مولفه های همبندی گراف :

در یک گراف به دو گره u و v همبند گفته می شود اگر که مسیری بین این دو راس موجود باشد و برعکس. و گراف همبند نامیده می شود اگر بین هر زوج از رئوس گراف مسیر وجود داشته باشد

مولفه همبندی زیرگرافی از گراف است که همبند باشد

الگوریتم های پیدا کردن مولفه های همبندی در گراف غیرجهتدار:

الگوریتم 1 : با استفاده از پیمایش سطحی BFS

اگر گراف همبند باشد با یکبار فراخوانی تابع BFS کل رئوس گراف پیمایش خواهد شد. و اگر نه به تعداد دفعات فراخوانی تابع که همان تعداد مولفه های دیده نشده بعد از هربار فراخوانی می باشد مولفه همبندی داریم.

کد :

```
Void BFT( G, n)
{
    For I = 1 to n do
        Visited [I] = false
    For I = 1 to n do
        If Visited [I] == fasle
            BFS(I)
}
```

الگوریتم 2 : با استفاده از پیمایش عمقی DFS

کاملا مانند بالا ! با این تفاوت که هر بار به جای پیمایش سطحی از پیمایش عمقی استفاده می کنیم

بررسی و تحلیل الگوریتم

اگر از لیست همسایگی برای ذخیره سازی گراف استفاده شده باشد داریم :

پیچیدگی زمانی : $\Theta(V + E)$

پیچیدگی حافظه : $\Theta(V + E)$

اگر از ماتریس مجاورت برای ذخیره سازی گراف استفاده کنیم داریم :

پیچیدگی زمانی : $\Theta(V + E)$

پیچیدگی حافظه : $\Theta(V * V)$

پیدا کردن مولفه های قویا همبند در گراف جهتدار :

مولفه قویا همیند زیرگرافی است که بین هر راس **v** و **u** هم از اولی به دومی راه باشد هم از دومی به اول

الگوریتم :

ابتدا برای راسی دلخواه مثل **v** یکبار تابع پیمایش عمقی DFS را اجرا می کنیم. اگر از **v** به تمام رئوس راه بود به مرحله بعد می رویم و اگرنه (راسی بود که دیده نشده بود) این مولفه قویا همبند نیست

سپس ترانهاده گراف را بدست میاوریم! یعنی جهت تمام بالها را بر عکس می کنیم. اینبار هم از **v** پیمایش عمقی رو اجرا می کنیم. و اگر همه رئوس دیگر دیده شدند یعنی گراف قویا همبند است

توضیح :

در مرحله اول از **v** به تمام رئوس میرویم! و در سری بعد هر مسیر از **v** به بقیه دقیقا مسیری از بقیه رئوس به **v** در درخت اصلی است!

راس برشی :

در ریاضیات و علوم کامپیوتر، **راس برشی** راسی از گراف است که حذف آن باعث افزایش تعداد مولفه‌های همبندی گراف می شود. اگر گراف قبل از حذف آن راس همبند باشد، بعد از حذف ناهمبند می شود. راس برشی در شبکه‌های کامپیوتر (به عنوان گره) اهمیت ویژه ای دارد.

نکته : به طور کلی یک گراف غیر جهت دار همبند با n راس، حداکثر $n - 2$ راس برشی دارد (حالت زنجیر مانند که دقیقا $n - 2$ راس برشی دارد)

طبیعتا ممکن است گرافی راس برشی نداشته باشد.

نکته بدیهی : رئوسی که دارای راس مجاوری از درجه 1 باشند برشی هستند (بجز در گراف با 2 راس).

یال برشی نیز مشابه راس برشی است، به طوری که حذف آن باعث افزایش تعداد مولفه‌های همبندی گراف می شود.

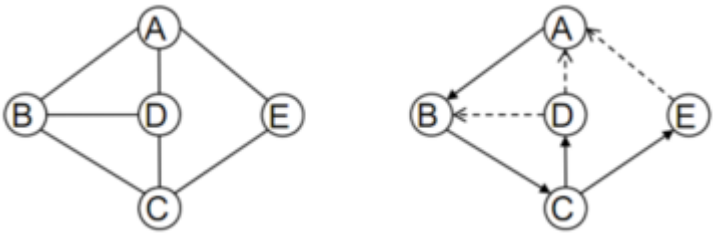
نکته : در درخت همه راس ها به جز برگ ها برشی هستند.

