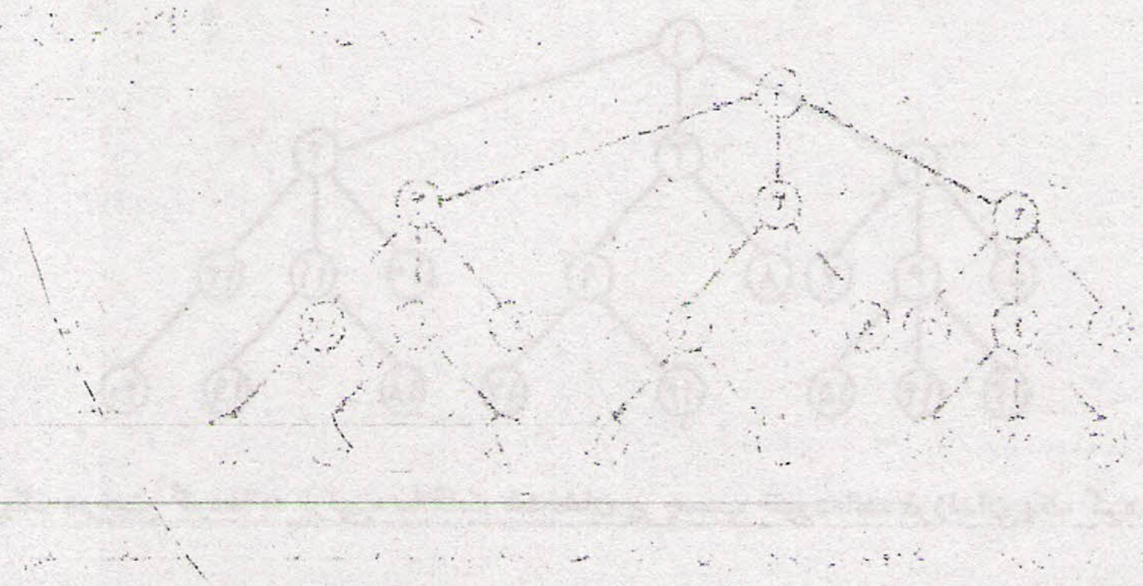


فصل ۶

روش‌های انشعاب و تحدید

تاکنون دو روش برای حل مسالهٔ کوله‌پشتی 0/1، به نام‌های برنامه‌نویسی و پویا و پسگرد ارایه کردیم. دیدیم که مرتبهٔ زمانی هر کدام از آن‌ها در بدترین حالت نمایی است. در این فصل روش سومی به نام انشعاب و تحدید ارایه می‌شود که بهبود یافتهٔ روش پسگرد است. هر چند مرتبهٔ زمانی آن نیز در بدترین حالت نمایی است، ولی برای یک نمونهٔ داده شده، می‌تواند خیلی کارآتر از دو روش قبل عمل کند.

روش انشعاب و تحدید، بسیار مشابه روش پسگرد است که از درخت فضای حالت برای حل مسایل استفاده می‌کند. دو اختلاف زیر بین این دو روش وجود دارد. در یک روش انشعاب و تحدید:

۱) محدودیتی برای استفاده از روش خاص برای پیمایش درخت فضای حالت وجود ندارد.

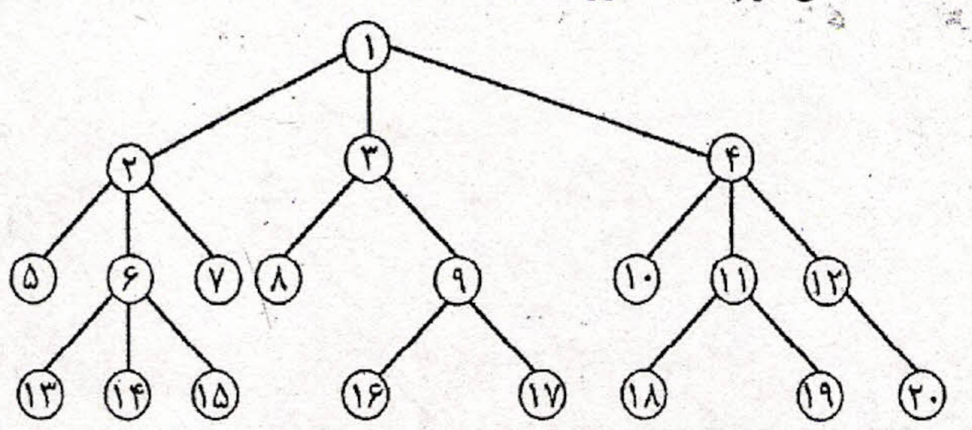
۲) فقط برای مسایل بهینه‌سازی به کار می‌رود.

در یک الگوریتم انشعاب و تحدید، برای هر گره، عددی (کرانه‌ای) برای تعیین امیدبخش بودن آن محاسبه می‌شود. عدد مزبور، کرانه برای جوابی است که در صورت گسترش گره مربوطه می‌تواند به آن رسید. اگر این کرانه، بهتر از مقدار بهترین جواب حاصل تاکنون نباشد، در این صورت، گره **غیر امیدبخش** و در غیر این صورت، **امیدبخش** است. از آن‌جا که منظور از مقدار بهینه، در بعضی از مسایل مینیمم و در بعضی دیگر ماکزیمم است، لذا مفهوم «بهتر» به‌کار رفته در این‌جا می‌تواند به معنی کوچک‌تر یا بزرگ‌تر باشد.

الگوریتم پسگرد به کار رفته در فصل قبل برای حل مسالهٔ کوله‌پشتی 0/1 را می‌توان یک روش انشعاب و تحدید به شمار آورد. با وجود این، روش پسگرد از مزایای انشعاب و تحدید بهره نمی‌جوید. علاوه بر استفاده از کرانه برای امیدبخش بودن یک گره، می‌توان کرانه گره‌های امیدبخش را نیز با هم مقایسه و اول، فرزندان گرهی را ملاقات کرد که بهترین کرانه را داراست. در این روش، اغلب سریع‌تر از حالتی به جواب می‌رسیم که در گره‌ها به صورت روشمند و با یک ترتیب خاص از پیش تعریف شده (مثل جستجوی در عمق) ملاقات بشوند. این روش انشعاب و تحدید را روش **جستجوی اول ـ بهترین** می‌نامیم. این روش، صورت بهبود یافته‌ای از روش روشمند انشعاب و تحدید دیگری به‌نام **جستجو در پهنا** است. بنابراین، هر چند این روش آخر، هیچ مزیتی بر روش جستجوی در عمق (روش پسگرد) ندارد، با وجود این، نخست به توضیح آن می‌پردازیم. این کار، ما را قادر خواهد ساخت تا روش انشعاب و تحدید جستجوی اول ـ بهترین برای حل مسالهٔ کوله‌پشتی 0/1 را راحت‌تر بیان کنیم.

در یک درخت، نخست ریشه را ملاقات می‌کند. سپس گره‌های واقع در سطح 1، بعد از آن گره‌های واقع در سطح 2 و الی آخر ملاقات

می‌شوند. شکل 1 درختی را نشان می‌دهد که گره‌های آن به روش جستجوی در پهنا ملاقات شده‌اند. گره‌های این درخت از چپ به راست ملاقات شده و به ترتیب ملاقات شدنشان برچسب خورده‌اند.



شکل 1. جستجوی در پهنای یک درخت، گره‌ها به ترتیب ملاقات شدنشان بر چسب خورده‌اند. فرزندان یک گره نیز از چپ ملاقات شده‌اند.

حل مسالهٔ کوله‌پشتی 0/1 به وسیلهٔ روش انشعاب و تحدید

در این بخش نشان می‌دهیم که چگونه می‌توان از روش انشعاب و تحدید برای حل مسالهٔ کوله‌پشتی 0/1، استفاده کرد. نخست، نسخهٔ سادهٔ این روش یعنی جستجوی در پهنا و سپس نسخهٔ بهبود یافته آن یعنی روش جستجوی اول ـ بهترین را ارایه می‌دهیم.

الف) جستجوی در پهنا

این روش را با یک مثال به تصویر می‌کشیم.

مثال ۱: نمونه مسالهٔ کوله‌پشتی 0/1 ارایه شده در فصل پسگرد را در نظر بگیرید. در این مثال داشتیم:

$M=16 \ , \ n=4$

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

یادآوری می‌شود که بسته‌ها به ترتیب نزولی P_i / W_i ها مرتب شده‌اند. برای حل این مسالهٔ با استفاده از روش جستجوی در پهنا، به همان صورت انجام شده برای حل این مثال توسط روش پسگرد، عمل خواهیم کرد. فقط با این تفاوت که به جای انجام جستجوی در عمق، جستجوی در پهنا خواهیم داشت. فرض کنید که weight و profit به ترتیب وزن و ارزش کل حاصل از انتخاب بسته ها تا یک گره مفروض باشند. برای تعیین این‌که آیا گره مزبور امیدبخش است یا نه، نخست weight و profit را به عنوان مقادیر اولیه به ترتیب برای totweight و bound منسوب می‌کنیم. سپس، بسته‌ها را مرحله به مرحله به روش حریصانه انتخاب کرده، وزن آن‌ها را به totweitght و ارزش آن‌ها را به bound اضافه می‌کنیم تا این‌که به بسته‌ای برسیم که حاصل جمع وزن آن با totweight بیشتر از M بشود. با توجه به وزن باقی‌مانده، کسر لازم از وزن این بستهٔ آخر را برداشته و ارزش مربوط به این کسر برداشته شده را به bound اضافه می‌کنیم. Bound، هنوز یک کرانه بالایی برای ارزشی است که می‌توان از گسترش گره مزبور به دست آورد. اگر این گره در سطح i, و گرهی که حاصل جمع وزن‌ها را از M بیشتر می‌کند در سطح k باشد، در این صورت:

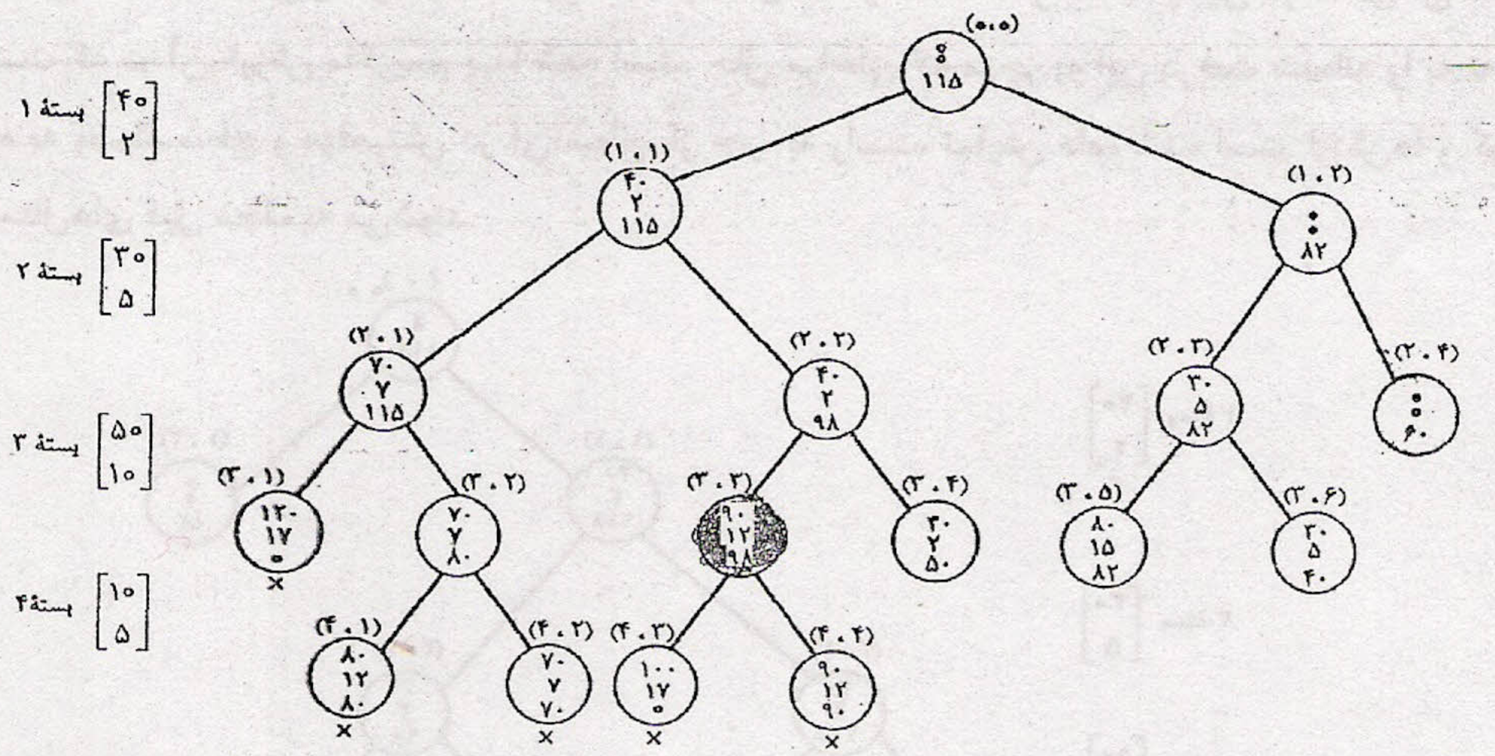
$$\text{totweight} + \sum_{j=i+1}^{k-1} w_j$$

$$\text{bound} = \left(\text{profit} + \sum_{j=i+1}^{k-1} P_i \right) + (M - \text{totweight}) * \frac{P_k}{W_k}$$

گرهی غیر امیدبخش است هر گاه کرانه آن کوچک‌تر یا مساوی maxprofit، بهترین جواب حاصل تا گره مزبور باشد. البته شرط دیگری نیز برای غیرامیدبخش بودن یک گره وجود دارد که به صورت زیر است:

$$\text{weight} \geq M$$

درخت فضای حالت هرس شدهٔ حاصل از اعمال روش جستجوی در پهنا برای مثال ۱، در شکل 2 آمده است. مقادیر weight, profit و bound برای یک گره، به ترتیب از بالا به پایین در داخل آن گره، آمده‌اند. گره سایه‌دار، نقطه‌ای است که ارزش ماکزیمم پیدا شده است. گره‌های این درخت بر اساس سطح و ترتیب قرار گرفتنشان در آن سطح، از چپ به راست، برچسب خورده‌اند.



