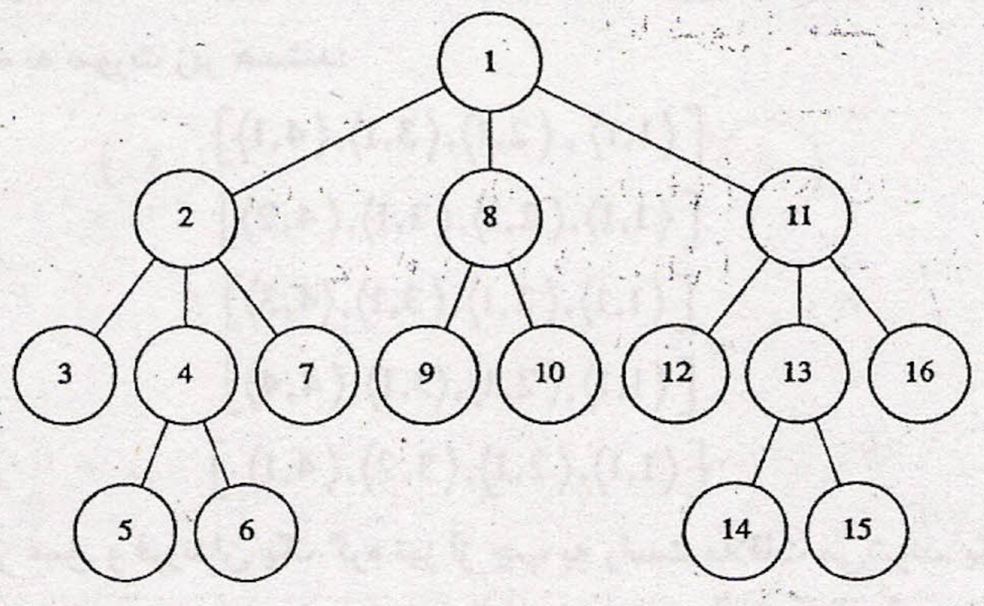
فصل ۵

روش‌های پسگرد

اگر در داخل یک هزار توی شمشادی قرار گرفته باشید، راهی جز دنبال کردن مسیری کورکورانه به امید این‌که به خروجی برسید ندارید. اگر به یک بن‌بست رسیدید بر می‌گردید و راه دیگری را امتحان می‌کنید. حال اگر در این هزار توی شمشادی یا معمای هزار تو، تابلوهایی وجود داشته باشند که به شما بگویند ادامهٔ مسیری که در پیش گرفته‌اید به بن‌بست منتهی می‌شود، پیدا کردن خروجی یا حل معما (البته اگر دیگر بشود اسم آن را معما گذاشت !) چقدر آسان‌تر خواهد شد. هر چقدر این تابلوها در ابتدای راه قرار داده شوند، زمان صرفه‌جویی شده، چشمگیرتر خواهد شد. زیرا دیگر نیازی به بررسی هیچ‌کدام از چند راهی‌های بعدی نخواهد بود. بدین‌ترتیب از ادامهٔ چندین مسیر منتهی به بن‌بست جلوگیری خواهد شد. در یک هزار توی شمشادی و معماهای هزار تو چنین تابلوهایی وجود ندارند، ولی در الگوریتم‌های پسگرد این تابلوها وجود دارند.

از روش پسگرد برای حل مسایلی استفاده می‌شود که در آن‌ها یک دنباله از اشیاء با شرایط داده شده باید از یک مجموعهٔ مشخص انتخاب شود. یک مثال کلاسیک از روش پسگرد، مساله n وزیر است. هدف از این مساله، چیدن n مهرهٔ وزیر در یک صفحه شطرنجی $n \times n$ به گونه‌ای است که هیچ دو وزیر هم‌دیگر را نزنند. به عبارت دیگر، هیچ دو مهره نباید در یک سطر، ستون یا قطر یکسان چیده شوند. در این مساله، دنباله عبارت از n موقعیتی است که در آن‌ها باید مهرهٔ وزیرها قرار داده شوند. مجموعه برای هر انتخاب، عبارت است از n^2 موقعیت روی صفحه شطرنج و شرط داده شده این است که هیچ دو وزیر هم‌دیگر را نزنند. مساله n وزیر، صورت عمومی مساله 8 - وزیر است که در آن از صفحهٔ شطرنجی استاندارد استفاده می‌شود.

روش پسگرد، حالت اصلاح شدهٔ جستجوی در عمق یک درخت است. پیمایش پیش ترتیب یک درخت، جستجوی در عمق آن است. این بدان معنی است که در ابتدای ریشهٔ درخت ملاقات می‌شود و پس از ملاقات یک گره، همه نواده‌های آن نیز بلافاصله ملاقات می‌شوند. اگرچه در جستجوی در عمق، نیاز به رعایت هیچ اولویت خاصی برای ملاقات فرزندان یک گره نیست، با وجود این، فرض خواهیم کرد که فرزندان یک گره از چپ به راست ملاقات می‌شوند. شکل 1 جستجوی در عمق یک درخت را به شیوهٔ ذکر شده در بالا، نشان می‌دهد. گره‌ها به ترتیب ملاقات شدنشان شماره‌گذاری شده‌اند. در جستجوی در عمق، یک مسیر را آن قدر در عمق پیش می‌رویم تا به بن‌بست برسیم. در بن‌بست، دوباره آن قدر به عقب بر می‌گردیم تا به یک گره با یک فرزند ملاقات نشده برسیم. سپس، از این فرزند ملاقات نشده، پیشروی در عمق را مجدداً تا رسیدن به بن‌بست دیگر ادامه می‌دهیم.



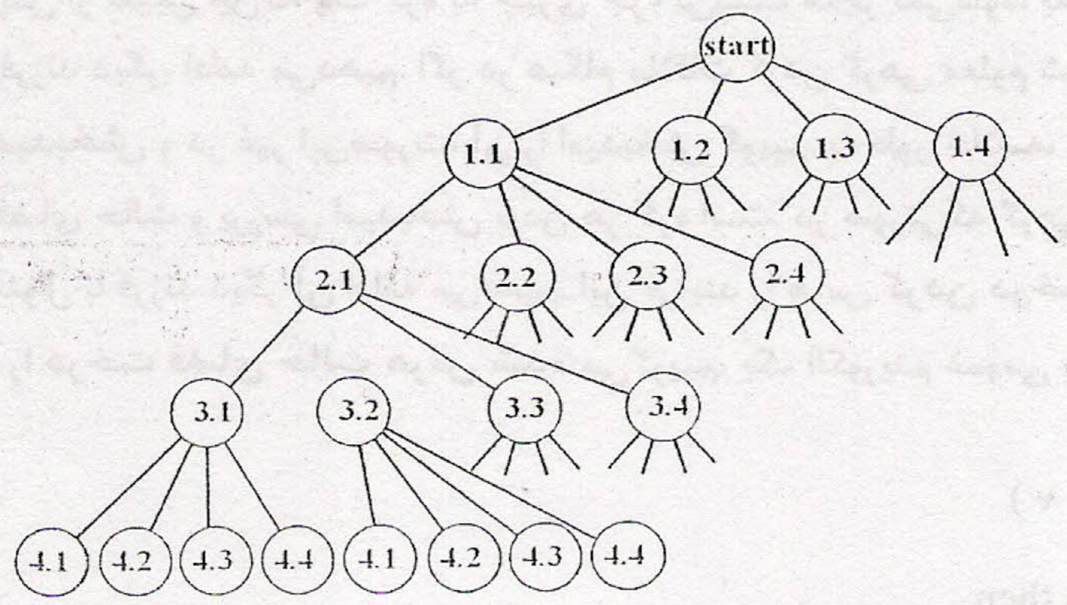
شکل ۱

مساله n – وزیر

روش پسگرد را با نمونه‌ای از مساله n وزیر که در آن $n = 4$ است، نشان می‌دهیم. هدف قرار دادن چهار وزیر بر روی یک صفحهٔ شطرنجی 4×4 است به طوری که هیچ دو وزیری هم دیگر را نزنند. با توجه به این‌که هیچ دو وزیر نمی‌توانند در یک سطر قرار بگیرند، لذا مساله را می‌توان به صورت زیر ساده کرد. می‌توان با نسبت دادن هر وزیر به یک سطر متفاوت و بررسی این‌که چه ترکیبی از ستون‌ها جواب است، مساله را حل کرد. چون هر وزیر را می‌توان در یکی از چهار ستون قرار داد، تعداد جواب‌های ممکن، $4\times4\times4\times4=256$ است.

جواب‌های ممکن را می‌توانیم با تشکیل درختی که در آن انتخاب‌های ستون برای نخستین وزیر (وزیر سطر اول) در گره‌های سطح 1 درخت ، انتخاب‌های ستون برای دومین وزیر (وزیر سطر دوم) در گره‌های سطح 2 و غیره نگهداری می‌شوند، تولید بکنیم (سطح ریشهٔ درخت برابر 0 فرض شده است). یک مسیر از ریشه به یک برگ، یک جواب ممکن است. این درخت را **درخت فضای حالت** می‌نامیم. بخش کوچکی از این درخت در شکل 2 آمده است. کل درخت 256 برگ دارد که هر کدام مربوط به یک جواب ممکن است.

مشاهده می‌کنید که در هر گره یک زوج مرتب $< i , j >$ نگهداری می‌شود. معنای این زوج مرتب این است که وزیر سطر i ام در ستون j ام قرار داده شده است.

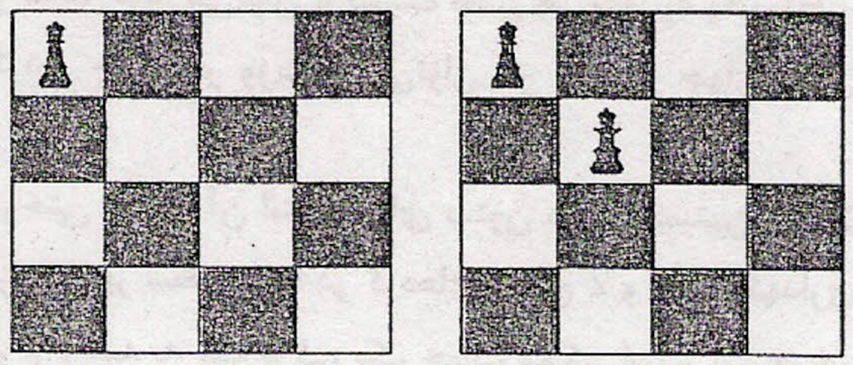


شکل ۲ ـ بخشی از درخت فضای حالت برای مساله 4- وزیر. هر زوج $< i , j >$ به این معنی است که وزیر i ام در ستون j ام قرار داده شده است. هر مسیر از ریشه به یک برگ یک جواب ممکن برای مساله است.