تعیین راس برشی:

یک الگوریتم بدیهی از مرتبه $O(V * E)$:

```
1  C = empty set (at the end of the algorithm it will
contain the cut vertices)
 2  a = number of components in G (found using DFS/BFS)
 3  for each i in V with incident edges
 4      b = number of components in G with i removed
 5      if b > a
 6          i is a cut vertex
 7          C = C + {i}
 8      endif
 9  endfor
```

الگوریتم بهینه شده :



راه حل از مرتبه O(V + E) می باشد و در صورت چند بخشی بودن گراف الگوریتم را برای هر بخش به کار می‌بریم.

نخست با شروع از یک راس دلخواه، الگوریتم DFS را اعمال می کنیم. از آنجا که ترتیب پیمایش در اینجا مهم است، برای یال‌های درخت حاصل از جستجو جهت تعیین می کنیم. در هنگام اعمال الگوریتم اگر به راسی رسیدیم که قبلا ملاقات شده از یال جهت دار نقطه چین شده استفاده می کنیم (به آنها یال برگشت Back - Edgeمی گوییم)

با رسیدن به هر راس، آن را شماره گذاری کنید (شماره هر(Num(v) ، برای هر راس، Low(v) را به صورت زیر تعریف می کنیم:

```
Low(v) = min{ قانون 1 : Num(v), قانون 2 : lowest Num(w)
among all back edges (v,w), قانون 3 : lowest Low(w) among
all tree edges (v,w)}
```

چگونگی مقدار دهی راس ها :

کلید B 2/1 : نشان دهنده Low(B) = 1 و Num(B) = 2 است. همان طور که در تعریف Low(v) آمد برای هر راس با توجه به اینکه کدام یک از سه شرط کمینه است Low(v) را تعیین می کنیم.

ابتدا به راس A ، مقدار Low(A) = Num(A) = 1 می دهیم- طبق قانون 1 - (طبیعتا ریشه همیشه 1 می گیرد). از آنجا بنا بر قانون دوم، Low(D) = Num(A) = 1 (چون از راس D به A یک بال برگشت وجود دارد)

حال برای راس C با توجه به قانون سوم، $Low(C) = Low(D) = 1$ و از آنجا طبق همان قانون $Low(B) = Low(C) = 1$.

از راس F به راس D یال برگشت وجود دارد پس طبق قانون دوم، $Low(F) = Num(D) = 4$ در نتیجه $Low(E) = Low(F) = 4$.

و در آخر چون G هیچ یال برگشتی ندارد پس $Low(G) = Num(G) = 1$

تعیین راس های برشی :

با توجه به اعداد بدست آمده برای هر راس:

ریشه یک راس برشی است اگرو تنها اگر بیش از یک فرزند داشته باشد

هر راس v غیر ریشه، برشی است اگر و تنها اگر فرزندی مانند w داشته باشد بطوریکه $Low(w) \leq Num(v)$

توضیح : حالت دوم تداعی کننده اینست که راس v فرزندی مانند w دارد که نمی تواند قبل از اینکه v پیمایش شود به راس دیگری دسترسی داشته باشد، یعنی حذف کردن v باعث انفصال w می شود.

در مثال بالا برای راس C ، $Low(G) = 7 \geq 3 = Num(C)$ و برای راس $Low(E) = 4 \geq 4 = Num(D)$، یعنی C و D راس‌های برشی هستند که رجوع به خود گراف این را تایید می کند.

شماره گذاری و تایین والدین :

```
1  // Assign Num and compute parents
2  Void Graph::assignNum(Vertex v)
3  {
4       v.Num = counter++;
5       V.visited = true;
6       for each Vertex w is adjacent to v
7               if(!w.visited)
8               {
9                       w.parent = v;
10                      assignNum(w);
11              }
```

```
12  }
```

تعیین Low و چاپ رأس های برشی :

```
1  //Assign Low and check for cut vertex
2  //Check for Root omitted
3  void Graph::assignLow(Vertex v)
4  {
5       v.Low = v.Num; // Rule 1
6       for each Vertex w adjacent to v
7       {
8               if( w.Num> v.Num) // Forward edge
9               {
10                      assignLow(w);
11                      if(w.Low>= v.Num)
12                              cout <<v <<” is a cut
vertex” <<endl;
13                      v.Low = min(v.Low, w.Low);
14              }
15              else
16              if(v.parent != w) // Back edge
17                      v.Low = min(v.Low, w.Num); //
Rule 2
18      }
19  }
```

راه حل 3 :

همانند راه حل بالا، الگوریتم DFS را اعمال می کنیم و درخت جستجوی جهت دار را بدست می آوریم. همان طور که گفته شد اگر راس v فرزندw ای داشته باشد که به هیچ ترتیب پیمایشی نتواند قبل از v ملاقات شود، حذف v باعث منفصل شدن w می‌شود یعنی v راس مفصل است. با تکیه بر این مطلب، در درخت جست و جوی بدست آمده:

ریشه مفصل است اگر و تنها اگر دارای بیش از 1 فرزند باشد.

