مرکز آموزش الکترونیکی دانشگاه علم و صنعت ایران

طراحی و تحلیل الگوریتمها

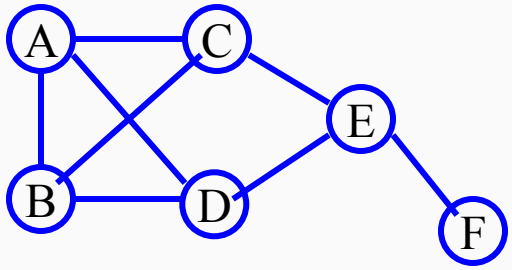
فصل چهارم: روشهای حریصانه

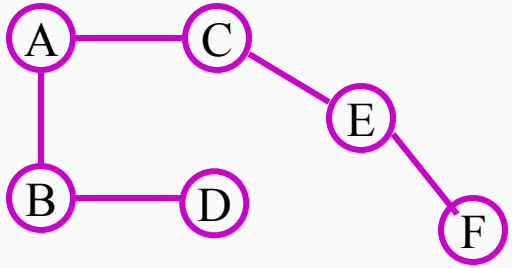
(جلسه دوم)

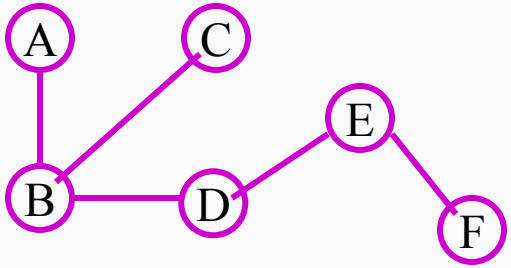
درختهای پوشای مینیمم

- فرض کنید که $G=(V,E)$ یک گراف همبند و بدون جهت است
- زیر گراف $T=(V',E')$ را یک درخت پوشا برای G گوییم هرگاه $V=V'$ و T یک درخت باشد
- زیر درخت پوشا دربردارنده همه رئوس گراف و بخشی از یالها می باشد به قسمی که یالها دور ایجاد نمی کند

مثال





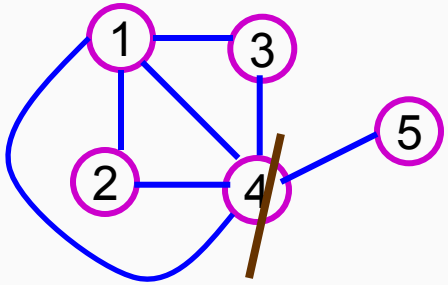


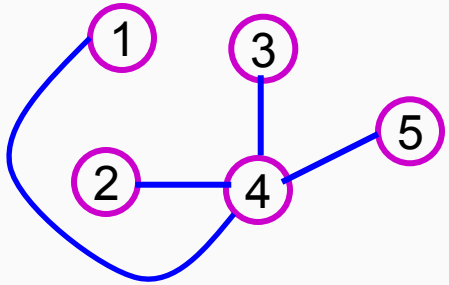
درختهای پوشا در گرافهای کامل

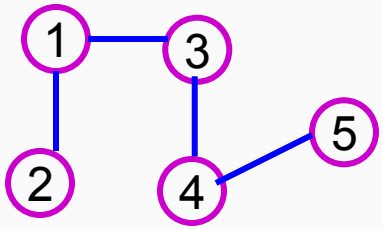
- گراف کامل با n راس گرافی است که بین هر دو راس متفاوت یک یال وجود دارد
- گراف کامل با n راس را با K_n نمایش می دهند
- هر درخت پوشا با n راس دارای n-1 یال می باشد
- تعداد درختهای پوشا برای گراف کامل K_n برابر n^{n-2} است.