برای تعیین جواب‌ها، هر جواب ممکن (هر مسیر از ریشه به یک برگ) را با شروع از مسیر واقع در منتهی‌الیه سمت چپ، بررسی می‌کنیم. چند مسیر اولیهٔ بررسی شده به صورت زیر هستند:

$$[\langle 1,1\rangle , \langle 2,1\rangle, \langle 3,1\rangle, \langle 4,1\rangle]$$
$$[\langle 1,1\rangle, \langle 2,1\rangle, \langle 3,1\rangle, \langle 4,2\rangle]$$
$$[\langle 1,1\rangle, \langle 2,1\rangle, \langle 3,1\rangle, \langle 4,3\rangle]$$
$$[\langle 1,1\rangle, \langle 2,1\rangle, \langle 3,1\rangle, \langle 4,4\rangle]$$
$$[\langle 1,1\rangle, \langle 2,1\rangle, \langle 3,2\rangle, \langle 4,1\rangle]$$

گره‌ها مطابق با الگوریتم جستجوی در عمق و فرزندان یک گره نیز از چپ به راست ملاقات می‌شوند. یک جستجوی ساده و در عمق درخت فضای حالت، مانند دنبال کردن تمامی مسیرهای موجود در یک هزار توی شمشادی و رسیدن به بن‌بست است و از مزایای هیچ علامت موجود در مسیر بهره نمی‌برد. جستجوی را می‌توان با دقت کردن به چنین علایمی، کارآمدتر کرد. برای مثال همان‌طور که در شکل ۳ (الف) نشان داده شده است. هیچ دو وزیری را نمی‌توان در یک ستون قرار داد. بنابراین هیچ امتیازی در تشکیل و بررسی مسیرهای واقع در شاخهٔ منشعب از گره $< 1 , 2 >$ در این شکل وجود ندارد (زیرا که وزیر 1 را قبلاً در ستون 1 قرار داده‌ایم و دیگر نمی‌توانیم وزیر 2 را در آن ستون قرار دهیم). این علامت به ما می‌گوید که این گره به چیزی جز بن‌بست ختم نمی‌شود. به‌طور مشابه، همان طور که در شکل 3 (ب) نشان داده شده است، هیچ دو وزیری نمی‌توانند بر روی یک قطر چیده شوند. بنابراین هیچ امتیازی در تشکیل و بررسی مسیرهای واقع در شاخه منشعب از گره $< 2 , 2 >$ در این شکل وجود ندارد.



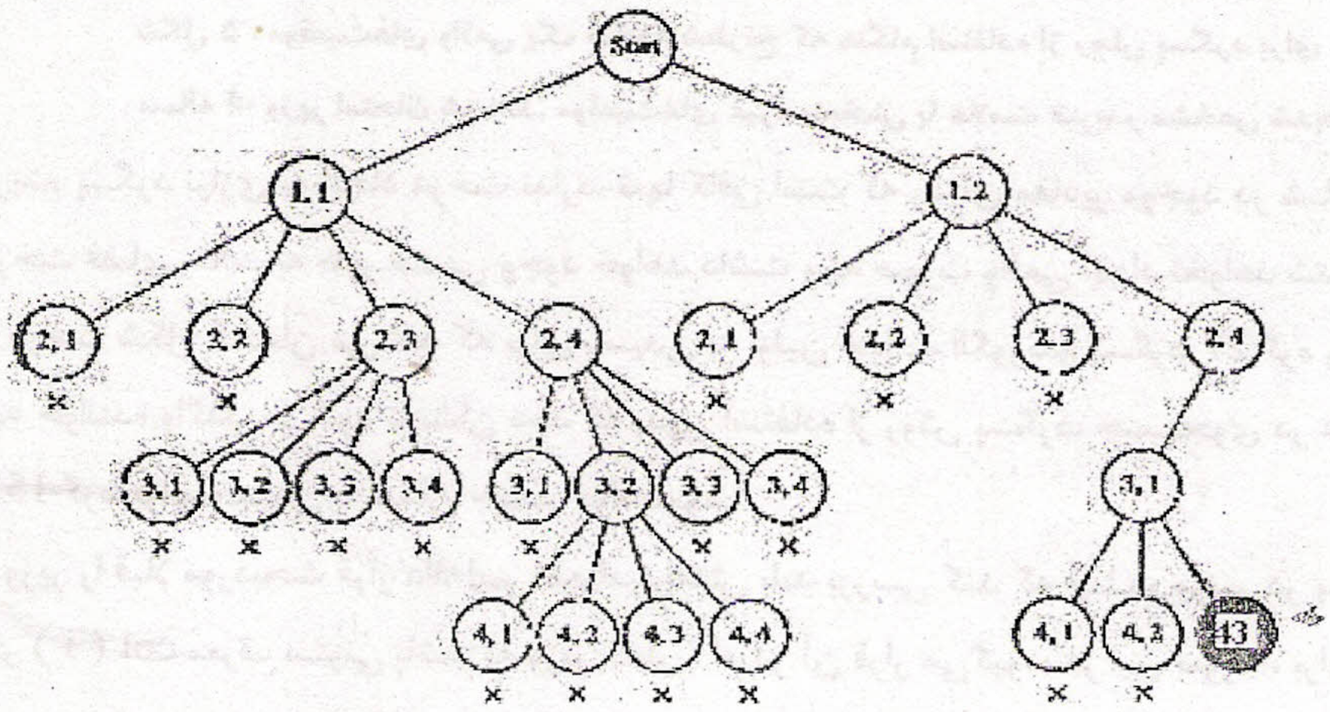
شکل ۳ : نموداری که نشان می‌دهد که اگر وزیر 1 را در ستون 1 قرار بدهیم، دیگر نمی‌توانیم وزیر 2 را در ستون 1 (الف) یا ستون 2 (ب) قرار دهیم.

پسگرد، روشی است که توسط آن پس از تعیین این‌که یک گره به چیزی جزء بن‌بست منجر نمی‌شود، به گره پدر بازگشته (عقب‌گرد کرده) و عمل را با جستجو بر روی فرزند دیگر، ادامه می‌دهیم. اگر در هنگام ملاقات کردن گرهی معلوم شود که این گره احتمالاً منجر به یک جواب نخواهد شد آن را **نا امیدبخش** و در غیر این‌صورت، آن را امیدبخش گوییم. به طور خلاصه، روش پسگرد، عبارت از انجام یک جستجوی در عمق در درخت فضای حالت و بررسی امیدبخش بودن هر گره است. در صورتی‌که گرهی امیدبخش نبود به گره پدر آن بازگشته و جستجو را با همین منوال با فرزند دیگر آن ادامه می‌دهیم. این فرآیند را هرس **کردن درخت فضای** حالت نامیده و زیر درخت شامل گره‌های ملاقات شده را **درخت فضای حالت هرس شده** می‌گوییم. یک الگوریتم عمومی برای روش پسگرد، به صورت زیر است:

```
Algorithm checknode ( v )
{
    if (promising ( v ) ) then
        if (there is a solution at v ) then
            write the solution ;
    else
    for (each child u of v )
    checknode ( u ) ;
}
```

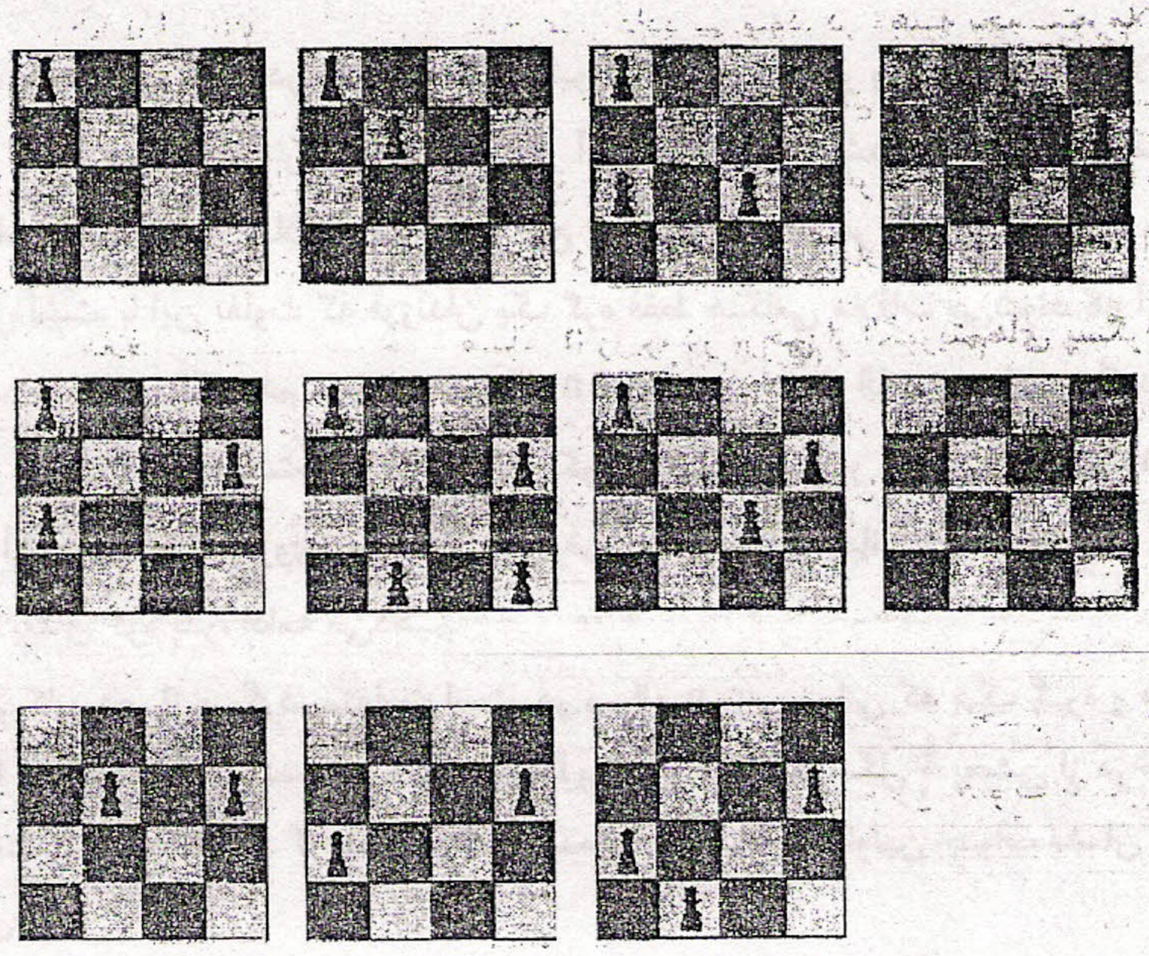
ریشهٔ درخت فضای حالت به checknode در بالاترین سطح عبور داده می‌شود. در وهلهٔ نخست، ملاقات کردن یک گره، بررسی امیدبخش بودن آن است. اگر گرهی امیدبخش بوده و جوابی در آن وجود داشته باشد،آن جواب چاپ می‌شود. اگر جوابی در یک گره امیدبخش وجود نداشته باشد، فرزندان آن ملاقات می‌شوند. تابع promising در هر کاربرد از پسگرد، متفاوت است. الگوریتم پسگرد همانند جستجوی در عمق است. با این تفاوت که فرزندان یک گره فقط هنگامی ملاقات می‌شوند که آن گره امیدبخش بوده و جوابی در آن وجود نداشته باشد. (بر خلاف الگوریتم مربوط به مساله n وزیر، در برخی از الگوریتم‌های پسگرد، جواب مساله ممکن است قبل از رسیدن به یک برگ در درخت فضای حالت، به دست آید). الگوریتم پسگرد مزبور به‌جای checknode , backtrack نامیده شده است زیرا موقع فراخوانی آن، عمل پسگرد روی نمی‌دهد. در عوض، عمل پسگرد زمانی رخ می‌دهد که گرهی امیدبخش نباشد که در این صورت کار را با فرزند بعدی گره پدر، ادامه می‌دهیم.