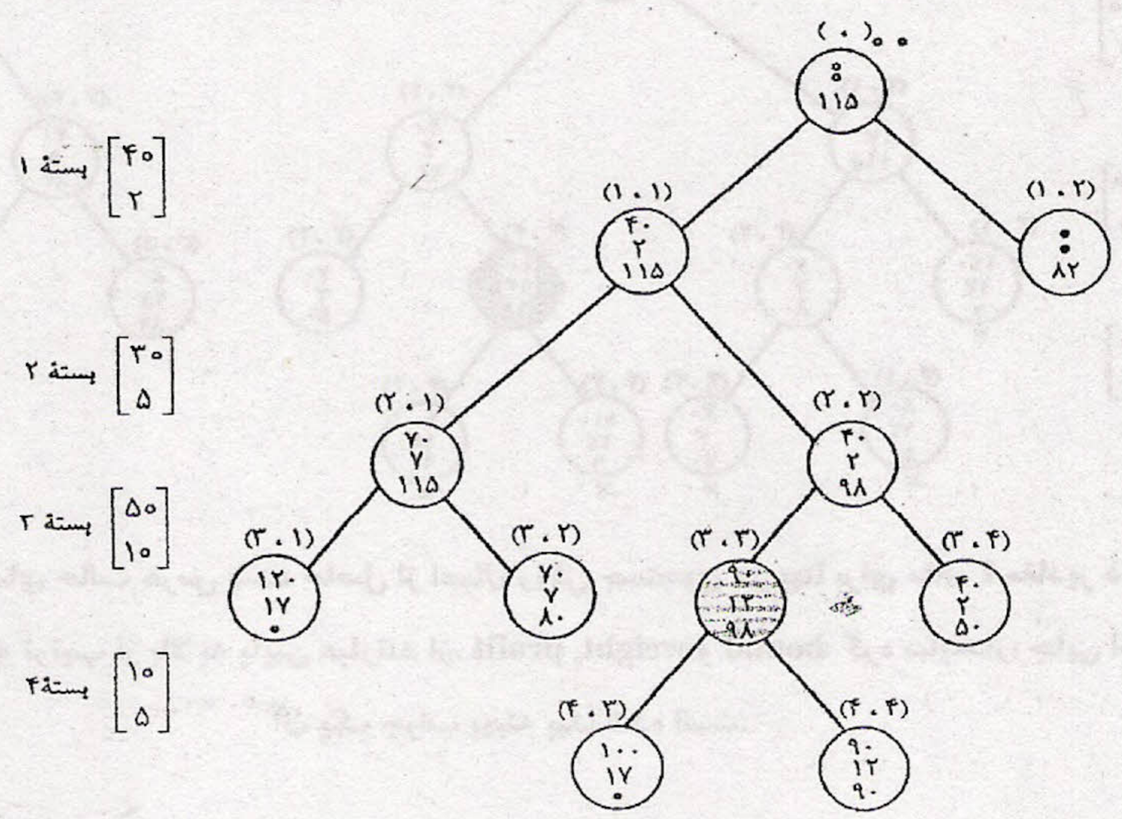
شکل 2 درخت فضای حالت هرس شده حاصل از اعمال روش جستجوی در پهنا برای مثال 1 مقادیر ذخیره شده در داخل هر گره به ترتیب از بالا به پایین عبارتند از، weight, profit و bound، گره سایه‌دار، جایی است که در آن یک جواب بهینه پیدا شده است.

چون مراحل کار، مشابه مثال پسگرد است. لذا از توضیح آن‌ها صرف‌نظر کرده فقط به ذکر چند نکته بسنده می‌کنیم. نخست توجه کنید که کرانه برای گره‌های (3,1) و (4,3) برابر صفر است. در روش انشعاب و تحدید، یک گره وقتی گسترش می‌یابد که کرانه آن بهتر از بهترین جواب حاصل تا آن گره باشد. بنابراین زمانی که وزن گرهی بیشتر از M و در نتیجه غیر امیدبخش باشد، کرانه آن را مساوی صفر قرار می‌دهیم. دوم، به خاطر بیاورید که موقع استفاده از روش پسگرد (جستجوی در عمق) برای حل همین مثال، گره (1,2) غیرامیدبخش بود و لذا آن را گسترش ندادیم، در صورتی که در این‌جا، مشاهده می‌کنید که همین گره امیدبخش تشخیص، و گسترش داده شده است. دلیل این امر آن است، که در روش جستجوی در پهنا، وقتی که این گره ملاقات می‌شود، مقدار maxprofit برابر 40 و bound این گره برابر 82 است. نکتهٔ آخر این‌که امیدبخش بودن یک گره به هنگام، ملاقات کردن همان گره، اتخاذ می‌شود. به عبارت دیگر اگر قرار است شاخه‌ها برای فرزندان گرهی را ملاقات کنیم، تصمیم برای این هرس کردن در موقع ملاقات آن گره گرفته می‌شود. در موقع ملاقات این گره، مقدار maxprofit برابر 70، کوچک‌تر از کرانه گره مزبور یعنی عدد 82، بوده است. برخلاف روش جستجوی در عمق، در روش جستجوی در پهنا، مقدار maxprofit می‌تواند تا زمان ملاقات عملی فرزند یک گره تغییر بکند. برای گره (2,3) اشاره شده در بالا، زمانی که بخواهیم فرزند آن را به صورت عملی ملاقات بکنیم، مقدار maxprofit به 90 افزایش پیدا کرده است. بنابراین، بررسی فرزندان آن، چیزی جز وقت تلف کردن نخواهد بود. این وقت تلف نکردن، کاری است که در الگوریتم جستجوی اول ـ بهترین که در بخش بعد به آن می‌پردازیم، انجام می‌شود.

ب) روش انشعاب و تحدید اول ـ بهترین

در حالت عمومی، راهبرد انشعاب و تحدید مزیتی بر روش جستجوی در عمق (پسگرد) ندارد. با وجود این اگر از تابع bound فراتر از آن چیزی که صرفاً برای تعیین امیدبخش بودن یک گره به کار رفت، استفاده کنیم، می‌توانیم جستجو را بهبود بخشیم. بعد از ملاقات همهٔ فرزندان یک گره، می‌توان از بین آن‌ها، فقط گره گسترش نیافته‌ای را گسترش داد که امیدبخش بوده و بهترین کرانه را نیز داشته باشد. یادآوری می‌شود که گرهی امیدبخش است که کرانه آن بهتر از مقدار بهترین جواب پیدا شده تا حال حاضر باشد. در این حالت، معمولاً خیلی سریع‌تر از حالتی که بخواهیم به صورت کورکورانه و به این ترتیب از پیش تعیین شده پردازش کنیم، به یک جواب بهینه می‌رسیم. مثالی که در ادامه می‌آید، این روش را به تصویر می‌کشد.

مثال ۲: نمونه مسالهٔ کوله‌پشتی 0/1 ارایه شده در مثال 1 را در نظر بگیرید. یک جستجوی اول ـ بهترین، درخت فضای حالت هرس شدهٔ شکل 3 را تولید می‌کند. مجدداً برای هر گره، مقادیر weight, profit و bound از بالا به پایین در داخل آن مشخص شده‌اند. گره سایه‌دار، جایی است که در آن، ارزش ماکزیمم پیدا شده است. حال مراحلی که منجر به این درخت شده‌اند را به تصویر می‌کشیم. در این‌جا نیز، هر گره به وسیله سطح و موقعیتش در آن سطح، از چپ به راست، نمایش داده شده است. ارزش‌ها و کرانه‌ها به همان صورت ارایه شده در مثال‌های قبل محاسبه می‌شوند.



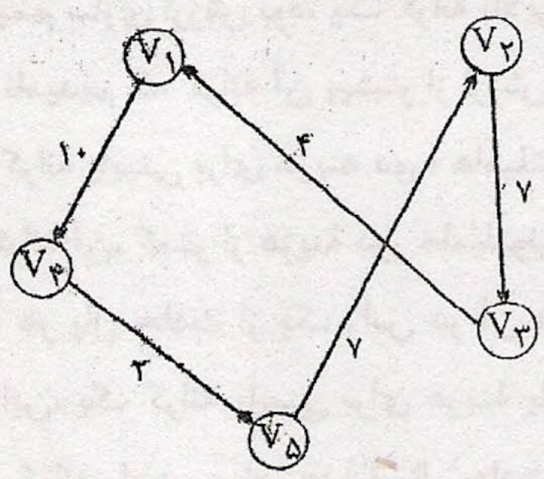
شکل 3: درخت فضای حالت هرس شده حاصل از اعمال روش جستجوی اول ـ بهترین برای مثال 2. مقادیر ذخیره شده در داخل هر گره به ترتیب از بالا به پایین عبارتند از: profit (ارزش کل بسته‌های انتخاب شده)، weight (وزن کل بسته‌های مزبور) و bound (کرانه برای کل ارزش که شاید بتوان از گسترش آن به دست آورد). گره سایه‌دار، جایی است که در آن یک جواب بهینه پیدا شده است.

مشاهده می‌شود تعداد گره‌های بررسی شده توسط جستجوی اول ـ بهترین برابر 11 است. این مقدار، 6 گره کمتر از مقدار گره‌های بررسی شده توسط جستجوی در پهنا (شکل 2) و 2 تا کمتر از تعداد گره‌های بررسی شده توسط جستجوی در عمق (فصل قبل) است. شاید بگوییم که صرفه‌جویی در بررسی 2 گره، خیلی تاثیرگذار نیست، ولی در یک درخت فضای حالت بزرگ، زمانی که از جستجوی اول ـ بهترین استفاده کنیم، این صرفه‌جویی برای رسیدن به یک جواب بهینه می‌تواند خیلی چشمگیر باشد. با وجود این، این نکته را یادآوری می‌کنیم که گره‌هایی که به ظاهر بهترین به نظر می‌رسند، ممکن است در عمل به جواب بهینه‌ای نرسند. برای مثال، در مثال 2، گره (2,1) بهتر از گره (2,2) به نظر می‌رسد، ولی دیدیم که این گره (2,2) بود که منجر به جواب بهینه شد. در حالت عمومی در جستجوی اول ـ بهترین نیز ممکن است، برای یک نمونه خاص، اکثر گره‌ها و یا همهٔ آن‌ها در درخت فضای حالت، قبل از رسیدن به یک جواب بهینه، بررسی می‌شوند.

مسالهٔ فروشنده دوره‌گرد

دیدیم که در این مسأله، هدف پیدا کردن یک دور هامیلتونی با هزینهٔ مینیمم در یک گراف وزن‌دار است. یادآوری می‌شود که یک دور هامیلتونی دوری است که از همهٔ رئوس گراف فقط و فقط یک بار عبور کند. قبلاً الگوریتمی به روش برنامه‌نویسی پویا از مرتبهٔ زمانی $\theta\left(2^n n^2\right)$، که در آن n تعداد رئوس گراف موردنظر است، برای این مسالهٔ ارایه دادیم.

