درس ۱۵:

الگوریتم فراابتکاری جستجوی ممنوع

تهیه شده توسط گروه بهینه‌یاب

www.behinehyab.com

مقدمه

محاسبه راه حل بهینه *Optimal solution* برای اکثر مسایل بهینه سازی کـه در خیلـی از زمینـه‌هـای کاربردی و عملی مشاهده می‌گردند، کار دشوار و سختی است. در عمل، معمولا به راه حل‌های خـوب کـه از الگوریتم‌های هیوریستیک *Heuristic* یا متاهیوریستیک (همان فراابتکاری)*Metaheuristic* بدست مـی‌آیـد، اکتفا می‌گردد. متاهیوریستیک‌ها مجموعه ای از فنون بهینه سـازی تقریبـی *Approximate optimization techniques* را که عمدتا در طول دو دهـه گذشـته شـهرت پیـدا کـرده انـد، در بـر مـی‌گیرنـد. روش‌هـای فراابتکاری راه حل‌های قابل قبول در زمان معقول را برای مسایل پیچیده و سخت، در زمینه‌های مهندسی و علوم پایه ارایه می نمایند. برخلاف الگـوریتم‌هـای بهینـه سـازی دقیـق *Exact optimization algorithms*، فراابتکاری ها بهینه بودن جواب‌های بدست آمده را ضمانت نمی نمایند.

کلمه متاهیوریستیک از کلمه یونانی *"Heuriskein"* به معنای هنر کشف قواعد جدید برای حل مسـایل گرفته شده است. پیشوند "متا" نیز از یک کلمه یونانی به معنای "متدولوژی" سطح بالا گرفته شـده اسـت. واژه " متاهیوریستیک " اولین بار توسط گلوور در سال ۱۹۸۶ ارایه گردید. روش جستجوی متاهیوریستیک را می توان به صورت متدولوژی‌های عمومی سطح بالایی که می توانند به عنوان یـک اسـتراتژی راهنمـا در طراحی هیورستیک‌های اختصاصی برای حل مسایل بهینه سازی تخصصی به کار روند، تعریف کرد.

برخلاف روش‌های دقیق، متاهیوریستیک‌ها (فراابتکاری‌ها) برای مسایل با اندازه‌های بزرگ کاربرد دشوار دارد و راه حل‌هایی راضی کننده ای در زمان معقولی ارایه می نمایند. در این الگوریتم‌ها، هیچ گونه ضمانتی برای یافتن جواب بهینه سراسری یا حدودی از آن وجود ندارد. متاهیوریستیک‌ها (فراابتکـاری‌هـا) در طـول بیست سال گذشته شهرت زیادی پیدا کرده اند. کاربرد و استفاده از آن‌ها در خیلی از مسایل، کـارایی و اثـر بخشی آن‌ها را برای حل مسایل پیچیده و بزرگ نشان می دهد. فراابتکاری‌ها در خیلی از زمینه‌هـا از قبیـل موارد زیر کاربرد دارند:

- ✓ طراحی مهندسی، بهینه سـازی توپولـوژی، بهینـه سـازی مسـایل الکترونیـک، آئرودینامیـک، دینامک سیالات، مخابرات، رباتیک

- ✓ یادگیری ماشین و کاوش داده‌ها
- ✓ مدل سازی سیستم‌ها، شبیه سازی و تحقیق در شیمی، فیزیک، بیولوژی، کنترل، سیگنال، و پردازش تصویر
- ✓ مسایل مسیردهی و برنامه ریزی، برنامه ریزی ربات، مسایل تولید و زمان بندی، حمل و نقل و لجستیک، مدیریت زنجیره تامین

روش‌های مختلف الگوریتم‌های فرابتکاری یا متاهیوریستیک تا به حال پیشنهاد شده است که به صورت زیر است:

- ✓ بهینه سازی کلونی مورچگان *Ant colony optimization*
- ✓ بهینه سازی کلونی زنبوران *Bee colony*
- ✓ الگوریتم‌های ترتیبی *cultural algorithms*
- ✓ الگوریتم‌های با هم تکاملی *Co-evolutionary algorithms*
- ✓ الگوریتم ژنتیک *Genetic algorithm*
- ✓ جستجوی محلی تکراری *Iterated local search*
- ✓ بهینه سازی توده ذرات *particle swarm intelligent*
- ✓ انجماد تدریجی *Simulated Annealing*
- ✓ جستجوی ممنوع *Taboo search*
- ✓ جستجوی همسایگی متغیر *Variable neighboar search*

در طراحی یک فرابتکاری، دو معیار متناقض شامل کاوش *(Exploration)* و در فضای جستجو (گوناگونی و تنوع) و تبعیت (*Exploitation*) از بهترین راه حل‌های پیدا شده، باید در نظر گرفته شوند. در کاوش در ناحیه‌های جستجو نشده بررسی صورت می‌گیرد. در تبعیت، در ناحیه‌های امید بخش که تا به حال در آن ناحیه یک جواب خوب پیدا شده است بررسی بیشتر صورت می‌گیرد. در صورتیکه به کاوش اهمیت بیشتری داده شود، الگوریتم رفتار تصادفی بیشتری خواهد داشت و به سمت رفتار تصادفی میل می

کند و در صورتی که به رفتار تبعیت توجه بیشتری شود، الگوریتم از رفتار تصادفی فاصله می‌گیرد و جستجو تنها در محدوده راه حل‌های خوب به جستجو می پردازد.

معیارهای طبقه بندی زیادی ممکن است برای طبقه بندی فراابتکاری‌ها استفاده شود که در زیر به بعضی از آن‌ها اشاره می‌شود:

الهام گرفته از طبیعت در مقابل عدم الهام از طبیعت

خیلی از الگوریتم‌های فراابتکاری از فرایندهای طبیعی الهام گرفته شده اند: از قبیل الگوریتم‌های اجتماع مورچگان و زنبور عسل که از هوش توده ای از گونه‌های مختلف مورچگان و زنبوران استفاده می کنند.

نحوه استفاده از حافظه

بعضی از الگوریتم‌های فراابتکاری از قبیل انجماد تدریجی بدون حافظه هستند و حرکت‌های قبلی را در جایی ذخیره نمی کنند. در مقابل الگوریتم انجماد تدریجی از یک حافظه که بعضی از اطلاعات را در طول جستجو بدست می‌آورد، ذخیره می کند.