تابع promising برای هر کاربردی از پسگرد، متفاوت است. در مساله n وزیر، زمانی که یک گره و هر یک از نیاهای آن بخواهند وزیرها را در یک ستون یا یک قطر قرار دهند، باید مقدار false برگردانیده شود. شکل 4 بخشی از درخت فضای حالت هرس شده را برای حل مساله 4- وزیر نشان می‌دهد. فقط گره‌های بررسی شده برای رسیدن به اولین جواب نشان داده شده‌اند. شکل 5، صفحه شطرنج واقعی را نشان می‌دهد.



شکل ۴ : بخشی از درخت فضای حالت هرس شده، ایجاد شده توسط روش پسگرد برای حل مساله 4- وزیر. فقط گره‌های بررسی شده برای رسیدن به اولین جواب نشان داده شده‌اند. این جواب در گره سایه‌دار پیدا شده است. گره‌های غیر امیدبخش با علامت × مشخص شده‌اند

در این شکل، گره‌های غیر امیدبخش با علامت × گذاشته شده است. گره سایه‌دار در شکل 4 گرهی است که در آن نخستین جواب حاصل شده است.



شکل ۵ : موقعیت‌های واقعی یک صفحه شطرنج که هنگام استفاده از روش پسگرد برای حل مساله 4- وزیر امتحان شده‌اند. موقعیت‌های غیرامیدبخش با علامت ضربدر مشخص شده‌اند.

در واقع یک الگوریتم پسگرد نیازی به ایجاد درخت ندارد. تنها کافی است که ردپای مقادیر موجود در شاخهٔ مورد بازرسی را داشته باشیم. بنابراین درخت فضای حالت به طور ضمنی وجود خواهد داشت و به صورت واقعی ایجاد نخواهد شد.

شمارش گره‌های درخت شکل 4 نشان می‌دهد که برای رسیدن به اولین جواب، الگوریتم پسگرد 27 گره را برسی می‌کند. این مطلب به عنوان تمرین به خواننده واگذار می‌شود تا نشان دهد که بدون استفاده از روش پسگرد، جستجوی در عمق درخت فضای حالت، مستلزم بررسی 155 گره، برای رسیدن به همان جواب خواهد بود.

هدف از مساله n وزیر را قبلاً موردبحث قرار داده‌ایم. تابع امیدبخش باید بررسی کند که آیا دو وزیر در یک ستون یا در یک قطر هستند یا خیر. اگر col (i) معرف ستونی باشد که وزیر سطر i ام در آن قرار می‌گیرد، در این صورت، برای تعیین این‌که آیا با وزیر سطر k ام در یک ستون است یا نه، مقایسهٔ زیر را انجام می‌دهیم:

$$col(i)=col(k)$$

حال ببینیم که قطرها چگونه بررسی می‌شوند. شکل 6 نمونهٔ $n = 8$ را نشان می‌دهد. در این شکل، وزیر سطر 6، در قطر چپ، وزیر سطر 3 ،در قطر راست وزیر سطر 2 را می‌زند. توجه کنید که:

$$col(6) - col(3) = 4 - 1 = 3 = 6 - 3$$

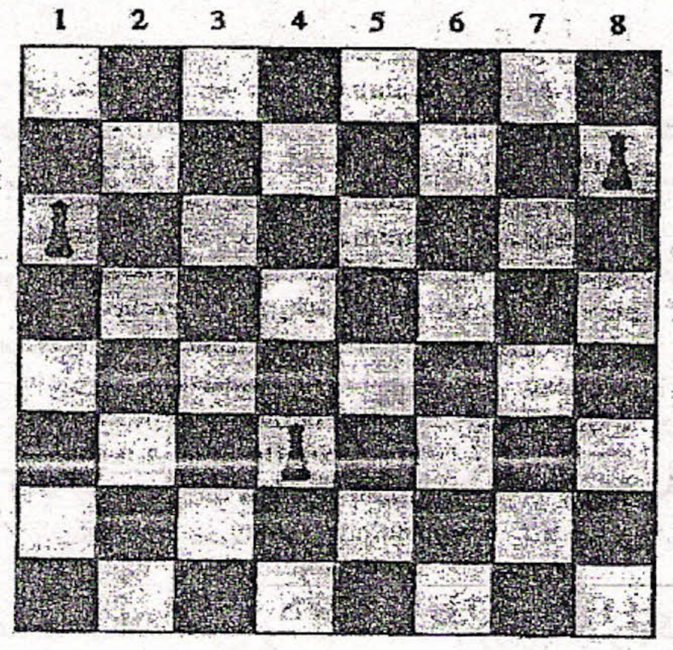
یعنی برای دو وزیری که بر روی یک قطر چپ قرار داشته باشند، اختلاف در ستون‌ها همانند اختلاف در ردیف‌ها است، به‌علاوه،

$$col(6) - col(2) = 4 - 8 = -4 = 2-6$$

یعنی، برای دو وزیری که در قطر راست هم‌دیگر را بزنند، اختلاف ستون‌ها، مساوی منفی اختلاف سطرها است.

این‌ها مثال‌هایی از این نتیجه کلی هستند که اگر وزیری در سطر k ام، وزیر دیگری در سطر i ام را در راستای یکی از قطرهایش بزند، در آن صورت:

$$col(i) - col(k) = i - k \quad \text{یا} \quad col(i) - col(k) = k - i$$



شکل ۶ : وزیر سطر 6 ، در قطر چپ، وزیر سطر 3، در قطر راست وزیر سطر 2 را می‌زند.

مساله حاصل جمع زیر مجموعه‌ها

مساله کوله‌پشتی 0/1 فصل پیش را به‌خاطر بیاورید. در این مساله، n بسته وجود داشت که هر کدام دارای وزن و ارزش خاصی بود. اگر ظرفیت کوله‌پشتی برابر M فرض شود، هدف این بود که بسته‌هایی در کوله‌پشتی قرار داده شوند که اولاً مجموع وزن آن‌ها از M بیشتر نبوده و دوماً مجموع ارزش بسته‌های قرار داده شده در کوله‌پشتی هم ماکزیمم بشود. حال اگر فرض کنیم که ارزش بسته‌ها در واحد وزن، یکسان است. در این صورت، جواب بهینه برای مساله کوله‌پشتی 0/1، صرفاً به دست آوردن زیر مجموعه‌ای از بسته‌ها خواهد بود که مجموع وزن آن‌ها ماکزیمم باشد، با این شرط این مجموع وزن آن‌ها از M بیشتر نشود. بنابراین در این حالت، بهترین جواب، زیر مجموعه‌ای از بسته‌ها خواهد بود که کوله‌پشتی را پر کند (مجموع وزن آن‌ها برابر M باشد). مساله تعیین چنین زیر مجموعه‌هایی را مساله حاصل‌جمع زیر مجموعه‌ها می‌گویند.

به صورت دقیق‌تر، در مساله حاصل‌جمع زیرمجموعه‌ها، n عدد صحیح و مثبت w_i (وزن‌ها) و یک عدد صحیح مثبت M داده شده و هدف، یافتن همهٔ زیرمجموعه‌هایی از این اعداد صحیح است که حاصل‌جمع آن‌ها برابر M بشود. مسلماً اگر این مساله را به مساله دزد و کوله‌پشتی تشبیه کنیم، تنها یافتن یک جواب کافی خواهد بود !

