به نام خدا

طراحی الگوریتم

نمونه‌ی امتحان پایان‌ترم (سه بخش آخر)

مدت امتحان: ۳ ساعت

توجه

- امتحان پایان‌ترم از کل مباحث تدریس شده می‌باشد. برای نمونه سوال سه بخش اول، به نمونه سوالات میان‌ترم مراجعه کنید.
- توصیه می‌شود قبل از خواندن پاسخ‌ها، سعی کنید سوال را خودتان حل نمایید.

۱. یک گراف وزن‌دار و درخت پوشای کمینه آن داده شده است. الگوریتم بهینه‌ای (به همراه شبه کد) برای هریک از حالات زیر ارائه دهید تا پس از تغییر گراف، درخت پوشای کمینه را به روزرسانی کند (به عبارتی درخت پوشا، کمینه باقی بماند.) تحلیل پیچیدگی زمان اجرای الگوریتم را نیز برای هر حالت بدست آورید.

الف) وزن یک یال گراف که جزئی از درخت پوشا کمینه داده شده نیست، کاهش بیابد.

ب) وزن یک یال گراف که جزئی از درخت پوشا کمینه داده شده است، کاهش بیابد.

پاسخ:

الف) یال با وزن کاهش یافته را به MST اضافه کنید. به کمک یک الگوریتم پیدا کردن دور (نظیر DFS)، دور ایجاد شده به واسطه یال جدید را پیدا کنید. یال با کمترین وزن را حذف کرده تا MST گراف جدید را بیابید.

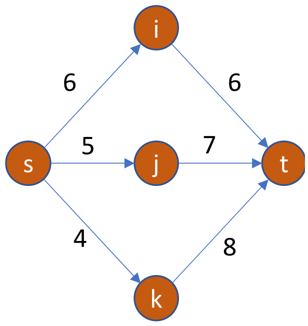
```
UpdateMST(MST, w, e) {
    MST_new = MST + e
    cycle = Find_Cycle(MST_new)

    max_weight = -∞
    max_edge = null
    for each edge in cycle {
        if max_weight < w(edge) {
            max_weight = w(edge)
            max_edge = edge
        }
    }

    return MST_new - max_edge
}
```

زمان اجرای الگوریتم، برابر $O(V+E)$ می‌باشد.

ب) چون MST شامل یال‌هایی با کمترین وزن است، کاهش وزن یال‌های آن باعث تغییر MST نمی‌شود.

۲. گراف جهت‌دار G با وزن‌های مثبت داده شده است. می‌خواهیم از بین کوتاه‌ترین مسیرهای از راس s به راس t، مسیری را پیدا کنیم که بیشترین وزن یال‌ها در این مسیر کمینه باشد. الگوریتم دایکسترا را طوری تغییر بدهید (به همراه ارائه شبه کد) که این مسیر را پیدا کند. پیچیدگی زمان اجرای الگوریتم خود را بدست آورده و نشان دهید پاسخ صحیح را پیدا می‌کند.

مثال: در گراف زیر، کوتاه‌ترین مسیر مورد نظر، $s \to i \to t$ می‌باشد. مسیرهای جایگزین، یعنی $s \to k \to t$ و $s \to j \to t$، هرکدام دارای یالی هستند که از بیشترین وزن یال‌های مسیر مورد نظر، وزن بیشتری دارد.