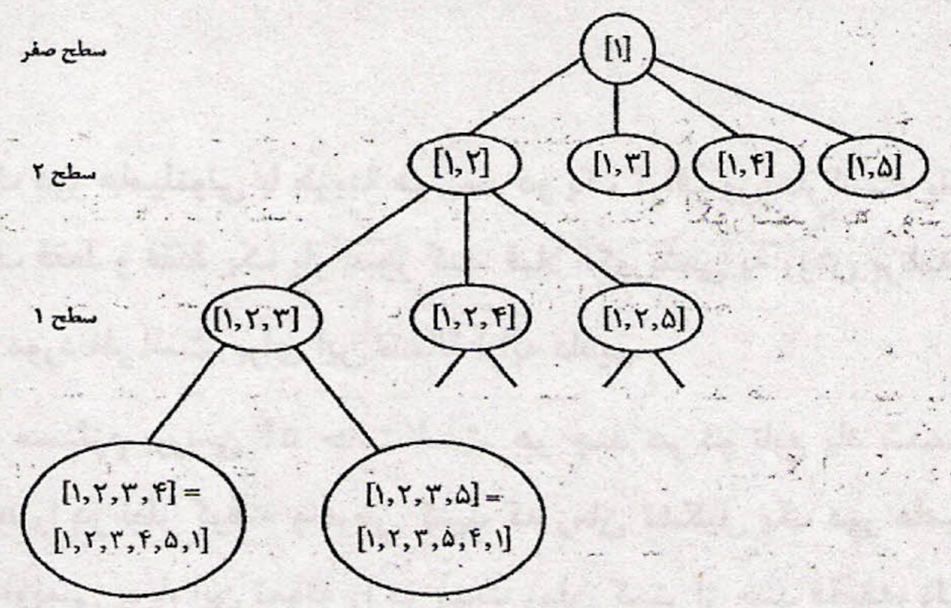
دیدیم که حل مسالهٔ به روش سعی و خطا، مستلزم بررسی !n حالت است. هر چند هر دو تابع یاد شده، نمایی هستند، ولی، برای درک بهتر اختلاف بین این دو، حالت n=20 را در نظر گرفته و فرض کنید که زمان تشکیل یک دور هامیلتونی در هر دو روش یک میکرو ثانیه است. در این صورت روش برنامه‌نویسی پویا، این نمونه را در مدت زمان کمتر از چند دقیقه، ولی روش سعی و خطا آن را حدوداً بعد از 3800 سال حل خواهد کرد. حال ببینیم که روش انشعاب و تحدید، چگونه آن را حل می‌کند. چون دور هامیلتونی را می‌شود از هر راس شروع کرد، در این‌جا نیز برای سهولت امر، فرض می‌کنیم که راس شروع برای دور هامیلتونی برابر راس 1 است. شکل 4 یک گراف با 5 راس تعریف شده توسط ماتریس هزینه‌ها، و نیز یک دور هامیلتونی با هزینه مینیمم را برای آن نشان می‌دهد:



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 14 & 4 & 10 & 20 \\ 2 & 14 & 0 & 7 & 8 & 7 \\ 3 & 4 & 5 & 0 & 7 & 16 \\ 4 & 11 & 7 & 9 & 0 & 2 \\ 5 & 18 & 7 & 17 & 4 & 0 \end{array}$$

شکل 4 : نمایش یک گراف وزن‌دار توسط ماتریس هزینه‌های آن (الف) و یک دور هامیلتونی با هزینه مینیمم برای آن (ب)

یک درخت فضای حالت بدیهی برای این مسالهٔ را می‌توان به صورت زیر تشکیل داد. در سطح 1 ، هر راس غیر از راس شروع را به عنوان اولین راس (بعد از راس شروع)، مورد بررسی قرار می‌دهیم. در سطح 2، هر راس شروع و راس بررسی شده در سطح 1 را به عنوان دومین راس مورد آزمایش قرار می‌دهیم و الی آخر. یک قسمت از درخت فضای حالت برای یک گراف دارای 5 راس که در آن از هر راس به راس دیگر یالی وجود دارد، در شکل 5 آمده است. در آن‌چه می‌آید، اصطلاح «گره» به معنی یک گره در درخت فضای حالت و اصطلاح «راس» به مفهوم یک راس در گراف خواهد بود. در داخل هر گره در شکل 5، مسیر انتخاب شده تا آن گره نوشته شده است. برای سهولت در نوشتن، فقط اندیس رئوس قید شده‌اند. گرهی که برگ نباشد، معرف همه دوره‌های هامیلتونی است که با مسیر ذخیره شده در آن گره شروع می‌شوند. برای مثال، گره شامل [1,2,3]، معرف همه دورهای هامیلتونی است که با مسیر [1,2,3] شروع می‌شوند. این دوره‌ها عبارتند از: [1,2 ,3,4,5,1] و [1,2,3,5,4,1] هر برگ، معرف یک دور هامیلتونی است. بنابراین نیاز به پیدا کردن برگی داریم که حاوی دور هامیلتونی با هزینهٔ مینیمم است. زمانی که مسیر، شامل چهار راس باشد، گسترش درخت را متوقف می‌کنیم، زیرا در این موقع، راس پنجم مسیر، به صورت منحصر به فردی مشخص است. برای مثال، برگ سمت چپ و پایین این درخت معرف دور هامیلتونی [1,2 ,3,4,5,1] است، زیرا زمانی که مسیر [1,2,3,4] مشخص شد، بدیهی است که راس بعدی، راس پنجم خواهد بود.



شکل 5 یک درخت فضای حالت برای نمونه‌ای از مسالهٔ فروشنده دوره‌گرد که در آن 5 راس وجود دارد.

برای استفاده از روش جستجوی اول ـ بهترین برای حل این مسالهٔ، نیاز به تعیین کرانه‌ای برای هر گره داریم. در مسالهٔ کولهٔ‌پشتی 0/1، چون هدف ماکزیمم سازی ارزش بود، یک کرانه بالایی برای ارزشی که می‌شد از گسترش یک گره مفروض به دست آورد، حساب و گرهی را امیدبخش نامیدیم که کرانه آن بیشتر از ارزش ماکزیمم فعلی باشد. در این‌جا، چون با مسالهٔ مینیمم‌سازی مواجه هستیم، لذا نیاز به تعیین یک کرانه پایینی برای هزینه دوره هامیلتونی داریم که می‌توان از گسترش یک گره داده شده به آن رسید، و گرهی را امیدبخش گوییم که کرانه آن، کمتر از هزینهٔ دور هامیلتونی مینیمم فعلی باشد. این کرانه را می‌شود به صورت زیر به‌دست آورد. در هر دور هامیلتونی، هزینهٔ هر یال حادث از یک راس در آن دور، بزرگ‌تر یا مساوی هزینهٔ یال با کمترین هزینهٔ حادث از همان راس در خود گراف است. بنابراین، یک کرانه پایینی برای هزینهٔ یال حادث از راس V_1 برابر است با مینیمم عناصر غیر صفر واقع در سطر 1 ماتریس هزینه‌ها، یک کرانه پایینی برای هزینهٔ یال حادث از V_2 برابر است با مینییم عناصر غیر صفر واقع در سط دوم و الی آخر. کرانه‌های پایینی برای هزینه‌های یال حادث از 5 راس گراف نمایش داده شده در شکل 4 به صورت زیر هستند:

برای راس V_1 داریم $(14,4,10,20)=4$ مینیمم

برای راس V_2 داریم $(14,7,8,7)=7$ مینیمم

برای راس V_3 داریم $(4,5,7,16)=4$ مینیمم

برای راس V_4 داریم $(11,7,9,2)=2$ مینیمم

برای راس V_5 داریم: $(18,7,17,4)=4$ مینیمم

چون یک دور هامیلتونی از هر راس گراف، فقط یک بار خارج می‌شود، بنابراین، یک کرانه پایینی برای هزینهٔ یک دور هامیلتونی، مجموع این مینیمم‌ها خواهد بود. در نتیجه، یک کرانه پایینی برای هزینهٔ دور هامیلتونی مثال بالا به صورت زیر خواهد بود:

$$4+7+4+2=21$$

البته، این بدین معنی نیست که یک دور هامیلتونی با این هزینه وجود دارد. برعکس، منظور این است که هیچ دور هامیلتونی با هزینهٔ کمتر از 21 برای گراف مزبور نمی‌توان پیدا کرد.

حال فرض کنید که گره شامل [1,2] را در شکل 5 ملاقات کرده‌ایم. در این حالت، راس V_2 به عنوان دومین راس برای دور هامیلتونی انتخاب شده است. بنابراین، هر دور هامیلتونی ناشی از گسترش گره مزبور، شامل یال (V_1, V_2) خواهد بود. هزینهٔ این یال برابر 14 است. در نتیجه، کرانه‌های پایینی زیر را برای هزینه‌های یال‌های حادث از رئوس چنین دوره‌های هامیلتونی خواهیم داشت:

برای راس V_1 14

برای راس V_2 $(7,8,7)=7$ مینیمم

برای راس V_3 $(4,7,16)=4$ مینیمم

برای راس V_4 $(11,9,2)=2$ مینیمم

برای راس V_5 $(18,17,4)=4$ مینیمم

برای به‌دست آوردن مینیمم برای راس V_2، یال (V_2,V_1) را نباید به حساب آورد. زیرا که در دور هامیلتونی ناشی از گسترش گره شامل $[1,2]$، از راس V_2 نمی‌توان به راس V_1 رفت. برای به‌دست آوردن مینیمم برای بقیهٔ رئوس نیز، یال‌های حادث به V_2 را در نظر نمی‌گیریم، زیرا که V_2 را قبلاً به عنوان دومین راس دور هامیلتونی برگزیده‌ایم. بنابراین، یک کرانه برای هر دور هامیلتونی که از گسترش گره شامل $[1,2]$ حاصل شود، حاصل‌جمع مینیمم‌های بالا خواهد بود:

$$14+7+4+2+4=31$$

به منظور توضیحات بیشتر برای تعیین کرانه، فرض کنید که گره شامل $[1,2,3]$ را در شکل 5 را ملاقات کرده‌ایم. چون V_2 را به عنوان راس دوم و V_3 را به عنوان راس سوم دور هامیلتونی انتخاب کرده‌ایم، هر دور هامیلتونی ناشی از گسترش گره مزبور، دارای کرانه‌های پایینی زیر برای یال‌های حادث از رئوس واقع در آن خواهد بود:

برای راس V_1 $14=$ هزینه یال (V_1,V_2)

برای راس V_2 $7=$ هزینه یال (V_2,V_3)

برای راس V_3 $(7,16)=7$ مینیمم

برای راس V_4 $(11,2)=2$ مینیمم

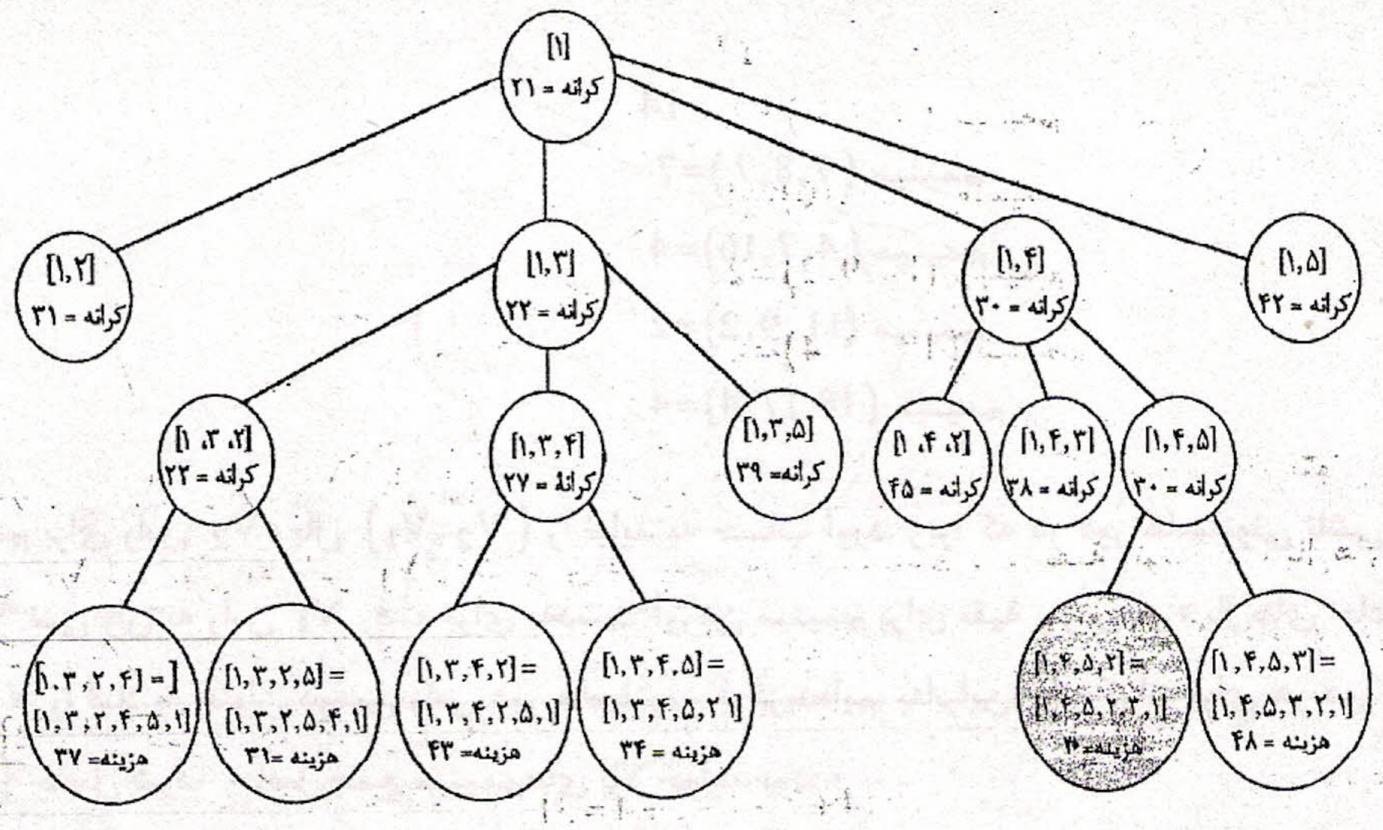
برای راس V_5 $(18,4)=4$ مینیمم

برای به‌دست آوردن مینیمم برای رئوس V_4 و V_5، یال‌های حادث به V_2 و V_3 را در نظر نمی‌گیریم، زیرا که قبلاً از رئوس V_2 و V_3 عبور کرده‌ایم. بنابراین، کرانه پایینی برای هر دور هامیلتونی که از گسترش گره شامل $[1,2,3]$ حاصل شود، برابر خواهد بود با:

$$14+7+7+2+4=34$$

به طریق مشابه، یک کرانه پایینی برای هزینهٔ یک دور هامیلتونی که می‌توان از گسترش یک گره در درخت فضای حالت به دست آورد را محاسبه کرده و این کرانه‌های پایینی را در الگوریتم جستجوی اول ـ بهترین خود مورد استفاده قرار می‌دهیم.

