درس طراحی الگوریتم
دانشکده ریاضی. دانشگاه صنعتی خواجه نصیرالدین طوسی **بهار ۹۹**
کتاب مرجع: طراحی الگوریتم، جان کلاینبرگ، اوا تاردش

الگوریتمهای پیمایش گراف

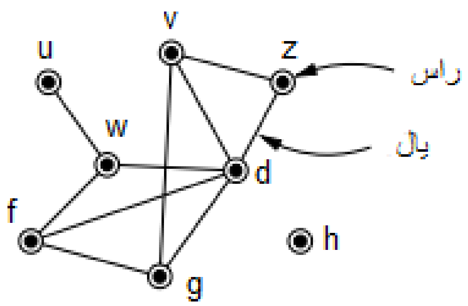
۱ مقدمه‌ای کوتاه بر گراف

گراف شامل مجموعه‌ای ناتهی از اشیاء است که به آنهای رأسهای گراف گفته می‌شود. معمولا مجموعه رئوس را با حرف V نشان می‌دهند. یک گراف همچنین شامل مجموعه‌ای از ارتباطات میان رئوس است که به هر ارتباط یک یال گفته می‌شود. یال‌ها در واقع رأس‌ها را به هم وصل می‌کنند. مجموعه یالها را معمولا با حرف E نمایش می‌دهند. با این توصیف گراف G بصورت یک زوج (V, E) تعریف می‌شود که V مجموعه رئوس و E مجموعه یالهای گراف G است.

دو رأس $a \in V$ و $b \in V$ که به هم متصل باشند یال $\{a, b\} \in E$ را تعریف می‌کنند. شکل زیر یک گراف با مجموعه رئوس $V = \{u, w, f, v, z, d, h, g\}$ و مجموعه یالهای

$$E = \{\{u, w\}, \{f, w\}, \{w, d\}, \{g, d\}, \{z, v\}, \{d, f\}, \{v, d\}, \{g, v\}\}$$

را نمایش می‌دهد.



چند تعریف:

۱. درجه راس: به تعداد رئوسی که به راس u متصل هستند، درجه راس u گفته می‌شود. در اینجا درجه راس u را با d_u نمایش می‌دهیم. درجه راس در واقع تعداد همسایه‌های یک راس در گراف است.

۲. مسیر: مسیر دنباله‌ای از راسهاست که بین هر دو راس متوالی در دنباله یالی وجود داشته باشد. برای مثال دنباله u, w, d, g یک مسیر در گراف شکل بالا است. همچنین v, d, w, f, w, u مسیری دیگر در این گراف است.

۳. گراف همبند: اگر بین هر دو راس گراف مسیری وجود داشته باشد، آنگاه گراف را همبند گویند. گراف شکل بالا همبند نیست چون راس h بصورت ایزوله است و مسیری به آن وجود ندارد.

۴. مولفه همبند: مجموعه‌ای حداکثری از رئوس گراف که همبند باشند (بین هر دو راس مسیری باشد) را یک مولفه همبند گراف گویند. گراف بالا دو مولفه همبند دارد:

$$\{u, w, f, v, z, d, g\} \qquad \{h\}$$

۲ نمایش گراف در حافظه کامپیوتر

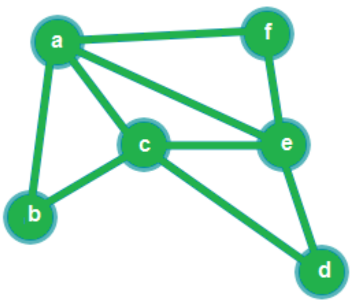
برای ذخیره گراف در حافظه کامپیوتر دو روش متداول وجود دارد: ماتریس مجاورتی و لیست مجاورتی.

