به نام خدا

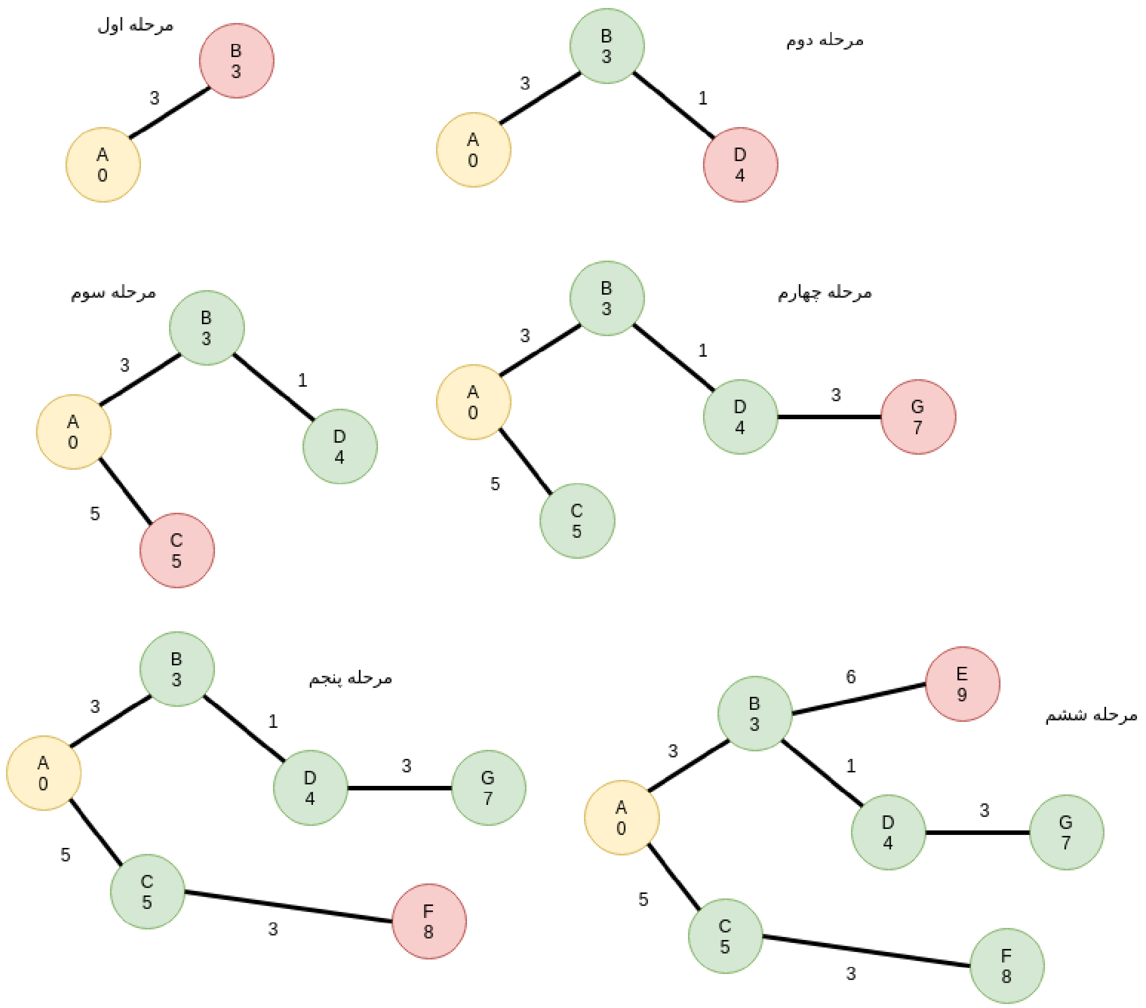
دانشگاه تهران، دانشکده مهندسی برق و کامپیوتر
تحلیل و طراحی الگوریتم‌ها

