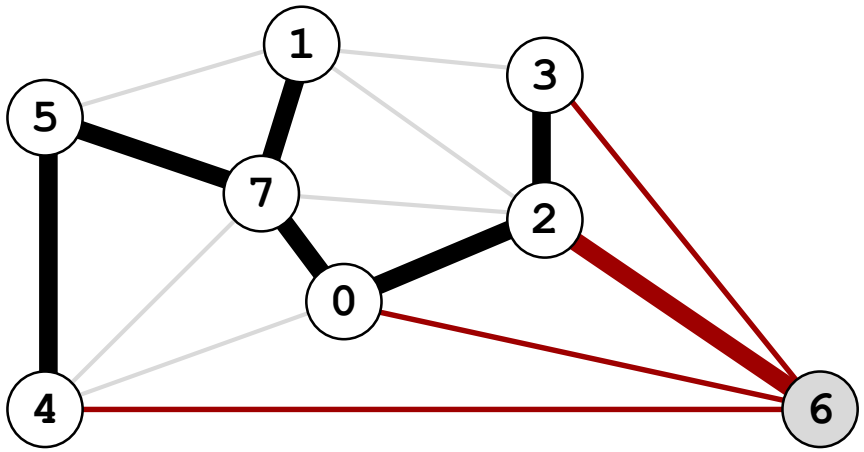
یالهای MST **0–7 1–7 0–2 2–3 5–7 4–5**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **6-2** *را از صف اولویت حذف و به درخت اضافه کن*

یال‌های موجود در صف اولویت	
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93



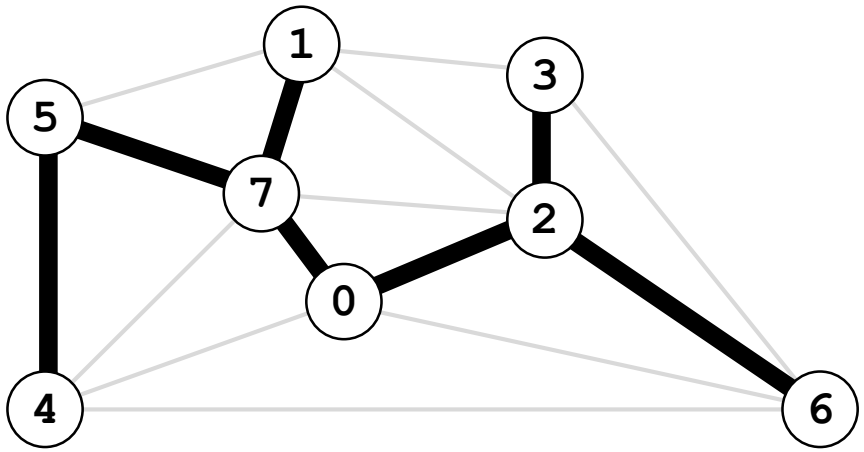
یالهای MST **0-7 1-7 0-2 2-3 5-7 4-5**

الگوریتم پریم (نسخه تنبل): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

یال **6-2** *را از صف اولویت حذف و به درخت اضافه کن*

یال‌های موجود در صف اولویت	
3-6	0.52
6-0	0.58
6-4	0.93



یالهای MST **0-7 1-7 0-2 2-3 5-7 4-5 6-2**

الگوریتم پریم (نسخه تنبل): پیاده‌سازی


```
public class LazyPrimMST {

   private boolean[] marked;          // MST vertices
   private Queue<Edge> mst;           // MST edges
   private MinPQ<Edge> pq;            // PQ of edges

   public LazyPrimMST(EdgeWeightedGraph G) {
      marked = new boolean[G.V()];
      mst = new Queue<Edge>();
      pq  = new MinPQ<Edge>();
      visit(G, 0);
      while (!pq.isEmpty())
      {
         Edge e = pq.delMin();
         int v = e.either(), w = e.other(v);
         if (marked[v] && marked[w]) continue;
         mst.enqueue(e);
         if (!marked[v]) visit(G, v);
         if (!marked[w]) visit(G, w);
      }
   }
```

فرض می کنیم G همبند است

حذف یال e با کمترین وزن از PQ

اگر دو سر e در T قرار دارند، آن را نادیده بگیر

یال e را به درخت اضافه کن

رأس v یا w را به درخت اضافه کن

الگوریتم پریم (نسخه تنبل): پیاده‌سازی


```
    private void visit(EdgeWeightedGraph G, int v)
    {
        marked[v] = true;
        for (Edge e : G.adj(v))
            if (!marked[e.other(v)])
                pq.insert(e);
    }

    public Iterable<Edge> mst()
    {   return mst;   }

}
```

افزودن رأس v به T

به ازای هر یال e = v-w، اگر رأس w در T نباشد، آن را به PQ اضافه کن

الگوریتم پریم (نسخه تنبل): زمان اجرا

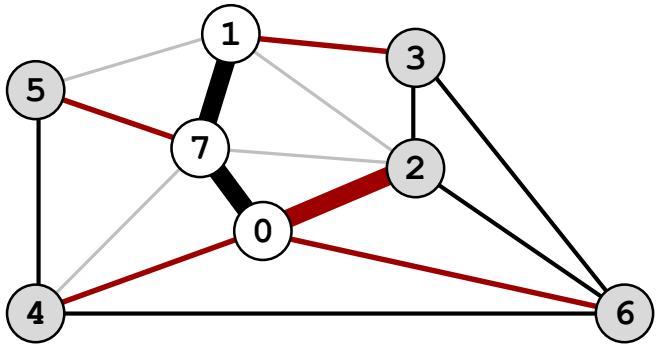
- گزاره. الگوریتم پریم درخت پوشای کمینه را در بدترین حالت در زمانی متناسب با $E \log E$ و حافظه‌ای متناسب با E محاسبه می‌کند.

- اثبات.

