فصل هشتم

انشعاب و تحدید (Branch and Bound)

مقدمه

تاکنون دو الگوریتم برای حل مسئله کوله‌پشتی ارائه داده‌ایم. الگوریتم اول با روش برنامه‌نویسی پویا ارائه شد، و دومی را با روش عقبگرد طراحی کردیم. ولی هر دو این الگوریتمها در بدترین حالت زمانی خود از درجه نمایی بودند. در این فصل، یک روش دیگر برای حل این مسئله بنام انشعاب و تحدید ارائه می‌دهیم. همانطور که ملاحظه خواهید کرد این روش، شکل بهبود یافته‌ای از روش عقبگرد می‌باشد. و در مواقعی که حتی روش‌های قبلی برای حل مسائل قادر نباشند، این روش می‌تواند این مسائل را حل نماید.

روش انشعاب و تحدید یکی دیگر از روش‌های پیمایش و جستجو درختها و گراف‌هاست) فضای حالت مسأله‌ای که قرار است به روش انشعاب و تحدید حل گردد، باید با یک گراف قابل نمایش باشد.

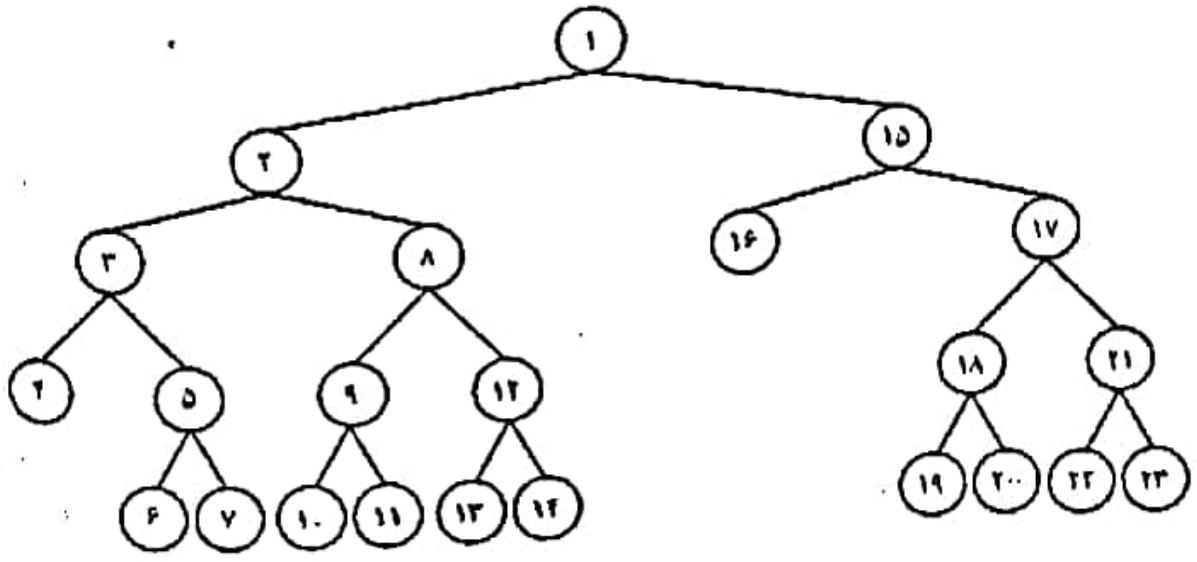
تا این فصل دانشجو با روش بازگشت به عقب که در فصل قبلی بررسی گردید آشنایی کامل دارد در این بخش یک روند مقایسه‌ای بین روشهای بازگشت به عقب و انشعاب و تحدید انجام می‌دهیم. تا تفاوت‌ها و شباهت‌های دو الگوریتم مشخص شود.

حال در اینجا نگاهی اجمالی به تفاوت‌ها و شباهت‌های دو روش طراحی می‌اندازیم تا موضوع بیشتر روشن شود:

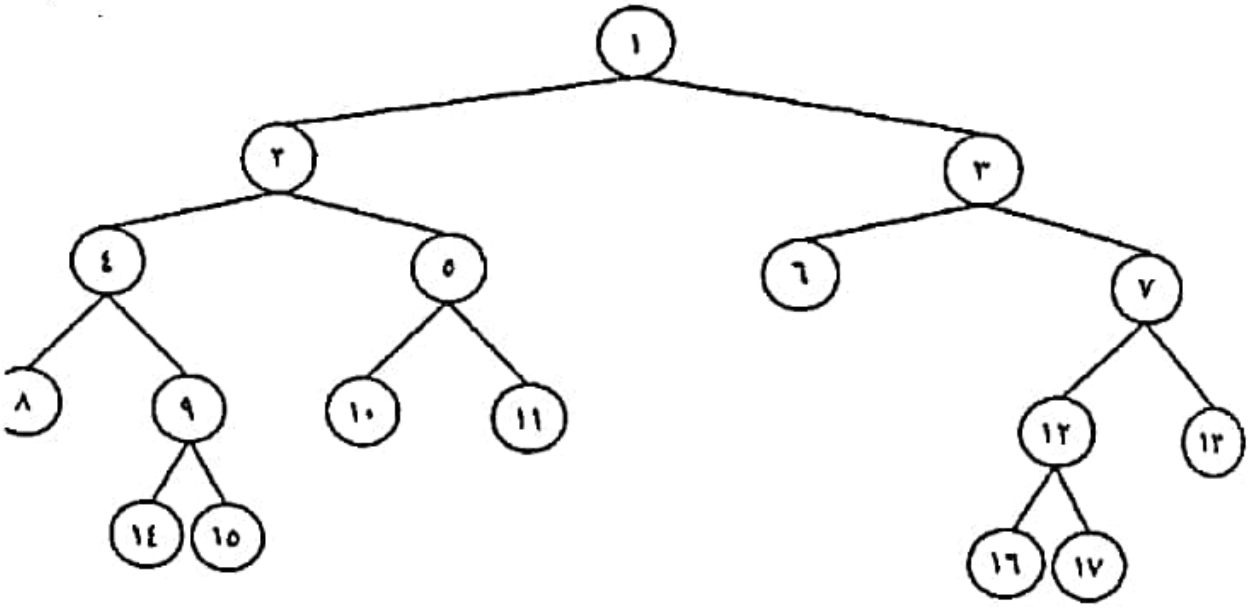
• روش پیمایش درخت(یا گراف)