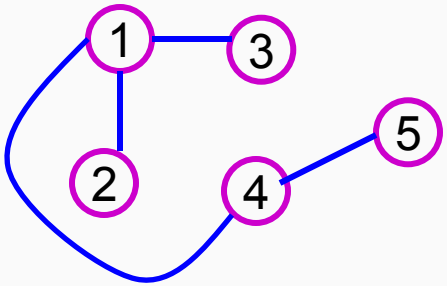
مثال

4^2 ❊





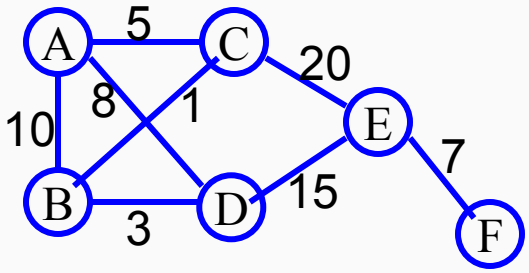


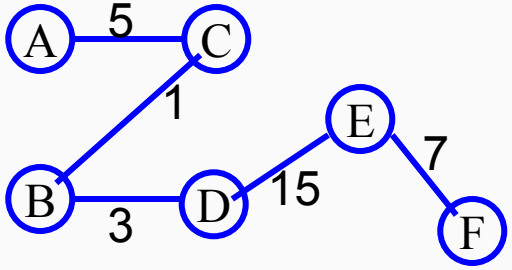


درخت پوشای وزن دار

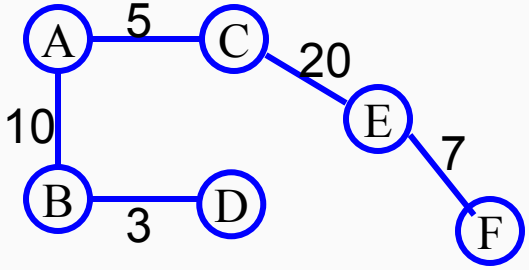
- اگر گراف وزن دار باشد، آنگاه درخت پوشای بدست آمده نیز وزن دار است
- وزن کلی یک درخت پوشا برابر مجموع همه اوزان یالهاست
- هدف یافتن درخت پوشای با وزن مینیمم است

مثال





31 وزن درخت پوشا



45 وزن درخت پوشا

الگوریتم کراسکال

* ایده: هر بار یالی را انتخاب کنید که بین یالهای باقی مانده دارای وزن کمتری باشد

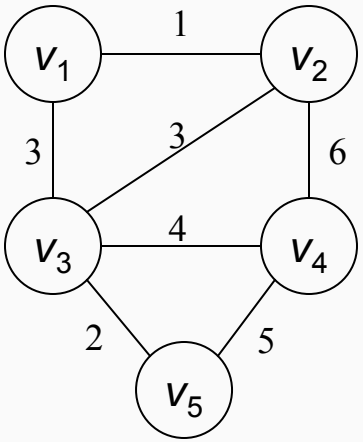
۱) یالها را به صورت صعودی مرتب کنید

۲) n-1 یال با وزن کمتر را مرحله به مرحله انتخاب کنید

۱_۲) در هر مرحله یال با وزن کمتر به شرطی انتخاب شود که دور ایجاد نکند

مثال

۱. مرتب سازی یال ها بر حسب طول



(v_1, v_2) 1

(v_3, v_5) 2

(v_1, v_3) 3

(v_2, v_3) 3

(v_3, v_4) 4

(v_4, v_5) 5

(v_2, v_4) 6

مثال(ادامه)

۲. ایجاد مجموعه های مجزا

(v_1, v_2) 1

(v_3, v_5) 2

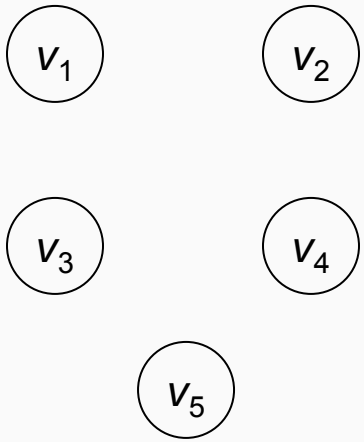
(v_1, v_3) 3

(v_2, v_3) 3

(v_3, v_4) 4

(v_4, v_5) 5

(v_2, v_4) 6



یک مثال

۳. انتخاب یال (v_1, v_2)

(v_1, v_2) 1

(v_3, v_5) 2

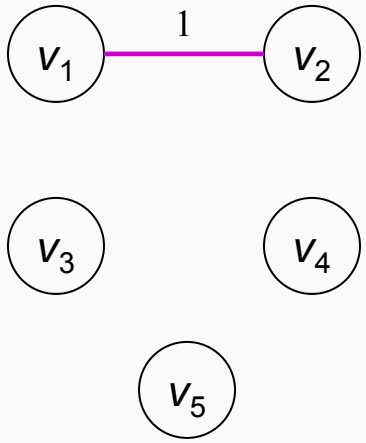
(v_1, v_3) 3

(v_2, v_3) 3

(v_3, v_4) 4

(v_4, v_5) 5

(v_2, v_4) 6



یک مثال

۴. انتخاب یال (v_3, v_5)