مثال 3: با فرض گراف شکل 4 و در نظر گرفتن ملاحظات بالا در مورد کرانه، یک الگوریتم جستجوی اول ـ بهترین، درخت فضای حالت هرس شده شکل 6 را تولید خواهد کرد. گره‌های داخلی، حاوی کرانه‌ها هستند، حال آن‌که گره‌های خارجی (برگ‌ها) نشان‌دهندهٔ هزینهٔ دور هامیلتونی ذخیره شده در آن‌ها می‌باشند.



شکل 6 درخت فضای حالت هرس شده حاصل از اعمال روش جستجوی اول ـ بهترین بر روی مثال 3، برای هر گره غیر برگ در این درخت ففضای حالت هرس شده، دور هامیلتونی جزیی در بالای گره و کرانه برای هزینه دور هامیلتونی که می‌توان از گسترش آن گره به دست آورد. در پایین آن نوشته شده است. برای هر گره برگ، دور هامیلتونی در بالا و هزینه آن در پایین آن نوشته شده است. گره سایه‌دار، گرهی است که در آن یک دور هامیلتونی با هزینه مینیمم به‌دست آمده است.

سوال‌های کارشناسی ارشد کنکور سراسری

۱ـ آنچه در زیر آمده است یک شبه کد اشتباه برای الگوریتمی است که باید متوازن بودن یا نبود رشته‌ای از پرانتزها را تعیین کند:

```
declare a character stack
while (more input is available)
    {read a character
      if (the character is a '(')
      then push it on the stack
      else if (the character is a ')' and the stack is not empty )
          then pop a character off the stack
          else print "unbalanced" and exit
}
```

کدام‌یک از رشته ها نامتوازن زیر توسط الگوریتم فوق، متوازن در نظر گرفته می‌شوند؟

۱) ((( )) ۲) ( ))(( )

۳) ( )( )) ((۴) (( )( )))

۲ـ فرض کنید اعداد 1 تا 1000 در یک درخت دودویی جستجو ذخیره شده‌اند و ما می‌خواهیم عدد 363 را پیدا کنیم. کدام‌یک از ترتیب‌های زیر (از چپ به راست) نمی‌تواند بیانگر ترتیب دسترسی به عناصر درخت در این جست و جو باشد؟

۱) 925,202,911,240,912,245,363

۲) 924,220,911,244,898,258,362,363

۳) 2,252,401,398,330,344,397,363

۴) 2,399,387,219,266,382,381,278,363

۳ـ در یک درخت دودویی جستجوی T ، اگر x برگ و y پدر x باشد، کدام گزینه در مورد مقادیر $a=key[x]$ و $b=key[y]$ درست است؟

۱) b بزرگ‌ترین کلید در T است که کوچک‌تر از a باشد.

۲) a کوچک‌ترین کلید در T است که بزرگ‌تر از b باشد.

۳) b کوچک‌ترین کلید در T است که بزرگ‌تر از a باشد.

۴) هیچ‌کدام از گزینه‌های فوق همواره درست نیست.

۴ـ کدام‌یک از گزینه‌های زیر در مورد الگوریتم Quicksort درست است؟ فرض کنید که عنصر اول همواره به عنوان محور (Pivot) انتخاب می‌شود (n تعداد عناصر آرایهٔ ورودی است). اگر آرایهٔ ورودی از قبل مرتب باشد، ....

۱) زمان اجرای الگوریتم $O(n)$ است.

۲) زمان اجرای الگوریتم $O(n^2)$ است.

۳) زمان اجرای الگوریتم $O(n \log n)$ است.

۴) متوسط زمان اجرا $O(n \log n)$ است.

۵ـ آرایهٔ زیر یک heap است. برای درج عدد 95 در آرایه، به گونه‌ای که آرایهٔ نهایی نیز وضعیت heap داشته باشد، چند عمل exchange (تعویض دو کمیت) لازم است؟

100
90
82
85
74
75
73
68
70

۱) دو

۲) چهار

۳) شش

۴) هشت

۶ـ تابع ذیل برای محاسبه بزرگ‌ترین مقسوم علیه دو عدد نوشته شده است.

```
int gcd(int a , int b)
{
 if (b==0)
        return a ;
 else
        return gcd (b,a mod b)
}
```

در این صورت می‌توان گفت: مرتبه زمانی الگوریتم..... است.

۱) $O(\log a/b)$

۲) $O(\log_b a)$

۳) $O(\log_2 a)$

۴) $O(\log_2 (a-b))$

۱ـ برای به‌دست آوردن بزرگ‌ترین و کوچک‌ترین عنصر در یک لیست غیر مرتب، بهترین الگوریتمی که می‌توان ارایه داد چند مقایسه خواهد داشت؟ (برای n های زوج)

۱) $2(n-1)$ ۲) $\frac{3n}{2}-2$ ۳) $\frac{(n-1)}{2}+1$ ۴) $(n-1)+(n-2)$

۲ـ یک ماشین انتزاعی که در مبنای 10 کار می‌کند، اعداد را به راحتی با یکدیگر جمع و تفریق می‌کند و ضرب در اعداد 10^x را از طریق Shift به راحتی عمل می‌کند، اما در ضرب تنها قادر است اعداد یک رقمی را در هم ضرب نماید. به نظر شما برای ضرب دو عدد 1234*5618 حداقل تعداد عملیات ممکن ضرب چه تعداد می‌باشد تا حاصل ضرب فوق به دست آید؟ (راهنمایی: از طریق ضرب اعداد صحیح بزرگ استفاده کنید.)

۱) 8 ۲) 12 ۳) 13 ۴) 16

۳ـ برای محاسبه عبارت زیر، حداقل از چند عمل ضرب می‌توان استفاده کرد؟

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + .. + a_1 x + a_0$$

۱) n ۲) n^2 ۳) $n+\frac{n(n+1)}{2}$ ۴) $n+\frac{n(n-1)}{2}$

۴ـ به دست آوردن تعداد سیکل‌های ساده در یک گراف مسطح با الگوریتمی با درجه..... قابل محاسبه می‌باشد؟

۱) 1 ۲) n ۳) n^2 ۴) 2^n

۵ـ کدام‌یک از عبارات زیر غلط است؟

۱) $10^n + n^{20} \notin \theta(n^n)$ ۲) $\log_2 n \in \theta(\log_{10} n)$

۳) $(\log_2 n)! \in \Omega(n!)$ ۴) $4n^3 + 5n^2 + 7n \in \Omega(\log_2 n)$

۶ـ تابع زیر را در نظر بگیرید:

```
int f(int n)
{
  if n < 1 return 1
  else return f(n-1)+ g(n);
}

int g(int n)
{
  if n < 2 return 1
  else return f(n-1)+g(n/2);
}
```

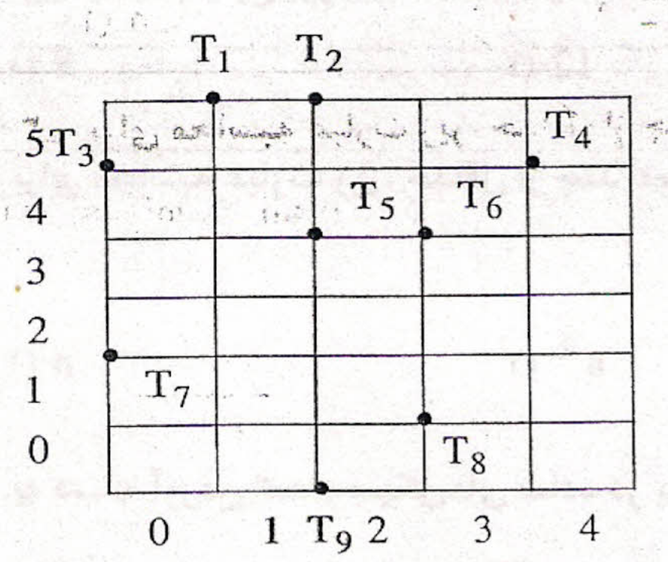
کدام گزینه، بهترین نمایش برای تابع رشد $f(n)$ برحسب n است؟

۱) خطی ۲) نمایی ۳) درجه دو ۴) لگاریتمی

۷ـ N یک عدد صحیح 16 بیتی است که در متغیر صحیح A قرار دارد. الگوریتم α که هدف آن بررسی اول (Prime) بودن متغیر A می‌باشد دارای چه درجه‌ای است؟

۱) N ۲) N^2 ۳) $\text{Log}\,N$ ۴) نمایی

۸ـ یک مهندس برق، مداری را طراحی نموده است دارای 9 ترمینال که باید ولتاژی معادل 5 ولت به آن‌ها متصل گردد. فرض کنید 5 ولت به یکی از ترمینال‌ها وصل است. برای این‌که کمترین سیم‌بندی در مدار به‌کار رود، بگویید حداقل چقدر سیم لازم دارد؟ (فاصله هر سطر و ستون را یک سانتیمتر در نظر بگیرید.)



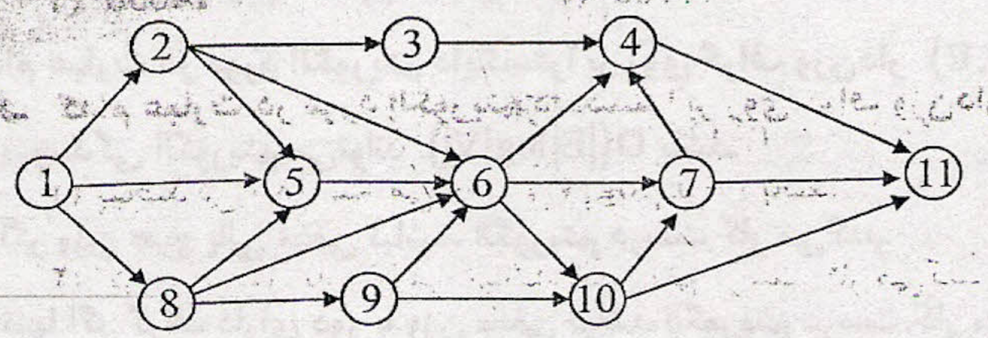
۱) $8+2\sqrt{2}+2\sqrt{5}$

۲) $6+2\sqrt{2}+2\sqrt{5}$

۳) $5+2\sqrt{2}+3\sqrt{5}$

۴) $7+2\sqrt{2}+2\sqrt{5}$

۱ـ در جستجوی عمق اول (Depth–First search) گراف جهت‌دار زیر، فرض کنید که گره 1 ، گره شروع باشد و گره‌های مجاور یک گره به ترتیب مقدار عددی شان ملاقات می‌شوند، کدام گزینه (از چپ به راست) ترتیب گره‌های ملاقات شده را نشان می‌دهد؟



۱) 1,2,3,4,11,5,6,7,10,8,9

۲) 1,2,8,5,3,9,6,4,11,10,7

۳) 1,2,5,8,3,9,6,4,10,7,11

۴) 1,2,3,4,5,6,7,8,9,10,11

۲ـ N بازه $[l_i, u_i]$ برای $i=1,\dots,n$ داده شده‌اند. می‌خواهیم S مجموعه‌ی بازه‌های دو به دو ناهمپوشان را با بیشترین طول (مجموع طول بازه‌های انتخاب شده) پیدا کنیم. برای این کار الگوریتم زیر پیشنهاد شده است:

بازه‌ها را به یک ترتیب انتخاب کن، برای هر بازه انتخابی، بازه‌های دیگر را که با آن هم‌پوشانی دارند حذف کن و این کار را تکرار کن.

کدام گزینه ترتیب درست برای این الگوریتم حریصانه است؟

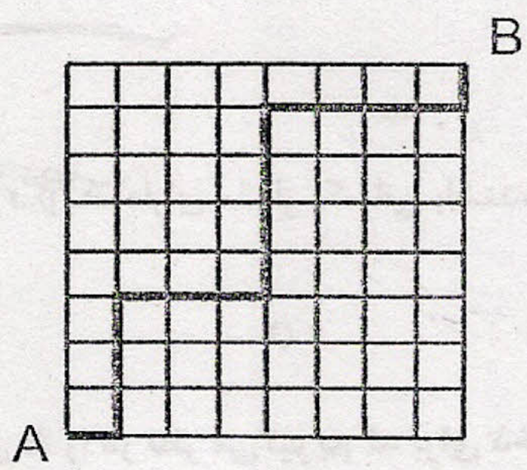
۱) ترتیب برحسب l_i بازه‌ها

۲) ترتیب برحسب u_i بازه‌ها

۳) ترتیب برحسب طول بازه‌ها

۴) هیچ‌کدام از ترتیب‌های فوق درست نیست.