۱.۲ ماتریس مجاورتی

فرض کنید گراف G با n راس داده شده است. مجموعه یالهای گراف G را می‌توان با یک ماتریس n در n نمایش داد. اگر اسم این ماتریس را A بگذاریم، متناظر با هر راس گراف G یک سطر در ماتریس A وجود دارد. درایه $A_{i,j}$ ماتریس را 1 قرار می‌دهیم اگر یالی بین راس i و راس j وجود داشته باشد در غیر این صورت مقدار $A_{i,j}$ صفر خواهد بود. ماتریس A را ماتریس مجاورتی گراف G گویند.

$$A_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

شکل زیر یک نمونه از ماتریس مجاورتی یک گراف را نشان می‌دهد.



$$\mathbf{A} = \begin{array}{c|c} & \begin{matrix} \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} & \mathbf{e} & \mathbf{f} \end{matrix} \\ \begin{matrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \\ \mathbf{e} \\ \mathbf{f} \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

بطور معمول ماتریس مجاورتی در قالب یک آرایه دو بعدی ذخیره می‌شود. در نتیجه یک درایه دلخواه ماتریس را می‌توان بسرعت از حافظه خواند (زمان دسترسی به آن $O(1)$ است). از این نتیجه می‌شود، جواب دادن به اینکه یالی بین دو راس دلخواه i و j وجود دارد یا نه در زمان $O(1)$ قابل انجام است. اما اگر بخواهیم همه همسایه‌های یک راس دلخواه را پیدا کنیم، باید کل سطر مربوطه را بخوانیم (حتی اگر این راس همسایه‌ای نداشته باشد!). با این توصیف، در نمایش ماتریس مجاورتی زمان پیدا کردن همسایه‌های یک راس برابر با $\Theta(n)$ است.

۲.۲ لیست مجاورتی

علاوه بر هزینه‌بر بودن محاسبه همسایه‌های یک راس، یکی از معایب دیگر نمایش ماتریس مجاورتی میزان حافظه‌ای است که برای ذخیره گراف استفاده می‌شود. برای گرافی با n راس باید n^2 عدد ذخیره شود. اگر گراف تعداد کمی یال داشته باشد، مثلا ($|E| = O(n)$) این اتلاف حافظه خواهد بود. یک روش کارا و مقرون به صرفه برای نمایش گراف، استفاده از لیست مجاورتی است. در این روش، برای هر راس مجموعه همسایه‌های آن در لیستی ذخیره می‌شود. برای مثال گراف شکل قبلی را می‌توان بصورت زیر ذخیره کرد.

a		b	c	e	f
b		a	c		
c		a	b	d	e
d		c	e		
e		a	c	d	f
f		a	e		

در روش لیست مجاورتی، میزان حافظه مورد نیاز برای ذخیره گراف متناسب با تعداد یالهاست. در واقع اگر گراف m یال داشته باشد، ذخیره لیست همسایه‌ها به $2m$ عدد نیاز دارد (هر یال دو بار آمده است.) در کل گراف را می‌توان با استفاده از $O(n+m)$ واحد حافظه ذخیره کرد. همچنین در این روش، زمان مورد نیاز برای محاسبه همسایه های یک راس دلخواه متناسب با تعداد همسایه‌ها خواهد بود. اما اگر بخواهیم چک کنیم آیا یالی بین دو راس i و j وجود دارد یا نه باید لیست همسایه‌های i (یا لیست همسایه‌های j) را بگردیم. اگر راس i یا j همسایه‌های زیادی داشته باشند، این کار زمانبر خواهد بود. دقت کنید در نمایش ماتریس مجاورتی می‌توانیم در زمان $O(1)$ به این سوال پاسخ دهیم.

۳ الگوریتمهای پیمایش گراف

منظور از پیمایش گراف، شروع از یک راس و ملاقات همه رئوسی است که از راس مبدا قابل دسترسی هستند (به آنها مسیری وجود دارد). برای مثال ممکن است دنبال مقداری خاص باشیم که در رئوس یا یالهای گراف ذخیره شده است، یا اینکه بخواهیم چک کنیم مسیری بین دو راس گراف وجو دارد یا نه. برای پیمایش گراف دو الگوریتم متداول وجود دارد که هر کدام دارای ویژگیها و کاربردهای خاص خود می‌باشد. در این قسمت به معرفی و بررسی این دو الگوریتم می‌پردازیم.

۱.۳ الگوریتم BFS

در الگوریتم جستجوی اول سطح یا Breadth First Search یا اختصارا BFS ، گراف بصورت سطحی پیمایش می‌شود. یعنی رئوسی که به راس مبدا نزدیکتر هستند زودتر پیمایش می‌شوند. بطور کلی زمان ملاقات یک راس بستگی به فاصله از راس مبدا دارد.