(v_1, v_2) 1

(v_3, v_5) 2

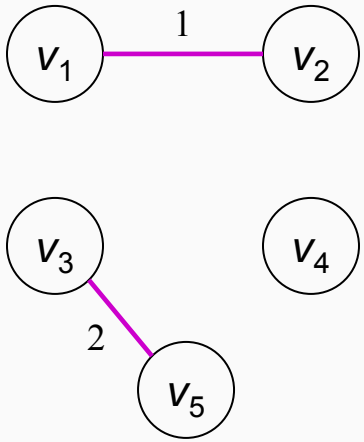
(v_1, v_3) 3

(v_2, v_3) 3

(v_3, v_4) 4

(v_4, v_5) 5

(v_2, v_4) 6



یک مثال

5. انتخاب یال (v_1, v_3)

(v_1, v_2) 1

(v_3, v_5) 2

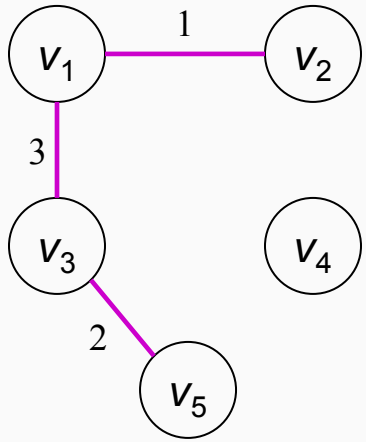
(v_1, v_3) 3

(v_2, v_3) 3

(v_3, v_4) 4

(v_4, v_5) 5

(v_2, v_4) 6



یک مثال

6. انتخاب یال (v_2, v_3)

(v_1, v_2) 1

(v_3, v_5) 2

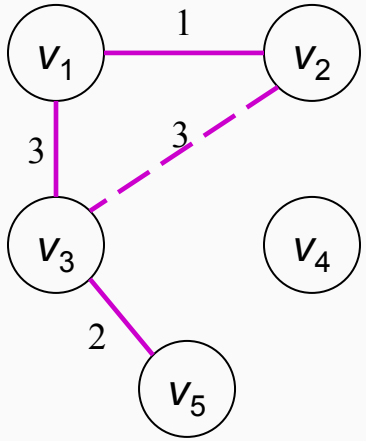
(v_1, v_3) 3

(v_2, v_3) 3

(v_3, v_4) 4

(v_4, v_5) 5

(v_2, v_4) 6



یک مثال

7. انتخاب یال (v_3, v_4)

(v_1, v_2) 1

(v_3, v_5) 2

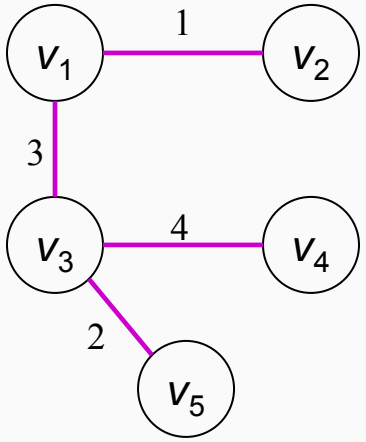
(v_1, v_3) 3

(v_2, v_3) 3

(v_3, v_4) 4

(v_4, v_5) 5

(v_2, v_4) 6



الگوریتم کراسکال

```
Algorithm Kruskal(Edge, n, T, Mincost)
Mincost=0;  Nedge=0;   i=1;   T={}
While Nedge<n-1 do
{  e=(u,v)=(Edge[i,1],Edge[i,2])
        اگر اضافه کردن e به  Tتشکیل دور ندهد آنگاه
    { T=T+{e}
      Mincost=Mincost+Cost[u,v]
      Nedge=Nedge+1
    }
i=i+1
}
```

→ O(|E| log|E|)

→ O(|E|)

→ O(log n)

→ O(|E| log|E|)