پاسخ:

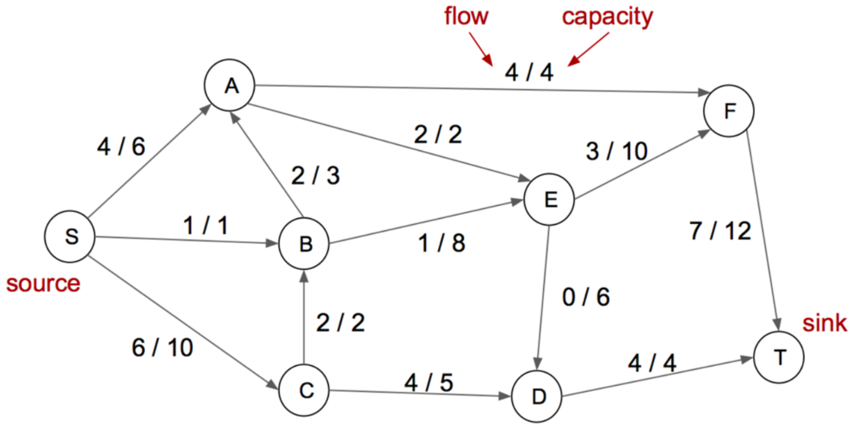
مشابه مثال حل شده در کلاس، می‌توان متغیری به نام max_edge[v] تعریف کرد که طول بلندترین یال را در کوتاه‌ترین مسیر پیدا شده از s به v نگه دارد. هرگاه مسیری با طول مشابه و با یال کوتاه‌تر پیدا شد، max_edge[v] آپدیت شده و مسیر جدید به عنوان مسیر مورد نظر نگهداری شود. زمان اجرای این الگوریتم مشابه الگوریتم دایکسترا $O(|E|\log|V|)$ (در پیاده‌سازی زیر) یا $O(|V|^2)$ می‌باشد. برای چاپ جواب می‌توانید از مقدار ذخیره شده در متغیر parent استفاده کنید.

```
Dijkstra(G, w, s) {
    S = {}
    for each node v ∈ V
        dist[v] = ∞
        max_edge[v] = ∞  // new variable init
        parent[v] = null
    dist[s] = 0
    max_edge[s] = 0

    for each node v ∈ V
        Add v to Q with priority dist[v] **

    while Q ≠ {} {
        u = ExtractMin(Q) **
        S = S + {u}
        for each neighbor of u in V - S {
            // Updated condition
            if dist[v] > dist[u] + w(u, v) OR
             (dist[v] == dist[u] + w(u, v) AND max_edge[v] > max(max_edge[u], w(u,v)))
{
```

```
                dist[v] = dist[u] + w(u, v)
                parent[v] = u
                // New variable update
                max_edge[v] = max(max_edge[u], w(u,v))
                Decrease priority of v in Q with dist[v]
            }
        }
    }
}
```

---

۳. فرض کنید شبکه جریان زیر و جریان ممکن f داده شده است.



الف) اندازه‌ی جریان f در گراف بالا چقدر است؟

ب) از جریان f در گراف بالا شروع کرده و یک مرحله از الگوریتم فورد-فالکرسون را روی آن اجرا کنید. رئوس روی مسیر تجمیعی (augmenting path) را از S به T به ترتیب ذکر کنید. جریان جدید را در گراف نشان دهید.

ج) اندازه‌ی جریان بیشینه در این گراف چقدر است؟

د) min-cut را در این گراف نشان داده و لیست رئوس سمت S و سمت T را ذکر کنید.

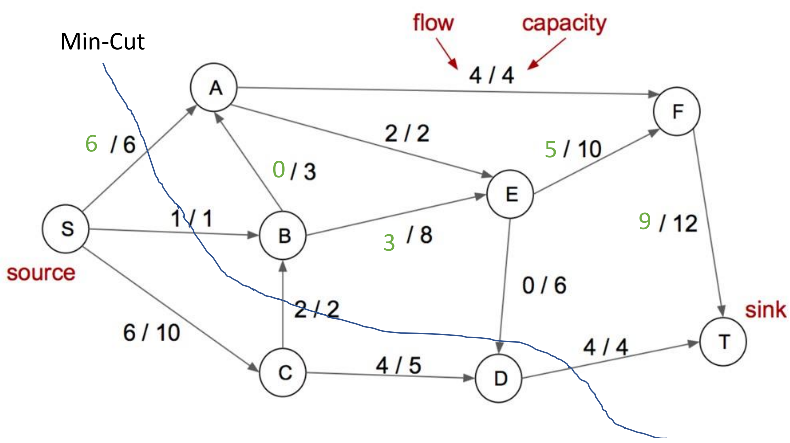
ه) ظرفیت min-cut در این گراف چقدر است؟

نکته: نیازی به نوشتن الگوریتم در مراحل بالا نیست.