قطعی در مقابل احتمالی

یک الگوریتم قطعی، یک مسئله بهینه سازی را از طریق تصمیم گیری قطعی حل می نماید(برای مثال الگوریتم جستجوی محلی و جستجوی ممنوع). در الگوریتم‌های فراابتکاری احتمالی، بعضی از قواعد احتمالی در فرایند جستجو مورد استفاده قرار می‌گیرد که می توان به الگوریتم انجماد تدریجی و الگوریتم ژنتیک اشاره کرد. در الگوریتم‌های قطعی، با داشتن یک راه حل اولیه و اجراهای متفاوت، تنها یک جواب نهایی بدست می‌آید و در حالی که در الگوریتم‌های تصادفی، با داشتن یک راه حل اولیه و اجراهای متفاوت، ممکن است که جواب‌های نهایی متفاوتی بدست آید.

تکراری در مقابل حریصانه

در الگوریتم‌های تکراری، الگوریتم با یک راه حل کامل شروع شده و در هر تکرار راه حل یا راه حل‌ها تغییر پیدا می کنند. در الگوریتم‌های حریصانه یک راه حل کامل در اختیار نبوده بلکه با یک راه حل ساخته نشده شروع شده و در هر مرحله، یک متغیر تصمیم از مسئله یک قسمت از راه حل را می سازد. اغلب الگوریتم‌های فراابتکاری، الگوریتم‌های تکراری هستند.

در ادامه الگوریتم جستجو ممنوع که الگوریتمی است که از طبیعت الهام گرفته است و در رده الگوریتم‌های قطعی، و تکراری با حافظه قرار می‌گیرد صحبت می‌شود.

الگوریتم جستجوی ممنوع

الگوریتم جستجو ممنوع یا ***Taboo search*** اولین بار توسط فرد گلوور در مقاله ای که در سال ۱۹۸۶ منتشر گردید، ارایه شد. همچنین بعدا دو مقاله از ایشان با نام ساده "جستجو ممنوع" در سال‌های ۱۹۸۹ و ۱۹۹۰ منتشر گردید که در آن، خیلی از کاربردهای جستجو ممنوع را معرفی کرد.

اساس نامگذاری این روش، استفاده آن از لیستی به نام لیست ممنوع *Tabu list*می باشد. این لیست برای جلوگیری از افتادن الگوریتم در نقطه بهینه‌های محلی طراحی شده است. به طور خلاصه، این روش به صورت مقابل بیان می‌شود. این الگوریتم از یک نقطه یا راه حل شروع کرده و در اطراف آن نقطه، به جستجوی همسایگی می پردازد. در بین همسایه‌ها، بهترین را انتخاب و به آن نقطه حرکت می نماید. این جستجوی را تا زمانی ادامه می دهد که یک معیار توقف برآورده گردد. در پایان جستجو، نقطه بهینه گزارش می‌شود. این الگوریتم فراابتکاری برای جلوگیری از افتادن در بهینه محلی، از یک حافظه کوتاه مدت استفاده می نماید. وظیفه این حافظه کوتاه مدت نگهداری از آخرین حرکت‌ها است که تعداد آن‌ها محدود شده است. این حرکت‌ها در یک لیست (لیست ممنوع) نگهداری می‌شوند. در هر حرکت، در صورتی که حرکت در لیست ممنوع قرار گرفته باشد، از آن انتقال جلوگیری می‌گردد مگر آنکه حرکت ممنوع دارای شرایط خاصی باشد.

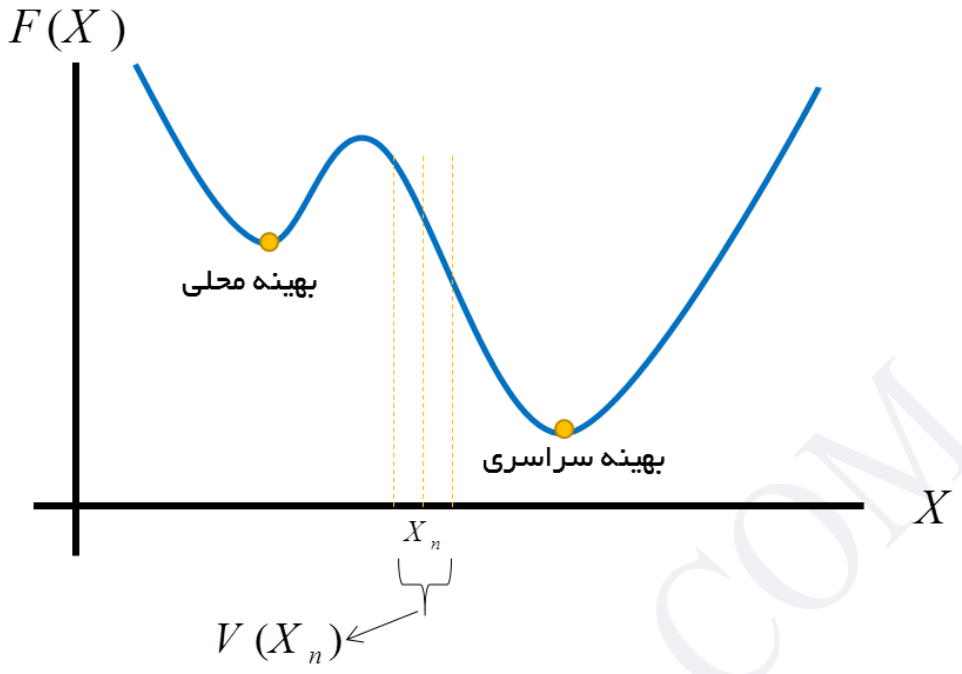
اصول کلی الگوریتم جستجوی ممنوع

الگوریتم فراابتکاری جستجوی ممنوع را می توان به عنوان یک استراتژی جستجوی محلی در نظر گرفت. این جستجو شامل حرکت از یک محل (همان جواب یا نقطه) به راه حل دیگری در همسایگی او مطابق با بعضی از قواعد تعریف شده می باشد. برای مثال، مسئله حداقل سازی یک تابع همچون *F(x)* روی یک مجموع محدود از نقاط *X* را به عنوان یک حالت عمومی از یک مساله بهینه سازی ترکیبی در نظر بگیرید. روش جستجوی ممنوع از یک راه حل اختیاری $x_1 \in X$ شروع شده و در هر مرحله، به جستجوی همسایگی می پردازد. در این جستجو همسایگی، به هر $x \in X$ یک زیر مجموعه $V(x) \subset X$ اختصاص داده می‌شود که همسایه *x* نامیده می‌شود. در مرحله *n*، یک راه حل جدید x_{n+1} در همسایگی $V(x_n)$ از راه حل جاری x_n را انتخاب می نماید. معمولا این راه حل بهترین راه حل در بین همسایه‌ها بوده و دارای شرط زیر می باشد.