عمل	فراوانی	هرم دودویی
حذف کوچک‌ترین	E	$\log E$
درج	E	$\log E$

الگوریتم پریم: پیاده‌سازی مشتاق

- چالش. یافتن یال با کمترین وزن که دقیقاً یک سر آن در T باشد.
- راه‌حل مشتاق. رئوسی را که به وسیله‌ی یک یال به T متصل هستند در یک صف اولویت نگهداری کن، به طوری که اولویت رأس v برابر است با وزن کوتاه‌ترین یالی که v را به T وصل می‌کند.
  - نزدیک‌ترین رأس (v) به درخت را از صف حذف و یال مرتبط با آن را (یال e = v-w) به درخت اضافه کن.
  - صف اولویت را با در نظر گرفتن همه‌ی یالهای متلاقی با v (به صورت e = v-x) به روز رسانی کن:
    - اگر x در صف اولویت قرار دارد، آن را نادیده بگیر؛
    - در غیر این صورت x را به صف اولویت اضافه کن؛
    - اگر v-x کوتاه‌ترین یالی باشد که x را به درخت وصل می‌کند، اولویت x را افزایش بده.



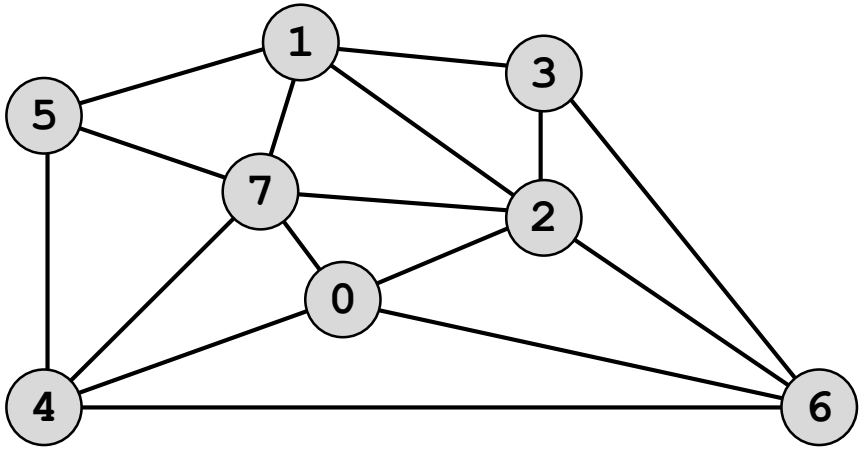

```
0
1   1-7   0.19
2   0-2   0.26
3   1-3   0.29
4   0-4   0.38
5   5-7   0.28
6   6-0   0.58
7   0-7   0.16
```

قرمز: در PQ ←

سیاه: در MST →

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

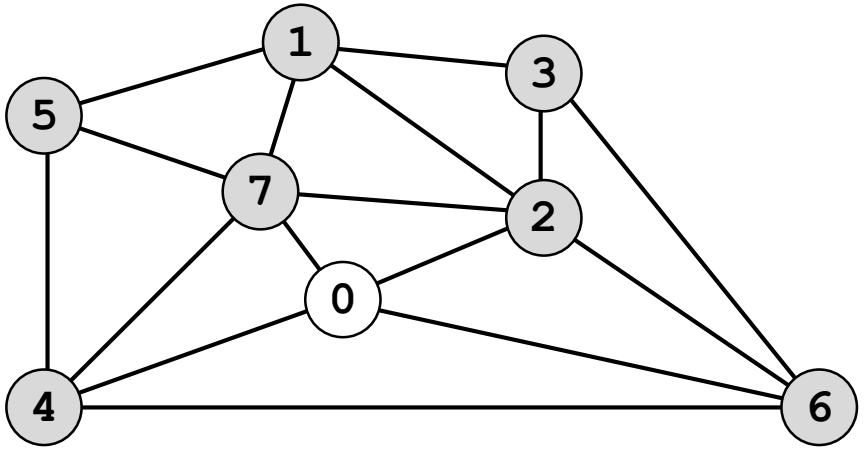
- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.




```
0-7   0.16
2-3   0.17
1-7   0.19
0-2   0.26
5-7   0.28
1-3   0.29
1-5   0.32
2-7   0.34
4-5   0.35
1-2   0.36
4-7   0.37
0-4   0.38
6-2   0.40
3-6   0.52
6-0   0.58
6-4   0.93
```

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

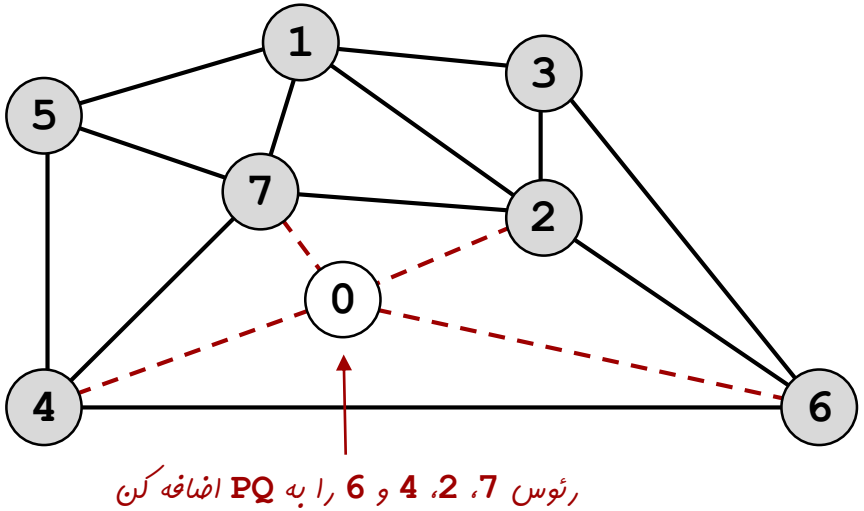
- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



v	edgeTo[]	distTo[]
0	-	-

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



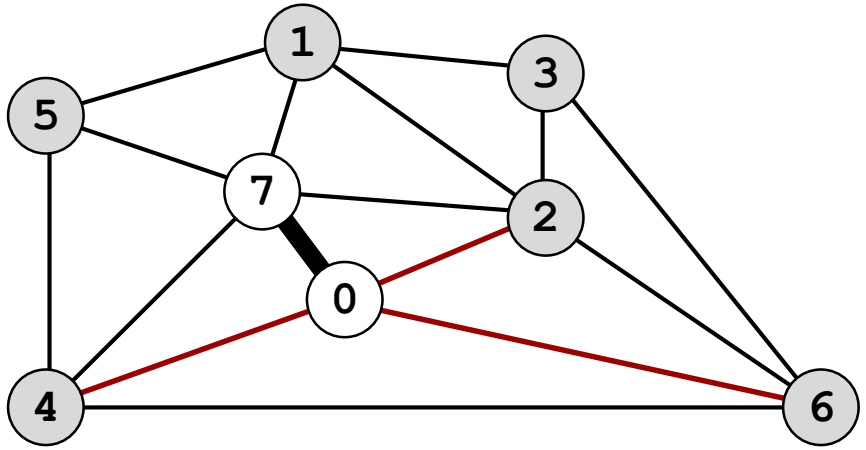
رئوس 7، 2، 4 و 6 را به PQ اضافه کن

v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
2	0-2	0.26
4	0-4	0.38
6	6-0	0.58