پاسخ تمرین کتبی چهارم

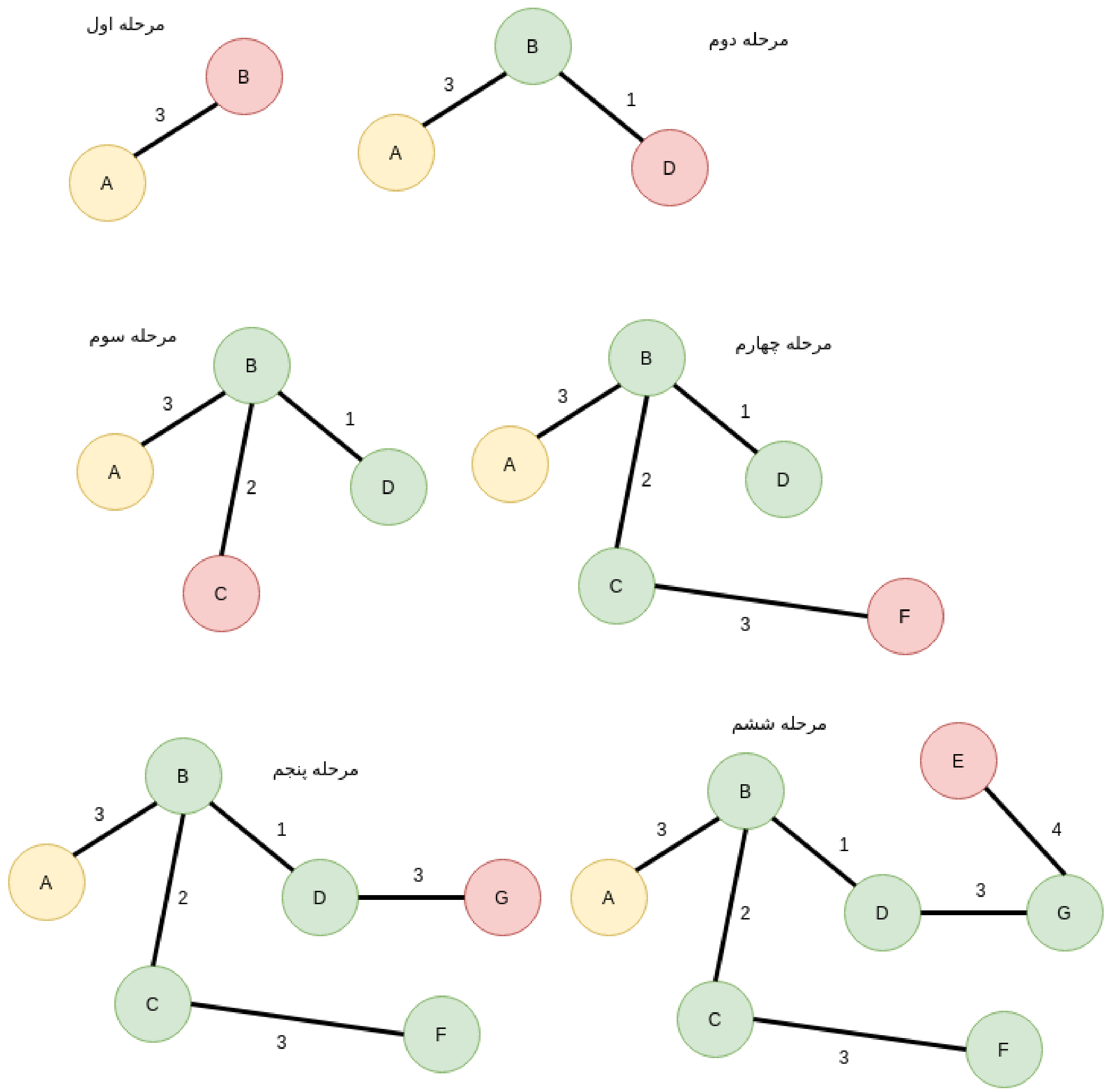
طراح: علیرضا سالمی، alirezasalemi7@gmail.com

---

۱. (آ) مراحل اجرای الگوریتم در تصویر زیر آمده است.



(ب) مراحل اجرای الگوریتم در تصویر زیر آمده است.



(ج) همانگونه که مشاهده کردید درخت حاصل از اجرای دو الگوریتم فوق با یکدیگر متفاوت است. این دو الگوریتم در اجرا شباهت زیادی به یک دیگر دارند اما درخت تولید شده از آن‌ها دارای خواص متفاوتی است. درخت کوتاه ترین فاصله، کوتاه ترین فاصله هر گره از گره مبدا را مشخص می‌کند. از طرف دیگر، درخت پوشای کمینه، کم وزن ترین درختی است که همه گره‌های گراف را به هم وصل می‌نماید و لزوما شامل کوتاه ترین فواصل نیست.