$$F(x_{n+1}) \le F(x) \quad \forall x \in V(x_n)$$

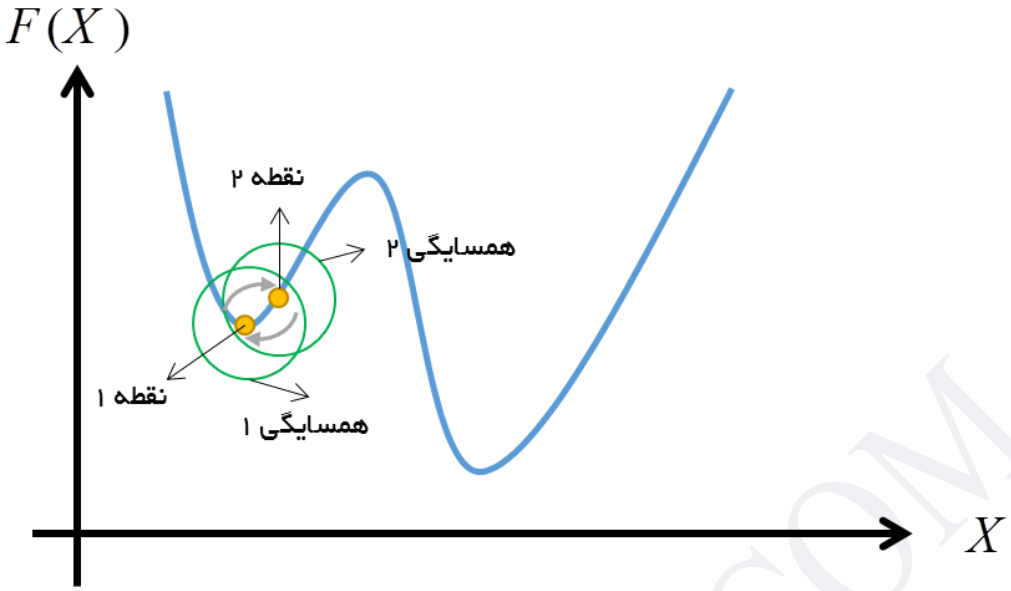
الگوریتم، بعد از انتخاب راه حل x_{n+1} از نقطه x_n به x_{n+1} حرکت می نماید و آنگاه راه حل x_{n+1} به عنوان راه حل جاری بعدی شناخته می‌شود. این جستجو تا زمانی ادامه می یابد که یک شرط توقف تعریف شده برآورده شود.

ممکن است به هنگام حرکت از یک نقطه به نقطه دیگر مشکلی پیش آید که به نام افتادن در بهینه محلی معروف است. شکل زیر را در نظر بگیرید.



همان طور که در این شکل دیده می‌شود، فضای جواب مساله دارای دو نقطه مینیم یا بهینه می باشد. نقطه بهینه، نقطه ای تعریف می‌شود که آن نقطه بهترین نقطه در تمامی نقاط اطراف خود باشد. به عبارت دیگر، نقطه ای مینیمم است که تمامی راه حل‌های اطراف آن بدتر از خود آن باشند. در این شکل، یک نقطه بهینه محلی و یک نقطه بهینه سراسری نمایش داده شده است.

مشکل افتادن در بهینه محلی، زمانی اتفاق می افتد که الگوریتم به نقطه بهینه محلی می‌رسد. در این نقطه، الگوریتم به جستجوی همسایگی پرداخته و یک همسایه را انتخاب می نماید. لازم به توضیح است که الگوریتم در جستجوی همسایگی، حتما باید یک نقطه را از بین همسایگان و غیر از خود انتخاب کند و به آن نقطه حرکت نماید، حتی اگر کیفیت آن نقطه پایین تر باشد. زمانی که الگوریتم در بهینه محلی قرار دارد، نقاط همسایه او دارای مقدار تابع یا کیفیت بدتر از خود نقطه بهینه محلی می باشند. از روی اجبار، الگوریتم به یکی از نقطه بین آن‌ها انتقال می یابد. بعد از حرکت به آن نقطه، دوباره به جستجوی همسایگی می پردازد و با توجه به اینکه نقطه بهینه محلی قبلی در همسایگی نقطه جدید قرار دارد و از نقطه فعلی بهتر است مجددا به بهینه محلی بر می‌گردد. در این حالت الگوریتم در یک دور یا سیکل افتاده و از آن خارج نمی‌گردد. این وضعیت به افتادن در بهینه محلی معروف می باشد که در شکل زیر نشان داده می‌شود.



در این شکل بهینه محلی با نقطه ۱ نمایش داده شده و همسایه انتخاب شده برای حرکت، نقطه ۲ می باشد. بعد از حرکت به نقطه ۲، بهترین نقطه در همسایگی او، نقطه ۱ می باشد و به آن بر می‌گردد. این روند همواره تکرار شده و الگوریتم از این دور یا سیکل خارج نمی‌گردد.

الگوریتم جستجوی ممنوع برای جلوگیری از این مشکل از ابزار لیست ممنوع استفاده می کند. در هر مرحله، در هنگام حرکت از یک نقطه به نقطه دیگر، مشخصاتی از حرکت (برای مثال فاصله و جهت) را به حافظه سپرده و در نقطه جدید از انجام حرکتی که منجر به برگشت به عقب می‌شود خودداری می کند. به منظور کارایی بیشتر، به جای یک حرکت قبلی، مجموعه ای از حرکت‌های قبلی را به حافظه سپرده و آن‌ها در لیستی به نام لیست ممنوع ذخیره می نماید. این لیست پویا بوده و در طول الگوریتم به هنگام می‌گردد. به عبارت دیگر، در هر حرکت، مشخصه حرکت جدید وارد لیست شده و مشخصه حرکت‌های قدیمی از لیست حذف می‌گردد. یک قاعده ساده این است که لیست طول محدودی داشته و زمان ورود یک حرکت به لیست، قدیمی ترین حرکت از آن خارج گردد.