رئوس موجود در PQ
(مرتب بر اساس وزن)

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

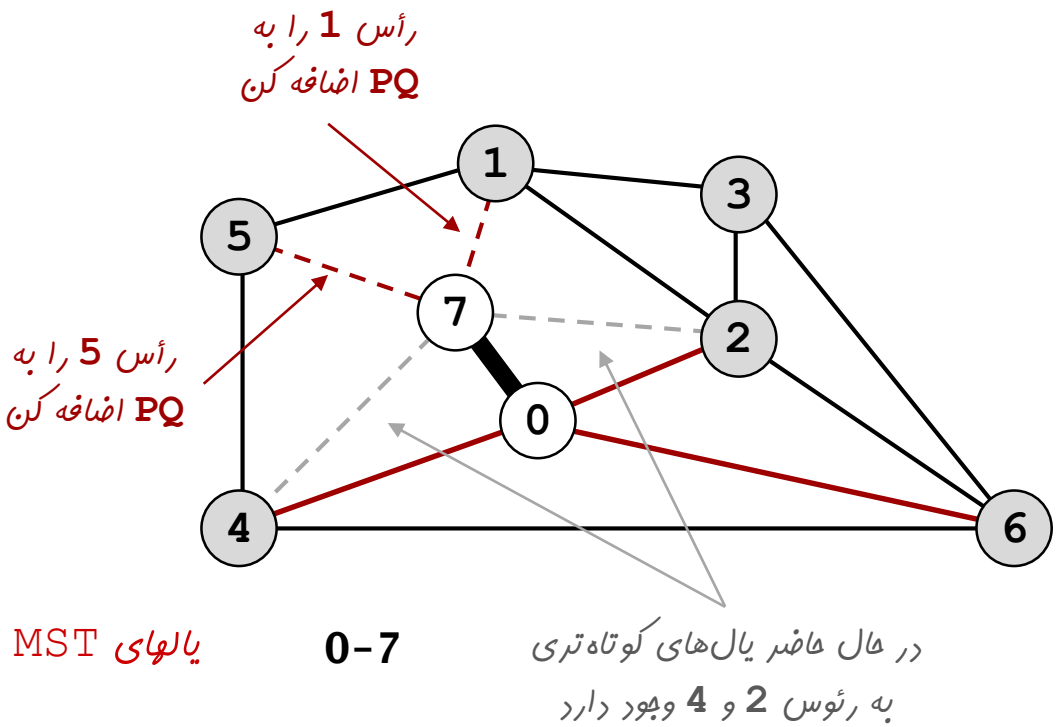


v	edgeTo[]	distTo[]
0	-	-
→ 7	0-7	0.16
2	0-2	0.26
4	0-4	0.38
6	6-0	0.58

یالهای MST 0-7

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

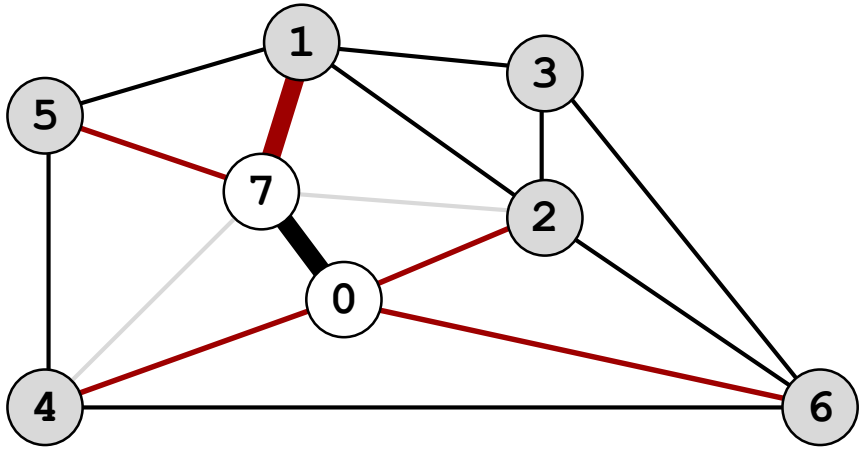
- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



v	edgeTo[]	distTo[]
0	-	-
→ 7	0-7	0.16
1	1-7	0.19
2	0-2	0.26
5	5-7	0.28
4	0-4	0.38
6	6-0	0.58

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

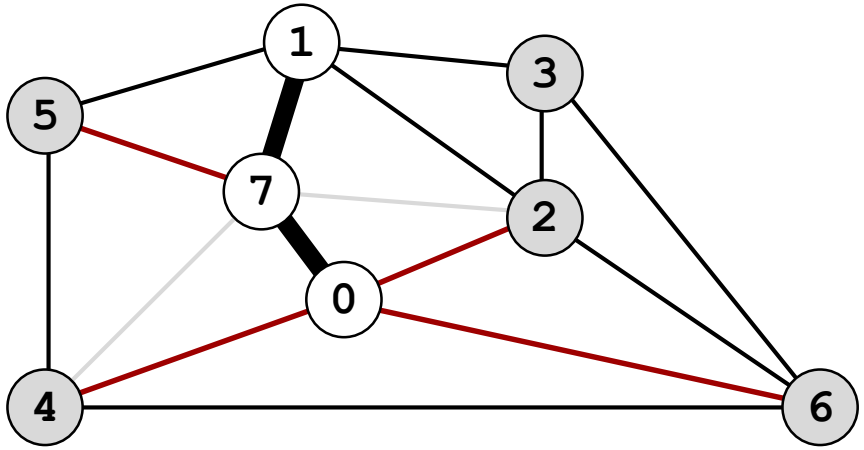


	v	edgeTo[]	distTo[]
	0	-	-
	7	0-7	0.16
→	1	1-7	0.19
	2	0-2	0.26
	5	5-7	0.28
	4	0-4	0.38
	6	6-0	0.58