پاسخ:

الف) ۱۱

ب) رئوس مسیر تجمیعی $S \rightarrow A \rightarrow B \rightarrow E \rightarrow F \rightarrow T$ می‌باشند. بر این اساس، جریان جدید به صورت زیر خواهد بود.



ج) پس از استفاده از مسیر تجمیعی قسمت ب، دیگر مسیر تجمیعی دیگری وجود نخواهد داشت. پس جریان بیشینه برابر ۱۳ خواهد بود.

د) C و D در سمت S و A، B، E و F در سمت T خواهند بود. min-cut در شکل نشان داده شده است.

ه) ۱۳ (برابر قسمت ج)

NOT APPROVED ۴. فرض کنید n مشتری و m محصول متفاوت دارید. هرکدام از مشتری‌ها تعداد متفاوتی از محصولات را خریداری کرده‌اند. از شما خواسته شده پرسشنامه‌ای با شرایط زیر طراحی کنید:

- هر مشتری فقط سوالاتی در مورد محصولاتی که خریداری کرده دریافت می‌کند.
- تعداد سوالاتی که هر مشتری دریافت می‌کند باید بین حداقل و حداکثر مشخصی باشد. این میزان برای هر مشتری متفاوت است.
- تعداد مشتری‌هایی که برای هر محصول مورد پرسش واقع می‌شوند باید بین حداقل و حداکثر مشخصی باشند. این میزان برای هر محصول متفاوت است.

به عنوان مثال، اطلاعات دریافتی شما برای طراحی پرسشنامه‌ها به صورت زیر خواهد بود.

حداکثر تعداد سوالات	حداقل تعداد سوالات	نوع محصول خریداری شده	شماره مشتری
۴	۲	B, C, D, F	۱
۳	۳	A, B, D	۲
۴	۲	B, C, E, F	۳
۲	۲	C, E	۴
۲	۲	A, D	۵

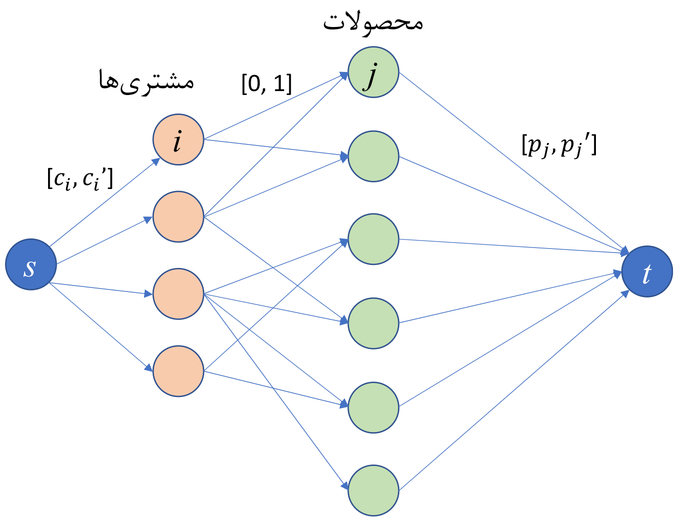
حداکثر تعداد مشتریانی که باید مورد پرسش واقع شوند	حداقل تعداد مشتریانی که باید مورد پرسش واقع شوند	نوع محصول
۲	۱	**A**
۳	۲	**B**
۳	۳	**C**
۳	۲	**D**
۲	۲	**E**
۲	۲	**F**

الگوریتم کارآمدی طراحی کنید که با دادن اطلاعاتی شبیه جداول بالا به عنوان ورودی، پرسشنامه‌های هر مشتری را طراحی کند. پیچیدگی محاسباتی الگوریتم شما چقدر است؟

راهنمایی: سعی کنید این مسئله را به یکی از انواع مسائل شبکه جریان تبدیل کنید. نیازی به نوشتن شبهه کد نیست ولی باید گام‌های الگوریتم شما به طور کامل مشخص باشد.