۲. (آ) برای یالی که وزن آن کم شده است دو حالت وجود دارد:

i. اگر یالی که وزن آن کاهش یافته درون درخت باشد، در این حالت وزن درخت کمتر شده است و چون خود درخت کمینه بوده است، هر درخت دیگری که داشته باشیم وزنی بیشتر یا مساوی با درخت فعلی دارد. پس درخت حاصل خود درخت پوشای کمینه است. یعنی اگر درخت داده شده را T بگیریم:

$$\forall T' \in Trees : weight(T') \geq weight(T) \Rightarrow \forall T' \in updatedTrees : weight(T') \geq weight(T) - e \quad (۱)$$

که e مقدار کاهش وزن یال درون درخت است.

ii. اگر یالی که وزن آن کاهش یافته درون درخت نباشد، یالی که وزن آن کاهش یافته را به درخت اضافه می‌کنیم. در گراف حاصل حتما یک دور وجود خواهد داشت. با استفاده از BFS پر وزن ترین یال درون دور ایجاد شده را پیدا کرده و حذف می‌کنیم. به این ترتیب درخت حاصل درخت پوشای کمینه جدید است.

اثبات: برحان خلف) فرض کنید درختی وجود دارد که وزن آن از درخت ایجاد شده توسط الگوریتم داده شده کمتر باشد و آن را T' می‌نامیم (درخت حاصل از الگوریتم خود را T می‌نامیم و یالی که وزن آن کاهش یافته را e می‌نامیم).

(i) $e \in T', e \in T$

در این حالت داریم:

$$w_2(e) + w_2(T-e) > w_2(e) + w_2(T'-e) \Rightarrow w_1(e) + w_2(T-e) > w_1(e) + w_2(T'-e) \quad (۲)$$

از آنجا که وزن سایر یال‌ها ثابت بوده است، پس درخت T' قبل از تغییر نیز وزن کمتری داشته که بر خلاف فرض کمینه بودن T است.

(ii) $e \notin T', e \notin T$

این حالت به این معنا است که درخت T' بدون توجه به تغییر وزن نیز وزن کمتری از T داشته که با کمینه بودن T قبل از تغییر متناقض است.

(iii) $e \notin T', e \in T$

در این حالت چون با افزودن e به T و حذف سنگین ترین یال از دور ایجاد شده وزن T کاهش یافته که و e به آن وارد شده، پس درخت T' بدون توجه به کاهش وزن e قبل از تغییر نیز وزن کمتری داشته است که با کمینه بودن T در تناقض است.

(iv) $e \in T', e \notin T$

$$w_1(T'-e) + w_2(e) < w_1(T) \,,\, w_1(T'-e) + w_2(e) \geq w_1(T) \quad (۳)$$

از تفریق دو رابطه فوق داریم که:

$$w_1(e) \leq w_2(e) \quad (۴)$$

که با فرض کاهش وزن در تضاد است.

بنابراین فرض خلف باطل است و T درخت کمینه برای گراف جدید است. هزینه اجرای الگوریتم نیاز به اندازه هزینه اجرای BFS است که برای درخت برابر با $O(n)$ است.

(ب) اگر هر اتاقک را به یک گره و هر تونل را به یک یال تصویر کنیم، هزینه بازگشایی هر تونل برابر با وزن یال معادل خواهد بود. از طرفی قصد داریم فقط همه اتاقک‌ها را به یک دیگر وصل کنیم پس با استفاده از اجرای الگوریتم کروسکال درخت کمینه را محاسبه می‌کنیم. این درخت کمترین هزینه وصل کردن همه گره‌ها به یک دیگر را به ما می‌دهد. هزینه اجرای این الگوریتم نیز به اندازه هزینه اجرای کروسکال است.

۳. (آ) کافی است درایه‌های موجود در قطر اصلی ماتریس حاصل از اجرای الگوریتم را بررسی کنیم. چون این الگوریتم کوتاه ترین مسیر بین هر گره و سایر گره‌ها را می‌دهد، عدد موجود در قطر اصلی فاصله هر گره از خودش است. اگر این فاصله عددی منفی باشد یعنی گراف داده شده دوری منفی داشته که از گره مربوط به آن درایه عبور می‌کرده است.

اثبات:

i. اگر یکی از درایه‌های قطر اصلی منفی باشد، با توجه به فلوید وارشال حتما یک مسیر از آن گره به خودش وجود داشته است که هزینه آن منفی بوده است. این مسیر یک دور است چون ابتدا و انتهای آن یک گره مشخص است پس گراف دور منفی دارد.

ii. فرض کنید گراف دارای دور منفی باشد و دور منفی با کمترین تعداد گره را در نظر بگیرید.