مثال : فرض کنید $M = 21 , n = 5$ و

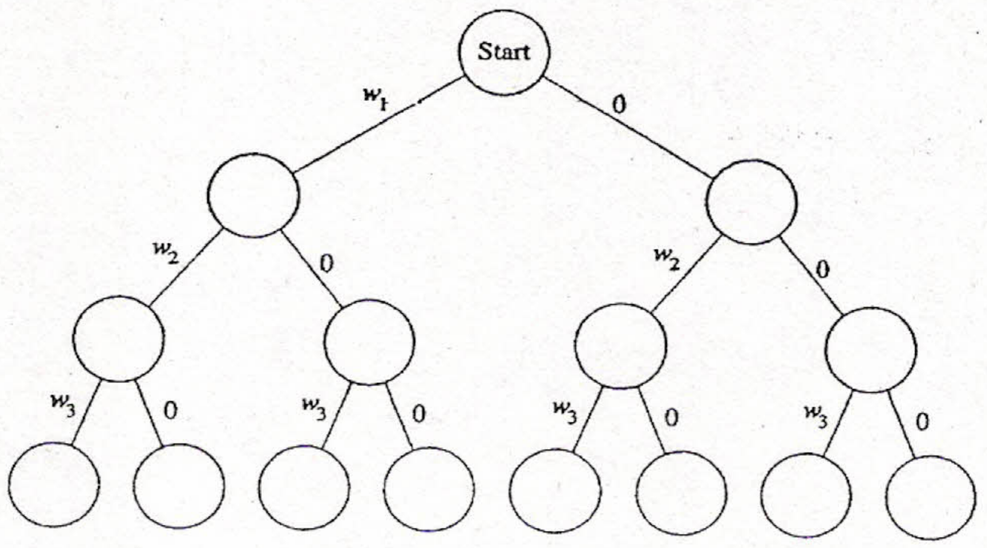
$w_1 = 5$ $w_2 = 6$ $w_3 = 10$ $w_4 = 11$ $w_5 = 16$

چون

$$w_1 + w_2 + w_3 = 5 + 6 + 10 = 21$$
$$w_1 + w_5 = 5 + 16 = 21$$
$$w_3 + w_4 = 10 + 11 = 21$$

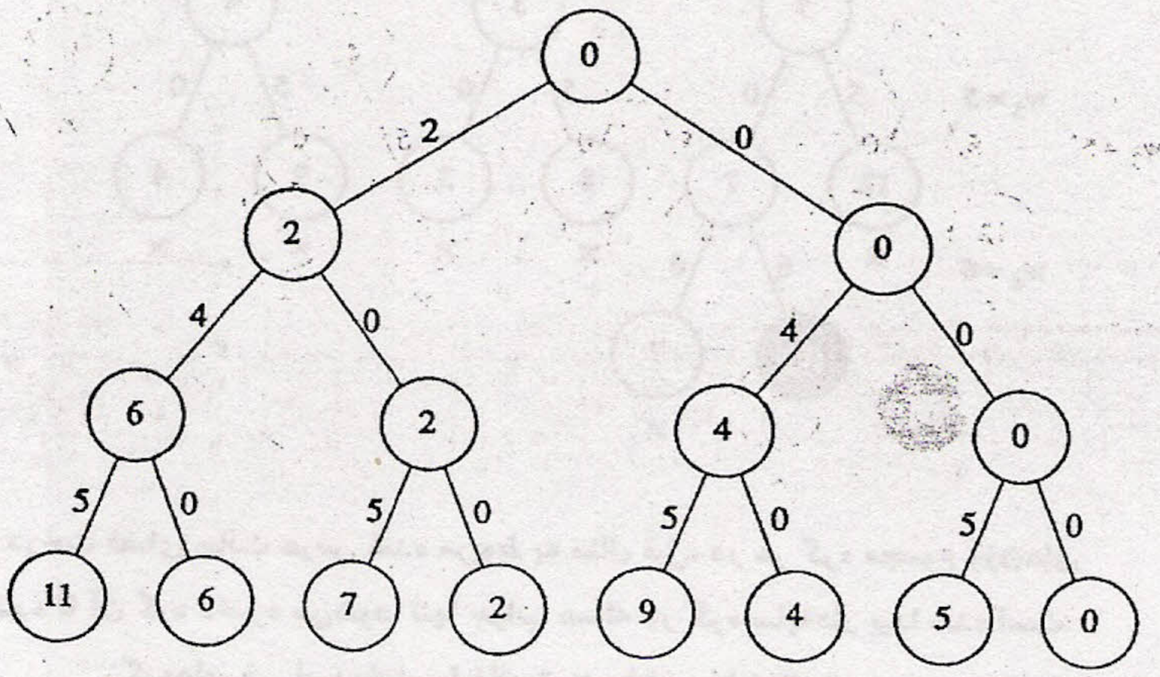
لذا، جواب‌ها عبارت‌اند از $\{w_1, w_2, w_3\}$ ، $\{w_1, w_5\}$ و $\{w_3, w_4\}$

هر چند این نمونه مساله را می‌توان با یک نگاه حل کرد، ولی برای n های بزرگ یک راه روشمند مورد نیاز است. یک روش، تشکیل درخت فضای حالت است.یک حالت ممکن برای ساختار چنین درختی مطابق شکل 7 است . به خاطر سادگی، درخت این شکل تنها به سه وزن مربوط می‌شود. اگر از ریشه به طرف چپ، برویم مفهومش این خواهد بود که w_1 را انتخاب می‌کنیم و اگر به طرف راست برویم به این معنی خواهد بود که w_1 انتخاب نشده است. به طریق مشابه، اگر از یک گره در سطح 1 ، به سمت چپ رفتیم به این معنی خواهد بود که w_2 را انتخاب کرده‌ایم و اگر به طرف راست رفتیم، آن‌را انتخاب نکرده‌ایم و الی آخر. هر مسیر از ریشه به برگ، معرفی یک زیر مجموعه است. اگر w_i انتخاب شده باشد، بر روی یال مربوطه این عدد را نوشته و در غیر این‌صورت عدد 0 را خواهیم نوشت.



شکل ۷ : یک درخت فضای حالت برای مساله حاصل‌جمع زیر مجموعه‌ها برای $n = 3$

مثال : شکل 8 یک درخت فضای حالت را برای $M = 6 , n = 3$ و $w_3 = 5$ $w_2 = 4$ $w_1 = 2$ را نشان می‌دهد. در هر گره این درخت ، حاصل‌جمع وزن‌های انتخاب شده تا آن گره، نوشته شده است. بنابراین، هر برگ، حاوی حاصل‌جمع اوزان زیر مجموعه‌ای است که به آن برگ منتهی می‌شود. برگ دوم از طرف چپ تنها برگی است که حاوی عدد 6 است. چون مسیر منتهی به این برگ نشانگر زیر مجموعهٔ $\{ w_1 , w_2 \}$ است، لذا این زیر مجموعه تنها جواب ممکن است.



شکل ۸ : یک درخت فضای حالت برای مساله ارایه شده در مثال قبل برای حاصل‌جمع زیر مجموعه‌ها

اگر اوازان را پیش از شروع جستجو، به ترتیب صعودی مرتب کنیم یک علامت آشکار مبنی بر غیر امیدبخش بودن یک گره وجود خواهد داشت. اگر وزن‌ها را به این ترتیب مرتب کنیم، هنگامی که در سطح i ام باشیم، w_{i+1} سبک‌ترین وزن باقیمانده خواهد بود. فرض کنید که weight، حاصل‌جمع اوزان انتخاب شده برای یک گره واقع در سطح i باشد. اگر برای w_{i+1}، مقدار weight از M بیشتر شود، در این صورت برای هر وزن دیگر بعد از آن نیز چنین خواهد بود. بنابراین به‌جز حالتی که weight برابر M است (که مفهومش این است که یک جواب در آن گره وجود دارد) یک گره در سطح i ام غیر امیدبخش است اگر:

$$Weight + w_{i+1} > M$$

یک علامت دیگر نیز با وضوح کمتر، برای تشخیص غیر امیدبخش بودن یک گره وجود دارد. اگر در یک گره مفروض با افزودن مجموع وزن‌های باقیمانده به weight ، حاصل از M کمتر شود، در آن صورت با گسترش آن گره، هرگز مقدار weight برابر با M نخواهد شد. این بدان معنی است که اگر total را برابر با مجموع وزن اعداد باقی‌مانده تعریف کنیم در این صورت یک گره غیر امیدبخش است هر گاه:

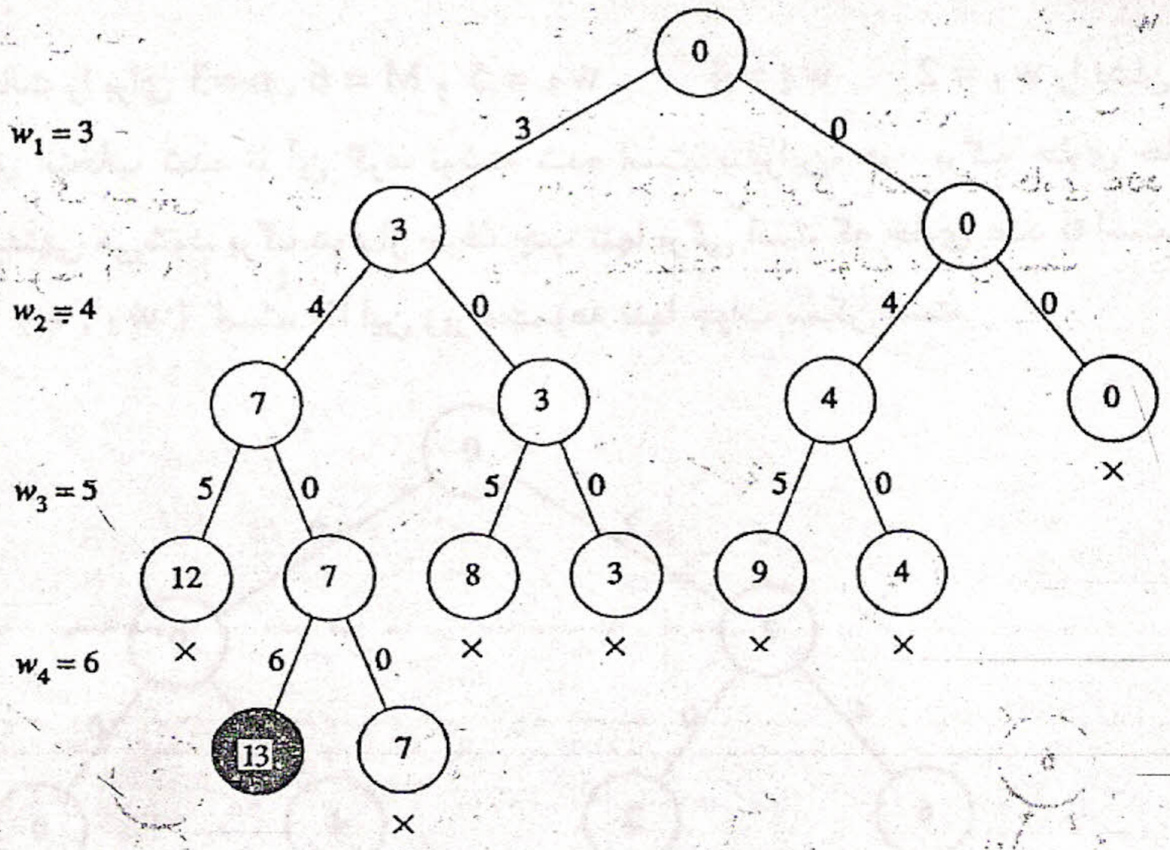
$$Weight + total < M$$

مثال زیر این راهبرد روش پسگرد را نشان می‌دهد.