پاسخ: this question i think is like hw5 99 spring

ابتدا یک گراف دو بخشی می‌سازیم. بخش اول n راس دارد که نشان‌دهنده‌ی مشتری‌ها و بخش دیگر m راس دارد که نشان‌دهنده‌ی محصولات است. از مشتری i به محصول j یک یال وجود دارد اگر مشتری iام محصول jام را خریداری کرده باشد. دو راس s و t نیز به این گراف اضافه کرده و راس s را به همه n راس متناظر با مشتری‌ها وصل کرده و همه‌ی m راس متناظر با محصولات را به راس t متصل می‌کنیم. حد بالا و پایین جریان شبکه برای یال‌های (s, i) برابر $[c_i, c_i']$ است که به ترتیب برابر حداقل و حداکثر سوالاتی است که از مشتری iام باید پرسیده شود. حد بالا و پایین برای یال‌های (i, j) برابر [0, 1] می‌باشد. همچنین حد بالا و پایین یال‌های (j, t) برابر $[p_j, p_j']$ است که به ترتیب برابر حداقل و حداکثر مشتریانی است که برای محصول j سوال می‌شوند. در این گراف تقاضای هر راس (demand) برابر صفر است. این گراف یک گراف circulation را تشکیل می‌دهد. حل این مسئله برابر با یافتن یک circulation در گراف است. زمان اجرا نیز به کمک روش ادموندز-کارپ برابر $O(|m+n||m \times n|^2)$ می‌باشد. دقت کنید که الگوریتم فورد-فالکرسون کارآمد نبوده و مسئله را در زمان چندجمله‌ای حل نمی‌کند.



حال نشان می‌دهیم:

۱) هر راه‌حلی برای مسئله‌ی اصلی، معادل راه‌حلی در مسئله circulation است. این بخش از نحوه‌ی ساختن گراف مشخص است. سوالاتی که از مشتری iام پرسیده شده متناظر با محصولاتی با جریان ۱ و سوالات پرسیده نشده برابر با جریان ۰ است. به علاوه، جریان (s, i) برابر تعداد سوالات پرسیده شده از مشتری iام و جریان

(j, t) برابر با تعداد مشتریانی است که در مورد محصول jام از آنها پرسش شده است.

۲) هر راه‌حلی برای مسئله circulation معادل راه حلی برای مسئله‌ی اصلی است. وجود یک circulation به معنای وجود یک راه حل با مقدار صحیح برای جریان است (ر.ک. به اسلایدهای درس) پس بر این اساس می‌توان پرسشنامه‌ها را برای هر مشتری طراحی کرد: از مشتری iام در مورد محصول jام فقط و فقط موقعی سوال می‌شود که یال (i, j) جریان واحد داشته باشد. به دلیل قانون بقای جریان و اینکه

محدودیت سوالات در مسئله‌ی circulation رعایت شده، پرسشنامه‌های طراحی شده قیود مسئله را رعایت می‌کنند.