الگوریتم‌ها در هر مرحله، بهترین نقطه را در بین همسایگان جستجو نموده و در صورتی به آن نقطه حرکت می نماید که آن حرکت ممنوع نباشد. به عبارت دیگر، حرکت مدنظر را با لیست ممنوع مقایسه نموده و در صورتی که حرکت ممنوع باشد، از آن حرکت صرفه نظر نموده و بهترین حرکت بعدی را در بین همسایگان انتخاب می نماید. نقطه بعدی انتخاب شده نیز با لیست ممنوع مقایسه می‌گردد.

لیست ممنوع با وجود مقید بودن، محدودیت‌هایی را برای الگوریتم به وجود می‌آورد. در طول جستجو الگوریتم، ممکن است یک حرکت ممنوع باشد، ولی انجام حرکت با وجود ممنوع بودن تاثیر بالایی در بهبود جهت حرکت و کیفیت جواب های الگوریتم دارد. برای این منظور، الگوریتم از یک معیار به نام معیار آرمانی برای رهایی از این محدودیت در مواقع لزوم استفاده می نماید. یک معیار آرمانی ساده این است که ممنوعیت حرکتی که موجب رسیدن به نقطه ای گردد که از تمامی نقاطی که تاکنون بدست آمده است بهتر باشد، در نظر گرفته نمی‌شود.

قاعده توقف پایان الگورتیم جستجو را تعیین می کند. یک قاعده ساده می توان به صورت محدودیتی برای کل تعداد حرکت‌ها باشد. به عبارت دیگر، وقتی تعداد حرکت‌ها به عدد خاصی رسید الگوریتم متوقف گردد.

مراحل الگوریتم جستجوی ممنوع

همان طور که قبلا شرح داده شد، این الگوریتم از یک نقطه یا راه حل شروع کرده و در اطراف آن نقطه، به جستجوی همسایگی می پردازد. در بین همسایه‌ها، بهترین را انتخاب و به آن نقطه حرکت می نماید. این جستجو را تا زمانی ادامه می دهد که یک معیار توقف براورده گردد. در پایان جستجو، نقطه بهینه گزارش می‌گردد. الگوریتم برای جلوگیری از افتادن در کمینه محلی از یک حافظه کوتاه مدت استفاده می نماید. وظیفه این حافظه کوتاه مدت نگهداری از اخرین حرکت‌ها می باشد. این حرکت‌ها در یک لیست ممنوع نگهداری می‌شوند. در هر حرکت، در صورتی که حرکت در لیست ممنوع قرار گرفته باشد، از آن حرکت جلوگیری می‌گردد مگر آن که حرکت ممنوع دارای شرط آرمانی باشد. در ادامه فرایند این الگوریتم به صورت جزیی بیان می‌شود.

قدم اول: تنظیمات الگوریتم

ساختار ساده الگوریتم جستجو ممنوع، در هنگام استفاده برای حل یک مسئله بهینه سازی، نیاز به تنظیمات خاصی دارد. در صورتی که این تنظیمات به صورت خوبی انجام شود، الگوریتم دارای کیفیت بالایی خواهد بود و در غیر این صورت کیفیت الگوریتم در یافتن جواب بهینه کاهش می یابد. یکی از این

تنظیمات، نحوه رد کردن جواب یا راه حل‌ها می باشد. کد باید به نوحی طراحی گردد که قابل برنامه نویسی با کامپیوتر بوده و ضمنا در صورت استفاده در الگوریتم، کارایی خوبی هم داشته باشد. همچنین، جواب‌های امکان پذیر را معرفی نماید. یکی از دیگر تنظیماتی که باید انجام گردد، تعریف ساختار همسایگی می باشد. ساختار همسایگی نحوه بدست آوردن همسایه‌ها و تعداد آن‌ها را برای هر راه حل مشخص می نماید.

قدم دوم: تشکیل یک راه حل اولیه

الگوریتم جستجوی ممنوع با یک راه حل اولیه شروع می‌شود. در خیلی از مواقع، راه حل اولیه به صورت تصادفی انتخاب می‌شود. در صورتی که ساختار جواب ساده باشد این کار به سادگی انجام می‌شود اما در خیلی از مسئله‌ها به دلیل پیچیدگی فضای جواب و نحوه کدگذاری آن، ایجاد یک جواب اولیه امکان پذیر به صورت تصادفی ، کار ساده ای نیست و در برخی اوقات نیاز به الگوریتم‌های کناری برای یافتن جواب اولیه مناسب و امکان پذیر است.

قدم سوم: محاسبه مقدار تابع هدف

در این مرحله، مقدار تابع هدف یا تابع برازندگی برای راه حل اولیه در نظر گرفته شده، محاسبه می‌گردد. در شروع الگوریتم مقدار به دست آمده از جواب اولیه به عنوان مقدار بهینه نیز در نظر گرفته می‌شود. این مقدار در طول الگوریتم به هنگام می‌گردد. همچنین، نقطه اولیه به عنوان راه حل جاری نیز مدنظر قرار می‌گیرد.

قدم چهارم: جستجوی همسایگی

در هر مرحله الگوریتم، جستجوی همسایگی به جستجوی همسایه‌های جواب جاری می پردازد. در این مرحله، با توجه به ساختار معرفی شده برای همسایگی، کلیه همسایه‌های راه حل جاری مورد بررسی قرار می‌گیرند.

قدم پنجم: یافتن بهترین همسایه

در جستجوی همسایگی، کلیه همسایه بررسی می‌گردد. در بین همسایه‌ها، بهترین همسایه انتخاب می‌گردد.

قدم ششم: ارزیابی ممنوعیت حرکت

بعد از انتخاب بهترین همسایه، حرکت به آن راه حل یا لیست ممنوع مقایسه می‌گردد. در صورتی که این حرکت جز لیست ممنوع باشد، به **قدم هفتم** رفته و در صورتی که حرکت ممنوع نباشد، الگوریتم به **قدم هشتم** خواهد رفت

قدم هفتم: معیار آرمانی

در این قدم، حرکت ممنوع شده با معیار ارمانی مقایسه می‌گردد. در صورتی که معیار ارمانی براورده شده باشد، حرکت انجام شده و در صورتی معیار ارمانی براورده نشود، الگوریتم به **قدم پنجم** بر می‌گردد و بهترین راه حل بعدی را انتخاب می کند.

قدم هشتم: انجام حرکت

در این قدم، راه حل جاری جدید به عنوان راه حل جاری در نظر گرفته می‌شود. به عبارت دیگر، از راه حل جاری به راه حل جدید حرکت می‌شود و راه حل جدید به عنوان راه جاری شناخته می‌شود.