این الگوریتم از راس s گراف $G=(V,E)$ شروع کرده و همه رئوسی را که از s قابل دسترسی هستند به نوبت علامت می‌زند. ترتیب علامت خوردن بستگی به فاصله از s دارد. ابتدا راسهایی که مستقیما به s وصل هستند علامت می‌خورند (لایه اول)، سپس راسهایی که قبلا علامت نخوردند و به لایه اول متصل هستند علامت می‌خورند (لایه دوم)، و به همین ترتیب لایه‌های بعدی شکل می‌گیرند.

کار الگوریتم را می‌توان به یک سری مرحله تقسیم کرد. در شروع مرحله iام، لیست $L[i]$ موجود است که در مرحله قبلی ساخته شده است. در این مرحله راسهای لایه $i+1$ام شناسایی شده و به لیست $L[i+1]$ اضافه می‌شوند. در شروع هر مرحله یک لیست جدید (برای نگهداری راسهای لایه جدید) ساخته می‌شود. اگر در مرحله iام لیست $L[i]$ تهی باشد (وقتی اتفاق می‌افتد که در مرحله قبلی هیچ راسی علامت نخورده باشد) الگوریتم پایان می‌پذیرد.

از آرایه منطقی D برای ثبت علامت راسها استفاده می‌کنیم به این معنی که $D[u]=\text{true}$ اگر و فقط اگر راس u علامت خورده باشد است. الگوریتم در حین پیمایش گراف درخت BFS را نیز می‌سازد.

دقت کنید هر یالی که باعث شود راسی علامت نخورده ملاقات شود به درخت BFS اضافه می‌شود.

۱. $D[s] \leftarrow \text{true}$

۲. $\forall v \in V/\{s\} \quad D[v] \leftarrow \text{false}$

۳. $L[0]$ را ایجاد کن. سپس $L[0] \leftarrow (s)$ (لایه صفرم تنها شامل راس s است)

۴. $i \leftarrow 0$ (شمارنده مرحله)

۵. $T \leftarrow \emptyset$ (T درخت BFS در آغاز کار تهی است)

۶. تا زمانیکه $L[i]$ تهی نیست اعمال زیر را انجام بده:

۱.۶ لیست جدید $L[i+1]$ را ایجاد کن. در شروع کار لیست خالی است.

۲.۶ برای هر راس u در لیست $L[i]$ اعمال زیر را انجام بده:

- هر بار یک یال (u,v) که روی راس u قرار دارد را در نظر بگیر.
- اگر $D[v]=\text{false}$ آنگاه:

$D[v] \leftarrow \text{true}$

یال (u,v) را به درخت T اضافه کن.

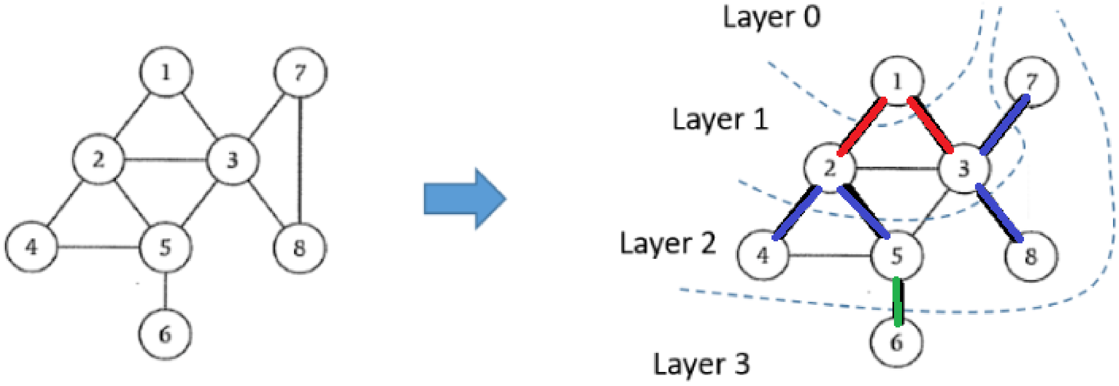
راس v را به لیست $L[i+1]$ اضافه کن.

۳.۶ $i \leftarrow i+1$

۷. پایان

یک مثال

در مثال الگوریتم BFS را از راس شماره 1 شروع کرده‌ایم. لایه‌ها در شکل نشان داده شده‌اند. درخت BFS با خطوط پررنگ مشخص شده است.