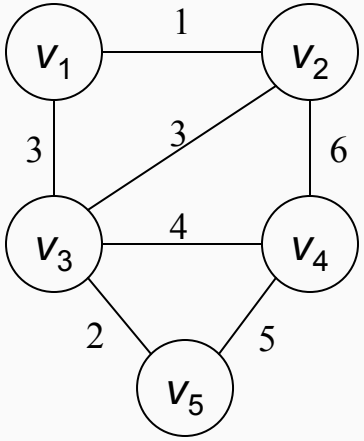
- Ω(n logn)
- $O(n^2 \log n)$

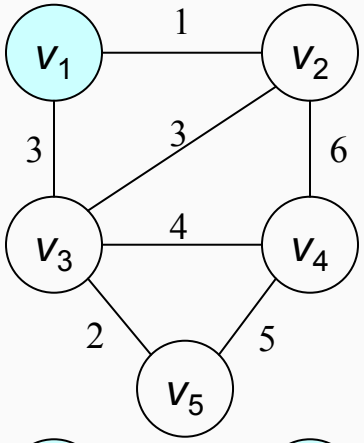
الگوریتم پریم

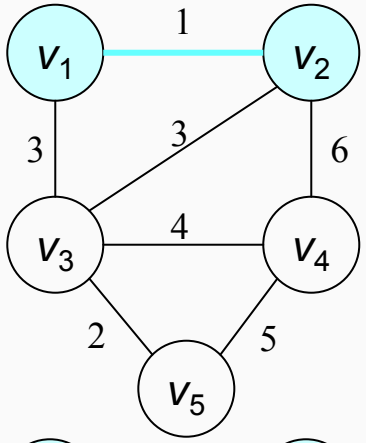
* ایده:

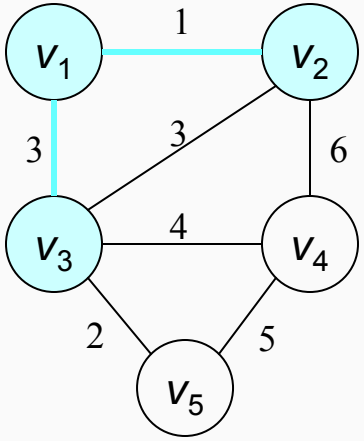
- از یک راس دلخواه شروع کنید. این راس را به عنوان راس پردازش شده تلقی می کنیم
- سپس تا زمانیکه همه رئوس پردازش نشده اند، راسی از بین رئوس پردازش نشده به گونه ای انتخاب خواهیم کرد که با حداقل هزینه به یکی از رئوس قبلاً پردازش شده، وصل شده باشد

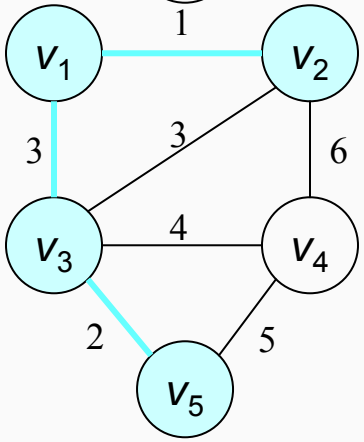
مثال

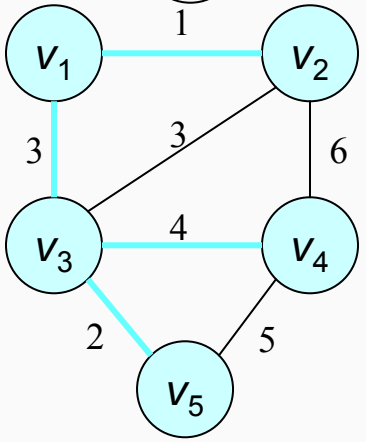

v_1
v_2
v_3
v_4
v_5
1
3
3
6
4
2
5


v_1
v_2
v_3
v_4
v_5
1
3
3
6
4
2
5


v_1
v_2
v_3
v_4
v_5
1
3
3
6
4
2
5


v_1
v_2
v_3
v_4
v_5
1
3
3
6
4
2
5


v_1
v_2
v_3
v_4
v_5
1
3
3
6
4
2
5


v_1
v_2
v_3
v_4
v_5
1
3
3
6
4
2
5

الگوریتم پریم

```
Algorithm Prim(cost,n,T,Mincost)
Mincost=0;
For i=2 to n do Near[i]=1;
Near[1]=0;
For i=1 to n-1 do {
   راس j را به گونه ای انتخاب کن که Near[j]<>0 و cost[j,Near[j]] هم مینیمم باشد      → O(n)
(T[i,1],T[i,2])=(j,Near[j])
Mincost=Mincost+cost[j,Near[j]]
Near[j]=0;
For k=2 to n do{                                                                      → O(n)
   if( Near[k]<>0  and  cost[k,Near[k]]>cost[k,j]) then  Near[k]=j
 }}
```