۳ـ صفحهٔ شطرنجی زیر مفروض است. می‌خواهیم از نقطهٔ A به B برویم. حرکت‌های مجاز فقط به سمت راست یا به سمت بالا هستند. تعداد مسیرهای ممکن برای رسیدن از A به B چقدر است؟ یک نمونه مسیر در شکل آمده است؟



۱) $8!$

۲) 8×8

۳) C_8^{16}

۴) $(8+8)^2$

۴ـ مساله برج هانوی را با سه میلهٔ C,B,A در نظر می‌گیریم. در این‌جا هیچ حلقه‌ای را نمی‌توان مستقیماً از A به B یا از B به A منتقل کرد. یعنی چنین انتقال‌هایی تنها به کمک میلهٔ C انجام پذیرند. اگر در ابتدا n حلقه در میلهٔ A داشته باشیم و $T(n)$ حداقل تعداد عملیات لازم برای انتقال n حلقه از A به B باشد، $T(n)$ با کدام رابطه مشخص می‌شود؟ (فرض کنید $T(1)=2$)(یادآوری می‌شود که در هیچ مرحله‌ای نمی‌توان حلقه‌ای را روی حلقه‌ای کوچک‌تر از خودش قرار داد و درهر مرحله فقط یک حلقه جابه‌جا می‌شود.)

۱) $T(n)=6T(n-1)+3$

۲) $T(n)=3T(n-1)+2$

۳) $T(n)=T(n-1)+T(n-2)+1$

۴) $T(n)=T(n-1)+T(n-2)+2$

۵ـ متنی شامل 7000 حرف از حروف e,c,d,b,a، و f با دفعات تکرار $a=1000, b=1200, c=800, d=1500, e=1800$ و $f=700$ موجود است. چنان‌چه کدی بهینه برای حروف بالا انتخاب نماییم، تعداد کل بیت‌های لازم برای تبدیل متن مذکور به مجموعه‌ای از بیت‌ها چقدر است؟

۱) 14600 ۲) 17700 ۳) 24300 ۴) 35200

۶ـ کدام عبارت در مورد الگوریتم دایکسترا بر روی گراف وزن‌دار $G=(V,E)$ نادرست است؟ (گراف $G=(V,E)$ وزن‌دار است.)

۱) پیچیدگی الگوریتم می‌تواند $O(|E|\log|V|)$ باشد.

۲) اگر وزن هیچ یالی منفی نباشد، الگوریتم درست کار می‌کند.

۳) تنها اگر گراف دارای دور با وزن منفی نباشد الگوریتم درست کار می‌کند.

۴) هم برای گراف‌های بدون جهت درست کار می‌کند و هم برای گراف‌های جهت‌دار.

۷ـ رویهٔ Partition در الگوریتم Quick Sort به‌صورت زیر است:

```
function partition (p,r:integer):integer;
        Var x,i,j:integer;
begin
    x=A[p];   i:=p-1;  j:=r+1
     while true do
   begin
      repeat  j:=j-1      until A[j]<=x;
      repeat  i:=i+1      until A[i]>=x;
      if i<j      then swap (A[i],A[j]) else return(j)
    end
end
```

اگر تمام درایه‌های $A[p..r]$ دارای مقدار یکسانی باشند، مقداری که رویهٔ فوق بر می‌گرداند چقدر است؟

۱) P ۲) r ۳) $\left\lceil \frac{p+r}{2} \right\rceil$ ۴) $\left\lfloor \frac{p+r}{2} \right\rfloor$

۸ـ درایهٔ n خانه‌ای A را در نظر می‌گیریم که برای ذخیره‌سازی عناصر یک درخت دودویی کامل مورد استفاده قرار گرفته است. فرض کنیم الگوریتمی کارآ برای بررسی این که این درخت دودویی یک heap است یا خیر در اختیار داریم. پیچیدگی این الگوریتم در بدترین حالت چقدر است؟

۱) $O(n)$ ۲) $O(n^2)$ ۳) $O(\log n)$ ۴) $O(n\log n)$

۱ـ فرض کنید $T(n)$ برابر تعداد پرانتزبندی‌های مختلف برای ضرب کردن n ماتریس درهم باشد، در این صورت، $T(1)=T(2)=1$ و

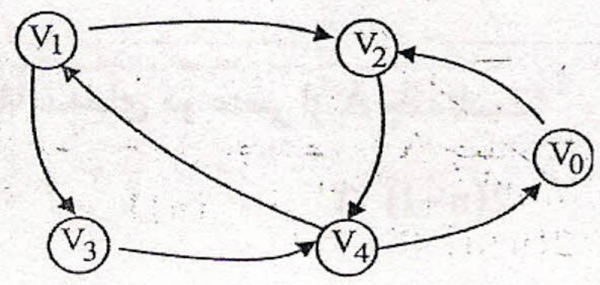
۱) $T(n)=\sum_{i=1}^{n} T(i)\times T(n-i)$

۲) $T(n)=\sum_{i=1}^{n-1} T(i)\times T(n-i)$

۳) $T(n)=\sum_{i=1}^{n} T(i)\times T(n-i+1)$

۴) $T(n)=\sum_{i=1}^{n-1} T(i)\times T(n-i+1)$

۲ـ اجزای قویاً هم بند گراف زیر کدام است؟



۱) $\{v_0, v_1, v_2, v_3, v_4\}$

۲) $\{v_0, v_2, v_3, v_4\}, \{v_0\}$

۳) $\{v_0, v_2 v_4\}, \{v_1, v_3, v_4\}$

۴) $\{v_1, v_3, v_4\}, \{v_2, v_1, v_4\}, \{v_0, v_2, v_4\}$

۳ـ کدام گزینه در مورد الگوریتم‌های کراسکال و پریم برای ایجاد درخت پوشای مینیمم درست است؟

۱) زمان اجرای هر دو الگوریتم روی گراف‌های یکسان، مساوی است.

۲) هر دو الگوریتم روی گراف‌های یکسان، درخت پوشای یکسان تولید می‌کنند.

۳) مجموع طول اضلاع درخت پوشا در هر دو الگوریتم یکسان است.

۴) هر دو الگوریتم با رشد و بهم پیوستن یک جنگل از درخت‌ها، درخت پوشا را تولید می‌کنند.

۴ـ دو مجموعهٔ A و B به ترتیب دارای m و n عنصر هستند با $m \le n$. در این صورت:

۱) اگر آرایه‌های A و B مرتب باشند، نمی‌توان اجتماع و اشتراک آنها را در زمان $\theta(n)$ به دست آورد.

۲) اگر آرایه‌های B,A مرتب باشند، نمی‌توان در زمانی کمتر از $\theta(n^2)$ اجتماع و اشتراک آن‌ها را به دست آورد.

۳) این دو آرایه چه مرتب باشند و چه نامرتب، نمی‌توان در زمانی کمتر از $\theta(n^2)$ اجتماع و اشتراک آن‌ها را به دست آورد.

۴) اگر آرایه‌های B,A مرتب باشند،می‌توان اجتماع و اشتراک آنها را در زمان $\theta(n)$ به دست آورد.

۵ـ پنج فایل مرتب شده به اندازه‌های 25,20,10,5 و 30 مفروض‌اند. می‌خواهیم از ادغام دو به دوی آن‌ها یک فایل مرتب شدهٔ واحد شامل همه رکوردها به دست آوریم. در هر ادغام رکوردهای فایل‌های ورودی ممکن است چند بار از یک فایل خوانده شده و در یک فایل دیگر نوشته شوند. به هر کدام از این نوشتن و خواندن یک جابه‌جایی می‌گوییم. حداقل تعداد کل این جابه‌جاییها برای ادغام همهٔ فایل‌ها چقدر است؟

۱) 195 ۲) 200 ۳) 185 ۴) 215

۶ـ الگوریتم زیر برای پیدا کردن اندیس‌های کوچک‌ترین و بزرگ‌ترین عناصر در یک آرایهٔ N تایی A داده شده است:

```
min ← 1; max ← 1
for    i ← 2 to N do
        choose B=true or B=false with equal probabities
        if B then if A[i] <A[min]
                  Then min ← i
              else if A[i]> A[max]
                      then max ← i
        ekse if A[i] > A[max]
                then max ← i
        else if A[i] < A[min]
                then min ← i
```

متوسط تعداد مقایسه‌های دو عنصر از A چندتاست؟

۱) n ۲) $2(n-1)$ ۳) $\frac{3n}{2}$ ۴) $\frac{3(n-1)}{2}$

۷ـ دومین کوچک‌ترین عنصر را بین n عنصر با چند مقایسه بین عناصر می‌توان به دست آورد؟ (بهترین جواب ممکن را انتخاب کنید.)

۱) $n+\lceil \lg_2 n \rceil-2$ ۲) $n+\lceil \lg_2 n \rceil-1$ ۳) $n+\lfloor \lg_2 n \rfloor-2$ ۴) $n+\lfloor \lg_2 n \rfloor-1$

۸ـ کدام‌یک از گزینه‌های زیر در یک گراف $G=(N,E)$ درست است؟

۱) درخت‌های عمق اول و سطح اول G با هم برابرند.

۲) وجود دور با وزن منفی در G را می‌توان در $O(|V||E|)$ تشخیص داد.

۳) اگر وزن هر یال را به اندازهٔ 1 واحد اضافه کنیم، کوتاه‌ترین مسیر بین هر دو راس دلخواه تغییر نمی‌کند.

۴) اگر M درخت پوشای مینیمم باشد،مسیری که در M بین هر دو راس u و v وجود دارد، کوتاه‌ترین مسیر بین آن دو راس در G است.

۱ـ اگر الگوریتم جستجوی دودویی را برای جستجوی عناصر آرایه $A[1..A]=[5,10,15,20,25,30,35,40]$ به کار ببریم، میانگین تعداد مقایسه‌ها برای جستجوی موفق تقریباً، کدام است؟

۱) 2.2 ۲) 2.4 ۳) 2.6 ۴) 2.8

۲ـ در الگوریتم پیدا کردن k امین عنصر از n عنصر، ابتدا همهٔ عناصر را به دسته‌های 5 تایی (به جز احتمالاً یک دسته) تقسیم می‌کنیم، میانه هر دسته را به دست می‌آوریم و سپس میانهٔ میانه‌ها را به صورت بازگشتی پیدا می‌کنیم. این عنصر را به عنوان محور انتخاب می‌کنیم و عمل Partition را بر روی آرایهٔ عناصر انجام می‌دهیم. پس از آن همین الگوریتم را به صورت بازگشتی (و برای یک k دیگر) بر روی یکی از بخش‌ها اجرا می‌کنیم تا عنصر مورد نظر پیدا شود. زمان اجرای این الگوریتم توسط کدام یک از رابطه‌های بازگشتی زیر بیان می‌شود؟

۱) $T(n)=T\left(\left\lceil\frac{n}{5}\right\rceil\right)+T\left(\frac{3n}{10}-6\right)+O(n)$

۲) $T(n)=T\left(\left\lfloor\frac{n}{5}\right\rfloor\right)+T\left(\frac{3n}{10}-6\right)+O(n)$

۳) $T(n)=T\left(\left\lfloor\frac{n}{5}\right\rfloor\right)+T\left(\frac{7n}{10}+6\right)+O(n)$

۴) $T(n)=T\left(\left\lceil\frac{n}{5}\right\rceil\right)+T\left(\frac{7n}{10}+6\right)+O(n)$

۳ـ حداکثر طول یک کد برای n عنصر که به روش هافمن کدگذاری می‌شوند چه‌قدر است؟

۱) $n-2$ ۲) $\lceil \lg n \rceil$ ۳) $n-1$ ۴) $\left\lceil\frac{n}{2}\right\rceil$

۴ـ آرایهٔ $S[1..N]$ از اعداد صحیح داده شده است. با داشتن K می‌خواهیم i و j ای را (در صورت وجود) پیدا کنیم که $S[i]+S[j]=K$. برای این کار، الگوریتم زیر پیشنهاد شده است:

```
i ← 1; j ← n
while i ≤ j do
    if S[i]+S[j]=K then return i , j
    if S[i]+S[j] < K then i ← i+1
    if S[i]+S[j] > K then j ← j−1
end
return not − possible
```

۱) همواره درست کار می‌کند.

۲) اگر S به‌صورت نزولی مرتب باشد درست کار می‌کند.

۳) اگر S به صورت صعودی مرتب باشد، درست کار می‌کند.

۴) برای هر حالت ورودی، ممکن است هیچ‌گاه جواب پیدا نشود.

۵ـ کدام گزینه نادرست است؟

۱) تنها یک درخت دودویی کامل با n گره می‌توان رسم کرد.

۲) ارتفاع درخت دودویی کاملی که n برگ دارد برابر است با $\lfloor \log_2 n \rfloor$

۳) اگر یک درخت کلی n برگ داشته باشد تعداد کل گره‌های آن 2n-1 است.