$L[0] = (1) \qquad L[1] = (2,3) \qquad L[2] = (4,5,7,8) \qquad L[3] = (6) \qquad L[4] = \emptyset$

تحلیل زمان اجرای BFS. در الگوریتم BFS برای هر راس، زمانی می‌رسد که همسایه‌هایش بررسی می‌شوند و آنهایی که علامت نخورده‌اند به لایه جدید اضافه می‌شوند. اگر گراف بصورت لیست مجاورتی ذخیره شده باشد، در صورتی که راس دلخواه x به تعداد d همسایه داشته باشد، چک کردن همسایه‌های x متناسب با $\Theta(d)$ زمان خواهد برد. اما اگر گراف بصورت ماتریس مجاورتی ذخیره شده باشد، این کار $\Theta(n)$ زمان می‌برد. اگر d_i درجه راس iام باشد، در حالت لیست مجاورتی زمان پردازش همسایه‌ها برابر است با

$$\Theta(d_1) + \Theta(d_2) + \ldots + \Theta(d_n) = \Theta(\sum_{i=1}^{n} d_i) = \Theta(2m) = \Theta(m)$$

علاوه بر این، به یک آرایه به طول n برای ثبت علامت رئوس نیاز داریم (آرایه D در الگوریتم بالا) که مقداردهی اولیه آن $\Theta(n)$ زمان می‌برد پس در کل زمان اجرای الگوریتم BFS برابر با $\Theta(n+m)$ است.
در حالت ماتریس مجاورتی زمان اجرا برابر است با:

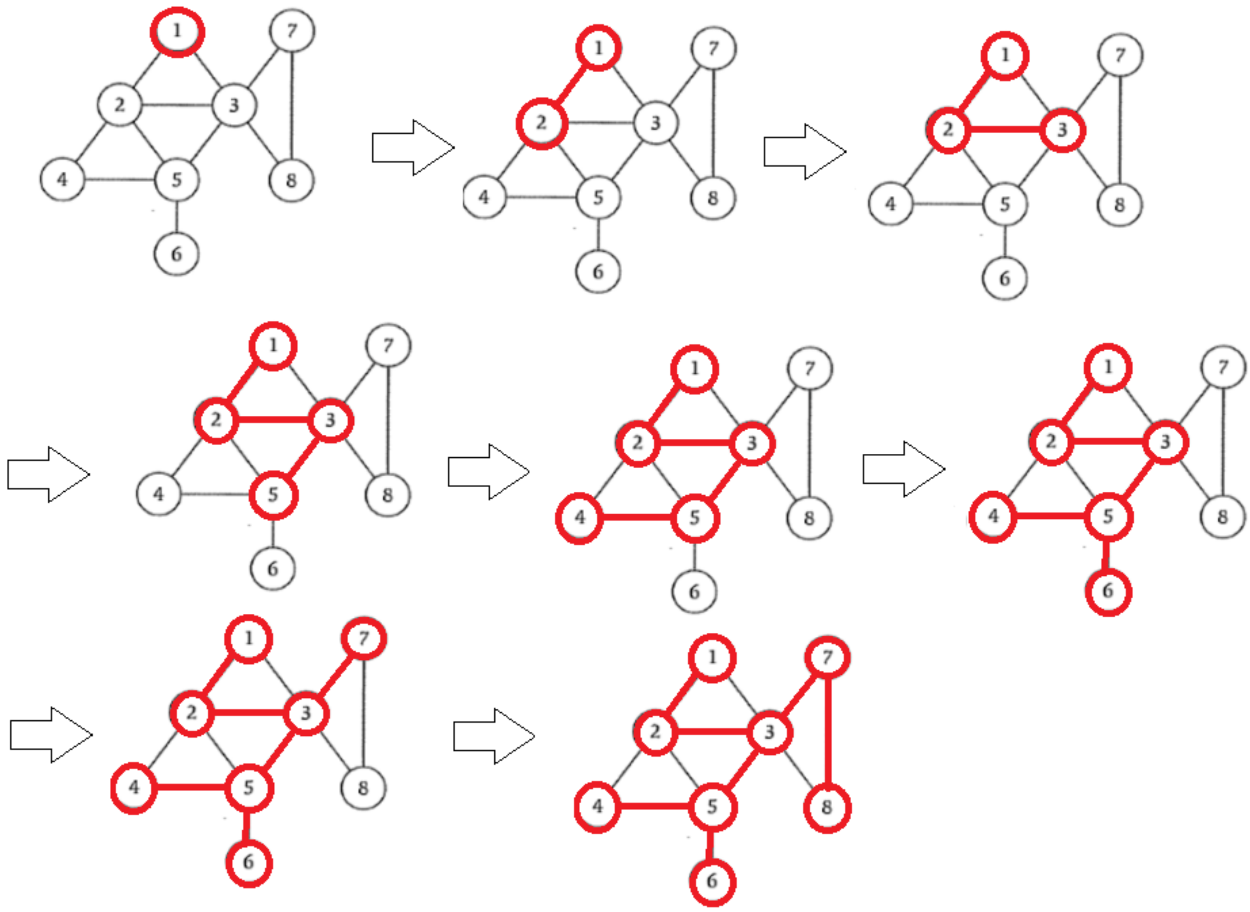
$$\sum_{i=1}^{n} \Theta(n) = \Theta(n^2)$$