قدم نهم: بهنگام سازی بهترین جواب

در فرایند الگوریتم، همواره بهترین جواب ذخیره می‌گردد. در ابتدا نقطه اولیه، به عنوان بهترین جواب در نظر گرفته می‌شود و هر زمان که حرکتی انجام می‌گردد، جواب جدید با بهترین جوابی که تا آن لحظه به دست امده است، مقایسه می‌گردد. در صورتی که راه حل جدید راه حل بهتری باشد، راه حل جدید به عنوان بهترین راه حل ذخیره می‌شود و در غیر این صورت، بهترین راه حل تغییر نمی کند.

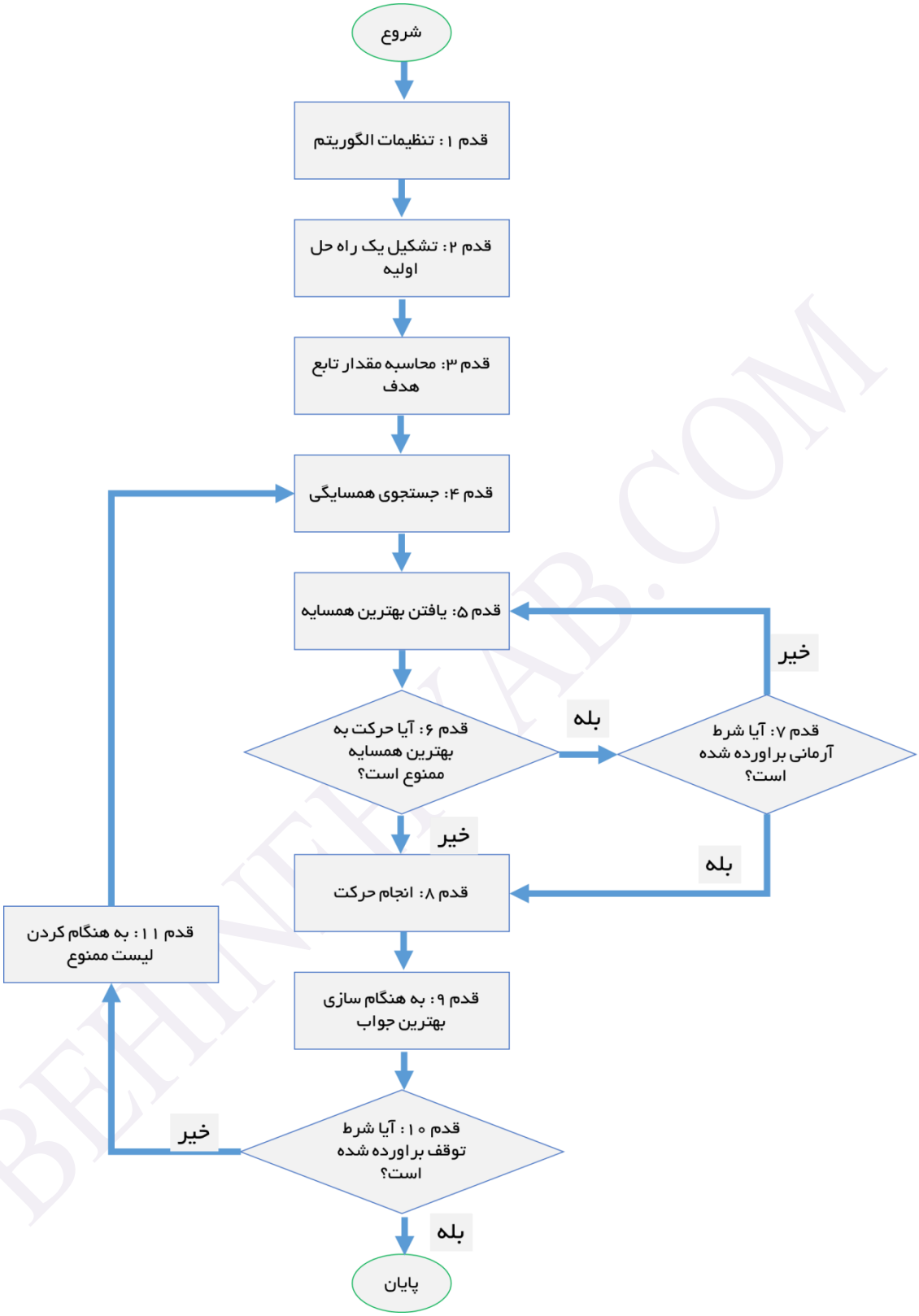
قدم دهم: شرط توقف:

بعد از هر حرکت، شرط توقف مورد ارزیابی قرار می‌گیرد. در صورتی که شرط توقف براورده شود، الگوریتم پایان یافته و نتایج آن گزارش می‌شود و در غیر این صورت، به **قدم یازده** منتقل خواهد شد.

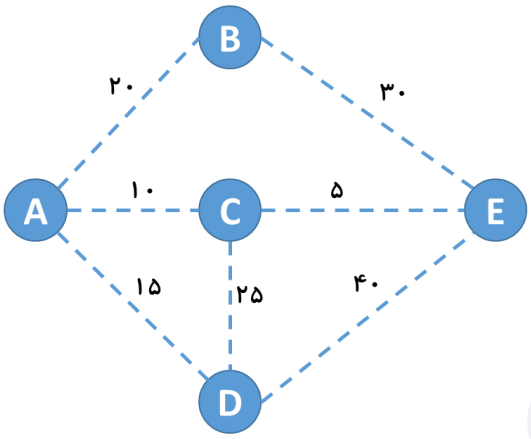
قدم یازدهم: بهنگام سازی لیست ممنوع

بعد از هر حرکت، مشخصه حرکت جدید وارد لیست ممنوع می‌گردد و سپس لیست ممنوع به هنگام می‌گردد. براساس این قاعده، در زمان انتقال یک حرکت به لیست ممنوع، ممکن است یک یا چند حرکت قدیمی از لیست ممنوع حذف گردد. بعد از بهنگام سازی لیست ممنوع، الگوریتم به **قدم چهارم** رفته و به جستجوی همسایگی در اطراف راه حل جدید می پردازد.