۴) اگر یک درخت دودویی n گره غیر برگ داشته باشد، تعداد کل گره‌های آن 2n + 1 است.

۶ـ سه آرایهٔ B,A و C هر کدام با عنصر را در نظر بگیرید به قسمی که A و C مرتب شدهٔ صعودی و B مرتب شدهٔ نزولی است. هم‌چنین A و B عنصر تکراری ندارند، ولی C دارد. به منظور مرتب‌سازی صعودی، الگوریتم‌های Heap sort , Straight Insertion Sort, Selection Sort ,Quick Sort را بر روی هر کدام از این آرایه‌ها اجرا می‌کنیم. در این صورت می‌توان گفت:

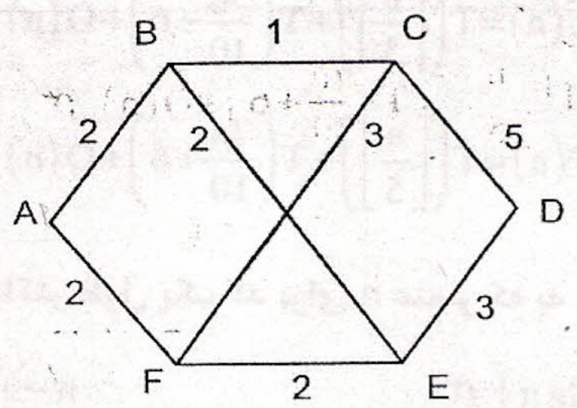
۱) برای B پیچیدگی زمانی Heap Sort و Quick Sort یکسان است.

۲) برای C ،برنامهٔ Quick Sort سریع‌تر از Heap Sort به پایان می‌رسد.

۳) برای C ،پیچیدگی زمانی Selection Sort و Straight Insertion sort یکسان است.

۴) برای A اجرای Straight Insertion sort سریع‌تر از Selection Sort و Quick Sort به پایان می‌رسد.

۷ـ گراف بدون جهت و وزن‌دار روبرو را در نظر می‌گیریم. اگر نقطهٔ A را به عنوان مبدأ انتخاب کنیم و الگوریتم‌های کروسکال و پریم را جداگانه بر روی این گراف اجرا کنیم، لبه (یال) های انتخاب شده به ترتیب مطابق با کدام گزینه می‌باشد؟



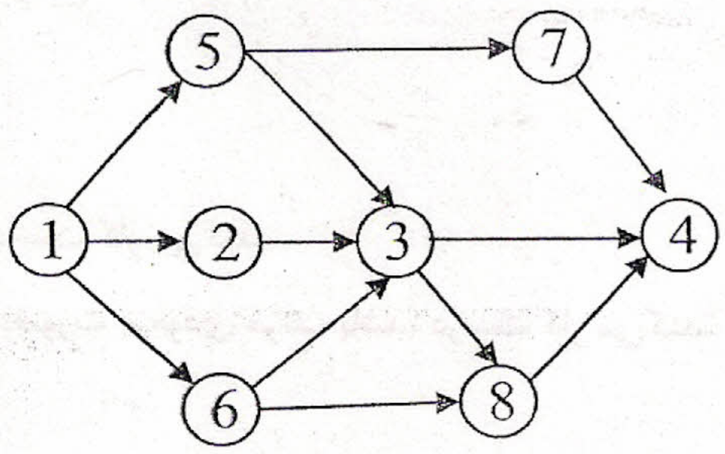
۱) کراسکال: BC, BE, FE, AF, ED
پریم: AF, FE, BE, BC, ED

۲) کراسکال: BC, BA, AF, FE, ED
پریم: AF, FE, EB, BC, ED

۳) کراسکال: BC, FE, AF, AB, ED
پریم: AB, AF, FE, BC, ED

۴) کروسکال: BC, BE, AB, ED, FE
پریم: AF, FE, ED, AB, BC

۸ـ اگر راس‌های گراف زیر را به‌صورت topological مرتب کنیم، چند دنبالهٔ متفاوت به دست می‌آید؟



۱) 6

۲) 1

۳) 2

۴) 48

۱ـ کدام‌یک از شرایط زیر برای مقادیر صحیح و مثبت a_1 تا a_n تضمین می‌کند که $\theta(n)$ حل رابطه بازگشتی $T(n)=\sum_{i=1}^{k} T(n/a_i)+\Theta(n)$ است؟

۱) $\sum_{i=1}^{k} a_i=1$

۲) $\sum_{i=1}^{k} a_i=k$

۳) $\sum_{i=1}^{k} a_i<k$

۴) $\sum_{i=1}^{k} a_i>k$

۲ـ مقدار تابع زیر به ازای $n\geq 2$ چقدر است؟

```
Function g(n)
begin
    if  n≤1 then g=n
      else g=5*g(n-1)-6*g(n-2);
end
```

۱) 5^n-6^n

۲) 3^n-2^n

۳) 3^n+2^n

۴) 5^n+6^n

۳ـ فرض کنید $f(n)=O(g(n))$. کدام یک از گزینه‌های زیر برای هر تابع با مقادیر مثبت مانند $f(n)$ و $g(n)$ برقرار است؟

۱) $g(n)=\Theta(f(n))$

۲) $2^{f(n)}=O\left(2^{g(n)}\right)$

۳) $2^{g(n)}=\Omega\left(2^{f(n)}\right)$

۴) $\log(g(n))=O(\log(f(n)))$ با فرض $\log(g(n))\geq 1$

۴ـ $G=(V,E)$ یک گراف بدون جهت و همبند و وزن‌دار است و a و b دو راس مجزا در آن. فرض کنید P_1 مساله پیدا کردن کوتاه‌ترین مسیر بین a و b و P_2 مساله پیدا کردن بلندترین مسیر ساده بین a و b باشد. کدام یک از گزینه‌های زیر در مورد P_1 و P_2 درست است؟

۱) P_1 و P_2 را می‌توان در زمان چند جمله‌ای حل کرد.

۲) P_1 و P_2 را نمی‌توان در زمان چندجمله‌ای حل کرد.

۳) P_1 را می‌توان در زمان چند جمله‌ای حل کرد، ولی P_2 را نمی‌توان.

۴) P_2 را می توان در زمان چند جمله‌ای حل کرد، ولی P_1 را نمی‌توان.

۵ـ در الگوریتم فلوید برای پیدا کردن طول کوتاه‌ترین مسیرها بین هر جفت گره از یک گراف جهت‌دار n گره‌ای، از روش پویا استفاده می‌شود و در هر خانه $[i,j]$ از ماتریسی به نام D طول کوتاه‌ترین مسیرها بین گره‌های i تا j ثبت می‌شود. همچنین ماتریسی به نام P تشکیل می‌شود. در این ماتریس مقدار $P[i,j]$ برابر گره‌ای است که در مسیر بهینه از i به j باید از آن عبور کرد. فرض کنیم که برای گرافی با 4 گره، ماتریس P به شکل زیر باشد. کدام یک از گزینه‌های زیر، مسیر بهینه را برای رفتن از گره 1 به 3 مشخص می‌کند؟

$$P=\begin{bmatrix} 0 & 0 & 4 & 2 \\ 4 & 0 & 4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

۱) 1 4 3

۲) 1 2 3

۳) 1 4 2 3

۴) 1 2 4 3

۶ـ در مورد الگوریتم « دایکسترا» بر روی گرافی با E یال و V راس کدام یک از گزینه‌های زیر نادرست است؟

۱) پیچیدگی آن $O(E \lg V)$ است.

۲) اگر وزن هیچ یالی منفی نباشد الگوریتم درست کار می‌کند.

۳) فقط اگر گراف دور با وزن منفی نداشته باشد، الگوریتم درست کار نمی‌کند.

۴) هم برای گرافهای جهت‌دار است و هم برای گرافهای بدون جهت درست کار می‌کند.

۷ـ در یک آرایهٔ A به اندازهٔ n، اگر $i<j$ و $A[i]>A[j]$ می‌گوییم که زوج (i,j) یک «زوج ـ معکوس» (inversion) در A است. بیشترین تعداد زوج ـ معکوسها در یک آرایهٔ n عضوی چند تاست؟

۱) $\frac{n(n-1)}{2}$ ۲) n^2 ۳) n^2-n ۴) $\frac{n^2}{2}$

۸ـ می‌خواهیم یک آرایهٔ A با n عنصر مجزا را به $k=2^r$ زیر آرایهٔ $A_1,...A_k$ هر یک با تعداد عناصر $\lfloor n/k \rfloor$ یا $\lceil n/k \rceil$ تقسیم کنیم به طوری که برای همهٔ $i<j$ و هر $a_i \in A_i$ و $a_j \in A_j$ داشته باشیم $a_i<a_j$. بهترین الگوریتم برای این کار چه ویژگی دارد؟

۱) همیشه از $O(n \lg k)$ است.

۲) همیشه از $O(n \lg n)$ است.

۳) در حالت متوسط از $O(n \lg k)$ است.

۴) در حالت متوسط از $O(n)$ است.

۱ـ n دانشجو به صورت تصادفی در یک ردیف نشسته‌اند. فرض کنید دانشجوی i در مختصات $(x_i, 0)$ قرار دارد. مربی می‌خواهد در نقطه‌ای بایستد که مجموع فاصله‌اش تا همهٔ دانشجویان کمینه شود. با داشتن همهٔ n مختصات، محل قرار گرفتن مربی را در چه زمانی می‌توان به دست آورد؟

۱) $O(n)$ ۲) $O(n^2)$ ۳) $O(n \lg n)$ ۴) $O(\lg n)$

۲ـ رویهٔ زیر را در نظر بگیرید:

```
Function power (x,k)
        y ← x
        i ← k
        a ← 1
while i>0  do
if  i  is odd  then
 {
    a ← a×y
    y ← y×y
    i ← ⌊i/2⌋
 }
return a
```

این الگوریتم قخرار است مقدار x^k را محاسبه کند. مقدار مستقل از حلقه (loop in variant) این الگوریتم چیست؟ (یعنی چه عبارتی همواره در ابتدای حلقه درست است؟)

۱) $y^i a = x^k$ ۲) $y^i a^i = x^k$ ۳) $a(ay)^i = x^k$ ۴) $y(ay)^i = x^k$

۳ـ در گونهٔ جدیدی از مرتب‌سازی سریع (quick sort) برای انتخاب محور از میان n عنصر، $2\sqrt{n}+1$ عنصر اول آرایه را انتخاب می‌کنیم و با یک الگوریتم ساده مانند مرتب‌سازی درجی آن‌ها را مرتب می‌کنیم. عنصر میانهٔ این تعداد عنصر مرتب را به عنوان محور الگوریتم انتخاب می‌کنیم، بقیهٔ الگوریتم مانند قبل عمل می‌کند.

بدترین زمان اجرای این الگوریتم با کدام رابطهٔ بازگشتی نشان داده می‌شود؟

۱) $T(n)=2T(\sqrt{n})+n$ ۲) $T(n)=T(n-1)+n$

۳) $T(n)=2T(n-\sqrt{n})+n$ ۴) $T(n)=T(n-\sqrt{n})+T(\sqrt{n})+n$

۴ـ برای ضرب دو عدد A و B به ترتیب با n و 4n رقم برای $5000 \le n \le 10000$

۱) بهتر است از روش معمولی و کلاسیک برای ضرب آن‌ها استفاده کرد.

۲) بهتر است از الگوریتم تقسیم و حل استفاده کرد و فرقی نمی‌کند به چه ترتیب.

۳) بهتر است پس از قسمت کردن B به 4 بخش n رقمی، هر کدام از این بخش‌ها را در A و با روش تقسیم و حل ضرب کند.

۴) بهتر است پس از اضافه کردن 3n صفر به سمت چپ A ،عدد حاصل را با کمک الگوریتم تقسیم و حل در B ضرب کند.