مثال : شکل ۹ درخت فضای حالت هرس شده را برای $M = 13 , n = 4$

$W_1 = 3$ $w_2 = 4$ $w_3 = 5$ $w_4 = 6$

نشان می‌دهد. تنها یک جواب، $\{ w_1 , w_2 , w_4 \}$ در گره سایه‌دار پیدا شده است. گره‌های غیرامیدبخش با علامت × مشخص شده‌اند. گره‌های حاوی 12 , 8 و 9 غیر امیدبخش هستند، زیرا که افزودن وزن بعدی (6) به weight باعث می‌شود که مقدار آن از M تجاوز کند. گره‌های حاوی 7 , 3 , 4 و 0 نیز غیر امیدبخش هستند، زیرا که مجموع کل وزن‌های باقیمانده برای رساندن مقدار weight تا M کافی نیست. اگر گرهی در درخت فضای حالت حاوی جوابی نباشد، به صورت خودکار غیرامیدبخش است، زیرا که دیگر وزنی برای رسانیدن weight به M باقی نمانده است. برگ حاوی 7 این مطلب را نشان می‌دهد. 15 گره در درخت فضای حالت هرس شده وجود دارد، حال آن‌که تعداد گره‌های درخت فضای حالت برابر 31 است.



شکل ۹ : درخت فضای حالت هرس شده مربوط به مثال قبل، در هر گره مجموع وزن‌های انتخاب شده تا آن گره ذخیره می‌شود. تنها جواب مساله در گره سایه‌دار پیدا شده است.
گره‌های غیر امیدبخش با غلامت × مشخص شده‌اند.

هنگامی که حاصلِ‌جمع وزن‌های انتخاب شده تا یک گره، برابر با weight بشود، جوابی در آن گره وجود دارد. بنابراین، با انتخاب وزن‌های بیشتر، به جواب دیگری نخواهیم رسید، این بدان معنی است که اگر

$$M = \text{weight}$$

باید جواب را چاپ کرده و پسگرد کنیم.

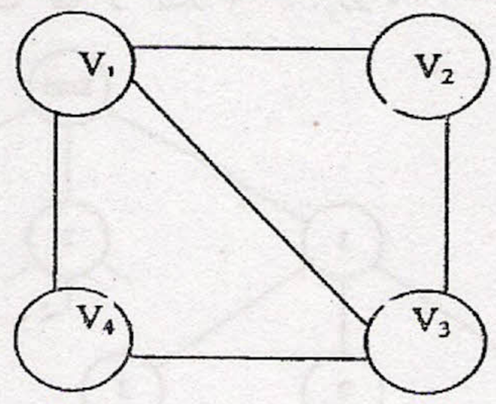
مساله رنگ‌آمیزی گراف‌ها

مساله m - رنگ‌آمیزی، یافتن همهٔ جواب‌های ممکن برای رنگ‌آمیزی یک گراف بدون جهت، با استفاده از حداکثر m رنگ متفاوت است، به‌طوری که هیچ دو راس مجاور، هم رنگ نباشند. معمولاً برای هر مقدار m، مساله m- رنگ‌آمیزی، یک مساله جداگانه‌های به حساب می‌آید.

مثال : گراف شکل 10 را در نظر بگیرید. برای مساله 2- رنگ‌آمیزی، جوابی برای این گراف وجود ندارد، زیرا اگر برای مثال راس v_1 را با رنگ اول رنگ‌آمیزی کنیم، چون این راس مجاور با 3 راس دیگر است، لذا ریوس v_2 ، v_3 و v_4 را باید با رنگ دوم رنگ‌آمیزی بکنیم، اما ریوس v_2 ، v_3 مجاور بوده و هر دو با رنگ دوم رنگ شده‌اند. یک جواب برای مساله 3- رنگ‌آمیزی برای این گراف به شرح زیر است:

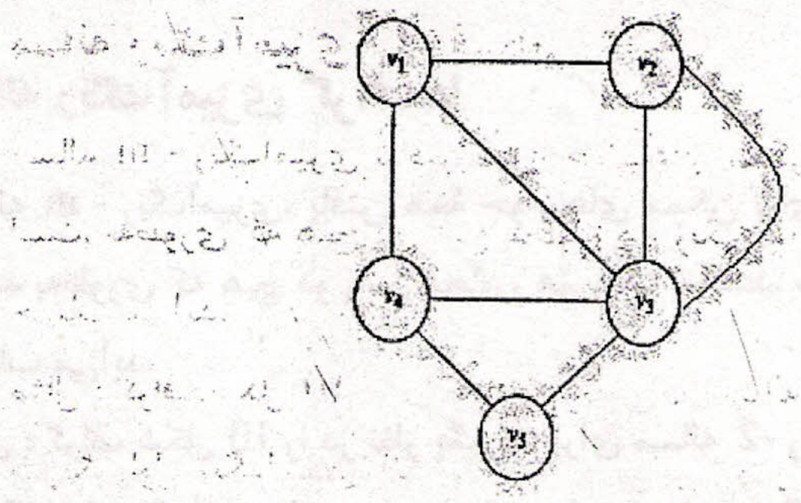
راس	رنگ
v_1	رنگ 1
v_2	رنگ 2
v_3	رنگ 3
v_4	رنگ 2

در کل شش جواب برای مساله 3- رنگ‌آمیزی برای این گراف وجود دارد. با وجود این، تفاوت این شش جواب فقط در شیوهٔ ترکیب رنگ‌ها است. برای مثال ، رنگ‌آمیزی v_1 با رنگ 2 ، v_2 و v_4 با رنگ 1 و v_3 با رنگ 3 ، یک جواب دیگر است.



شکل 10 : گرافی که برای آن جوابی برای مساله 2- رنگ‌آمیزی وجود ندارد. جواب مساله 3- رنگ‌آمیزی برای این گراف در مثال بالا توضیح داده شده است.

یک کاربرد مهم رنگ‌آمیزی گراف‌ها، رنگ‌آمیزی نقشه‌هاست. یک گراف را مسطح گوییم هر گاه بتوان آن را بر روی صفحه به گونه‌ای رسم کرد که هیچ دویالی یکدیگر را قطع نکنند. گراف شکل 11 ب یک گراف مسطح است، ولی اگر یال‌های (v_1 , v_5) و (v_2 , v_4) را به آن اضافه کنیم، گراف حاصل دیگر مسطح نخواهد بود متناظر با هر نقشه، یک گراف مسطح وجود دارد. هر ناحیه از نقشه توسط یک راس نشان داده می‌شود. اگر ناحیه‌ای مجاور با ناحیه دیگر است، ریوس متناظر با آن‌ها را به وسیله یک یال به هم وصل می‌کنیم. شکل 11 ، یک نقشه و گراف مسطح متناظر با آن را نشان می‌دهد. مساله m- رنگ آمیزی برای یک گراف مسطح، تعیین تعداد جواب‌هایی است که این نقشه را می‌توان با استفاده از حداکثر m رنگ، رنگ‌آمیزی کرد به گونه‌ای که هیچ دو ناحیهٔ مجاور، هم رنگ نباشند.

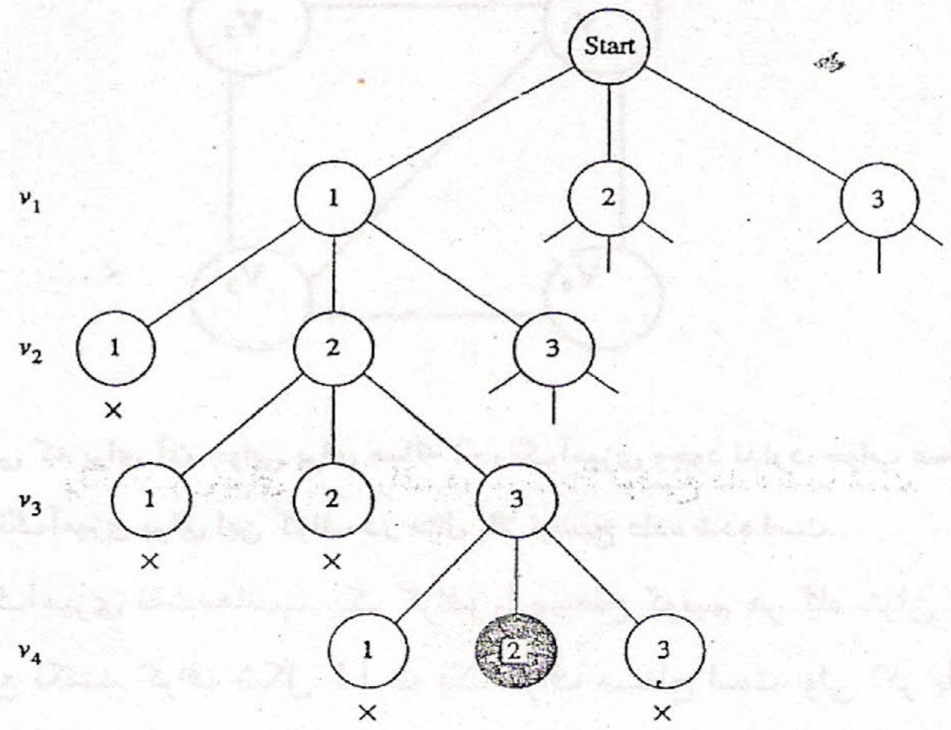


الف ب

شکل 11، نقشه (الف) و نمایش گراف مسطح آن (ب)

یک درخت فضای حالت صریح برای مساله m-رنگ‌آمیزی، درختی است که در آن هر رنگ‌آمیزی ممکن برای راس v_1 ، در سطح 1 ، هر رنگ‌آمیزی ممکن برای راس v_2 ، در سطح 2 و با ادامهٔ این روند، سرانجام هر رنگ‌آمیزی ممکن برای راس v_n در سطح n امتحان شود. هر مسیر از ریشه به یک برگ یک جواب ممکن خواهد بود. بررسی این‌که یک جواب ممکن، واقعاً جواب مساله است یا خیر، با تعیین همرنگ بودن هر دو راس مجاور به هم، امکان‌پذیر است.