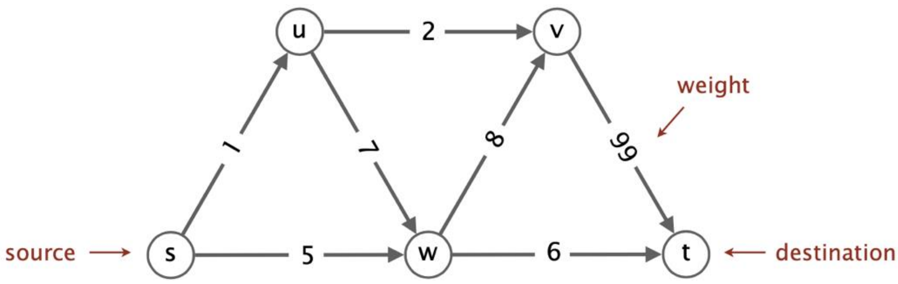
---

۵. دو مسئله‌ی یافتن کوتاه‌ترین مسیر در گراف را در نظر بگیرید:

مسئله‌ی A: گراف وزن‌دار و جهت‌دار G با وزن‌های غیرمنفی و دو راس مبدا s و مقصد t داده شده است. کوتاه‌ترین مسیر از s به t را پیدا کنید.

مسئله‌ی B: گراف وزن‌دار و جهت‌دار G با وزن‌های غیرمنفی و دو راس مبدا s و مقصد t داده شده است. کوتاه‌ترین مسیر از s به t را پیدا کنید اگر بتوانید از یکی از یال‌های این مسیر با وزن صفر عبور کنید. به عبارتی، وزن هر مسیر برابر با مجموع وزن یال‌های آن منهای وزن سنگین‌ترین (بزرگترین) یال است.

به عنوان مثال، کوتاه‌ترین مسیر از s به t در مسئله‌ی A برابر با $s \to w \to t$ است که وزن ۱۱ دارد. برای همین مثال، در مسئله‌ی B، مسیر $s \to u \to v \to t$ با وزن ۳ کوتاه‌ترین مسیر است.



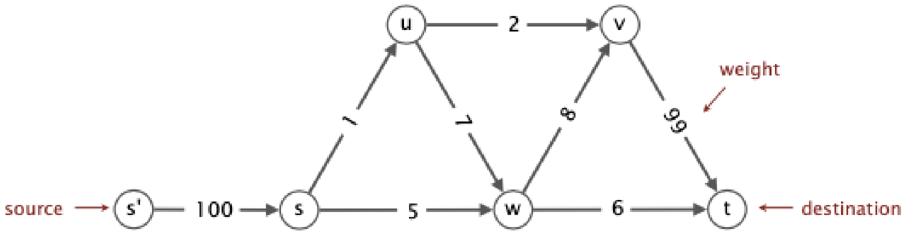
الف) یک تبدیل (reduction) با زمان خطی از مسئله‌ی A به B ارائه بدهید.

ب) یک تبدیل با زمان خطی از مسئله‌ی B به A ارائه بدهید. APPROVED

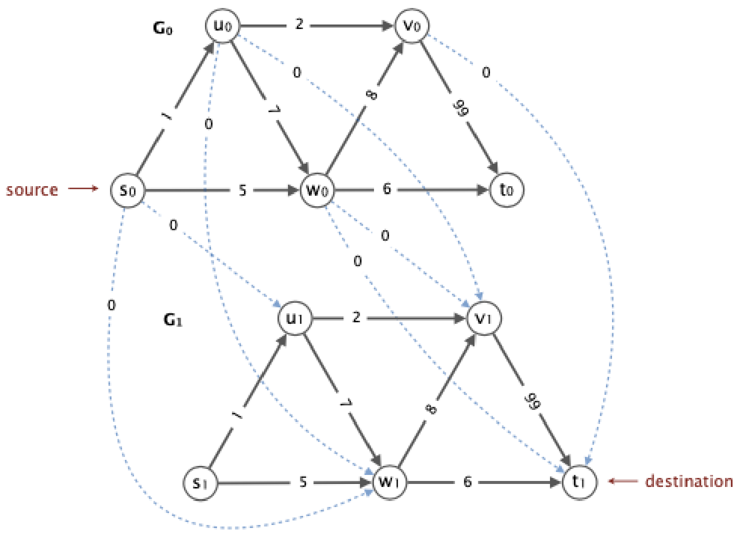
نکته: دقت کنید در تبدیل‌های خود، جهت تبدیل را رعایت کنید. به عبارتی، پاسخ قسمت الف و ب را به درستی ذکر کنید.