(i) اگر دور فقط شامل یک گره باشد پس باید از آن گره به خودش یالی منفی وجود داشته باشد، پس با همان وزن منفی مقدار اولیه می‌گیرد. پس در پایان اجرای الگوریتم چون مقادیر ماتریس در فرایند اجرای الگوریتم افزایش نمی‌یابند، حتما مقداری منفی و برابر با وزن یال نام برده دارد.

(ii) اگر تعداد گره‌ها ۲ یا بالاتر باشد، فرض کنید k بزرگ‌ترین شماره موجود برای یک گره در این دور باشد و i یک گره دیگر باشد. در این صورت d_{ik}^{k-1} و d_{ki}^{k-1} شامل مقدار صحیح کوتاه‌ترین فاصله از i به k و از k به i هستند زیرا هنوز در دور منفی قرار نگرفته اند و d^{k-1} شامل مقادیر برای مسیرهای دارای k-1 راس است و دور در نظر گرفته نیز کوتاه‌ترین

دور منفی است پس دور کوتاه‌تری که منفی باشد بین این راس‌ها نیست. حال چون دور بین $i \to k \to i$ منفی است، پس مقدار $d_{ii}^{k} = d_{ik}^{k-1} + d_{ki}^{k-1}$ منفی خواهد بود. از طرفی چون در فرایند اجرای الگوریتم هیچ‌گاه وزن‌ها افزوده نمی‌شوند، پس در پایان یکی از درایه‌های قطر اصلی ماتریس منفی خواهد بود.

(ب) چون فقط هزینه یکی از یال‌ها کاهش یافته است، برای محاسبه فاصله جدید کافی است که برای هر دو گره دلخواه فاصله آن‌ها تا دو سر یال uv محاسبه کنیم و با وزن کنونی آن‌ها مقایسه کنیم. هرکدام که کوچکتر بود به عنوان مسیر کمینه بین دو گره معرفی می‌کنیم. به این ترتیب داریم:

```
for i in 0...N-1
    for j in 0...N-1
        d[i][j] = min(d[i][j],d[i][u]+w(uv)+d[v][j])
        d[i][j] = min(d[i][j],d[j][u]+w(uv)+d[v][i])
```

هزینه این الگوریتم $O(n^2)$ است.

اثبات: برای دو راس دلخواه i و j:

i. اگر کوتاه ترین مسیر بین این دو راس دلخواه از یال uv عبور نکند چون فقط وزن همین یال عوض شده است پس مقدار کوتاه‌ترین مسیر با مقدار در گراف قبلی برابر است.
حال باید ثابت کنیم مسیر $i \to uv \to j$ یا عکس آن مقدار کمتری از کوتاه ترین مسیر ندارد. اگر مقدار عبارت $d[i][u] + w(uv) + d[v][j] < shortest\,path(i,j,G')$ برقرار باشد، در این صورت مسیر $i \to uv \to j$ کوتاه‌ترین خواهد بود که با فرض تناقض دارد که کوتاه‌تری مسیر شامل uv نیست.

ii. اگر کوتاه ترین مسیر بین این دو راس دلخواه از یال uv عبور کند، با فرض عدم وجود دور منفی خواهیم داشت که مسیرهای $i \to u$ و $v \to j$ مسیر ساده هستند و دوری ندارند. دو مسیر $i \to u$ و $v \to j$ کوتاه‌ترین مسیر بین جفت گره‌های نام برده هستند. این موضوع را فقط برای $i \to u$ ثابت می‌کنیم و برای دیگری نیز مشابه است. فرض کنید مسیر دیگری مثل P' باشد که از مسیر $i \to u$ که آن را P می‌نامیم و در گراف اولیه است کوتاه‌تر باشد. P' از uv عبور نمی‌کند چون اگر می‌کرد می‌شد با عدم عبور از آن اندازه مسیر را کوتاه تر کرد. بنابراین اندازه مسیر P و P' برابر است.
حال کافی است که نشان دهیم که $d[i][j] > d[i][u] + W(uv) + d[v][j]$ است. d[i][j] مقدار فاصله کمینه این دو راس در گراف قبلی است. از طرفی ثابت کردیم که مقدار فاصله $i \to u$ و $v \to j$ در دو گراف برابر و کمینه است، پس تنها تفاوت دو مسیر اندازه یال uv است که در گراف دوم کمتر است پس این مسیر کمینه است.