شکل 12 بخشی از درخت فضای حالت هرس شدهٔ حاصل از اعمال روش پسگرد برای رنگ‌آمیزی گراف شکل 10 را با 3 رنگ نشان می‌دهد. عدد درون هر گره شمارهٔ رنگ به کار رفته برای رنگ‌آمیزی راس مربوط به آن گره در گراف است. نخستین جواب، در گره سایه‌دار پیدا شده است. گره‌های غیر امیدبخش با علامت × مشخص شده‌اند. پس از آن‌که v_1 با رنگ 1 رنگ آمیزی شد، انتخاب رنگ 1 برای v_2 غیرامیدبخش است، زیرا v_1 مجاور با v_2 است. به‌طور مشابه پس از آن‌که v_1 ، v_2 و v_3 به ترتیب با رنگ‌های 1 ، 2 و 3 رنگ‌آمیزی شدند، انتخاب رنگ 1 برای v_4 غیرامیدبخش می‌شود زیرا v_1 مجاور v_4 است.



شکل ۱۲ : بخشی از درخت فضای حالت هرس شده حاصل از اعمال روش پسگرد برای رنگ آمیزی گراف شکل 10 با 3 رنگ. نخستین جواب در گره سایه‌دار پیدا شده است. هر گره غیرامیدبخش با علامت × مشخص شده است.

تعداد گره‌ها در درخت فضای حالت برابر است با

$$1+m+m^2+.....+m^n=\frac{m^{n+1}-1}{m-1}$$

برای یک m و n مفروض، می‌توان نمونه‌ای یافت که تعداد گره‌های بررسی شده در آن، حداقل نمایی (در حسب n) باشد. برای مثال، برای $m = 2$ و گرافی که در آن راس v_n به همهٔ ریوس دیگر گراف وصل شده و تنها یال دیگر گراف، یال میان v_{n-2} و v_{n-1} باشد، هیچ جوابی وجود ندارد، با وجود این، برای رسیدن به این نتیجه، تقریباً همه گره‌های درخت فضای حالت ملاقات می‌شوند.

مساله کوله‌پشتی 0/1

در فصل قبل این مساله را با استفاده از روش برنامه‌نویسی پویا حل کردیم. در این فصل، آن را با استفاده از روش پسگرد حل می‌کنیم. به خاطر دارید که در این مساله مجموعه‌ای از n بسته که هر یک دارای وزن و ارزش معینی است، مفروض است. وزن‌ها و ارزش‌ها اعدادی صحیح و مثبت هستند، هدف قرار دادن زیر مجموعه‌ای از این بسته‌ها در یک کوله‌پشتی به ظرفیت M است به گونه‌ای که اولاً، مجموع وزن‌های بسته‌های قرار داده شده در کوله‌پشتی از M بیشتر نشود و دوماً مجموع ارزش این بسته‌های انتخاب شده ماکزیمم بشود.

این مساله را می‌توان با استفاده از یک درخت فضای حالت، دقیقاً مشابه درخت به‌کار رفته در مساله حاصل جمع زیر مجموعه‌ها حل کرد. یعنی، رفتن از ریشه به طرف چپ مفهومش این خواهد بود که اولین بسته در کوله‌پشتی قرار داده می‌شود، ولی رفتن به سمت راست این معنی را خواهد داشت که این بسته در کوله‌پشتی قرار داده نمی‌شود. به طریق مشابه، اگر از یک گره واقع در سطح 1 ، به سمت چپ برویم، یعنی بستهٔ دوم را انتخاب می‌کنیم و اگر به سمت راست برویم، یعنی آن‌را انتخاب نمی‌کنیم و الی آخر. هر مسیر از ریشه به یک برگ، یک جواب ممکن به شمار می‌رود.

چون این مساله، یک مساله بهینه‌سازی است، از این لحاظ با دیگر مسایل مورد بحث در این فصل تفاوت دارد. یعنی تا جستجو به پایان نرسد، نمی‌توان دریافت که آیا گرهی حاوی یک جواب هست یا خیر. بنابراین شیوهٔ پسگرد اندکی متفاوت است. اگر مجموع ارزش بسته‌های انتخاب شده تا یک گره بیشتر از بهترین ارزش یافته شده تا آن گره باشد، مقدار بهترین ارزش را تغییر می‌دهیم. ولی، هنوز ممکن است در یکی از نواده‌های آن گره، ارزش بهتری، با انتخاب بسته‌های دیگر، پیدا شود. بنابراین در مسایل بهینه‌سازی، فرزندان گره امیدبخش، همواره ملاقات خواهند شد. الگوریتم زیر،یک الگوریتم عمومی پسگرد برای مسایل بهینه‌سازی است.

```
Algorithm checknode ( v )
{
      if ( value ( v ) is better than the best ) then best = value ( v ) ;
           best = value ( v ) ;
             if promising ( v ) then
               for ( each child u of v) do
                 checknode ( u ) ;
{
```

متغیر best دارای بهترین ارزش حاصل تا گره v بوده و (value (v نیز مقدار جواب در آن گره است.

مقدار اولیه‌ای که به best منسوب می‌شود باید بدتر از هر مقدار برای یک جواب ممکن باشد. توجه کنید که یک گره در صورتی امیدبخش است که مجبور به گسترش فرزندانش باشیم. یادآوری می‌کنیم که در دیگر الگوریتم‌های پسگرد، اگر جوابی در یک گره موجود باشد آن گره را امیدبخش می‌دانند.

حال این روش را برای مساله کوله‌پشتی 0/1 به کار می‌بریم. نخست نگاه به علایمی داشته باشیم که حاکی از غیر امیدبخش بودن یک گره هستند. یک علامت آشکار برای غیر امیدبخش بودن یک گره این است که ظرفیتی برای بسته‌های بیشتر، در کوله‌پشتی باقی

نمانده باشد. بنابراین، اگر weight ، حاصل جمع وزن‌های بسته‌های انتخاب شده تا یک گره باشد، این گره در صورتی غیر امیدبخش است که:

$$\text{weight} \geq W$$

حتی اگر weight با M برابر باشد گره غیر امیدبخش است، زیرا که در مورد مسایل بهینه‌سازی، امیدبخش بودن یک گره به این معنی است که فرزندان آن باید بسط داده شوند.

از ملاحظات مربوط به روش حریصانه می‌توان برای یافتن یک علامت نسبتاً غیر بدیهی استفاده کرد. در این‌جا از این ملاحظات صرفاً برای محدود کردن جستجو استفاده می‌شود. ابتدا بسته‌ها را بر اساس مقدار $\frac{p_i}{w_i}$ به ترتیب نزولی مرتب می‌کنیم، w_i و p_i به ترتیب وزن و ارزش بسته i هستند. فرض کنید سعی در تعیین امیدبخش بودن یک گره خاص داریم. بسته‌های باقیمانده را به هر نحوی که انتخاب بکنیم قادر نخواهیم بود ارزش بیشتری نسبت به حالتی به دست آوریم که محدودیت 0 یا 1 بودن بسته‌های باقیمانده را نداشته باشیم (مجاز به انتخاب هر کسری از بسته‌های باقیمانده باشیم). بنابراین می‌توان یک کرانهٔ بالایی برای ارزشی که می‌شود از گسترش دادن به یک گره به دست آورد، به صورت زیر ارایه داد. فرض کنیم که proft مجموع ارزش بسته‌های انتخاب شده تا یک گره باشد. به‌خاطر دارید که weight حاصل جمع وزن‌های این بسته‌ها است. به متغیرهای bound و totweight به ترتیب مقادیر اولیه proft و weight را منسوب می‌کنیم. سپس به روش حریصانه بسته‌ها را یکی یکی برداشته، ارزش آن‌ها را به bound و وزن آن‌ها را به totweight اضافه می‌کنیم. تا این‌که به بسته‌ای برسیم که انتخاب آن منجر به نامساوی $\text{totweight} > M$ بشود. در این صورت آن کسر از وزن بستهٔ مزبور را که وزن باقیمانده اجازه می‌دهد، برداشته و ارزش مربوط به آن کسر را به bound اضافه می‌کنیم. هر چند، اگر قادر نباشیم این وزن آخر را به‌طور کامل برداریم، این گره نمی‌تواند به ارزشی مساوی با bound منجر شود ولی bound هنوز یک کرانهٔ بالایی برای ارزشی است که می‌توان از بسط آن گره به دست آورد. فرض کنید گره مورد اشاره، در سطح i قرار دارد و گره واقع در سطح k گرهی است که مجموع وزن‌ها را از M بیشتر می‌کند. در این صورت،

$$\text{totweight} = \text{weight} + \sum_{j=i+1}^{k-1} w_i$$

و

$$\text{bound} = \left(\text{profit} + \sum_{j=i+1}^{k-1} p_j\right) + (M - \text{totweight}) \times \frac{p_k}{w_k}$$