۲.۳ الگوریتم DFS

بر خلاف الگوریتم BFS ، الگوریتم اول عمق یا Depth First Search و یا اختصارا DFS گراف را بصورت عمقی پیمایش می‌کند. پیمایش از یک راس شروع شده و در یک جهت بصورت عمقی ادامه پیدا می‌کند تا اینکه دیگر امکان پیشروی عمقی وجود نداشته باشد. در اینجا، الگوریتم به راس قبل از بن بست برمی‌گردد و پیمایش دصورت عمقی در یک جهت دیگر ادامه پیدا میکند.

بطور دقیقتر، زمانی که در راسی هستیم، همسایه‌ای که قبلا ملاقات نشده را انتخاب می‌کنیم و به آنجا می‌رویم و آن را علامت می‌زنیم. اگر چند همسایه باشند که علامت نخورده‌اند، مطابق ترتیبی که برای همسایه‌ها قائل شده‌ایم آنکه اولویت بیشتری دارد را انتخاب می‌کنیم. اگر همه همسایه‌ها علامت خورده باشند، به راس قبلی برمی‌گردیم و کار را از آنجا دنبال میکنیم. اینکار اینقدر ادامه پیدا می‌کند تا زمانی که به راس مبدا بازگشته‌ایم و دیگر همسایه‌ای برای علامت زدن وجود ندارد.

شکل زیر یک اجرا از الگورتیم DFS را نشان می‌دهد. دقت کنید ترتیب همسایه‌ها مهم است. هر راس لیستی مرتب شده از همسایه‌ها دارد، که برای انتخاب حرکت بعدی به آن مراجعه می‌شود. دقت همیشه در لیست اولین همسایه‌ای که علامت نخورده انتخاب می‌شود. مشابه پیمایش BFS ، اینجا هم یک درخت DFS داریم. یالی که باعث می‌شود راسی جدید ملاقات شود به درخت DFS اضافه می‌شود.



1		2	3			
2		1	3	4	5	
3		1	2	5	7	8
4		2	5			
5		2	3	4	6	
6		5				
7		3	8			
8		3	7			

جدول بالا برای هر راس، لیست همسایه‌ها را نشان می‌دهد. در مثال بالا راسها به ترتیب زیر ملاقات می‌شوند.

زمان اولین ملاقات پررنگ شده است.

$$\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{5}, \mathbf{4}, 5, \mathbf{6}, 5, 3, \mathbf{7}, \mathbf{8}, 7, 3, 2, 1$$

الگوریتم DFS یک توصیف بازگشتی دارد که در زیر مشاهده می‌کنید. فراخوانی $DFS(u)$ پیمایش DFS را از راس مبدا u آغاز می‌کند.

DFS(u)
۱. راس u را علامت بزن
۲. برای هر همسایه u کارهای زیر را انجام بده:
فرض کن v یک همسایه u باشد.
اگر v علامت خورده است، از آن صرف نظر کن،
در غیر این صورت $DFS(v)$ را فراخوانی کن.