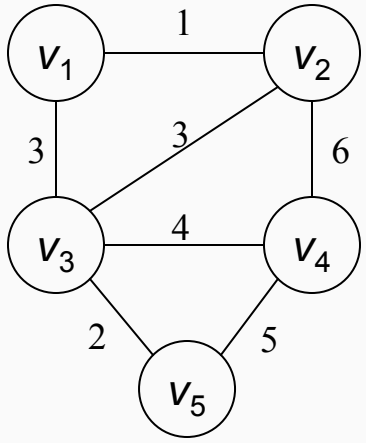
$O(n^2)$

مثال

j	Near[j]	Cost[j,Near[j]]
1	0	-
2	1	1
3	1	3
4	1	∞
5	1	∞

	1	2
1		
2		
3		
4		

Mincost=0

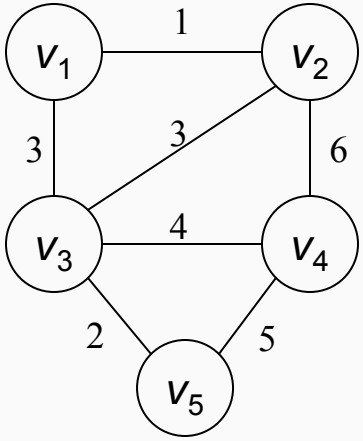


مثال (ادامه)

j	Near[j]	Cost[j,Near[j]]
1	0	-
2	0	-
3	1	3
4	2	6
5	1	∞

	1	2
1	2	1
2		
3		
4		

Mincost=1

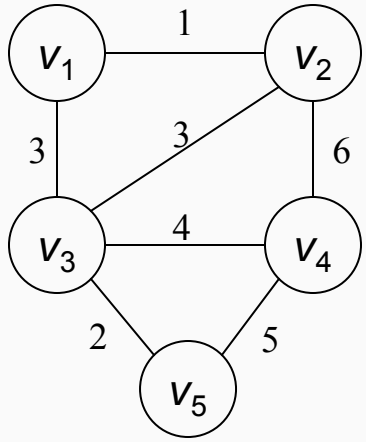


مثال (ادامه)

j	Near[j]	Cost[j,Near[j]]
1	0	-
2	0	-
3	0	-
4	3	4
5	3	2

	1	2
1	2	1
2	3	1
3		
4		

Mincost=4

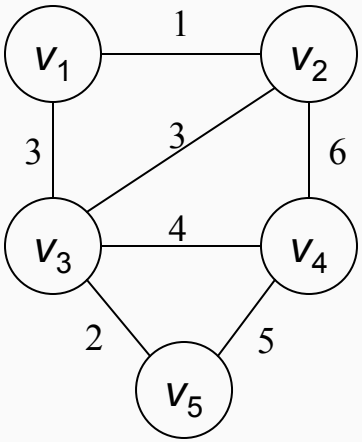


مثال (ادامه)

j	Near[j]	Cost[j,Near[j]]
1	0	-
2	0	-
3	0	-
4	3	4
5	0	-

	1	2
1	2	1
2	3	1
3	5	3
4	4	3

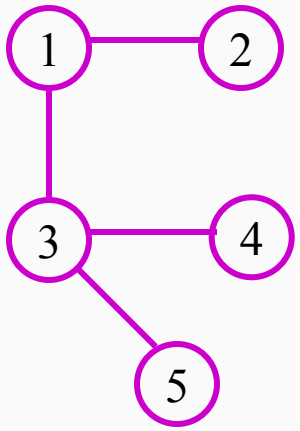
Mincost=6



Mincost=10

مثال (ادامه)

	1	2
1	2	1
2	3	1
3	5	3
4	4	3



کوتاهترین مسیرهای هم مبدأ

- در مساله کوتاهترین مسیرهای هم مبدأ، هدف تعیین کوتاهترین مسیرهای از یک راس مشخص مثل V_0، به عنوان راس مبدأ، به کلیه رئوس گراف است
- به عنوان مثال اگر گره های گراف را به منزله شهرها و یالها را به منزله جاده های بین شهرها در نظر بگیریم، هدف یافتن کوتاهترین مسیر از یک شهر خاص تا سایر شهر ها می باشد

الگوریتم دایکسترا (Dijkstra)

* ایده:

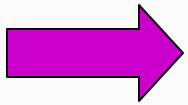
- در هر مرحله رئوس به دو دسته پردازش شده و نشده تقسیم می شوند
- در هر مرحله کوتاهترین مسیر از راس مبدأ به یک راس پردازش نشده بدست می آید به گونه ای که این مسیر کوتاهتر از سایر مسیرها تا رئوس دیگر پردازش نشده باشد