۴. هر بخاری را مانند یک گره در نظر بگیرید و هر مسیر یک طرفه را یک یال جهت دار فرض کنید. یک گراف جدید از روی گراف قبلی می‌سازیم که گره‌ها هما گره‌های قبلی هستند اما برای هر یال در گراف قبلی، هم خودش و هم برعکسش را قرار می‌دهیم. برای یال اصلی وزن صفر و برای یال برعکس وزن یک را در نظر بگیرید. حال با اجرای الگوریتم دایکسترا از گره شروع و هزینه رسیدن به گره نهایی را محاسبه می‌کنیم. به این ترتیب اگر از یالی که وجود داشته برویم هزینه صفر و اگر از یال برعکس اضافه شده برویم هزینه یک خواهیم داد. بنابراین هزینه کلی برابر کمینه تعداد عبور از یال‌های برعکس شده است. هزینه این عمل همانند هزینه دایکسترا است.

۵. روش اول:

یک گره شروع و یک گره پایان در نظر بگیرید. به ازای هر کلاس یک گره در نظر می‌گیریم. ابتدا گره‌ها را بر اساس زمان شروع و پایان مرتب می‌کنیم. از گره شروع به هر گره با زمان آغاز بیشتر یک یال جهت دار با وزن معادل با منفی مقدار جایزه دریافتی از گذراندن آن کلاس وصل می‌کنیم. از هر گره به گره پایانی نیز یک یال به وزن صفر وصل می‌کنیم. برای هر گره i به هر گره مانند j که زمان شروع آن از زمان پایان گره i بیشتر است یالی با وزن $-W_j$ وصل می‌کنیم. گراف حاصل یک گراف جهت‌دار بدون دور است زیرا برای هر کلاس فقط به کلاس‌هایی که بعد از آن باشد یال داریم پس به طور قطع در اجرای الگوریتم بلمنفرد با دور منفی مواجه نمی‌شویم. حال با استفاده از الگوریتم نام برده کوتاه ترین فاصله از گره شروع به گره پایان را می‌یابیم. منفی مقدار خروجی الگوریم به عنوان کوتاه ترین فاصله برابر با بیشترین مقدار ممکن است که بتوان از کلاس‌ها امتیاز گرفت. همچنین گره‌هایی که در این مسیر قرار دارند نیز کلاس‌هایی هستند که باید در آن‌ها شرکت کند.

دقت کنید با کمک منفی کردن وزن یال‌ها و عدم وجود دور منفی در گراف توانستیم طولانی ترین مسیر در گراف با وزن یال معادل با مقدار جایزه هر کلاس را محاسبه کنیم. هزینه اجرای این روش برابر $O(ne)$ است که معادل اجرای بلمنفرد است.

روش دوم:

ابتدا تمام زمان‌های شروع و پایان را در کنار یک دیگر مرتب می‌کنیم. برای هر کلاس یک گره شروع و یک گره پایان در نظر می‌گیریم. برای هر کلاس، از گره شروع به گره پایان آن کلاس یک یال با وزن مقدار جایزه آن کلاس قرار می‌دهیم. همچنین از هر گره شروع یالی

با وزن صفر به اولین گره شروع بعد وصل می‌کنیم. از هر گره پایانی نیز یالی به وزن صفر به اولین گره شروع بعد وصل می‌کنیم. گراف حاصل یک DAG است زیرا زمان‌ها همه رو به جلو هستند. خواسته سوال آن است که شما مسیری از گره شروع به گره پایانی بیابید که دارای بیشترین مقدار باشد به عبارتی بیشینه طول مسیر را داشته باشد. در DAG با کمک روش زیر می‌توانید بلندترین مسیر درون گراف را بیابید.

```
int dist[n] = {-inf, ..., -inf}
dist[s] = 0 #s is start vertex
sorted = topological_sort(graph)
for each u in sorted
    for each adjacent v of u
        if dist[v] < dist[u]+weight[u][v]
            dist[v] = dist[u]+weight[u][v]
# answer is dist[target]
```

این روش فاصله گره شروع از خودش را صفر گرفته و با توجه به یال‌های موجود، هزینه گره‌های بعدی را به گونه‌ای نگهداری می‌کند که در پایان پر هزینه ترین مسیر از راس شروع تا هر راس در آرایه بماند. هزینه اجرای این روش برابر $O(nlogn)$ است زیرا با توجه به نوع چینش یال‌ها تعداد یال‌ها از مرتبه n است و هزینه کل برابر هزینه مرتب سازی است.