پاسخ:

الف) از گراف داده شده‌ی G، یک گراف جدید G’ می‌سازیم بدین صورت که یک راس جدید به نام s’ به G اضافه کرده و یال (s, s’) را با وزن $max\{e \in E: w_e\} + 1$ به آن می‌افزاییم. حال ادعا می‌کنیم کوتاه‌ترین مسیری که مسئله‌ی B برای رفتن از s’ به t پیدا می‌کند حتما از یال اضافه شده عبور می‌کند و شامل کوتاه‌ترین مسیر از s به t نیز می‌باشد. صحت این ادعا با توجه به نحوه‌ی ساختن G’ واضح است زیرا تنها یالی که s’ را به بقیه‌ی گراف متصل می‌کند یال جدید است. این یال بزرگترین یال گراف است و حذف خواهد شد و در نتیجه باقی یال‌ها از s به t مانند مسئله‌ی A در نظر گرفته خواهند شد. بالعکس، اگر بخواهیم کوتاهترین مسیر برای G’ را به کمک مسئله‌ی A پیدا کنیم، می‌توانیم کوتاه‌ترین مسیر از s به t را یافته و یال s به s’ را به آن بیفزاییم چون یال (s, s’) بزرگ‌ترین یال در گراف است، در انتها در نظر گرفته نخواهد شد. پس مسئله‌ی A قابل تقلیل به مسئله‌ی B در زمان خطی است.



ب) فرض کنید گراف ورودی داده شده G است. یک گراف جدید G’ به کمک دو کپی G_0 و G_1 از G می‌سازیم و رئوس اولی را با اندیس ۰ و دومی را با اندیس ۱ نمایش می‌دهیم (ر.ک. به شکل زیر) اگر یال $v \rightarrow w$ در گراف G وجود داشته باشد، یک یال با وزن صفر از v_0 به w_1 به گراف G’ اضافه می‌کنیم. حال ادعا می‌کنیم برای یافتن کوتاه‌ترین مسیر از s به t در گراف G (مسئله‌ی B)، می‌توان در مسئله‌ی A کوتاهترین مسیر از s_0 به t_1 را در در گراف G’ پیدا کرد. دقت کنید که برای این منظور، وقتی کوتاه‌ترین مسیر در مسئله‌ی B پیدا شد، مسیری که از گراف G_0 به G_1 می‌رود معادل یالی است که الگوریتم B از آن با وزن صفر عبور می‌کند. باقی مسیر در گراف G_1 طی می‌شود که معادل آن در G_0 وجود دارد. به طور مشابه، کوتاه‌ترین مسیر در گراف G (در مسئله‌ی B) را می‌توان به جوابی در مسئله‌ی A تبدیل کرد بدین صورت که یالی که از آن با وزن صفر عبور کرده‌ایم یالی خواهد بود که از G_0 به G_1 می‌رویم و باقی مسیر را به طور متناظر در G_1 طی می‌کنیم. پس مسئله‌ی B قابل تقلیل به مسئله‌ی A در زمان خطی است.



۶. مسئله‌ی «تقسیم دوگانه مجموعه» به صورت زیر تعریف می‌شود: یک مجموعه از اعداد صحیح به صورت $X = \{x_1, x_2, \dots, x_n\}$ داده شده است. آیا این مجموعه را می‌توان به دو زیر مجموعه‌ی $X_۱$ و $X_۲$ افراز کرد به طوری که مجموع اعضای $X_۱$ و $X_۲$ با هم برابر باشند؟ دقت کنید که افراز مجموعه X بدین معنی است که $X_1 \cup X_2 = X$ و $X_1 \cap X_2 = \emptyset$.