الگوریتم دایکسترا (ادامه)

- dist[j] برابر طول کوتاهترین مسیر از V_0 به j است به گونه ای که کلیه رئوس داخلی آن از مجموعه رئوس پردازش شده باشد
- در هر مرحله گره با dist مینیمم را انتخاب و آن را به مجموعه رئوس پردازش شده اضافه می کنیم
- اگر گرهی که در مرحله قبل انتخاب شد x باشد، آنگاه باید برای سایر رئوس پردازش نشده(t) با استفاده از مقایسه مقابل، dist آنها را به روز کرد: dist[x]+cost[x,t]<dist[t]

الگوریتم Dijkstra

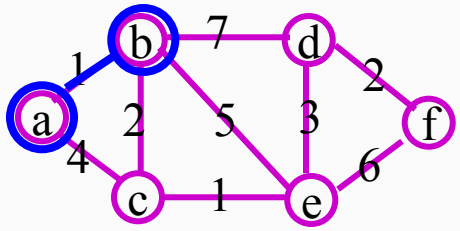
```
Algorithm Shortest_Paths(V0,cost,dist,n)
For i=1 to n do
   S[i]=False; dist[i]=cost[V0,i];
S[V0]=true; dist[V0]=0;
For num=1 to n-1 do{
        فرض xراسی باشد که S[x]=false و dist[x] هم مینیمم باشد
     S[x]=true;
     for(x  از حادث راس هر) do
            if S[t]=False and dist[x]+cost[x,t]<dist[t] then
                dist[t]=dist[x]+cost[x,t]
}
```



$O(n^2)$

مثال

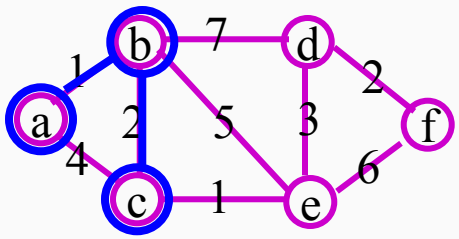
راس	S	dist	مسیر
a	1	-	-
b	0	1	a,b
c	0	4	a,c
d	0	∞	-
e	0	∞	-
f	0	∞	-



X=b

مثال (ادامه)

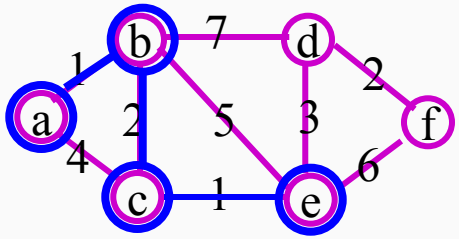
راس	S	dist	مسیر
a	1	-	-
b	1	1	a,b
c	0	3	a,b,c
d	0	8	a,b,d
e	0	6	a,b,e
f	0	∞	-



X=c

مثال (ادامه)

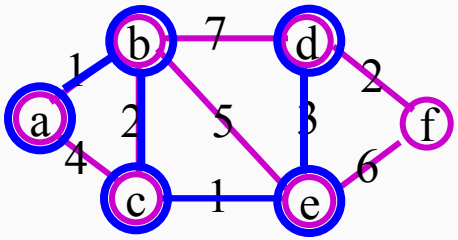
راس	S	dist	مسیر
a	1	-	-
b	1	1	a,b
c	1	3	a,b,c
d	0	8	a,b,d
e	0	4	a,b,c,e
f	0	∞	-



X=e

مثال (ادامه)

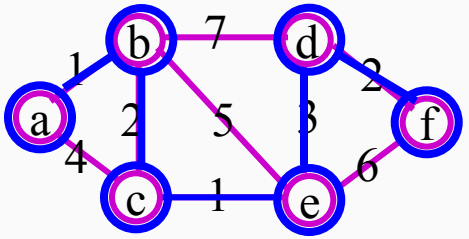
راس	S	dist	مسیر
a	1	-	-
b	1	1	a,b
c	1	3	a,b,c
d	0	7	a,b,c,e,d
e	1	4	a,b,c,e
f	0	10	a,b,c,e



X=d

مثال (ادامه)

راس	S	dist	مسیر
a	1	-	-
b	1	1	a,b
c	1	3	a,b,c
d	1	7	a,b,c,e,d
e	1	4	a,b,c,e
f	0	9	a,b,c,e,d,f



X=f