هر راس v غیر ریشه مفصل است اگر دارای فرزند یا فرزندهایی مانند w باشد بطوریکه هیچ راسی در زیر درخت به ریشه w به هیچ یک از اجداد v با استفاده از یال برگشت وصل نشده باشند.

برای مثال در درخت روبرو، Bمفصل نیست زیرا در زیر گراف به ریشه C ، از راس D به راس A به عنوان نیای B یال برگشت وجود دارد ولی D مفصل است زیرا در زیرگراف به ریشه E هیچ راسی را نمی توان پیدا کرد که یال برگشتی به یکی از اجداد D داشته باشد C .نیز مفصل است زیرا در زیر گراف به ریشه G هیچ یال برگشتی وجود ندارد.

خوبی این روش نسبت به روش بالا در عدم نیاز به محاسبه Low و Num برای راس‌ها است.

یال برشی :

در نظریه گراف، یال برشی یالی از گراف است که حذف آن باعث افزایش تعداد مولفه های همبندی گراف می شود. اگر گراف قبل از حذف آن یال همبند باشد، بعد از حذف ناهمبند می شود. یال برشی در شبکه‌های کامپیوتری اهمیت ویژه ای دارد چرا که حذف آن کلا اتصال شبکه را مختل می کند.

طبیعتا ممکن است گرافی یال برشی نداشته باشد .راس برشی نیز مشابه یال برشی است، به طوری که حذف آن باعث افزایش تعداد مولفه‌های همبندی گراف می شود.

هر یال برشی حداقل یکی از راس هایش برشی است.(جز در حالات خاصی که یال حذف شده باعث ایجاد مولفه ای با یک راس می شود(در این حالات امکان دارد همچین چیزی رخ ندهد. مثلن اگر گراف فقط یک یال باشد))

نکته : همه یال‌های درخت برشی هستند.

قضیه : یالی از گراف برشی است اگر و فقط اگر در هیچ دوری قرار نگرفته باشد.

اثبات :

اثبات با برهان خلف: فرض کنید $e = uv$ یال برشی از گراف همبند G است و e در دور C شامل u، x,w,v,u...قرار گرفته است. گراف $G - e$ دارای یک مسیر u-v به شکلu، x،...،w، vاست، یعنی u به v متصل است. حال a و b را دو راس دلخواه از $G - e$ در نظر می گیریم و نشان می دهیم که $G - e$ مسیری از a به b دارد. چون G همبند بود پس مسیر a-b مانند P در G یافت می شود. اگر یال e در آن مسیر قرار نداشته باشد مسلما باز هم همان مسیر P در $G - e$ نیز وجود دارد و a به b در $G - e$ راه دارد. فرض کنید e در مسیر P قرار دارد. بنابراین P را می توان بصورتa ،...،v,u،...، aیاb ،...،u,v،...، bنشان داد. قبلا ثابت کردیم در$G - e$ ، از v به u مسیر وجود دارد. بنابراین اگر e در دوری قرار گرفته باشد، با حذف آن هنوز هم بین هر دو راس دلخواه a و b از $G - e$ مسیر وجود دارد که یعنی e یال برشی نیست. پس به تناقض رسیدیم.

برعکس فرض کنید $e = uv$ یالی است که در هیچ دوری قرار نگرفته است. دوباره با برهان خلف فرض کنید eیال برشی نیست. پس $G - e$ همبند است و مسیری مثل P از u به v در $G - e$ وجود دارد. در این صورت P باضافه e دوری در G ایجاد می‌کند که شامل e است که به تناقض می رسیم.

الگوریتم بافتن یال برشی :

1 : از اعداد بدست آمده در الگوریتم بهینه شده در قسمت یافتن راس برشی استفاده می کنیم

هر راس v که $Low(v) = Num(v)$ باشد در این صورت v و پدرش یال برشی هستند