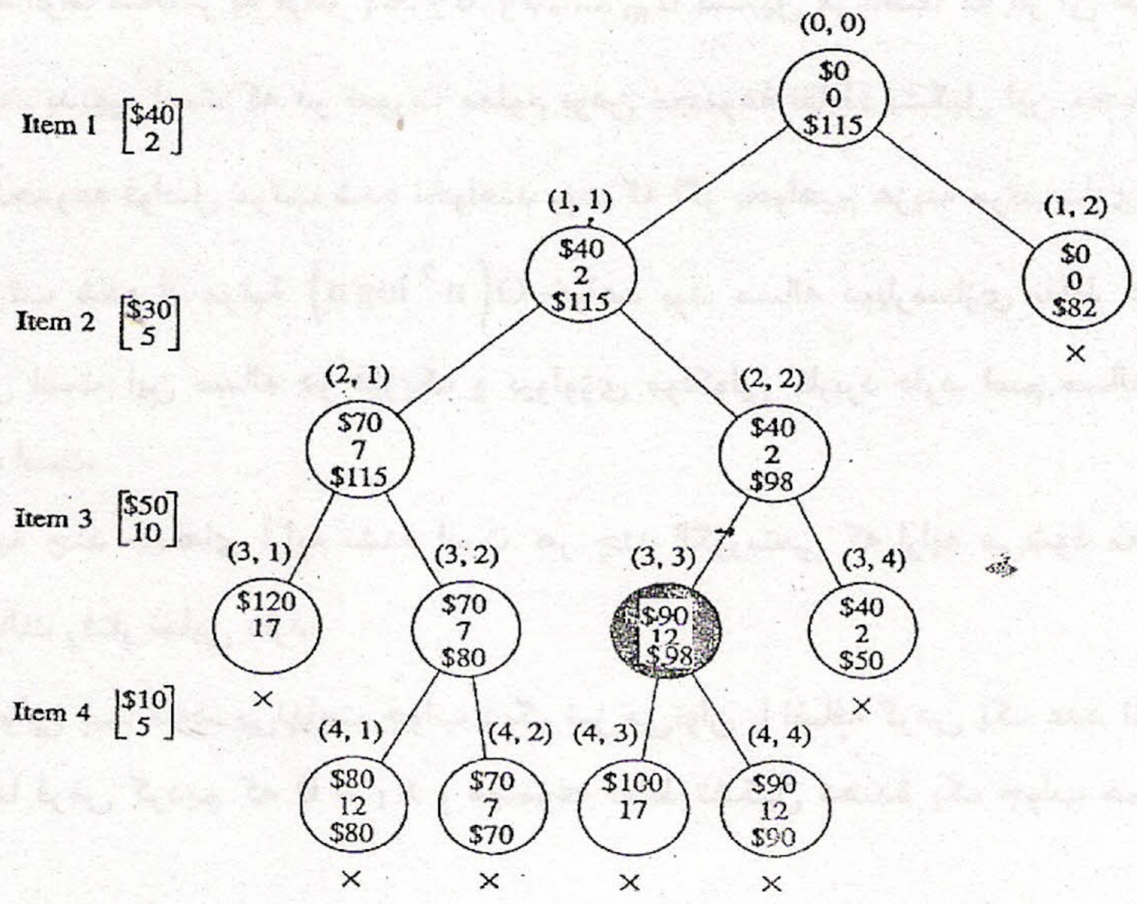
اگر maxprofit ارزش بهترین جوابی باشد که تاکنون پیدا شده است، در آن صورت یک گره در سطح i غیر امیدبخش است اگر:

$$\text{bound} \leq \max \text{profit}$$

مثال : فرض کنید $n = 4$ ، $M = 16$ و

i	p_i	w_i	p_i / w_i
1	40	2	20
2	30	5	6
3	50	10	5
4	10	5	2

بسته‌ها را قبلاً به ترتیب نزولی بر حسب (p_i / w_i) مرتب کرده‌ایم. برای سادگی، مقادیری از p_i و w_i انتخاب شده‌اند که حاصل p_i / w_i اعداد صحیحی باشند، مسلماً در حالت عمومی نیازی به این کار نیست. شکل 13 درخت فضای حالت هرس شده‌ای را نشان می‌دهد که از به کارگیری ملاحظات پسگرد مورد بحث به دست آمده است. ارزش کل، وزن و کرانهٔ بالایی برای هر گره از بالا به پایین در آن گره مشخص شده‌اند. این‌ها به ترتیب مقادیر متغیرهای profit , weight و bound مورد اشاره در قبل هستند. ارزش ماکزیمم در گره سایه‌دار پیدا شده است. هر گره با سطح و موقعیت آن از سمت چپ درخت بر چسب خورده است. برای مثال گره سایه‌دار به صورت (3 , 3) برچسب خورده است، زیرا که در سطح 3 قرار داشته و سومین گره از سمت چپ در آن سطح از درخت ا ست.



شکل 13 درخت فضای حالت هرس شده‌ای که از به‌کارگیری روش پسگرد در مثال قبل حاصل شده است.

در هر گره از بالا به پایین، ارزش کل بسته‌های انتخاب شده تا آن گره، وزن کل آن‌ها و کرانهٔ بالایی ارزش کل قابل حصول توسط گسترش آن گره نوشته شده است. جواب بهینه در گره سایه‌دار پیدا شده است. هر گره غیر امیدبخش با علامت × مشخص شده است.

مقدار کرانهٔ بالایی مادامی‌که در درخت فضای حالت به سمت چپ پیشروی می‌کنیم، تغییر نخواهد کرد. مگر این‌که به یک گره در سطح k برسیم. بنابراین هر بار که مقداری برای k معین شد، می‌توانیم مقدار آن را ذخیره کرده و با بررسی امیدبخش بودن به پیش برویم تا به گرهی در سطح $(k - 1)$ ام برسیم. می‌دانیم که فرزند چپ این گره غیر امیدبخش است، زیرا انتخاب بستهٔ k ، مقدار weight را بیشتر از M خواهد کرد. بنابراین از این گره فقط به سمت راست می‌رویم. فقط پس از حرکت به طرف راست است که نیاز به بررسی امیدبخش بودن و تعیین مقدر جدیدی برای k داریم.

درخت فضای حالت برای مساله کوله‌پشتی 0/1 همانند مساله حاصل جمع زیر مجموعه‌ها است. تعداد گره‌های آن درخت برابر

$$2^{n+1}-1$$

است. برای نمونه در مساله زیر، همه گره‌های موجود در درخت فضای حالت بررسی خواهند شد.

برای یک n مفروض فرض کنید $W = n$ و

$$P_i = 1 \quad w_i = 1 \qquad 1 \le i \le n-1$$
$$P_n = n \quad w_n = n$$

بدیهی است که جواب بهینه برای این نمونه مساله فقط انتخاب بسته n ام است و این جواب پیدا نخواهد شد، مگر آن‌که همهٔ راه‌ها را به طرف راست تا عمق n -1 طی کرده و سپس به طرف چپ برویم. به عبارت دیگر، قبل از یافتن جواب بهینه، هر گره غیر برگ، امیدبخش تشخیص داده شده و بنابراین همه گره‌های درخت فضای حالت بررسی خواهند شد.

مساله دوباره‌سازی نقاط عوارض

در این‌جا به یک مساله از هندسهٔ محاسباتی می‌پردازیم. n نقطهٔ $x_1, x_2, ..., x_n$ که در آن $0 = x_1 < x_2 < ..... < x_n$، مفروض هستند. این n نقطه، $m = \frac{n(n-1)}{2}$ فاصله (نه الزاماً منحصر به فرد) $d_1, d_2, d_3, ..., d_m$ تشکیل می‌دهند که در آن هر d_k فاصله یک زوج نقطه به صورت $|x_i - x_j|$, $i \neq j$ است . بدیهی است که در صورت معلوم بودن مجموعهٔ نقاط، تشکیل این مجموعه فواصل در زمان $O(n^2)$ امکان پذیر است. البته این مجموعه فواصل مرتب شده نخواهند بود. که اگر بخواهیم هزینه مرتب‌سازی را هم منظور کنیم، زمان تشکیل این مجموعه فواصل مرتب شده از مرتبهٔ $O(n^2 \log n)$ خواهد بود. مساله دوباره‌سازی نقاط عوارضی، دوباره‌سازی مجموعه نقاط از روی مجموعه فواصل است. این مساله در فیزیک و بیولوژی مولکولی کاربرد دارد. اسم مساله هم از ایستگاه‌های عوارضی واقع در یک اتوبان گرفته شده است.

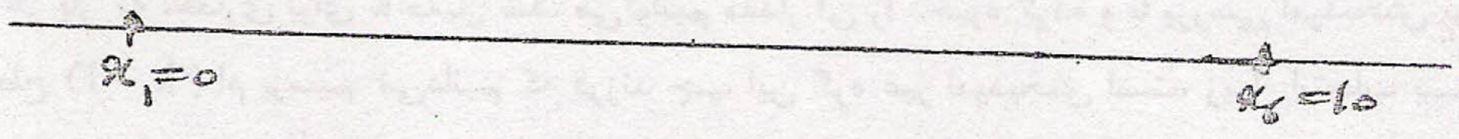
برای این مساله نیز الگوریتمی از مرتبهٔ چند جمله‌ای ارایه نشده است. هر چند الگوریتمی که ارایه می‌شود معمولا از مرتبه زمانی $O(n^2 \log n)$ است ولی در بدترین حالت رفتار نمایی دارد.

بدیهی است که برای این مساله اگر جوابی پیدا شود، بی‌نهایت جواب دیگر نیز می‌توان با اضافه کردن یک عدد ثابت به همه x_i ها به دست آورد. به همین دلیل است که ما فرض کردیم که $x_1 = 0$ و مجموعه نقاط تشکیل دهندهٔ یک جواب هم، به صورت صعودی مرتب شده‌اند.