اصولاً دو روش جستجوی اصلی برای پیمایش گرافها در حالت کلی وجود دارد. روش جستجو در پهنا یا جستجوی ردیفی (Breadth First Search) و جستجو در عمق یا جستجوی عمقی (Depth first Search) دو روش مذکور می‌باشند) الگوی جستجو برای روش بازگشت به عقب(عقبگرد) به صورت جستجو در عمق می‌باشد. همان‌گونه که قبلاً نیز گفته شد در این روش زیردرختهای هر گره یکی‌یکی تولید می‌شوند و تا بررسی کامل هر زیردرخت، زیردرخت دیگری ایجاد نمی‌شود. اما در روش انشعاب و تحدید یکی از روش‌های جستجوی درخت، جستجو به ترتیب پهنا می‌باشد. در این روش کلیه فرزندهای یک گره ایجاد می‌شوند و سپس هر یک از گره‌هایی که تازه ایجاد شده‌اند را به ترتیب ایجاد، برداشته و کلیه گره‌های فرزند آن را ایجاد می‌کنیم و این کار را تا آخر ادامه می‌دهیم. به این روش اصطلاحاً جستجوی FIFO (اولین ورودی ـ اولین خروجی) نیز می‌گویند)

شکل‌های الف. ۱-۸ و ب. ۱-۸ نحوه شماره‌گذاری گره‌های یک درخت برای جستجو، به روش ترتیب عمق (مناسب برای روش بازگشت به عقب) و جستجو به روش ترتیب پهنا (مناسب برای روش انشعاب و تحدید) را نشان می‌دهد.



(الف)



(ب)

شکل ۱-۸: الف) پیمایش درخت به صورت عمق ب) پیمایش درخت به صورت پهنا

• **هرس‌کردن شاخه‌ها**

در هر دو روش بازگشت به عقب و انشعاب و تحدید سعی می‌شود شاخه‌هایی از درخت هرس شود، در اینصورت تعداد حالتهای ایجاد شده کاهش می‌یابد و در نتیجه زمان لازم برای اجرای الگوریتم کاهش می‌یابد. گرچه این کار مرتبه زمانی را کاهش نمی‌دهد، اما برای تعداد داده‌های کم زمان اجرا را به حد قابل قبولی کاهش می‌دهد. در روش بازگشت به عقب امکان تغییر ترتیب بررسی گره‌ها پیش‌بینی نشده است و براساس روش ترتیب جستجو مشخص شده است. اما در روش انشعاب و تحدید امکان تغییر بررسی گره‌ها وجود دارد. در برخی از مسائل، با انجام محاسبات اضافی می‌توان میزان امید برای رسیدن به جواب را در هر شاخه برآورد کرد و سپس گره‌های ایجاد شده زنده را برحسب این برآورد مرتب کرد.

• مسأله‌ای که به روش بازگشت به عقب حل می‌گردد می‌تواند بیش از یک جواب داشته باشد و هیچ جوابی بر جواب دیگر امتیازی ندارد. در اغلب مسائلی که به روش انشعاب و تحدید حل می‌شوند مهم یافتن جواب بهینه است. برای مثال، مسأله n وزیر به روش بازگشت به عقب حل می‌شود، اما مسأله فروشنده دوره‌گرد که در آن هدف یافتن توری کامل با حداقل طول می‌باشد به روش انشعاب و تحدید حل

می‌گردد.

• همانند الگوریتم عقبگرد، زمان الگوریتمهای انشعاب و تحدید نیز معمولاً در بدترین حالت زمانی نمایی (یا بدتر) می‌باشد.

۸-۱ حل مسأله فروشنده دوره‌گرد با استفاده از روش انشعاب و تحدید

فرض کنید G یک گراف جهت‌دار وزن‌دار با n گره بوده و C_{ij} وزن ارتباطی i به j باشد. که در آن $C_{ij} \geq 0$ به ازاء کلیه مقادیر $i \neq j$ بوده و $C_{ii} = 0$ است و اگر میان دو گره لبه‌ای نباشد $C_{ij} = \infty$ می‌باشد. هدف این مساله همان‌طور که قبلا اشاره کردیم یافتن یک دور کامل است به گونه‌ای که از یکی از گره‌ها آغاز کرده و هر گره را تنها یکبار ملاقات کنیم و به گره اولیه برگردیم. این مسأله به روش برنامه‌نویسی پویا حل شد که دارای مرتبه زمانی $O(n^2 2^n)$ بود. حال آنرا با روش انشعاب و تحدید حل می‌کنیم و سعی می‌شود که با ارائه یک تابع حد مناسب زمان اجرای این الگوریتم برای بعضی حالات بهبود یابد. البته مرتبه زمانی بهتری حاصل نخواهد شد(قبلا این نکته را در فصل ۷ بحث کردیم).