یالهای MST 0-7

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



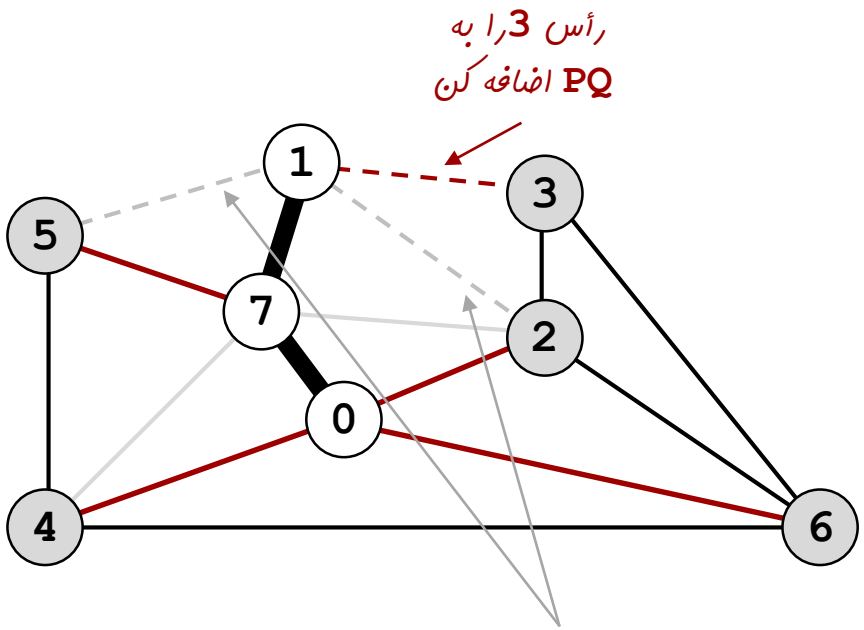
v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
→ 1	1-7	0.19
2	0-2	0.26
5	5-7	0.28
4	0-4	0.38
6	6-0	0.58

یالهای MST 0-7 1-7

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

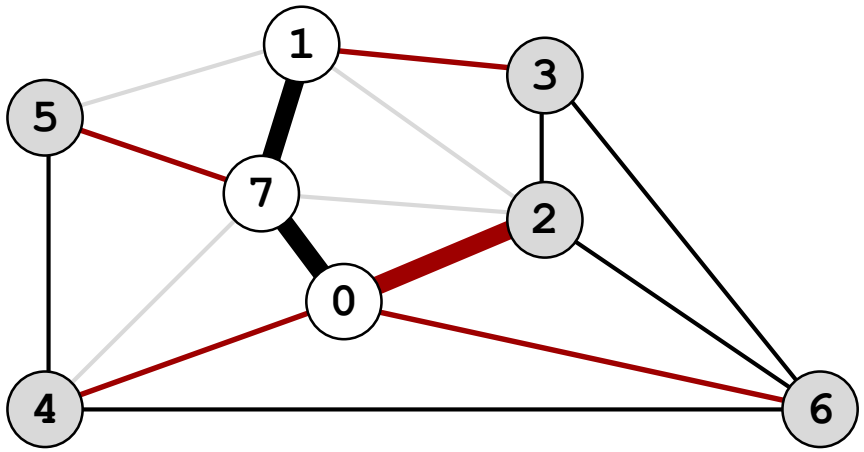
v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
→ 1	1-7	0.19
2	0-2	0.26
5	5-7	0.28
(3)	1-3	0.29
4	0-4	0.38
6	6-0	0.58



یالهای MST 0-7 1-7

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

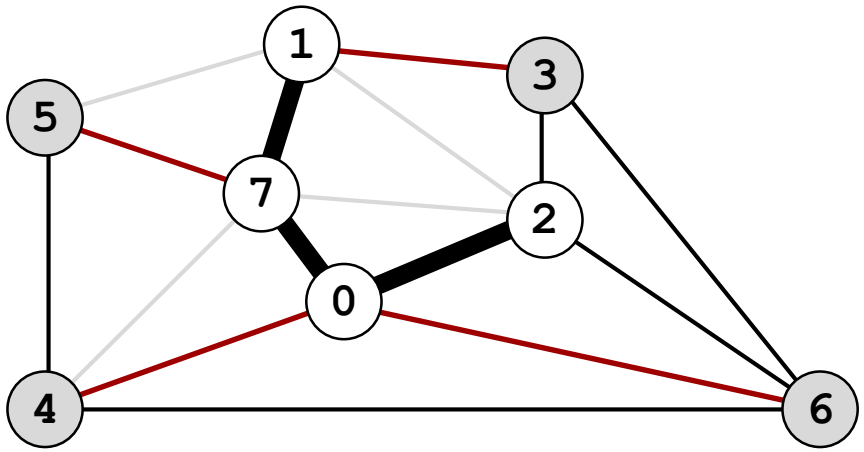


یالهای MST 0-7 1-7

v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
1	1-7	0.19
→ 2	0-2	0.26
5	5-7	0.28
3	1-3	0.29
4	0-4	0.38
6	6-0	0.58