تحلیل زمان اجرای DFS. زمان اجرای الگوریتم DFS بستگی به پیاده‌سازی آن و شیوه نمایش گراف در حافظه دارد. الگوریتم بالا یک توصیف سطح بالا و بازگشتی را از الگوریتم ارائه می‌دهد و به جزئیات پیاده‌سازی نمی‌پردازد. برای مثال، برای یک راس، چگونه یک همسایه علامت نخورده را پیدا کنیم؟ با فرض اینکه گراف بصورت لیست مجاورتی ذخیره شده است، یک پیاده‌سازی با استفاده از ساختارداده پشته وجود دارد که زمان اجرای آن $O(n+m)$ است. برای جزئیات پیاده‌سازی و تحلیل آن بخش ۳.۳ از کتاب مرجع را ببینید. البته واضح است که الگوریتم را می‌توان در زمان $O(n^2)$ پیاده‌سازی کرد. دانشجو باید این را براحتی مشاهده کند.

۴ چند کاربرد از الگوریتمهای پیمایش گراف

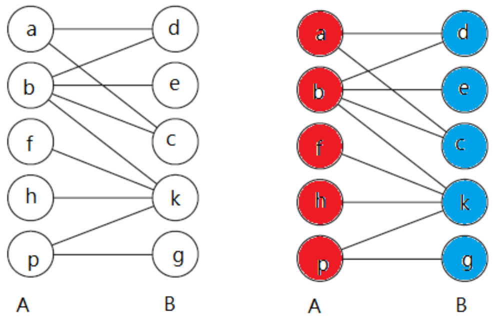
۱.۴ شمارش و محاسبه مولفه‌های همبند گراف

از الگوریتمهای BFS و DFS می‌توان برای محاسبه مولفه‌های همبند یک گراف استفاده کرد. کافی است از یک راس دلخواه u شروع کنیم و الگوریتم BFS را با شروع از u اجرا کنیم. با انجام این کار، مولفه همبندی که شامل u است پیدا می‌شود (رئوسی که علامت خورده اند.) حال دوباره یک راس دلخواه از مجموعه رئوس علامت نخورده را انتخاب می‌کنیم و با شروع از آن یک BFS می‌زنیم. با این کار یک مولفه همبند دیگر پیدا می‌شود. این کار را آنقدر ادامه می‌دهیم تا اینکه راس علامت نخورده‌ای باقی نماند. زمان اجرای این الگوریتم $O(n+m)$ است.

۲.۴ تشخیص گراف دوبخشی و رنگ آمیزی با دو رنگ

گراف $G=(V,E)$ دوبخشی است وقتی مجموعه رئوس آن V را بتوان به دو قسمت A و B افراز کرد بطوریکه همه یالها بین دو قسمت A و B باشند. در زیر یک گراف دو بخشی را مشاهده می‌کنید. یک تعریف دیگر برای گراف دوبخشی این است که بتوان رئوس گراف را با دو رنگ طوری رنگ آمیزی کرد که هیچ دو راس همرنگی همسایه نباشند. روشن است که این معادل با تعریف گراف دوبخشی است.

یک ویژگی مهم دیگر گرافهای دو بخشی این است که دور با طور فرد ندارند. در واقع می‌توان گفت گراف G دوبخشی است اگر و فقط اگر G دور با طول فرد نداشته باشد.



از الگوریتم BFS می‌توان برای تشخیص و رنگ آمیزی گراف دوبخشی با دو رنگ استفاده کرد. برای سادگی فرض کرده‌ایم که گراف ورودی همبند است (اگر همبند نباشد می‌توان اول مولفه‌های همبند را محاسبه کرد و الگوریتم زیر را برای هر مولفه همبند بطور مستقل اجرا کرد.)

۱. یک راس را به دلخواه از گراف انتخاب کن. فرض کن راس انتخابی u باشد.
۲. با شروع از u الگوریتم $BFS(u)$ را اجرا کن:
راسهایی که در لایه زوج قرار می‌گیرند را آبی رنگ کن و
راسهایی که در لایه فرد قرار می‌گیرند را قرمز رنگ کن.
۳. در انتها، بررسی کن اگر یالی وجود دارد که دو سر آن همرنگ باشند، گزارش بده که گراف ورودی دوبخشی نیست در غیر این صورت A را مجموعه رئوسی قرار بده که در لایه زوج هستند و B را مجموعه رئوسی قرار بده که در لایه فرد قرار گرفته‌اند.

تحلیل زمان اجرا و اثبات درستی الگوریتم به عنوان تمرین بر عهده دانشجو است.