فرض کنید ما می‌خواهیم مسیری روی گراف داده شده G پیدا کنیم که ماتریس وزن گراف G به‌صورت زیر باشد:

شماره	۱	۲	۳	٤	٥
۱	∞	۱٤	٤	۱۰	۲۰
۲	۱٤	∞	۷	۸	۷
۳	٤	٥	∞	۷	۱٦
٤	۱۱	۷	۹	∞	۲
٥	۱۸	۷	۱۷	٤	∞

شکل ۲-۸: ماتریس وزن یک گراف جهت‌دار

برای این ماتریس می‌خواهیم یک مسیر با کمترین هزینه (یا طول یا ...) پیدا کنیم

چون مسیر باید کامل باشد، باید همه گره‌ها را در برگیرد، پس می‌توان هر یک از گره‌ها را به عنوان نقطه شروع عملیات انتخاب کرد.

فرض کنید از گره شماره ۱ شروع کنیم. به خاطر داشته باشید که، در یک دور به هر یک از گره‌ها تنها یکبار وارد و تنها یکبار از آنها خارج می‌شویم. برای هر گره باید مقدار تابع حد را محاسبه کنیم. برای محاسبه آن بصورت زیر عمل می‌کنیم.

تعیین تابع حد: برای هر گره کمترین ارزش لبه ورودی و کمترین ارزش لبه خروجی را با هم جمع می‌کنیم سپس حاصل جمع برای کلیه گره‌ها را بدست آورده و با هم جمع می‌کنیم. حاصل جمع بدست آمده دو برابر کمترین ارزش ممکن برای یک مسیر خواهد بود. توجه کنید که، دلیل دو برابر بودن کمترین ارزش این است که، یک لبه ضمن خارج شدن از یک گره به گره دیگری وارد می‌شود.

برای پیداکردن کمترین ارزش ممکن برای یک مسیر، حاصل جمع بدست آمده را تقسیم بر ۲ می‌کنیم.

کمترین ارزش لبه خروجی از یک گره دلخواه i متناظر با مینیمم ارزش‌های موجود در ردیف iام می‌باشد. و کمترین ارزش لبه ورودی به یک گره دلخواه j متناظر با مینیمم ارزش‌های موجود در ستون jام می‌باشد. به عنوان مثال، کمترین ارزش لبه خروجی از گره ۱ حداقل مقادیر ۱٤، ٤، ۱۰ و ۲۰ می‌باشد و کمترین ارزش لبه ورودی به گره حداقل، مقادیر ۱٤، ٤، ۱۱ و ۱۸ یعنی ٤ می‌باشد.

شماره گره	min خروجی	min ورودی	min ورودی + min خروجی
۱	٤	٤	۸
۲	۷	٥	۱۲
۳	٤	٤	۸
٤	٤	۲	٦
٥	۲	٤	٦

بنابراین در ابتدای کار فرض می‌شود که ارزش مسیر ۲۰ باشد. حال از گره ۱ می‌توان به گره‌های ۲، ۳، ٤ یا ٥ رفت (با توجه به ماتریس وزن یا ارزش). به عنوان مثال فرض کنید به گره شماره ۲ برویم. بنابراین ارزش خروجی از گره ۱ و همچنین ارزش ورودی به گره ۲ برابر ١٤ خواهد بود. با توجه به انتخاب لبه(۲و۱) برای محاسبه کمترین ارزش ممکن باید به نکات زیر توجه کرد:

۱. لبه (۲و۱) بصورت قطعی انتخاب شده است. ارزش خروجی از گره ۱ و ورودی به گره ۲ هر دو برابر ١٤ است.

۲. خروجی از گره ۲ نمی‌تواند ورودی به گره ۱ باشد پس کمترین ارزش خروجی از ۲ برابر ۷ خواهد بود.

۳. ورودی به هیچ گره (جز ۲) نمی‌تواند از گره ۱ باشد پس کمترین ارزش ورودی مثلاً به گره ۳ برابر ۷ خواهد بود.

٤. خروجی از هیچ گره دیگری نمی‌تواند به گره ۲ باشد، چرا که ایجاد یک زیرمسیر خواهد نمود.

با رعایت موارد فوق کمترین ارزش یک مسیر پس از انتخاب قطعی لبه (۲و۱) برابر خواهد بود با:

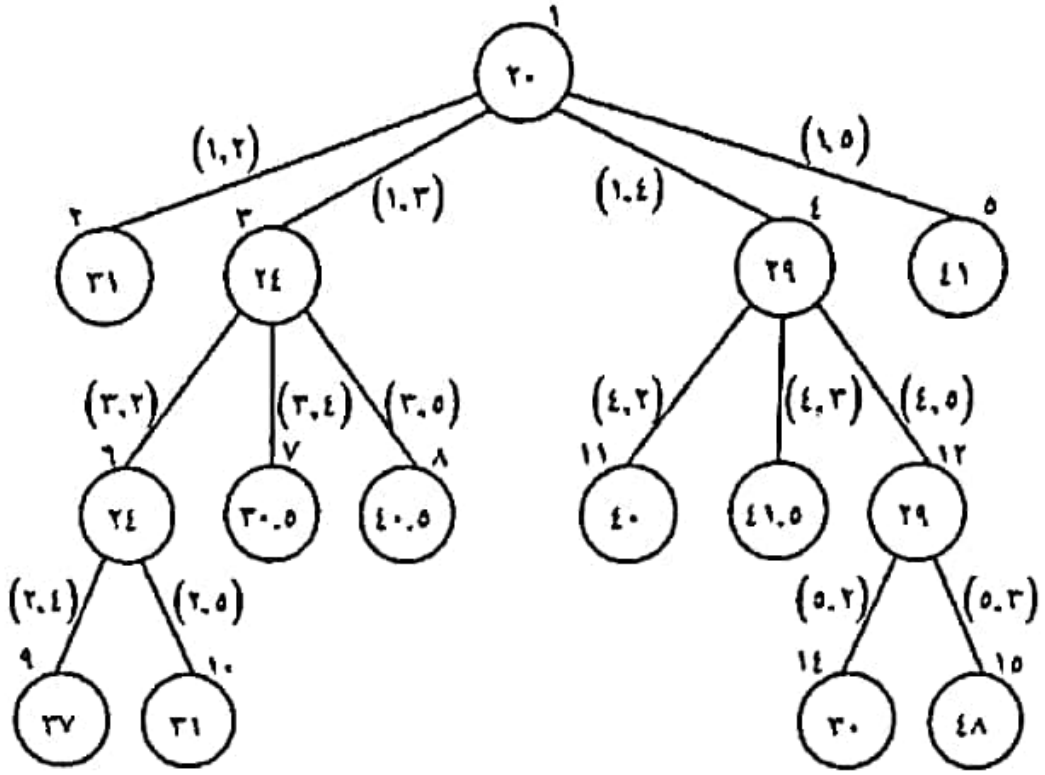
چون لبه (۲و۱) به صورت قطعی انتخاب شده است پس مینیمم خروجی از گره ۱ که به گره ۲ وارد می‌شود ١٤ می‌باشد.

شماره گره	min خروجی	min ورودی	Min ورودی + min خروجی
۲	۷	١٤	۲۱
۳	٤	۷	۱۱
٤	۲	٤	٦
٥	٤	۲	٦
			٤٤

پس ارزش مسیر با انتخاب قطعی (۲و۱) بصورت زیر می باشد:

$$(١٨+٤٤)/٢ = ٣١$$

به شکل ۳-۸ دقت کنید:



شکل ۳-۸: درخت فضای حالت

در این شکل، گره شماره ۲ دارای ارزش ۳۱ می‌باشد. حال از بین گره‌های موجود در شکل، که توسعه نیافته‌اند گره‌ای را که دارای کمترین ارزش است ادامه می‌دهیم تا جایی که به گره‌های ۹ و ۱۰ برسیم. با توجه به مسیر کامل با ارزش ۳۱ در گره ۱۰، دیگر لزومی به توسعه گره‌های توسعه‌نیافته‌ای که کمترین ارزش آنها از ۳۱ بیشتر است، نیست. مثلاً گره‌های ۵، ۲، ۸، ۹ و ۷ را در نظر بگیرید. با توجه به این گره‌ها، تنها گره ٤ را می‌توان بسط داد و بعد گره ۱۳، که در نهایت مسیرهای جدید حاصل می‌شود که ارزش آنها از همه گره‌های توسعه‌یافته کامل و توسعه‌نیافته کمتر می‌باشد. بنابراین گره ۱٤ جواب اصلی مسئله می‌باشد. به این ترتیب ارزش کمترین مسیر متعلق به مسیر۱، ٤، ٥، ۲، ۳، ۱ است (از راست به چپ).

۲-۸ مسأله کوله‌پشتی صفر و یک

الگوریتم عقب‌گرد ارائه شده برای حل مسأله کوله‌پشتی صفر و یک در فصل ۷، در واقع یک الگوریتم انشعاب و تحدید می‌باشد. که در آن تابع امیدبخش مقدار False را زمانی برمی‌گرداند که، مقدار حد کوچکتر از مقدار maxprofit باشد. به بیان دیگر، یک

گره غیرامیدبخش می‌باشد هرگاه داشته باشیم:

bound ≤ max profit

با توجه به مباحث قبلی و مقایسه دو روش عقبگرد و انشعاب و تحدید می‌دانیم که، در روش انشعاب و تحدید علاوه بر استفاده از حد برای تعیین اینکه گرهی امیدبخش است یا خیر، می‌توان حدهای گره‌های امیدبخش را نیز مقایسه کرد و فرزندان گرهی با بهترین حد را گسترش داد. بدین ترتیب سریعتر می‌توان به یک حل بهینه دست پیدا کرد. این روش بهترین جستجو با هرس کردن انشعاب و تحدید می‌باشد.

همان‌گونه که در ابتدای فصل اشاره گردید روش انشعاب و تحدید به جای عمقی به صورت عرضی درخت را مورد پیمایش قرار می‌دهد. لذا،قبل از ادامه بحث روش عرضی پیمایش گراف را مرور می‌کنیم. در یک درخت، جستجوی عرضی از ریشه آغاز می‌شود و سپس تمام گره‌های موجود در سطح ۱، سپس همه گره‌های سطح ۲ و غیرپیمایش می‌شود.