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

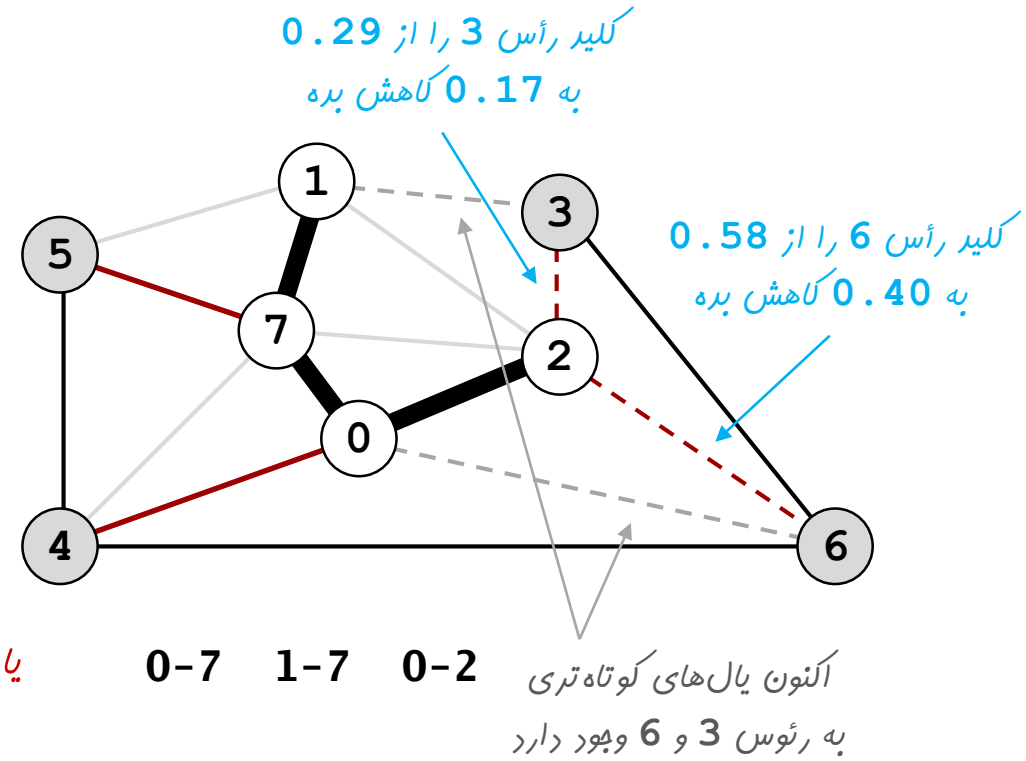


یالهای MST 0-7 1-7 0-2

v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
1	1-7	0.19
→ 2	0-2	0.26
5	5-7	0.28
3	1-3	0.29
4	0-4	0.38
6	6-0	0.58

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

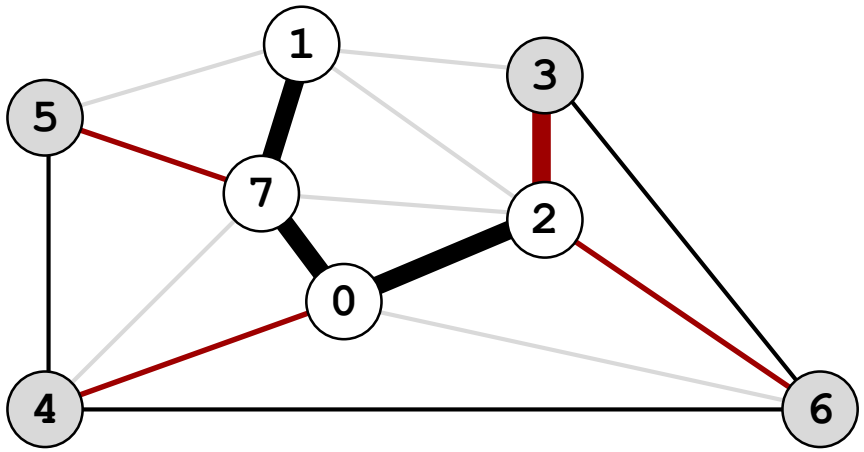
- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



	v	edgeTo[]	distTo[]
	0	-	-
	7	0-7	0.16
	1	1-7	0.19
→	2	0-2	0.26
	3	~~1-3~~ 2-3	~~0.29~~ 0.17
	5	5-7	0.28
	4	0-4	0.38
	6	~~6-0~~ 6-2	~~0.58~~ 0.40

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

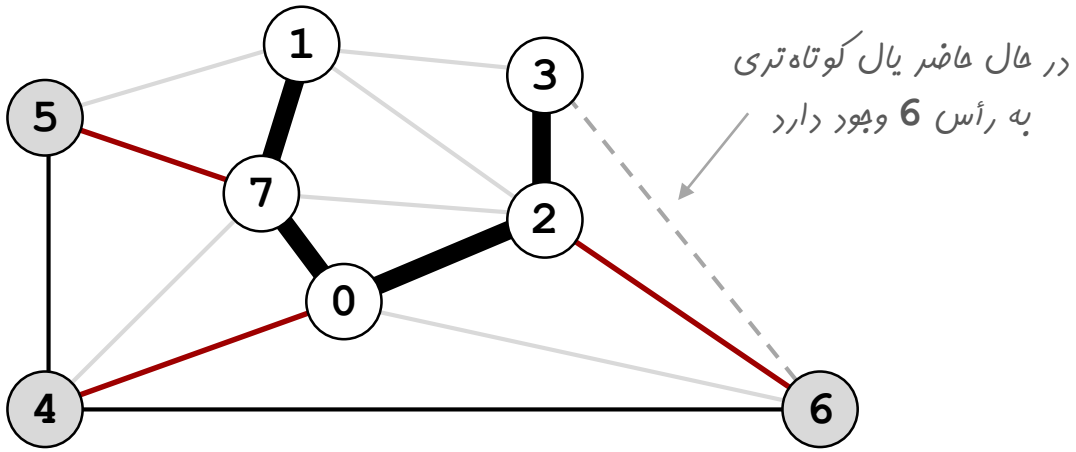


یالهای MST 0-7 1-7 0-2

	v	edgeTo[]	distTo[]
	0	-	-
	7	0-7	0.16
	1	1-7	0.19
	2	0-2	0.26
→	3	2-3	0.17
	5	5-7	0.28
	4	0-4	0.38
	6	6-2	.40