۶. روش اول:

دو لیست با طول n داده شده است که شماره موادی که خاصیت یکسان دارند را در اندیس‌های یکسان قرار داده است و همچنین خاصیت تعدی نیز وجود دارد یعنی اگر ماده اول با ماده دوم برابر بود و ماده دوم با ماده سوم برابر باشد، ماده اول نیز با ماده سوم برابر است و برعکس. به همین دلیل باید مجموعه‌هایی از مواد با خواص یکسان به دست آوریم و در پایان برای هر یک از اعضای لیست سوم، معادہ مادل با هزینه کمتر را استفاده کنیم تا هزینه لیست سوم طبق خواسته سوال کمینه شود. برای این کار از روش union-find استفاده می‌کنیم. ابتدا برای هر یک از اعضای درون دو لیست اول که در اندیس یکسان قرار دارند عملیات union را انجام می‌دهیم. دقت کنید در اینجا آن ماده‌ای به عنوان ریشه قرار می‌گیرد که هزینه همتری داشته باشد. به این ترتیب هماره در ریشه هر مجموعه کم هزینه ترین ماده قرار دارد. پس از آن برای هر یک از اعضای لیست سوم عملیات find را انجام می‌دهیم تا ریشه مجموعه‌ای که ماده به آن تعلق دارد را بیابیم که کم هزینه ترین ماده با خواص مشابه است. لیست حاصل کم هزینه‌ترین لیست است. هزینه اجرای این الگوریتم $O(nlogm)$ است زیرا در ابتدا n مرتبه عملیات union انجام شده که هزینه آن logm است. و پس از آن نیز n مرتبه find صورت گرفته که هزینه آن logm است.

روش دوم:

برای هر ماده یک گره در نظر بگیرید. دو ماده که در اندیس یکسان قرار گرفته باشند را با یک یال به هم وصل می‌کنیم. در پایان تمام موادی که خواص یکسان داشته باشند در یک گراف قرار گرفته‌اند و گراف اصلی ممکن است شامل چند مولفه هم بندی باشد. حال برای هر مولفه همبندی با اجرای DFS کم هزینه ترین ماده را پیدا کرده و برای تمام ماده‌های موجود در آن مولفه در یک آرایه شماره آن ماده را می‌نویسیم تا با هزینه O(1) به آن دسترسی داشته باشیم. سپس برای هر یک از مواد موجود در لیست سوم، عدد نوشته شده در آرایه که برای آن ماده، شماره ماده‌ای است که کم هزینه‌ترین ماده با خواص مشابه است. هزینه اجرای این روش O(m+n) که برابر با هزینه انجام عمل DFS روی گراف اصلی شامل همه مواد و روابط بین آن‌ها است.

۷. برای هر واحد پول یک راس در نظر می‌گیریم. برای هر واحد پول i به واحدهای دیگر مثل j یک یال جهت دار از i به j قرار می‌دهیم. چرخه‌ای سود آور است که ضرب یال‌های آن چرخه بزرگتر از 1 باشد یعنی:

$1 < w(a,b) \times w(b,c) \times ... \times w(z,a)$

حال برای تبدیل ضرب به جمع از یال‌ها لگاریم می‌گیریم پس به این ترتیب داریم:

$0 < log(w(a,b)) + log(w(b,c)) + ... + log(w(z,a))$

حال برای بررسی وجود چنین دوری، می‌توان آن را به یافتن دور منفی تصویر کرد به این صورت که وزن یال‌ها را قرینه کنیم:

$0 < (-log(w(a,b))) + (-log(w(b,c))) + ... + (-log(w(z,a)))$

$0 > log(w(a,b)) + log(w(b,c)) + ... + log(w(z,a))$

حال با استفاده از الگوریتم بلمنفرد وجود دور منفی را در گراف بررسی می‌کنیم. اگر دور منفی وجود داشته باشد، چرخه سودآور وجود دارد. برای بررسی وجود دور منفی یک بار بلمنفرد را اجرا می‌کنیم سپس یک بار دیگر نیز الگوریتم را اجرا می‌کنیم. اگر مقدار راسی تغییر کرد، پس دور منفی وجود دارد. هزینه اجرای این الگوریم $O(ne)$ است.