برخلاف جستجوی عمقی، هیچ الگوریتم بازگشتی ساده‌ای برای جستجوی عرضی یا جستجو در پهنا وجود ندارد. ولی می‌توان آن را با استفاده از یک صف پیاده‌سازی کرد. الگوریتم زیر این کار را فقط بر روی درختها انجام می‌دهد. در این الگوریتم روالی بنام enqueue عنصری در انتهای صف قرار می‌دهد و dequeue عنصر جلوی صف را حذف می‌کند.

```
void  breadth_first_tree_search ( tree  T )
{
     Queue_of_node Q ;
     node  U ,V ;
     V = root of T ;
     visit V ;
     enqueue ( Q ,V ) ;
     while ( !empty( Q ) )
     {
     dequeue ( Q , V ) ;
     for ( each child  U  of  V )
     {
          visit U ;
```

```
            enqueue (Q,U) ;
        }
    }
}
```

با اندکی تغییر در روال فوق می‌توان کارایی جستجوی عرضی را بهبود بخشید. به جای استفاده از یک صف از صف تقدم (یا صف اولویت‌دار) استفاده می‌کنیم. در صف اولویت، عنصری که دارای بالاترین تقدم است همواره حذف می‌شود. در کاربردهای بهترین جستجو، عنصری که دارای بالاترین تقدم است، گرهی با بهترین حد است. یک صف اولویت را می‌توان به‌صورت یک لیست پیوندی پیاده‌سازی کرد. الگوریتم بیان‌شده در زیر یک الگوریتم کلی برای روش بهترین جستجو(با هرس کردن انشعاب و تحدید) می‌باشد. در این الگوریتم insert (PQ,V) روالی است که V را به صف اولویت PQ می‌افزاید و remove (PQ,V) روالی است که گرهی با بهترین حد را حذف کرده، مقدار آن را به V نسبت می‌دهد.

```
void   best_first_branch_and_bound (State_Space_tree T,
                                    number  & best )
{
      priority - queue - of - node  PQ ;
      node U,V ;
      initialze (PQ) ;
      V= root of T ;
      best = value (V) ;
      insert (PQ,V) ;
      while (! Empty (PQ) )
      {
         remove(PQ,V) ;
         if ( bound (V)  is  better than  best )
           for ( each child U of V )
           {
              if (value (U) is better than best)
                 best = value(U);
              if (bound (U) is better than best)
                 insert (PQ,U) ;
           }
      }
}
```

در روال فوق پس از حذف گرهی از صف اولویت، شرطی اضافه شده است که تعیین می‌کند آیا حد مربوط به گره هنوز بهتر از best است یا خیر. به این ترتیب می‌توان تعیین کرد که، گرهی پس از ملاقات شدن، غیرامیدبخش است یا خیر.

۸-۲-۱ حل مسأله کوله‌پشتی صفر و یک با استفاده از روش انشعاب و تحدید

به طور کلی، روش جستجو عرضی مزیتی بر جستجوی عمقی (عقبگرد) ندارد. ولی می‌توانیم جستجوی خود را با استفاده از حد (bound) بهبود دهیم تا علاوه بر تعیین امیدبخش بودن یک گره، کار دیگری هم انجام دهد. لذا برای تحقق چنین امری، پس از ملاقات همه فرزندان گره مفروضی، می‌توانیم همه گره‌های امیدبخش و گسترش‌نیافته را مورد بررسی قرار دهیم و آنی را که دارای بهترین حد است گسترش دهیم. از روش عقبگرد به خاطر دارید که یک گره در صورتی امیدبخش است که حد آن بهتر از مقدار بهترین حل پیدا شده تا آن لحظه باشد. بنابراین همانطور که ملاحظه خواهید کرد، نسبت به حالتی که فرضاً کورکورانه در یک ترتیب از پیش تعیین شده عمل کرده و پیش می‌رفتیم، در این شیوه به مراتب سریعتر به حل بهینه خواهیم رسید. برای روشن شدن مطلب مثالی را ارائه می‌دهیم.

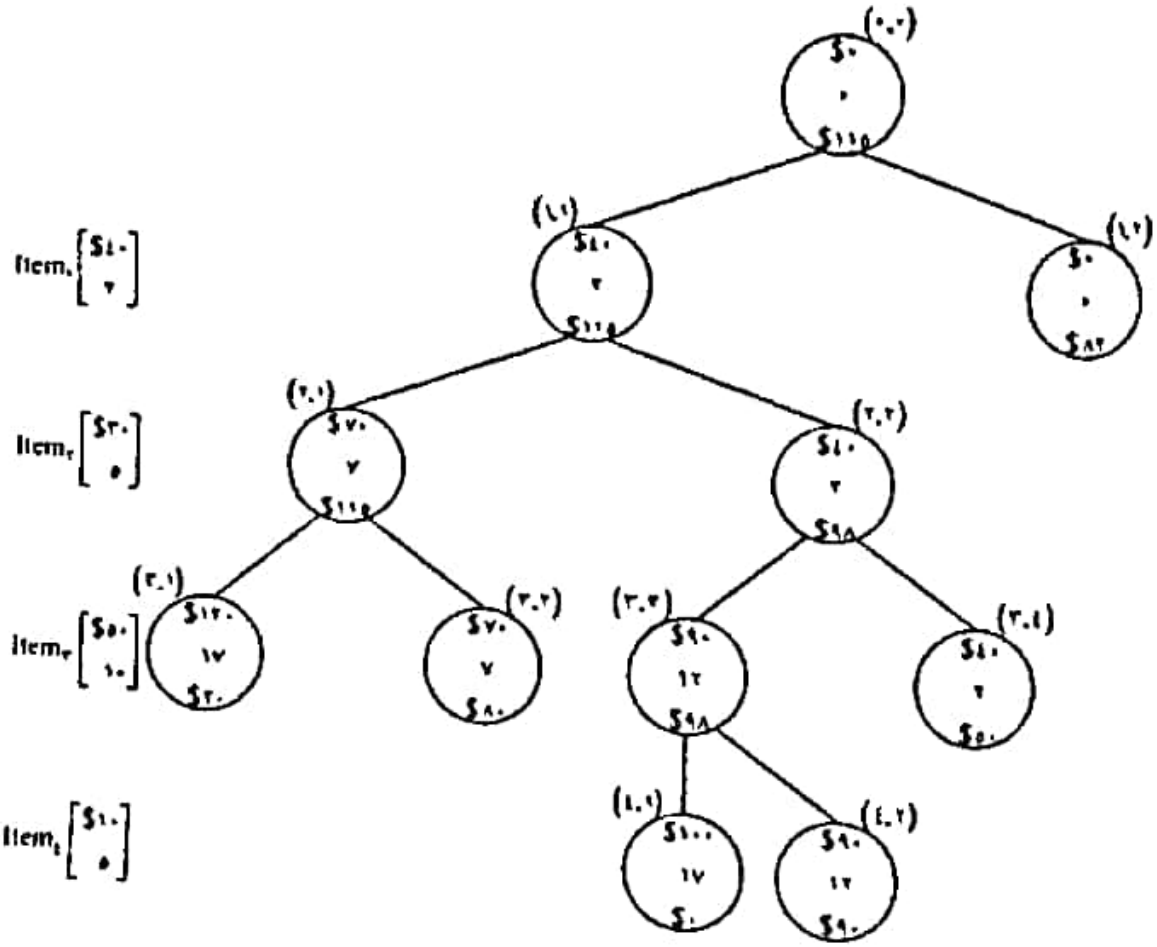
مثال ۸-۱: فرض کنید چهار قطعه با وزن‌ها و ارزشهای مشخص بصورت زیر داریم و حداکثر وزن تحملی کوله‌پشتی ۱۶ باشد یعنی:

$W=16$ و $n=4$

مشخصات قطعات در جدول زیر براساس P_i/W_i مرتب شده‌اند:

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

تمام تعریفهایی که در حل مسأله کوله‌پشتی در روش عقبگرد داشتیم، تمام آنها در روش انشعاب و تحدید نیز معتبر می‌باشد. و تنها تفاوتی که وجود دارد نحوه پیمایش درخت است. در الگوریتم عقبگرد، از روش جستجوی عمقی برای پیمایش درخت استفاده کردیم ولی در روش انشعاب و تحدید از روش بهترین جستجوی عرضی(هرس شده) استفاده می‌کنیم. درخت فضای حالت هرس‌شده که با استفاده از جستجوی عرضی بدست آمده است در شکل ٤-٨ نمایش داده شده است. گره سایه‌خورده گرهی است که حل بهینه در آن پیدا شده است.



شکل ٤-٨: درخت فضای حالت هرس‌شده با استفاده از بهترین جستجوی.

حال حل را مرحله به مرحله نمایش می‌دهیم:

۱. گره *(0,0) را ملاقات می‌کنیم:*

الف) ارزش و وزن آن را بترتیب برابر $0 و صفر قرار می‌دهیم.

ب) حد آن را برابر $115 قرار می‌دهیم.

ج) maxprofit = 0

۲. *گره (۱-۱) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را بترتیب برابر 40$ و 2 محاسبه می‌کنیم.

ب) چون وزن آن که ۲ است کوچکتر از ۱۶ می‌باشد و ارزش آن که 40$ است بزرگتر از 0$ (مقدار maxprofit) می‌باشد، بنابراین مقدار maxprofit را برابر 40$ قرار می‌دهیم.

ج) حد آن را برابر 115$ محاسبه می‌کنیم.

۳. *گره (۲-۱) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را بترتیب برابر 0$ و صفر محاسبه می‌کنیم.

ب) حد آن را برابر 82$ محاسبه می‌کنیم.

۴. *گره گسترش‌نیافته امیدبخش با بزرگترین حد را تعیین می‌کنیم.*

الف) چون گره (۱-۱) دارای حدی برابر 115$ و گره (۱،۲) دارای حد 32$ است پس گره (۱-۱)، گره با بزرگترین حد گسترش‌نیافته امیدبخش می‌باشد. حال فرزندان آن را ملاقات می‌کنیم.

۵. *گره (۱-۲) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را برابر 70$ و ۷ محاسبه می‌کنیم.

ب) چون وزن آن ۷ کوچکتر یا مساوی ۱۶ بوده و ارزش آن 70$بزرگتر از مقدار maxprofit یعنی 40$ است، بنابراین maxprofit را برابر 70$ قرار می‌دهیم.

۶. *گره (۲-۲) را ملاقات می‌کنیم.*

الف) ارزش و وزن ان را بترتیب برابر 40$ و ۲ محاسبه می‌کنیم.

ب) حد آن را برابر 98$ محاسبه می‌کنیم.

۷. *گره گسترش‌نیافته امیدبخش با بزرگترین حد را تعیین می‌کنیم.*

الف) گره (۱-۲)، گره مذکور می‌باشد. حال فرزندان آن را ملاقات می‌کنیم.

۸. *گره (۱-۳) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را بترتیب برابر 120$ و ۱۷ محاسبه می‌کنیم.

ب) وزن گره ۱۷ بوده که بزرگتر از ۱۶ می‌باشد پس گره غیرامیدبخش است.

۹. *گره (۲-۳) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را بترتیب برابر 70$ و ۷ محاسبه می‌کنیم.

ب) حد آن را برابر 80$ محاسبه می‌کنیم.

۱۰. *گره گسترش‌نیافته امیدبخش با بزرگترین قید را تعیین می‌کنیم.*

الف) گره (۲-۲) مذکورمی‌باشد. حال فرزندان آن را ملاقات می‌کنیم.

۱۱. *گره (۳-۳) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را بترتیب برابر 90$ و ۱۲ محاسبه می‌کنیم.