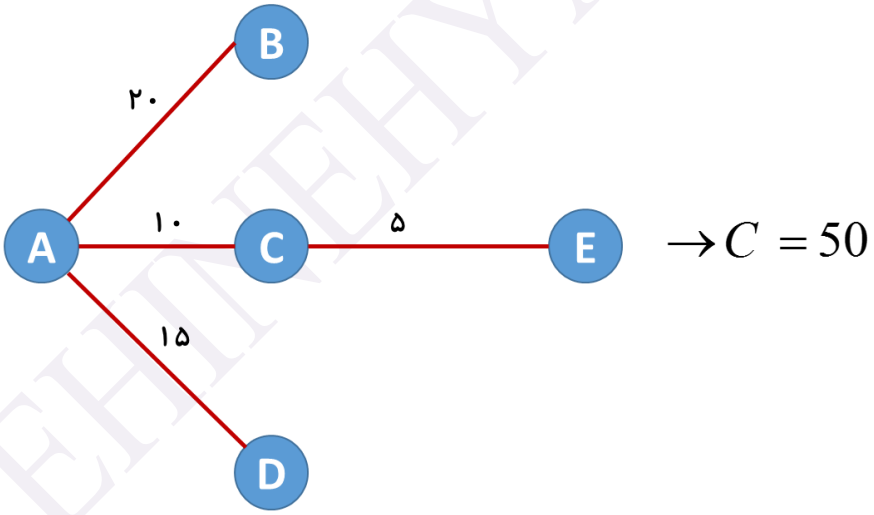
در شکل زیر مراحل این الگوریتم به صورت فلوچارت بیان می‌شود.



تمرین: درخت پوشای کمینه برای شبکه زیر را با استفاده از الگوریتم جستجوی ممنوع بیابید.



درخت پوشای کمینه در صورتی که محدودیتی در این مسئله وجود نداشته باشد به صورت زیر می شود.



در این مسئله دو محدودیت زیر در نظر گرفته می شود.

محدودیت ۱: یال *AD* زمانی می تواند در جواب بهینه حضور داشته باشد که یال *DE* هم وجود داشته باشد.

محدودیت ۲: یکی از سه یال *AD*، *CD* و *AB* می تواند در شبکه نهایی حضور داشته باشد.

شبکه ای که ازالگوریتم درخت پوشای کمینه بدون محدودیت بدست امـده اسـت دو محـدودیت فـوق الذکر را برآورده نمی کند.

روش های مختلفی برای در نظر گرفتن این دو محدودیت در مسئله وجود دارد. یکی از روش های ساده و رایج، استفاده از روش پنالتی است. در صورت نقض محدودیت های ۱ و ۲، هزینه پنـالتی بـه صـورت زیـر محاسبه می شود.

محدودیت ۱: ایجاد هزینه پنالتی ۱۰۰ اگر محدودیت ۱ نقض شود.

محدودیت ۲: ایجاد هزینه پنالتی ۱۰۰ در صورتی که از سه یال *AD*، *CD* و *AB* در جواب وجود داشته باشد و در صوریتکه هر سه کمان در جواب حضور داشته باشد، هزینه جریمه برابر ۲۰۰ می شود.

برای حل این مسئله با استفاده از روش جستجو ممنوع باید موارد زیر در نظر گرفته نشود.

۱- **فرایند جستجوی محلی:** در این حالت، به جوابی حرکت می کنیم که بهترین جواب کناری است به شرطی که محدودیت ها را نقض نکند.

۲-**حرکت به همسایگی:** در این قسمت، کمان انتخاب شده به شبکه اضافه می شود و کمانی که باعث دور یا *cycle* در شبکه می شود را حذف می کنیم.

۳-**لیست ممنوعه:** در این حالت کمان هایی که نباید حذف شوند قرار گیرد.

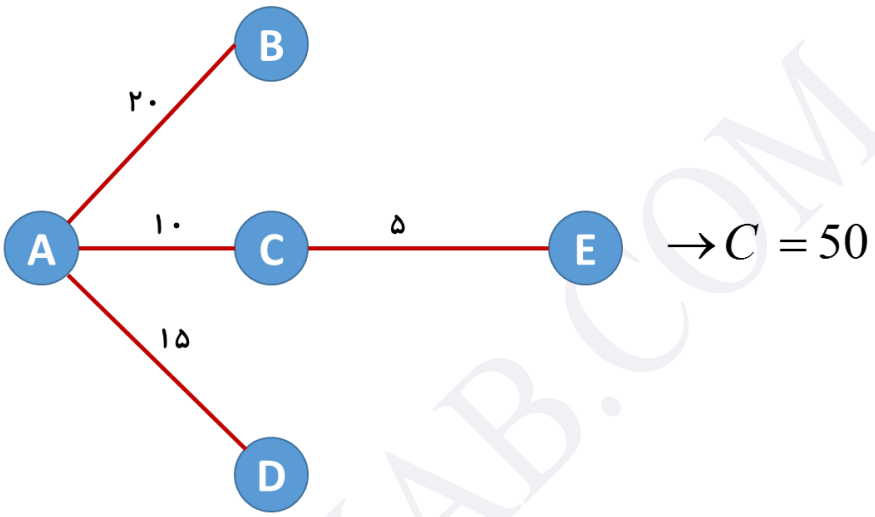
۴-**به هنگام سازی لیست تابو :** وقتی در یک مرحله یک یال به شبکه اضافه می شو ایـن لینـک وارد لیست تابو می شود.

۵-**حداکثر اندازه لیست:** اندازه این لیست برابر با ۲ (دو) در نظر گرفته می شود.

۶- **قاعده توقف:** توقف در الگوریتم زمانی رخ می دهد که پس از ۳ (سه) تکرار تغییری در بهترین تابع هدف رخ ندهد.

تکرار ۱:

در ابتدای حل مسئله نیاز است که با یک جواب اولیه کار را آغاز کنیم. جواب اولیه زیر انتخاب می شود که همان جواب روش ارایه شده از روش بهینه سازی درخت پوشای ارایه شده در درس ۷ است.



سه گزینه برای اضافه کردن لینک به شبکه بالا وجود دارد که عبارتند از *BE/CD/DE*. اگر لینک *BE* اضافه شود دور *BE-CE-AC-AB* ایجاد می شود و برای جلوگیری از این دور می توان یکی از سه لینک *CE*، *AC*، و*AB* را حذف کرد(تا حالا در لیست تابو لینکی اضافه نشده است).

اگر لینک *CE* حذف شود و لینک *BE* اضافه شود، هزینه کل ۲۵ واحد افزایش می یابد (۲۵=۵-۳۰)، و هزینه پنالتی ۲۰۰ واحد باقی می ماند و لذا کل هزینه از ۲۵۰ به ۲۷۵ افزایش می یابد.