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

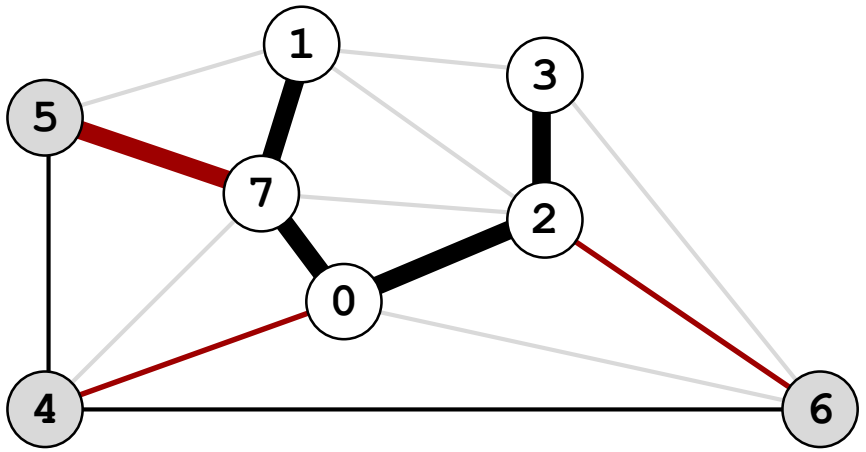


v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
1	1-7	0.19
2	0-2	0.26
→ 3	2-3	0.17
5	5-7	0.28
4	0-4	0.38
6	6-2	0.40

یالهای MST 0-7 1-7 0-2 2-3

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

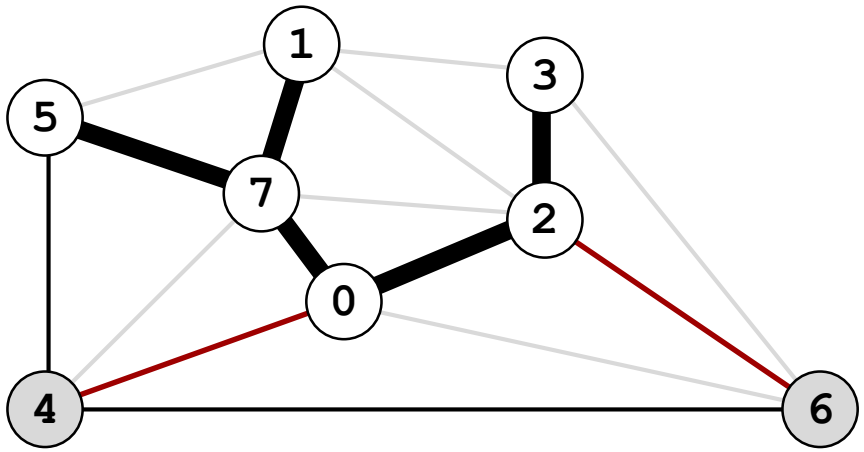


یالهای MST 0-7 1-7 0-2 2-3

	v	edgeTo[]	distTo[]
	0	-	-
	7	0-7	0.16
	1	1-7	0.19
	2	0-2	0.26
	3	2-3	0.17
→	5	5-7	0.28
	4	0-4	0.38
	6	6-2	0.40

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



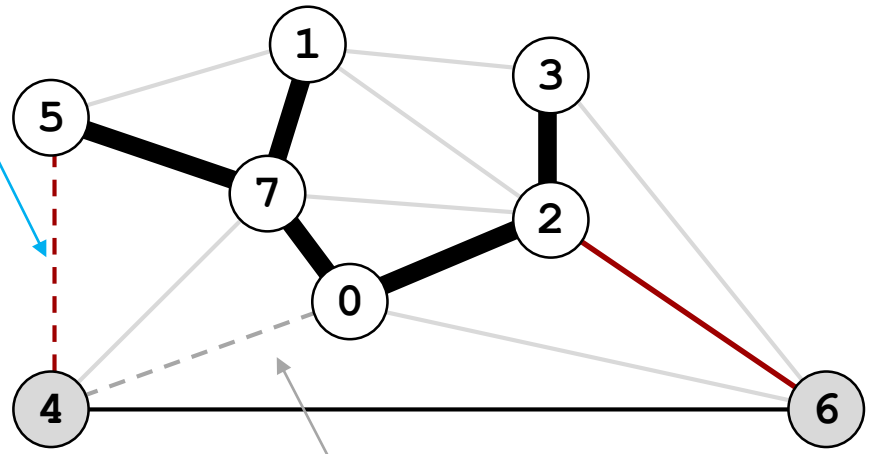
یالهای MST 0-7 1-7 0-2 2-3 5-7

	v	edgeTo[]	distTo[]
	0	-	-
	7	0-7	0.16
	1	1-7	0.19
	2	0-2	0.26
	3	2-3	0.17
→	5	5-7	0.28
	4	0-4	0.38
	6	6-2	0.40

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

کلید رأس 4 را از 0.38 به 0.35 کاهش بده



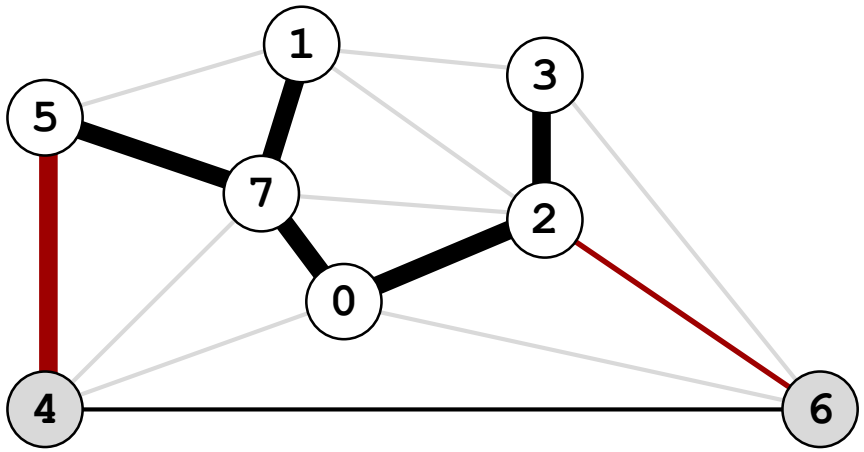
یالهای MST 0-7 1-7 0-2 2-3 5-7

اکنون یک یال کوتاه‌تر به رأس 4 وجود دارد

v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
1	1-7	0.19
2	0-2	0.26
3	2-3	0.17
→ 5	5-7	0.28
	4-5	0.35
(4)	~~0-4~~	~~0.38~~
6	6-2	0.40