ب) چون وزن آن که ۱۲ می‌باشد کوچکتر یا مساوی ۱۶ است و ارزش آن 90$ بوده که بزرگتر از 70$ مقدار کنونی maxprofit است، بنابراین مقدار maxprofit را به 90$ تغییر می‌دهیم.

ج) در این نقطه گره‌های (۲-۱) و (۲-۳) غیرامیدبخش می‌باشند، زیرا حدهای آنها به ترتیب 82$ و 80$ می‌باشد که کوچکتر از 90$ مقدار کنونی maxprofit هستند.

۱۲. *گره (٤-۳) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را بترتیب برابر 40$ و ۲ محاسبه می‌کنیم.

ب) حد آن را برابر 50$ محاسبه می‌کنیم.

ج) چون حد آن 50$ بوده و کوچکتر از 90$ می‌باشد پس غیرامیدبخش است.

۱۳. *گره گسترش‌نیافته امیدبخش با بزرگترین حد را تعیین می‌کنیم.*

تنها گره گسترش‌نیافته امیدبخش (۳-۳) است. حال فرزندان این گره را ملاقات می‌کنیم.

۱٤. *گره (۱-٤) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را بترتیب برابر 100$ و ۱۷ محاسبه می‌کنیم.

ب) چون وزن گره که ۱۷ است بزرگتر از ۱٦ می‌باشد بنابراین گره غیرامیدبخش می‌باشد.

۱۵. *گره (۲-٤) را ملاقات می‌کنیم.*

الف) ارزش و وزن آن را بترتیب برابر 90$ و ۱۲ محاسبه می‌کنیم.

ب) حد آن را برابر 90$ محاسبه می‌کنیم.

ج) چون حد گره که 90$ بوده و کوچکتر مساوی مقدار maxprofit که 90$ می‌باشد بنابراین گره غیرامیدبخش است.

چون اکنون دیگر هیچ گره گسترش‌نیافته امیدبخش باقی نمانده است. کار تمام است.

با استفاده از روش انشعاب و تحدید فقط ۱۱ گره را چک کردیم که ۲ گره کمتر از تعداد گره‌های چک شده با استفاده از جستجوی عمقی است. البته در درختهای با فضای حالت بزرگتر تعداد گره‌های بسیار کمتری نسبت به الگوریتم عقبگرد برای پیداکردن روش بهینه چک می‌شود.

حال در زیر قطعه کد الگوریتم ذکر شده را ارائه می‌دهیم، که در آن هر گره درخت فضای حالت باید فیلدهای زیر را برای حفظ ونگهداری حد داشته باشد:

```
Struct node
{
     int level , profit , weight ;
     float  bound ;
}
```

الگوریتم ۱-۸ بهترین جستجو با هرس کردن انشعاب وتحدید برای مسئله کوله‌پشتی صفر و یک

مساله: n جنس یا قطعه با وزن و سود مشخص داده شده‌اند و W ظرفیت کوله‌پشتی. هدف انتخاب تعدادی جنس با بیشترین سود ممکن، بطوریکه مجموع وزن‌های آنها از W کمتر باشد.

ورودی: اعداد مثبت و صحیح n و W، آرایه‌های اعداد صحیح و مثبت w و p که هر کدام از یک تا n اندیس‌گذاری شده‌اند و هر یک از آنها بترتیب غیر نزولی بر اساس مقادیر p[i]/w[i] مرتب شده‌اند.

خروجی: عدد صحیح maxprofit که حاصل جمع سود اجناس می‌باشد.

```
void      Knapsack ( int n , int  p[ ] , int w[ ] , int W , int &
```

```
{
        priority - queue - of - node  PQ ;
        node U,V ;
        initialze (PQ) ;
        V.level = V.profit = V.weight = 0 ;
        maxprofit = 0 ;
        V.bound = bound(V) ;
        insert (PQ,V) ;
        while (! Empty (PQ) )
        {
            remove(PQ,V) ;
            if ( V. bound > maxprofit )
            {
                U.level = V.level + 1 ;
                U.weight = V. weight + w[U.level] ;
                U.profit = V. profit + p[U.level] ;
                if (U.weight <= W  && U.profit > maxprofit )
                    maxprofit = U.profit ;
                U.bound = bound(U) ;
                if ( U.bound  > maxprofit )
                    Insert(PQ,U);
                U.weight = V.weight ;
                U.profit = V.profit ;
                U.bound = bound(U) ;
                if (  U.bound > maxprofit )
                    insert (PQ,U) ;
            }
        }
}
```

۳-۸ خلاصه فصل

۱. فضای مسأله‌ای که با استفاده از روش انشعاب و تحدید حل می‌شود باید با یک گراف قابل نمایش باشد.

۲. الگوی جستجو در درخت برای روش بازگشت به عقب روش جستجوی عمقی است در حالیکه برای روش انشعاب و تحدید جستجوی ردیفی یا جستجو در پهنا می‌باشد.

۳. برای روش انشعاب و تحدید می‌توان الگوریتم جستجوی بهینه‌ای بنام جستجو با هرس کردن انشعاب و تحدید ارائه داد. این الگوریتم با هرس کردن فضای حالات تعداد مقایسه‌ها در جستجو را بهینه می‌کند.

٤. در اغلب مسائلی که به روش انشعاب و تحدید حل می‌شوند هدف یافتن جواب بهینه است.
٥. همانند الگوریتمهای عقبگرد، زمان الگوریتمهای انشعاب و تحدید نیز در بدترین حالت، زمان نمایی یا بدتر می‌باشد.
٦. مرتبه زمانی مسأله فروشنده دوره‌گرد با استفاده از برنامه‌نویسی پویا $O(n^2 2^n)$ می‌باشد. در روش انشعاب و تحدید با ارائه یک تابع حد مناسب زمان اجرای الگوریتم کاهش می‌یابد ولی مرتبه زمانی تغییری نمی‌کند.