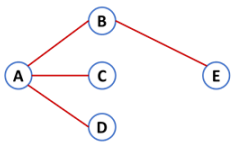
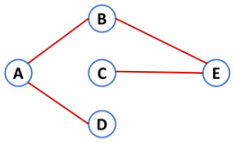
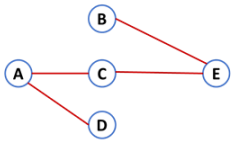
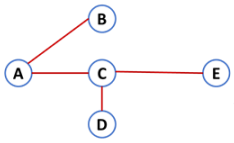
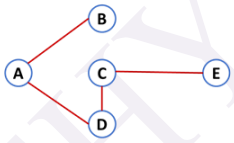
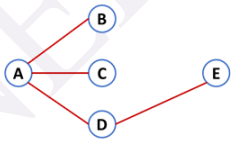
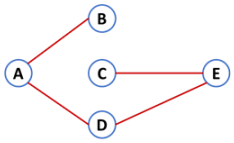
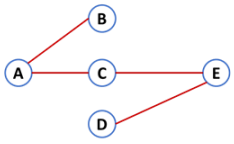
اگر لینک *AB* حذف شود هزینه لینک ها ۱۰=۲۰-۳۰ تغییر کرده و هزینه پنالتی از ۲۰۰ به ۱۰۰ کاهش می یابد و لذا هزینه کل برابر با ۱۶۰=۱۰۰+۱۰+۵۰ خواهد شد.

اگر لینک *AC* حذف شود، محدودیت های ۱ و ۲ نقض می شوند و کل هزینه برابر با ۲۷۰ می شود.

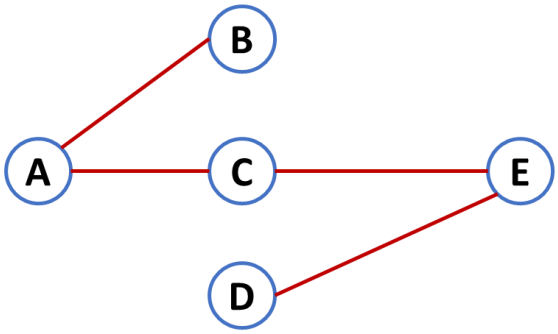
اگر لینک *CD* انتخاب شود که باید به شبکه اضافه شود، در این صورت *CD-AD-AC* ایجاد می شود و لینک های *AD* و *AC* جز گزینه های حذف خواهد بود. در صوریتکه لینک *AC* حذف شود، محدودیت ۱ و

محدودیت ۲ نقض خواهد شد و ۱۰۰ واحد هزینه به واسطه نقض محدودیت ۱ و ۲۰۰ واحد هزینه به واسطه نقض محدودیت ۲ وارد می شود. ولی در صورت حذف کردن *AD* محدودیت ۱ براورده می شود و فقط محدودیت ۲ باعث ۱۰۰ واحد هزینه پنالتی می شود.

اگر لینک *DE* به شبکه اضافه شود آنگاه دور *DE-CE-AC-AD* ایجاد می شود و می توان لینک های *AC*،*CE* و *AD* را حذف کرد. حذف هر یکی از این سه لینک باعث برقراری محدودیت ۱ خواهد شد ولی حذف لینک *AD*، محدودیت ۲ را هم برقرارکند و هزینه کل شبکه برابر با ۵۰+(۱۵-۴۰)=۷۵ می شود. جدول زیر خلاصه یافتن بهترین همسایگی را نشان می دهد. در ابتدای این تکرار لیست تابو تهی است.

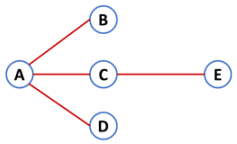
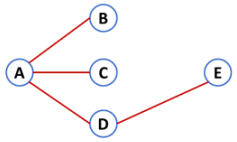
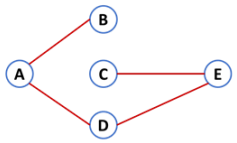
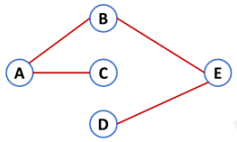
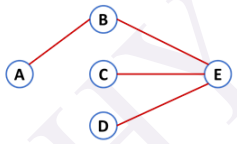
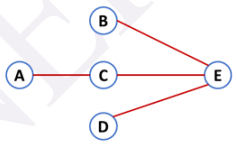
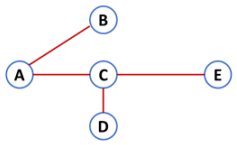
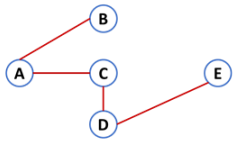
هزینه	شبکه	حذف کردن	اضافه کردن
محدودیت ۱ نقض می شود. محدودیت ۲ نقض می شود. ۲۰۰+۷۵=۲۷۵		CE	BE
محدودیت ۱ نقض می شود. محدودیت ۲ نقض می شود. ۲۰۰+۷۰=۲۷۰		AC	BE
محدودیت ۱ نقض می شود. محدودیت ۲ نقض نمی شود. ۱۰۰+۶۰=۱۶۰		AB	BE
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض می شود. ۱۰۰+۶۰=۱۶۰		AD	CD
محدودیت ۱ نقض می شود. محدودیت ۲ نقض می شود. ۳۰۰+۶۵=۳۶۵		AC	CD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض می شود. ۱۰۰+۸۵=۱۸۵		CE	DE
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض می شود. ۱۰۰+۸۰=۱۸۰		AC	DE
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۰+۷۵=۷۵		AD	DE

با توجه به جدول فوق، جواب فعلی به جواب همسایگی زیر حرکت می کند و لینک DE به لیست تابو اضافه می شود و لذا لیست تابو به صورت $\{DE\}$ می شود.



تکرار ۲

در این تکرار مشابه تکرار ۱ عمل می کنیم و در تکرار لینک BE به شبکه اضافه می شود و این کمان به لیست تابلو اضافه می شود و لینک AB از شبکه حذف می شود. در جدول زیر خلاصه یافتن بهترین همسایگی را نشان می دهد. در ابتدای این تکرار لیست تابو برابر $\{DE\}$ می شود.