حال مسئله‌ی «تقسیم سه‌گانه مجموعه» را در نظر بگیرید: یک مجموعه به صورت $X = \{x_1, x_2, \dots, x_n\}$ داده شده است. آیا این مجموعه را می‌توان به سه زیر مجموعه‌ی X_1 و X_2 و X_3 افراز کرد به طوری که مجموع اعضای هرکدام از X_1 و X_2 و X_3 با هم برابر باشند؟ دقت کنید که افراز مجموعه X بدین معنی است که اشتراک دو به دوی زیرمجموعه‌ها تهی بوده و $X_1 \cup X_2 \cup X_3 = X$.

اگر بدانیم مسئله‌ی «تقسیم دوگانه مجموعه»، **NP-Complete** است، اثبات کنید مسئله‌ی «تقسیم سه‌گانه مجموعه» نیز **NP-Complete** است.

پاسخ:

ابتدا باید نشان دهیم که این مسئله NP است. برای این منظور، به عنوان گواهی، سه زیرمجموعه‌ی افراز شده‌ی X_1 و X_2 و X_3 را در نظر می‌گیریم. سپس یک verifier می‌سازیم که شرایط زیر را بررسی کند:

۱. این زیرمجموعه‌ها هیچ اشتراکی نداشته باشند. این عمل در O(n) قابل انجام است (راه چندجمله‌ای هم قابل قبول است.)

۲. اجتماع این ۳ زیرمجموعه برابر X است. این عمل در O(n) قابل انجام است. (راه چندجمله‌ای هم قابل قبول است.)

۳. جمع اعضای هرکدام از این سه زیرمجموعه برابر است. این عمل در O(n) قابل انجام است. (راه چندجمله‌ای هم قابل قبول است.)

حال باید ثابت کنیم مسئله، NP-hard است. برای اینکار «تقسیم دوگانه مجموعه» را به «تقسیم سه‌گانه مجموعه» تقلیل می‌دهیم. دو حالت زیر را در نظر بگیرید:

حالت ۱: مجموع اعضای X زوج است. در این حالت، مقدار y را که به صورت $y = \frac{1}{2}\sum x_i$ تعریف می‌شود، به مجموعه اضافه می‌کنیم:

$$X' = \{x_1, x_2, \dots, x_n, y\}$$

حالت ۲: مجموع اعضای X فرد است. در این حالت، y را طوری انتخاب می‌کنیم که $\sum x_i + y$ بر ۳ بخش‌پذیر نباشد سپس آن را به X' اضافه می‌کنیم:

$$X' = \{x_1, x_2, \dots, x_n, y\}$$

اگر پاسخ مسئله‌ی «تقسیم سه‌گانه مجموعه» به ورودی X'، «بله» باشد، مجموعه به سه زیرمجموعه شکسته می‌شود. یکی از زیرمجموعه‌ها برابر $X_3=\{y\}$ و دو تای دیگر (X_1 و X_2) مجموع هرکدامشان برابر y می‌شود (حالت ۱). پس می‌توان گفت پاسخ به مسئله‌ی «تقسیم دوگانه مجموعه» نیز بله است چون توانستیم X_1 و X_2 را با مجموع برابر بیابیم.

برعکس، اگر پاسخ مسئله‌ی «تقسیم دوگانه مجموعه» بله بوده و خروجی آن X_1 و X_2 باشد، در نتیجه مجموعه عناصر زوج بوده (حالت ۱) و می‌توان X_1، X_2 و y را می‌توان ساخت تا پاسخ مسئله‌ی «تقسیم سه‌گانه مجموعه» نیز بله باشد.
دقت کنید حالت ۲ برای این ساخته شده که مواردی که پاسخ مسئله خیر است، در هر دو مسئله به طور مشابه خیر باشد.