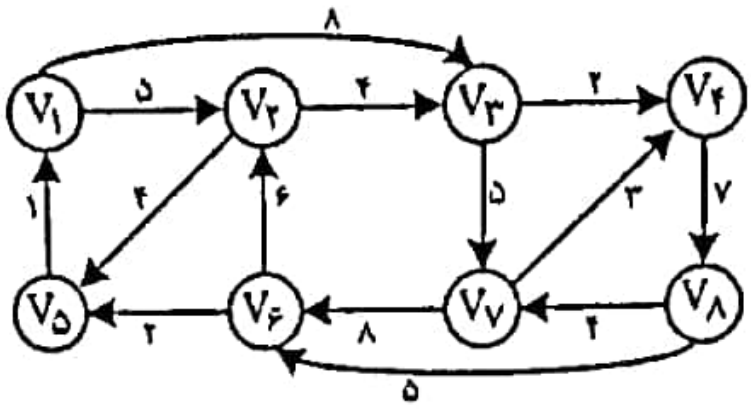
٤-٨ تمرینات

١. در بازی سه به سه قطار یک مربع به ۹ مربع کوچکتر تقسیم می‌شود. دو بازیکن به نوبت بازی می‌کنند و در هر نوبت یک بازیکن یک مربع کوچک را با علامت ویژه خود علامتگذاری می‌کند. بازیکنی که ابتدا سه مربع یک ردیف یا یک ستون یا یک قطر را با علامت ویژه خود علامتگذاری کند برنده خواهد بود. این بازی را تحلیل کنید و روش پیشنهادی خود را برای برنده شدن یکی از بازیکن‌ها براساس اصول انشعاب و تحدید بنویسید.
٢. با استفاده از روش انشعاب و تحدید برای مسأله کوله‌پشتی صفر و یک سود ماکزیمم قابل حصول از نمونه زیر را پیدا کنید. عملیات را مرحله‌به‌مرحله نشان دهید.

n=5 W=۲٥

i	P_i	W_i	P_i/W_i
1	$27	۳	۹
2	$30	٦	٥
3	$35	۷	٥
4	$18	۹	۲
5	$3	۳	۱

۳. با استفاده از روش انشعاب و تحدید برای مسأله فروشنده دوره‌گرد به یک تور بهینه و طول آن را برای گراف زیر پیدا کنید. مرحله به مرحله حل مسئله را نمایش دهید.



٤. یک الگوریتم انشعاب و تحدید برای مسأله زمان‌بندی با مهلت معیّن بنویسید.

۵. کارایی برنامه‌نویسی پویا و بازگشت به عقب را برای مسأله کوله‌پشتی مقایسه کنید.

٦. الگوریتمی به روش انشعاب و تحدید برای حل مسئله چهار وزیر ارائه دهید، سپس آنرا برای حالت n توسعه دهید.

۷. الگوریتم ارائه داده شده در مسئله ٦ را تحلیل زمانی کنید.

۸. درخت فضای حالات را برای مسئله ٦ ترسیم نموده سپس مراحل آنرا توصیف کنید.

۹. جستجو با هرس کردن انشعاب و تحدید را برای مسئله n وزیر بکار گرفته سپس آنرا با الگوریتم عقبگرد مقایسه نمائید.

۱۰. الگوریتم انشعاب و تحدید برای مسأله مدارهای هامیلتونی ارائه دهید، سپس آنرا با روش عقبگرد مقایسه نمائید.

۱۱. الگوریتم انشعاب و تحدید برای مسأله مدارهای هامیلتونی را چنان اصلاح کنید که به جای تولید همه حل‌های ممکن، فقط یک حل را پیدا کند.

۱۲. الگوریتمی به روش انشعاب و تحدید برای حل مسئله رنگ‌آمیزی با m رنگ ارائه دهید، سپس کارائی آنرا نسبت به الگوریتم ارائه شده به روش عقبگرد مقایسه نمائید.

۱۳. فرض کنید برای رنگ‌آمیزی مناسب یک گراف یک رأس آغازی و یک رنگ انتخاب کرده هر تعداد رأس ممکن را رنگ‌آمیزی می‌کنیم. سپس رنگ جدیدی را انتخاب کرده هر تعداد رأس ممکن از رئوس رنگ نشده را رنگ‌آمیزی می‌کنیم. این فرایند را چندان ادامه می‌دهیم که همه رئوس گراف رنگ‌آمیزی شوند. الگوریتمی به روش انشعاب و تحدید جهت رنگ‌آمیزی یک گراف با n رأس بنویسید. این الگوریتم را تحلیل کنید و نتیجه را با نماد مرتبه نشان دهید.

۱٤. الگوریتمی به روش انشعاب و تحدید برای حل مسئله مجموع زیر مجموعه‌ها ارائه دهید. حل حاصل را با حل بدست آمده برای مسئله به روش عقبگرد مقایسه نمائید. آیا زمان الگوریتم جدید بهتر شده است؟

۱۵. از الگوریتم انشعاب و تحدید برای مسأله مدارهای هامیلتونی جهت یافتن همه مدارهای هامیلتونی ممکن برای گراف شکل زیر استفاده کنید. عملیات را مرحله به مرحله نشان دهید. الگوریتم فوق روی کامپیوتر پیاده‌سازی کنید.