۵ـ تابع زیر را در نظر می‌گیریم. فرض کنیم که $T(n)$ تعداد سطرهایی است که در اثر اجرای این تابع روی صفحه، نمایش داده خواهند شد و داریم $T(n)=1\ ,\ n\le 1$. کدام یک از تعاریف بازگشتی زیر صحیح است؟

```
function Concow (n)
  if n≤1 then
    writeln ('STOP')
    return 1
  else
    for i ← 1 to n do
      writeln (Concow (n div 2))
    return n
```

۱) $T(n)=\sum_{i=1}^{n} T\left(\left\lfloor \frac{i}{2} \right\rfloor\right)$

۲) $T(n)=nT\left(\left\lfloor \frac{n}{2} \right\rfloor\right)+n$

۳) $T(n)=nT\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$

۴) $T(n)=T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)+n$

۶ـ در یک گراف جهت‌دار $G=(V;E)$ وزن همهٔ یال‌ها برابر است. می‌خواهیم طول کوتاه‌ترین مسیرها از یک راس به نام S را تا بقیهٔ راس‌های G به دست آوریم. کدام‌یک از گزینه‌های زیر مرتبهٔ یک الگوریتم کارا برای حل این مسأله است؟

۱) $O\left(|V|^2\right)$

۲) $O(|V||E|)$

۳) $O(|V|\log|E|)$

۴) $O(|V|+|E|)$

۱ـ عمق یک گره در درایه‌ی i در یک هیپ با n گره چقدر است؟

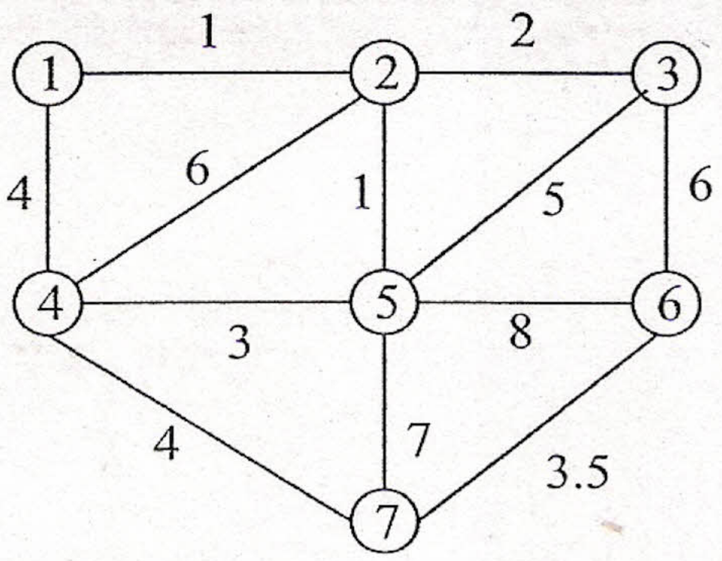
۱) $\lfloor \lg i \rfloor$ ۲) $i-1$ ۳) $\lfloor i/2 \rfloor$ ۴) $\lceil i/2 \rceil$

۲ـ میزان زمان لازم برای اجرای قطعه برنامهٔ زیر چگونه برآورد می‌شود؟

```
for i ← to n
  for j ← n to i
    for k ← 1 to n^2
      sum ← dum+A
```

۱) $O(n^3)$ ۲) $\theta(n^3)$ ۳) $\omega(n^4)$ ۴) $\theta(n^4)$

۳ـ الگوریتم Kruskal در مرحلهٔ چهارم خود، کدام کمان را از گراف به عنوان کمان درخت پوشای با حداقل هزینه در نظر می‌گیرد؟



۱) (1,4)
۲) (4,5)
۳) (6,7)
۴) هیچ‌کدام

۴ـ اگر A یک ماتریس 13×5 ، B ماتریس C,5×89 و D یک ماتریس 3×34 باشد، کمترین تعداد ضرب‌ها برای به دست آوردن حاصل‌ضرب A×B×C×D چقدر است؟

۱) 4055 ۲) 2856 ۳) 2820 ۴) 1742

۵ـ برای حذف یک عنصر از اول یک صف حلقوی که عناصر آن درون یک آرایه بنام queue از نوع ELEMENT ذخیره شده، برنامهٔ زیر ارایه شده است، ولی یک سطر از برنامه حذف شده است. کدام گزینه باید به جای سطر حذف شده قرار گیرد تا برنامه کامل شود؟

```
ELEMENT delete (int *front, int rear)
{
 if (*front = rear)
     return empty-q( );
 else
..................................... سطر حذف شده
     return queue[ *front];
}
```

۱) *front = *front − 1
۲) *front = *front + 1
۳) *front = *(front + 1)
۴) *front = (*front + 1) % MAXQSIZE

۶ـ آرایهٔ n عنصری A[1:n] را در نظر بگیرید. کدام‌یک از گزاره‌های ذیل، درست یا درست‌تر است؟

۱) الگوریتم INSERTION SORT، در بدترین حالت و به‌طور متوسط، زمان مصرفی مرتبهٔ n^2 دارد.

۲) الگوریتم QUICK SORT، در بدترین حالت و به‌طور متوسط زمانی در گروه n.logn دارد.

۳) الگوریتم QUICK SORT، همواره از الگوریتم INSERTION SORT از نظر زمان مصرفی، بهتر است.

۴) الگوریتم MERGE SORT، همواره از INSERTION SORT بهتر است، اما SELECTION SORT از هر دو، از نظر زمان مصرفی بهتر است.

۱ـ آرایهٔ n تایی A حاوی جایگشتی از اعداد 1 تا n است. برنامهٔ زیر این اعداد را مرتب می‌کند:

```
For i=1 to n -1 do
    While A[i] <> i do
        Swap (A[i] , A[A[i]])
```

بیشترین تعداد تعویض‌های برنامه فوق چندتاست؟

۱) n ۲) $n-1$ ۳) $n \log n$ ۴) $n(n-1)/2$

۲ـ در گونهٔ جدیدی از مرتب‌سازی سریع (quicksort)، برای انتخاب محور از میان n عنصر، $2\sqrt{n}+1$ عنصر اول آرایه را انتخاب می‌کنیم و با یک الگوریتم ساده (مانند Insertion Sort) آن‌ها را مرتب می‌کنیم. محور، عنصر میانهٔ این تعداد عنصر مرتب است. بقیهٔ الگوریتم مانند quicksort عمل می‌کند. کدام‌یک از رابطه‌های بازگشتی زیر، زمان اجرای این الگوریتم را در بدترین حالت نشان می‌دهد؟

۱) $T(n)\le T(n-\sqrt{n})+O(\sqrt{n})$ ۲) $T(n)\le T(2\sqrt{n})+T(n-2\sqrt{n})+O(n)$

۳) $T(n)\le T(\sqrt{n})+T(n-\sqrt{n})+O(n)$ ۴) $T(n)\le T(2\sqrt{n})+T(n-2\sqrt{n})+O(\sqrt{n})$

۳ـ در مورد گراف‌های جهت‌دار و وزن‌دار با وزن مثبت و منفی، ولی بدون دور منفی کدام‌یک از گزینه‌های زیر درست است؟

۱) الگوریتم بلمن فورد همیشه درست کار می‌کند. ۲) الگوریتم دایکسترا همیشه درست کار می‌کند.

۳) الگوریتم فلوید ممکن است در دور بیفتد. ۴) الگوریتم کراسکال ممکن است درست کار نکند.

۴ـ جواب رابطه‌ی بازگشتی $T(n)=8T\left(\frac{n}{9}\right)+n\lg n$، کدام‌یک از گزینه‌های زیر است؟

۱) $\theta(\lg n)$ ۲) $\theta(n)$ ۳) $\theta(n^2\lg n)$ ۴) $\theta(n\lg n)$

۵ـ $f(n)$ زمان اجرای الگوریتمی مطابق رابطهٔ بازگشتی زیر محاسبه شده است. کدام گزینه صحیح است؟

$$f(n)=\begin{cases} a & \text{اگر } n=1 \\ bn^2+n.f(n-1) & \text{در غیر این‌صورت} \end{cases}$$

۱) $f(n)=\theta(2^n)$ ۲) $f(n)=\theta(n!)$

۳) $f(n)=\theta(2^{n!})$ ۴) $f(n)=\theta(n!)^2$

۶ـ در یک گراف همبند بدون جهت و وزن‌دار، کدام‌یک از گزینه‌های زیر در مورد درخت پوشالی کمینهٔ ایجاد شده توسط الگوریتم‌های prim و kruskal درست است؟

۱) با هر وزنی این دو الگوریتم درخت‌های یکسانی تولید می‌کنند.

۲) درخت تولید شده با الگوریتم Prim مستقل از راس شروع در آن الگوریتم است.

۳) اگر وزن یال‌ها مجزا باشند، درخت‌های تولیدی این دو الگوریتم یکسان هستند.

۴) اگر وزن همه یال‌ها برابر باشند، این دو الگوریتم درخت‌های یکسانی را تولید می‌کنند.

۷ـ به منظور sort صعودی یک آرایه، از heapsort استفاده نموده و max − heap می‌سازیم. در مورد زمان اجرای این الگوریتم کدام گزینه صحیح است؟

۱) اگر آرایه از ابتدا به صورت صعودی مرتب شده باشد، زمان مورد نیاز $\theta(n)$ است.

۲) اگر آرایه از ابتدا به صورت نزولی مرتب شده باشد، زمان مورد نیاز $\theta(n)$ است.

۳) اگر آرایه از ابتدا به صورت نزولی مرتب شده باشد، زمان مورد نیاز $\theta(n^2 \log n)$ است.

۴) اگر آرایه از ابتدا به صورت صعودی مرتب شده باشد، زمان مورد نیاز $\theta(n \log n)$ است.

۸ـ در یک گراف جهت‌دار بدون دور و وزن‌دار می‌خواهیم طولانی‌ترین مسیر از یک راس مشخص را به بقیه راس‌ها به‌دست آوریم. این کار را با چه مرتبهٔ زمانی می‌توان انجام داد؟

۱) $O(E|E|)$

۲) $O(|E|+|V|)$

۳) $O(|V^3|)$

۴) این کار را در زمان چند جمله‌ای نمی‌توان انجام داد.

۱ـ n کار با زمان‌های پردازش $p_1<p_2<...<p_n$ قرار است در m پردازندهٔ کاملاً یکسان اجرا شوند. به این صورت که کارهایی را که هر کدام از پردازنده‌ها قرار است به ترتیب اجرا کند از قبل مشخص می‌کنیم. بدیهی است که زمان اتمام کارهای یک پردازنده جمع زمان‌های پردازش کارهایی است که به آن پردازنده تخصیص داده شده است. برای تخصیص، کارها را به ترتیب شمارهٔشان (از زمان کم تا زیاد) انتخاب می‌کنیم و هر کار را به پردازنده‌ای تخصیص می‌دهیم که همهٔ کارهای تخصیص داده شده به آن نسبت به بقیه پردازنده‌ها زودتر تمام شوند. حداکثر زمان اتمام همه کارهای چقدر است؟ (فرض کنید $A=\sum_{i=1}^{n} p_i$).

۱) $\frac{A}{m}$

۲) $\frac{A}{m}-p_n$

۳) $\frac{A}{m}+p_n$

۴) $\frac{A-p_n}{m}+p_n$

۲ـ الگوریتم فلوید کوتاه‌ترین مسیر بین همهٔ زوج نقطه‌ها را در گراف جهت‌دار و وزن‌دار G محاسبه می‌کند. اگر یالهای با وزن منفی هم داشته باشیم، کدام‌یک از گزینه‌های زیر بهترین توصیف این الگوریتم است؟

۱) با این الگوریتم می‌توان وجود یا عدم جود دور منفی را تشخیص داد.

۲) الگوریتم برای یالهای منفی، ولی بدون دور منفی، ممکن است در دور بیفتد.

۳) الگوریتم برای یالهای منفی، ولی بدون دور منفی متوقف می‌شود، ولی درست کار نمی‌کند.

۴) الگوریتم با یالهای منفی، ولی بدون دور منفی، متوقف می‌شود، ولی جواب آن همیشه غلط است.

۳ـ کدام‌یک از عبارت‌های زیر جواب رابطهٔ بازگشتی زیر است؟

$$T(n)\le T\left(\frac{n}{5}\right)+T\left(\frac{7n}{10}\right)+\theta(n)$$

۱) $T(n)\le \sum_{i=0}^{2\log_5^n} \left(\frac{9}{10}\right)^i n$

۲) $T(n)\le \sum_{i=0}^{\log_{10}^n} \left(\frac{1}{5}\right)^i n$

۳) $T(n)\le \sum_{i=0}^{2\log_5^n} \left(\frac{7}{10}\right)^i n$

۴) $T(n)\le \sum_{i=0}^{\log_{10}^n} \left(\frac{5}{10}\right)^i n$

۴ـ زیر برنامهٔ New Quick Sort را برای مرتب‌سازی آرایهٔ T به ترتیب نزولی در نظر می‌گیریم. Pivot(T,p,k) عناصر T را حول محور p طوری می‌چرخاند که پس از اتمام کار ،کلیهٔ عناصر بزرگ‌تر از p ،قبل از خانهٔ k و عناصر کوچک‌تر و مساوی، بعد از آن در T جای می‌گیرند. (k پارامتر خروجی است.) Insertion Sort به روش درجی آرایه را به ترتیب نزولی مرتب می‌کند. پیچیدگی زمانی این زیر برنامه برای کدام‌یک از آرایه‌های زیر مطمیناً کمتر از پیچیدگی زمانی الگوریتم معمول Quick Sort است؟

```
New Qucik Sort (T[1..n])
    p ← T[1]
    pivot(T,p,k)
    Insertion Sort (T[1..k-1])
    Insertion Sort (t[k+1..n])
```

۱) برای همه حالات ورودی.

۲) آرایهٔ ابتدایی مرتب شدهٔ نزولی باشد.

۳) ارایهٔ ابتدایی مرتب شدهٔ صعودی باشد.

۴) p دقیقاً میانهٔ آرایه باشد و آرایهٔ ابتدایی نامرتب باشد.

۵ـ می‌خواهیم k تا کوچک‌ترین عناصر یک آرایه را از کوچک به بزرگ به صورت مرتب به دست آوریم. کدام‌یک از الگوریتم‌های زیر درست و سریع‌تر از بقیه است؟

۱) عناصر را مرتب می‌کنیم و سپس k تای اول آن به ترتیب بر می‌داریم.