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

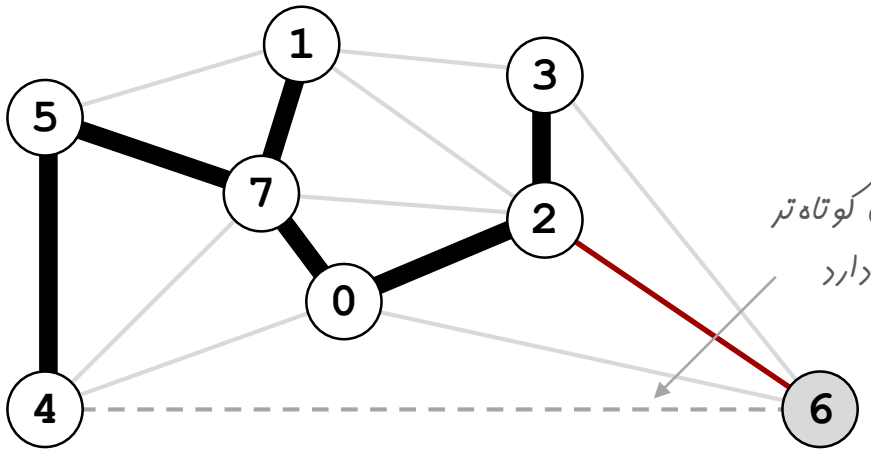


یالهای MST 0-7 1-7 0-2 2-3 5-7

	v	edgeTo[]	distTo[]
	0	-	-
	7	0-7	0.16
	1	1-7	0.19
	2	0-2	0.26
	3	2-3	0.17
	5	5-7	0.28
→	4	4-5	0.35
	6	6-2	0.40

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



در حال حاضر یک یال کوتاه‌تر به رأس 4 وجود دارد

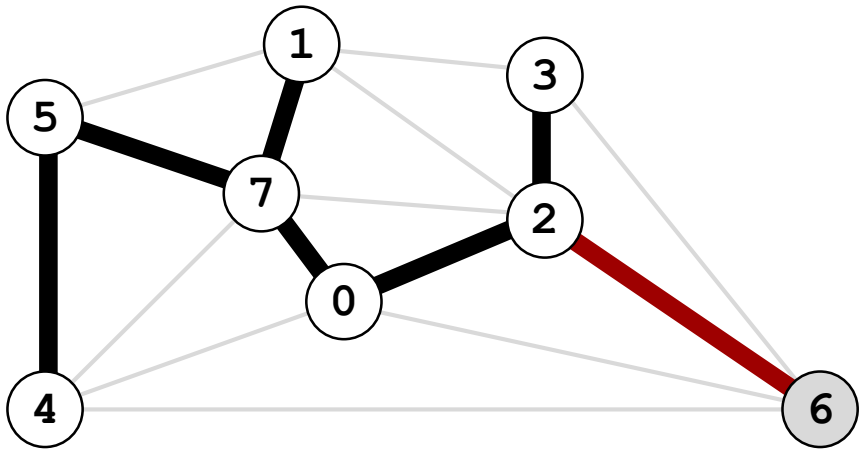
v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
1	1-7	0.19
2	0-2	0.26
3	2-3	0.17
5	5-7	0.28
→ 4	4-5	0.35
6	6-2	0.40

یالهای MST

0-7 1-7 0-2 2-3 5-7 4-5

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

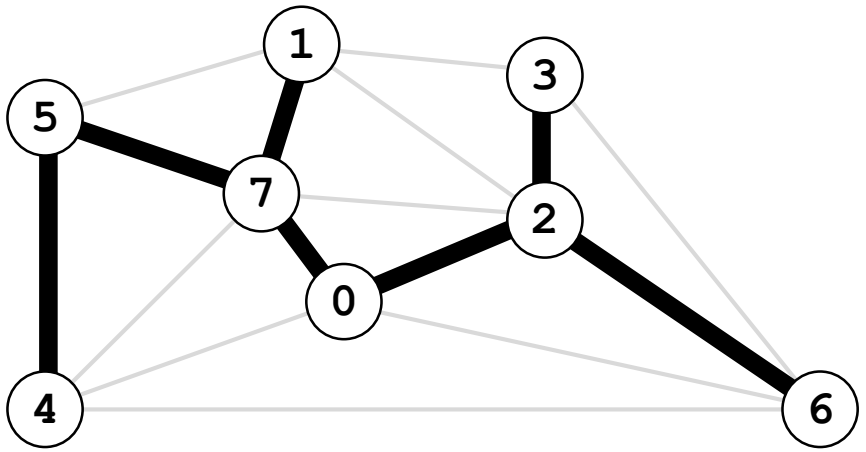


یالهای MST 0-7 1-7 0-2 2-3 5-7 4-5

v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
1	1-7	0.19
2	0-2	0.26
3	2-3	0.17
5	5-7	0.28
4	4-5	0.35
→ 6	6-2	0.40

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.

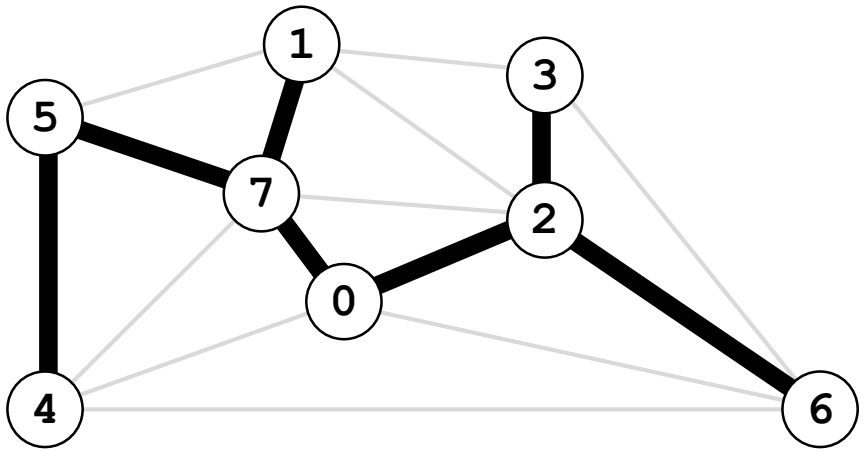


v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
1	1-7	0.19
2	0-2	0.26
3	2-3	0.17
5	5-7	0.28
4	4-5	0.35
→ 6	6-2	0.40