فرض کنید که D مجموعه فواصل و $|D| = m = \frac{n(n-1)}{2}$. به عنوان یک مثال فرض کنید که

$$D = \{1,2,2,2,3,3,3,4,5,5,5,6,7,8,10\}$$

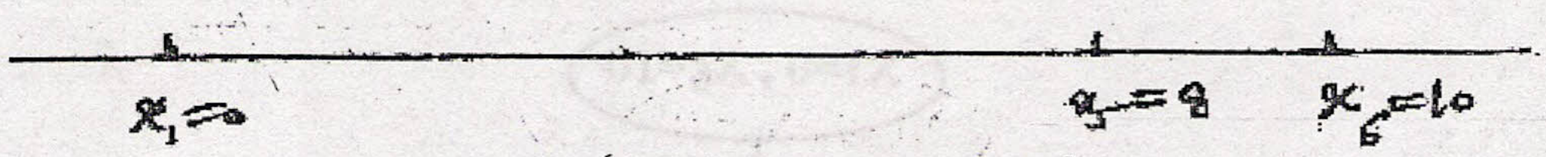
چون $|D| = 15$، لذا $m = 6$ نقطه داریم . الگوریتم را با قرار دادن $x_1 = 0$ شروع می‌کنیم. بدیهی است که در این صورت $x_6 = 10$ خواهد بود، زیرا که 10 بزرگ‌ترین عنصر D است. 10 را از مجموعه D حذف می‌کنیم. نقاط قرار داده شده و مجموعه فواصل باقیمانده برای بقیهٔ نقاط در شکل زیر به تصویر کشیده شده است:



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

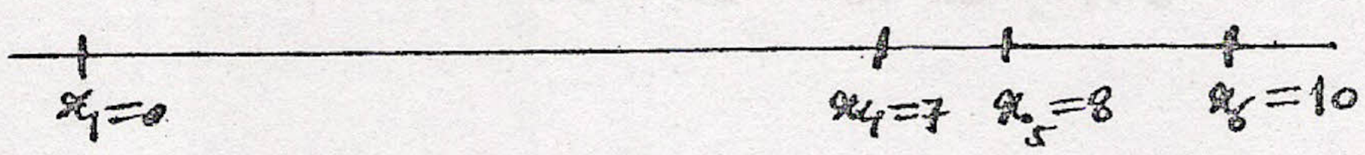
بزرگ‌ترین فاصله باقی‌مانده برابر 8 است این بدین معنی است که یا $x_2 = 2$ یا $x_5 = 8$. با توجه به متقارن بودن نقاط، می‌توان نتیجه گرفت که انتخاب مهم نیست، زیرا که یا هر دو انتخاب منجر به جواب خواهند شد (یکی تصویر آینه‌ای دیگری) یا هیچ‌کدام از این انتخاب‌ها منجر به جواب نخواهد شد.

بنابراین بدون لطمه زدن به جواب، فرض می‌کنیم که اگر $x_5 = 8$ با این انتخاب، فواصل $x_6 - x_5 = 2$ و $x_5 - x_1 = 8$ را از D حذف می‌کنیم، نتیجه به صورت زیر خواهد بود:



$$D=\{1,2,2,3,3,3,4,5,5,5,6,7\}$$

مرحلهٔ بعد الگوریتم خیلی بدیهی نیست. چون 7 بزرگ‌ترین مقدار مجموعه D است، بنابراین یا $x_4=7$ یا $x_2=3$. اگر $x_4=7$، در این صورت فواصل $x_6-7=3$ ، $x_5-7=1$ باید در مجموعهٔ D باشند با نگاهی سریع به مجموعه D بالا مشاهده می‌شود که این دو فاصله در آن قرار دارند. از طرف دیگر اگر $x_2=3$ در این صورت $3-x_1=3$ ، $x_5-3=5$ نیز باید در D باشند، که هستند. بنابراین در این‌جا، هیچ راهنمایی مبنی بر این‌که کدام یک را انتخاب کنیم، نداریم. بنابراین، یکی را برای مشاهدهٔ این‌که آیا منجر به جوابی می‌شود یا نه، آزمایش می‌کنیم. اگر به این نتیجه رسیدیم که این انتخاب منجر به نتیجه نخواهد شد، عمل پسگرد انجام داده و نقطه دوم را انتخاب خواهیم کرد. فعلاً انتخاب اول را آزمایش می‌کنیم. داشتیم $x_4 = 7$ که منجر به نتیجه زیر خواهد بود:

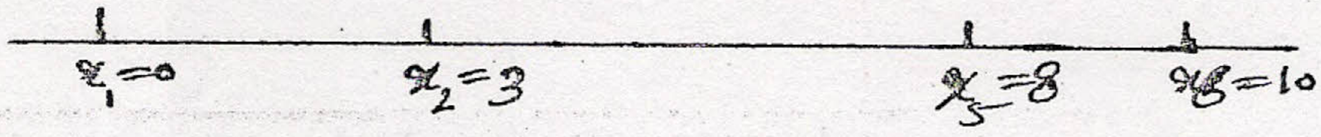


$$D=\{2,2,3,3,4,5,5,5,6\}$$

در این نقطه، داریم $x_1 = 0$ ، $x_4 = 7$ ، $x_5 = 8$ و $x_6 = 10$. حال بزرگ‌ترین فاصله برابر 6 است، لذا یا $x_3 = 6$ یا $x_2 = 1$. اما اگر $x_3 = 6$ انتخاب شود، آن‌گاه $x_4-x_3=1$ که ناممکن است زیرا 1 دیگر در مجموعه D نیست. از طرف دیگر، اگر $x_2 = 4$ انتخاب شود، آنگاه $x_2-x_0=4$ ، $x_5-x_2=4$ که این نیز امکان ندارد، زیرا که عدد 4 در مجموعه D فقط یک بار ظاهر شده است.

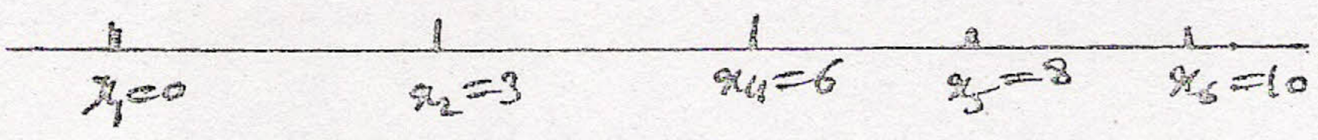
بنابراین چون این راه استدلال منجر به هیچ نتیجه‌ای نمی‌شود، پسگرد می‌کنیم.

چون انتخاب $x_4 = 7$ منجر به جوابی نشد، $x_2 = 3$ را می‌آزماییم. اگر این انتخاب نیز منجر به جوابی نشود، نتیجه خواهیم گرفت که این مساله جواب ندارد. اما، فعلاً داریم:



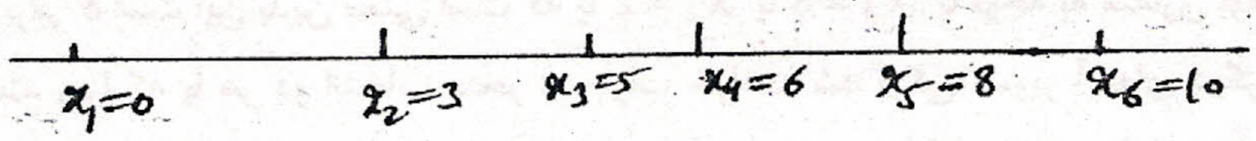
$$D=\{1,2,2,3,3,5,5,6\}$$

یک‌بار دیگر، یکی از دو انتخاب $x_4=6$ یا $x_3=4$ را باید انجام دهیم. انتخاب $x_3 = 4$ ناممکن است، زیرا در مجموعهٔ D فقط یک عدد 4 داریم در صورتی‌که $x_3-x_2=4$ ، $x_5-x_3=4$، هر دو باید در D باشند. انتخاب $x_4 = 6$ امکان دارد و منجر به نتیجه زیر می‌شود:



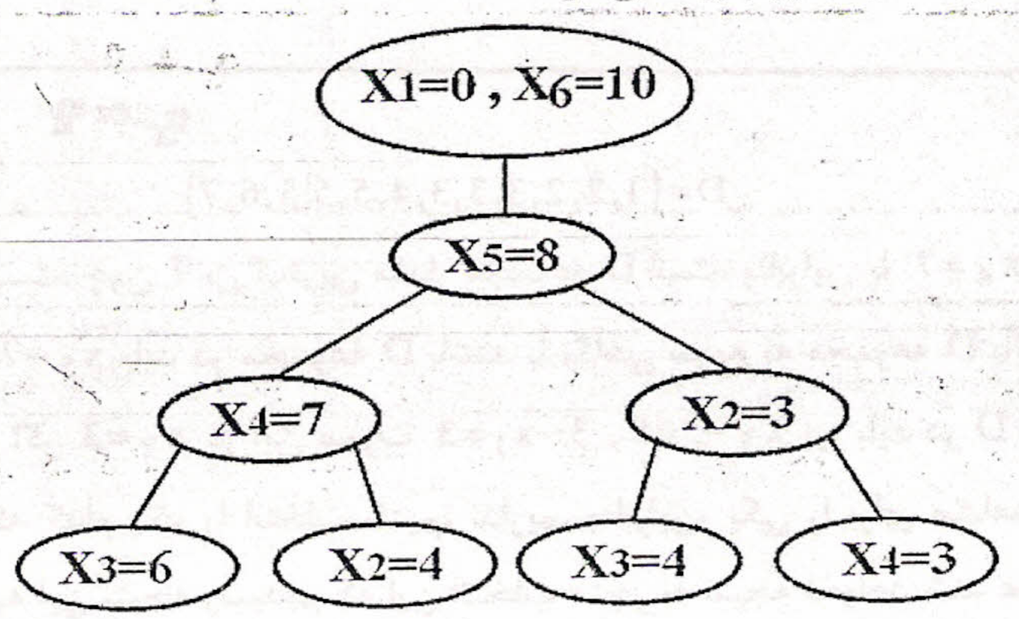
$$D=\{1,2,3,5,5\}$$

تنها انتخاب باقیمانده $x_3 = 5$ است، و این انتخاب درست عمل می‌کند، زیرا که منجر به مجموعهٔ تهی D و جواب زیر می‌شود:



D={ }

شکل 15 درخت فضای هرس شده را برای رسیدن به این جواب نشان می‌دهد. گره‌های غیر امیدبخش با علامت × مشخص شده‌اند. جواب مساله در مسیر از ریشه به گره‌ها سایه خورده پیدا می‌شود.



شکل 15 درخت فضای حالت هرس شده برای مثال دوباره‌سازی نقاط عوارضی