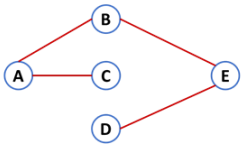
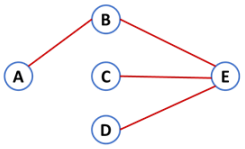
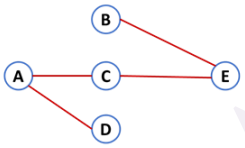
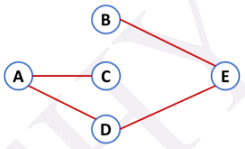
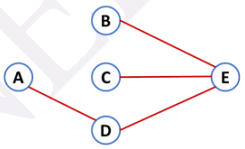
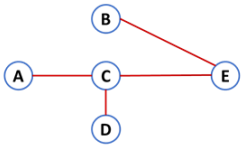
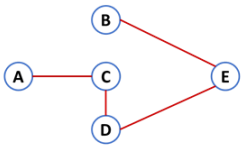
هزینه	شبکه	حذف کردن	اضافه کردن
حرکت غیرمجاز به علت اینکه لینک DE جز لیست تابو است.		DE*	AD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض می شود. ۱۰۰+۸۵=۱۸۵		CE	AD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض می شود. ۱۰۰+۸۰=۱۸۰		AC	AD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۰+۱۰۰=۱۰۰		CE	BE
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۰+۹۵=۹۵		AC	BE
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. کمینه ** ۰+۸۵=۸۵		AB	BE
حرکت غیرمجاز به علت اینکه لینک DE جز لیست تابو است.		DE*	CD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض می شود. ۱۰۰+۹۵=۱۹۵		CE	CD

*: زمانی لحاظ می شود که بتوان جواب بهتری از جواب فعلی بدست آورد.

**: این جواب از بهترین جواب فعلی ،۷۵، بدتر است و لذا به عنوان یک جواب محلی بهینه به حساب می آید. در انتهای این تکرار لیست تابو به صورت $\{DE,BE\}$ می شود.

تکرار ۳

در این تکرار لینک CD به شبکه اضافه می شود و این لینک به لیست تابو اضافه می شود و لینک از لیست تابو حذف می شود. در ابتدای این تکرار لیست تابو به صورت $\{DE,BE\}$ می شود.

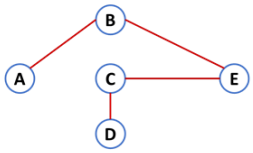
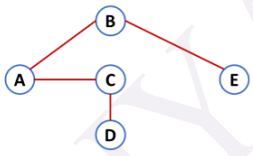
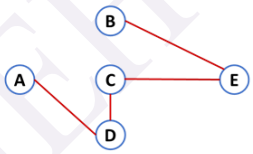
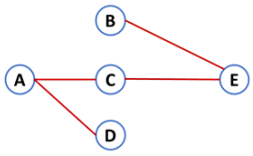
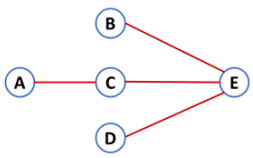
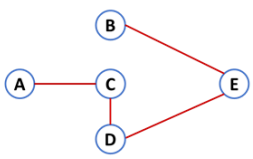
هزینه	شبکه	حذف کردن	اضافه کردن
حرکت ممنوعه	چون BE کمان در لیست تابو است این یک حرکت ممنونه است.	BE	AB
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۱۰۰+۰=۱۰۰		CE	AB
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۰+۹۵=۹۵		AC	AB
محدودیت ۱ نقض می شود. محدودیت ۲ نقض نمی شود. ۶۰+۱۰۰=۱۶۰		DE	AD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۰+۹۵=۹۵		CE	AD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۰+۹۰=۹۰		AC	AD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۰+۷۰=۷۰ کمینه **		DE	CD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۱۰۵+۰=۱۰۵		CE	CD

با وجود اینکه لینک *DE* عضو لیست تابواست، ولی چون منجر به جوابی شده است که از جـواب فعلـی بهتر است(یعنی ۷۰<۷۵) لذا این امکان وجود دارد که از لیست تابو انتخاب شـود. از طرفـی چـون حـداکثر

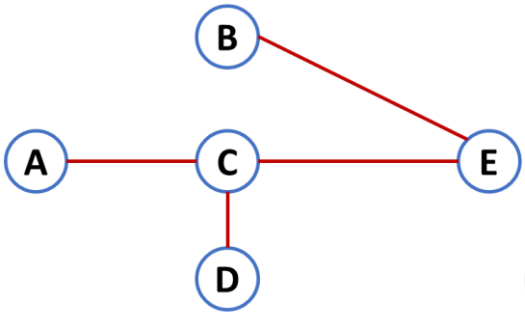
اعضای لیست تابو ۲ است و با اضافه کردن لینک *CD*، باید قدیمی ترین لینک خارج شود که در این صورت لینک *DE* از لیست خارج شود و می توان مورد بررسی قرار می گیرد و در حال حاضر لیست تابو بـه صـورت $\{BE,CD\}$ می شود.

تکرار ۴

در ابتدای این تکرار لیست تابو به صورت $\{BE,CD\}$ است.

هزینه	شبکه	حذف کردن	اضافه کردن
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض می شود. ۱۸۰=۱۰۰+۸۰		AC	AB
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض می شود. ۱۰۰=۸۵+۱۰۰		CE	AB
محدودیت ۱ نقض می شود. محدودیت ۲ نقض می شود. ۲۷۵=۷۵+۱۰۰+۱۰۰		AC	AD
محدودیت ۱ نقض می شود. محدودیت ۲ نقض نمی شود. ۱۶۰=۱۰۰+۶۰		CD*	AD
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. کمینه ** ۸۵=۸۵+۰		CD*	DE
محدودیت ۱ نقض نمی شود. محدودیت ۲ نقض نمی شود. ۱۰۵=۱۰۵+۰		CE	DE

مشاهده می شود که بهبودی در این تکرار رخ نداده و بهبودی در جواب رخ نداده و به جواب تکرار ۲ رسیدیم و در صورت ادامه دوباره به تکرار ۳ می رسیم. لذا به جواب بهینه رسیدیم و این جواب به صورت زیر است.



برای دریافت بسته‌های آموزشی گروه **بهینه‌یاب** به وب سایت ما به نشانی www.behinehyab.com مراجعه کنید.

در صورت هر گونه سوال از طریق ایمیل به نشانی behinehyab@gmail.com و یا با شماره ۰۹۰۲۷۷۷۰۶۰۱ با گروه **بهینه‌یاب** در تماس باشید.

با تشکر از توجه شما

گروه آموزشی **بهینه‌یاب**