یالهای MST 0-7 1-7 0-2 2-3 5-7 4-5 6-2

الگوریتم پریم (نسخه‌ی مشتاق): اجرای نمایشی

- الگوریتم پریم.
  - با رأس صفر شروع کن و درخت T را به طور حریصانه بزرگ کن؛
  - در هر مرحله، یال با کمترین وزن را که دقیقا یک رأس آن در T است، به T اضافه کن؛
  - مراحل فوق تا به دست آوردن V - 1 یال تکرار کن.



v	edgeTo[]	distTo[]
0	-	-
7	0-7	0.16
1	1-7	0.19
2	0-2	0.26
3	2-3	0.17
5	5-7	0.28
4	4-5	0.35
6	6-2	0.40

یالهای MST 0-7 1-7 0-2 2-3 5-7 4-5 6-2

صف اولویت اندیس‌دار

- به هر کلید در صف اولویت یک اندیس بین 0 تا N - 1 نسبت بده.
  - عملیات: حذف عنصر با کوچک‌ترین کلید و درج یک عنصر جدید.
  - یک عمل جدید: کاهش کلید یک عنصر با مشخص کردن اندیس آن.

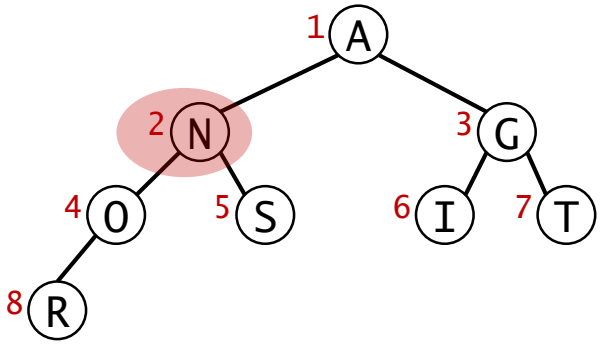
```
public class  IndexMinPQ <Key extends Comparable<Key>>

                IndexMinPQ(int N)        ایجاد یک صف اولویت اندیس‌دار با اندیس های 0 تا N - 1
           void insert(int k, Key key)   نسبت دادن اندیس k به کلید key
           void decreaseKey(int k, Key key)   کاهش کلید مرتبط با اندیس k
        boolean contains(int k)          آیا اندیس k در صف اولویت وجود دارد؟
            int delMin()                 حذف عنصر با کوچک‌ترین کلید و برگرداندن اندیس مرتبط با آن
        boolean isEmpty()                آیا صف اولویت خالی است؟
            int size()                   برگرداندن تعداد عناصر صف اولویت
```

پیاده‌سازی صف اولویت اندیس‌دار

- پیاده‌سازی.
  - با کدی مشابه با کد `MinPQ` شروع کن
  - برای پیاده‌سازی از سه آرایه `keys[]`، `pq[]` و `qp[]` استفاده کن به گونه‌ای که:
    - `Keys[i]` بیانگر اولویت `i` است.
    - `pq[i]` بیانگر اندیس کلیدی است که در مکان `i` از هرم قرار دارد.
    - `qp[i]` بیانگر مکان کلید با اندیس `i` در هرم است.
  - به منظور پیاده‌سازی عمل `decreaseKey(k, key)` از دستور `swim(qp[k])` استفاده کن.

i	0	1	2	3	4	5	6	7	8
keys[i]	A	S	O	R	T	I	N	G	-
pq[i]	-	0	6	7	2	1	5	4	3
qp[i]	1	5	4	8	7	6	2	3	-



الگوریتم پریم (نسخه مشتاق): پیاده‌سازی


```
public class PrimMST {

    private boolean[] marked;                  // true if v on tree
    private Edge[] edgeTo;                     // shortest edge from tree vertex
    private double[] distTo;                   // distTo[w] = edgeTo[w].weight()
    private IndexMinPQ<Double> pq;             // eligible crossing edges

    public PrimMST(EdgeWeightedGraph G) {
        marked = new boolean[G.V()];
        edgeTo = new    Edge[G.V()];
        distTo = new  double[G.V()];
        pq = new IndexMinPQ<Double>(G.V());

        for (int v = 0; v < G.V(); v++)
            distTo[v] = Double.POSITIVE_INFINITY;
        distTo[0] = 0;

        pq.insert(0, 0.0);
        while (!pq.isEmpty())
            visit(G, pq.delMin());
    }
```

الگوریتم پریم (نسخه مشتاق): پیاده‌سازی


```
    private void visit(EdgeWeightedGraph G, int v)
    {
        marked[v] = true;
        for (Edge e : G.adj(v))
        {
            int w = e.other(v);
            if (marked[w]) continue;
            if (e.weight() < distTo[w])
            {
                edgeTo[w] = e;
                distTo[w] = e.weight();
                if (pq.contains(w)) pq.decreaseKey(w, distTo[w]);
                else                pq.insert     (w, distTo[w]);
            }
        }
    }

    public Iterable<Edge> edges() {  /* exercise */  }
    public double weight()        {  /* exercise */  }
}
```

الگوریتم پریم: زمان اجرا

- زمان اجرای الگوریتم پریم بستگی به روش پیاده‌سازی صف اولویت دارد:
  - V مرتبه درج، V مرتبه حذف کوچک‌ترین، E مرتبه کاهش کلید.

پیاده‌سازی	درج	حذف کوچک‌ترین	کاهش کلید	زمان اجرا
آرایه	1	V	1	V^2
هرم دودویی	log V	log V	log V	E log V
هرم d طرفه	d $\log_d$ V	d $\log_d$ V	$\log_d$ V	E $\log_{E/V}$ V
هرم فیبوناچی	1 †	log V †	1 †	E + V log V

† هزینه‌ی سرشکن شده

- جمع‌بندی.
  - پیاده‌سازی با آرایه برای گراف‌های شلوغ بهینه است.
  - هرم دودویی برای گراف‌های خلوت بسیار سریع‌تر است.
  - هرم فیبوناچی به لحاظ نظری بهترین است، اما ارزش پیاده‌سازی ندارد.