۲) یک Heap بر روی این عناصر می‌سازیم و سپس k بار کوچک‌ترین عنصر را به دست آوریم.

۳) k امین عنصر را انتخاب می‌کنیم. این عنصر را محور قرار داده و عمل افراز را انجام می‌دهیم و سپس کار را دنبال می‌کنیم.

۴) عنصر میانه را به دست آورده بر اساس این محور عمل افراز را انجام می‌دهیم. سپس در یکی از بخش‌ها، کار را دنبال می‌کنیم.

۶ـ رابطهٔ بازگشتی زیر داده شده است:

$$\begin{cases} G_2=4 \quad G_1=2 \quad G_0=1 \\ G_n=G_{n-1}+2G_{n-2}+G_{n-3} \qquad n\geq 3 \end{cases}$$

برای $n\geq 3$ کدام یک از گزینه‌های زیر بهترین جواب این رابطه است؟

۱) $G_n\leq 2^n$ ۲) $G_n\leq 4^n$ ۳) $G_n\leq 2^{n+1}$ ۴) $G_n\leq 4^{n+1}$

۷ـ یک گراف همبند $G=(V,E)$ را در نظر بگیرید که وزن‌های یال‌های آن مساوی 1 است. می‌خواهیم درخت فراگیر کمینه (MST) این گراف را پیدا کنیم.

۱) این کار را می‌توان در $O(|E|\log|E|)$ انجام داد.

۲) این کار را می‌توان در $O(|V||E|)$ انجام داد.

۳) این کار را می‌توان در $O(|V|\log|E|)$ انجام داد.

۴) این کار را می‌توان در $O(|E|)$ انجام داد.

۱ـ کدام‌یک از گزاره‌های ذیل در مورد محاسبهٔ درخت پوشای می‌نیمم برای یک گراف همبند (Connected) درست است؟

۱) در روش Prim در هر مرحله، همبندی درخت رعایت می‌شود و زمان مصرفی الگوریتم Prim و الگوریتم Kruskal، یکسان است.

۲) در روش Prim در هر مرحله، همبندی درخت رعایت می‌شود، اما در روش Kruskal در مرحله پایانی، این امر اتفاق می‌افتد. اما زمان مصرفی روش Prim بهتر است.

۳) در روش Kruskal، در هر مرحله، همبندی درخت رعایت می‌شود، اما در روش Prim در مرحله پایانی، این امر اتفاق می‌افتد. اما زمان مصرفی روش Prim بهتر است.

۴) در روش Kruskal ، در هر مرحله، همبندی درخت رعایت نمی‌شود. اما در پایان الگوریتم، درخت همبند است، اما زمان مصرفی آن از روش Prim بهتر است.

۲ـ کدام گزینه زیر صحیح است؟

۱) الگوریتم بازگشتی، یک الگوریتم از بالا به پایین است که همهٔ زیر مسایل را حداقل یک بار حل می‌کند.

۲) الگوریتم طراحی شده توسط روش برنامه‌ریزی پویا، یک الگوریتم از پایین به بالاست که همهٔ زیر مسایل را تنها یک بار حل می‌کند.

۳) الگوریتم‌های طراحی شده توسط روش برنامه‌ریزی پویا (نوع استاندارد یا Memoized) زیر مسایل حل شده را فقط یک بار حل می‌کنند.

۴) الگوریتم طراحی شده توسط روش برنامه‌ریزی پویا و از نوع Memoized یک الگوریتم بالا به پایین است که همهٔ زیر مسایل لازم را تنها یک بار حل می‌کند.

۳ـ ۴ماتریس زیر را با ابعاد داده شده در نظر بگیرید:

$A_{10\times2}, B_{2\times25}, C_{25\times3}, D_{3\times4}$

کدام‌یک از گزینه‌های زیر مربوط به پرانتزبندی بهینه برای محاسبهٔ حاصل‌ضرب آن‌هاست؟

۱) $(A\times B)\times(C\times D)$

۲) $(A\times(B\times C))\times D$

۳) $A\times((B\times C)\times D)$

۴) $((A\times B)\times C)\times D$

۴ـ دومین عدد بزرگ یک آرایهٔ n عضوی را با چند مقایسه می‌توان پیدا نمود؟

۱) $n+\lfloor \log_2 n \rfloor$

۲) $(n-1)+(n-2)$

۳) $n+\lceil \log_2 n \rceil-2$

۴) $n+\lfloor \log_2 n \rfloor-2$

۵ـ کدام‌یک از گزینه‌های زیر صحیح است؟

۱) $n^2 \sin n \in \Omega(n)$

۲) $n^2 \sin n \in O(n)$

۳) $n \in \Theta(n^2 \sin n)$

۴) $n \in \Omega(n^2 \sin n)$

۶ـ در الگوریتم پیدا کردن دو عنصر بیشینه و کمینه در یک آرایه با N عنصر: (مطابق شکل روبرو)

```
MINMAX(A)
    min ← 1
    max ← 1
    for i ← 2 to N do
        if A[i]<A[min]
            then min ← i
            else if A[i]>A[max]
                then max ← i
```

حداکثر و حداقل تعداد مقایسه‌های دو عنصر از آرایه به ترتیب برابر است با:

۱) $2n$ و n ۲) هر دو $n-1$ ۳) هر دو n ۴) $2(n-1)$ و $(n-1)$

۷ـ می‌خواهیم دو هیپ مینیمم را با اندازه‌های n و m در یک هیپ به اندازهٔ $n+m$ ادغام کنیم. فرض کنید که هیپ خروجی می‌تواند به صورت درخت باشد. با فرض $n>m$، کدام یک از گزینه‌های زیر همیشه درست است؟

۱) این کار را همیشه می‌توان در $O(\lg^2 n)$ انجام داد.

۲) این کار را همیشه می‌توان در $O(\lg n)$ انجام داد.

۳) این کار را می‌توان در $O(\lg n)$ انجام داد و هیپ خروجی لزوماً به‌صورت آرایه، نیست.

۴) این کار را می توان در $O(\lg^2 n)$ انجام داد، ولی هیپ خروجی لزوماً به صورت آرایه، نیست.

۸ـ یک مسیر ساده در گراف مسیری است که از هیچ راسی بیش از یک بار عبور نکند. مسالهٔ پیدا کردن طولانی‌ترین مسیر ساده را در گراف بدون وزن و جهت‌دار G در نظر می‌گیریم. برای حل این مساله:

۱) می‌توان از روش برنامه‌ریزی پویا استفاده کرد چون اصل بهینه‌سازی برقرار هست.

۲) با وجودی که اصل بهینه‌سازی برقرار است، نمی‌توان از روش برنامه‌ریزی پویا استفاده کرد.

۳) نمی‌توان از روش برنامه‌ریزی پویا استفاده کرد چون اصل بهینه‌سازی برقرار نیست.

۴) با وجودی که اصل بهینه‌سازی برقرار نیست می‌توان از روش برنامه‌ریزی پویا استفاده کرد.

۹ـ در مسالهٔ کوله پشتی صفر و یک، با داشتن n شی و ظرفیت w برای کوله پشتی، کدام گزینه نادرست است؟

۱) کارآترین الگوریتم برای این مساله، الگوریتم حریصانه با $\theta(n)$ زمان مورد نیاز است.

۲) الگوریتم برنامه‌ریزی پویا در بدترین حالت برای این مساله، $\theta(n.w)$ زمان نیاز دارد.

۳) الگوریتم انشعاب و تحدید (branch and bound) در بدترین حالت برای این مساله $\theta(n^n)$ زمان نیاز دارد.

۴) الگوریتم پسگرد (backtracking) در بدترین حالت برای این مساله $\theta(2^n)$ زمان نیاز دارد.

۱۰ـ مرتبهٔ زمانی الگوریتمی با تابع زمانی زیر، برابر کدام گزینه است؟

$$T(n)=\begin{cases}0 & n=0\\ 1 & n=1\\ 3T(n-1)+4T(n-2) & \text{other wise}\end{cases}$$

۱) n^2 ۲) 4^n ۳) $2^n \log n$ ۴) $n^4 \log n$

۱ـ در یک درخت جستجوی باینری با n گره، تعداد مقایسه‌ها برای جستجوی یک عنصر، حداکثر برابر است با:

۱) $O\left(\frac{n}{2}\right)$ ۲) $O(n)$ ۳) $O(\sqrt{n})$ ۴) $O(\lg n)$

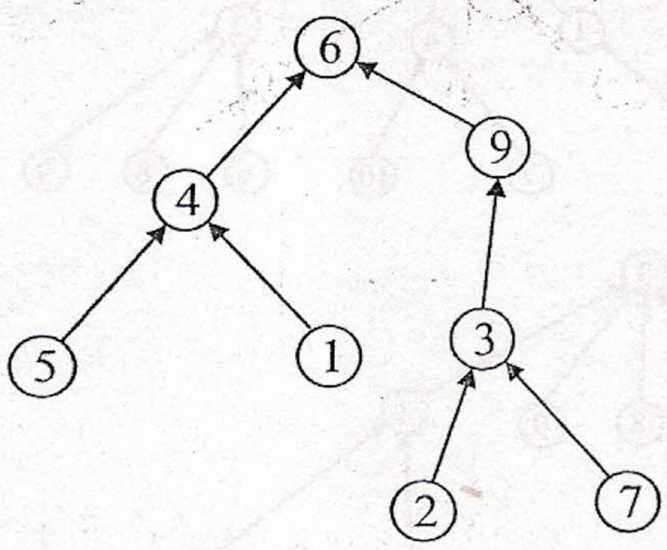
۲ـ یک الگوریتم مرتب‌سازی صعودی در زمان اجرا در چهارمین تکرار خود دارای ارزش‌های زیر است. این الگوریتم چه نوع مرتب‌سازی است؟ 15,18,30,34,41,0,91,32,49

۱) Bubble ۲) Selection ۳) Insertion ۴) Quicksort

۳ـ اگر دنبالهٔ درجهٔ یک گراف بدون جهت 3,3,3,3,2,2 باشد در آن صورت تعداد یال‌های پسین (backedge) این گراف در جستجوی DFS برابر است با:

۱) 2 ۲) 3 ۳) 4 ۴) 5

۴ـ اگر درخت زیر را به صورت آرایه نمایش دهیم، کدام آرایه بهترین ساختار برای انجام عملیات بر روی درخت است؟



۱)

1	2	3	4	5	6	7	8	9
6	4	5	1	9	3	2	7	−1

۲)

1	2	3	4	5	6	7	8	9
6	4	9	5	1	3	−1	2	7

۳)

1	2	3	4	5	6	7	8	9
6	4	9	5	1	3	2	7	−1

۴)

1	2	3	4	5	6	7	8	9
4	3	9	6	4	−1	3	−1	6

۵ـ فرض می‌کنیم n مجموعه تک عنصری متمایز از هم داریم. اگر u عمل union انجام دهیم، حداقل تعداد مجموعهٔ تک عنصری که باقی مانده برابر است با:

۱) $n-2u$ ۲) $\max\{n-2u,0\}$ ۳) $\max\{2u,n\}$ ۴) $\min\left\{u,\frac{n}{2}\right\}$

۶ـ الگوریتم پیمایش درخت در زمان $O(d)$ اجرا می‌شود. d چه می‌باشد؟

۱) تعداد برگ‌های درخت ۲) عمق درخت

۳) درجهٔ درخت ۴) تعداد گرههای درخت

۷ـ ماکزیمم تعدا مقایسه برای min−heap کردن یک max−heap با n گره برابر است با:

۱) $O(n+\log n)$ ۲) $O(\log n)$

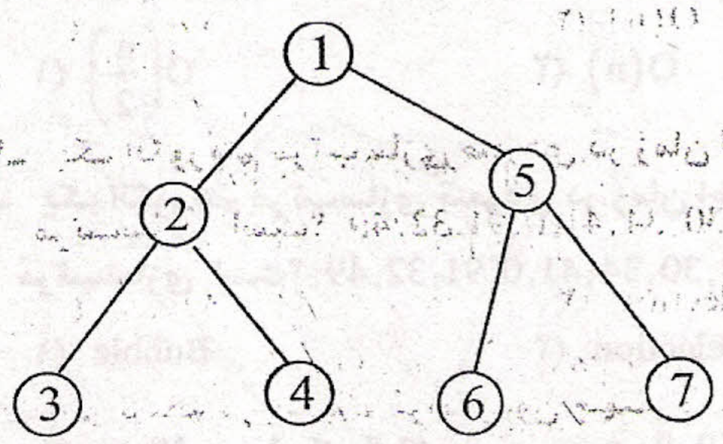
۳) $O(n\log n)$ ۴) $O(2n)$

۸ـ کدام گزینه صحیح است؟

۱) $0<\varepsilon<0.1$ $n^3\log n=O\left(n^{3+\varepsilon}\right)$ ۲) $\sqrt{n}=O(\log n)$

۳) $0<\varepsilon<0.1$ $n^{1+\varepsilon}=O(n\log n)$ ۴) $n^2=O\left(\frac{n^2}{\log n}\right)$

۹ـ اگر در پیمایش DFS گراف غیر جهت‌دار $G=(V,E)$، درخت فراگیر آن برابر با درخت زیر باشد در آن صورت کدام گزینه صحیح است؟ در گراف $G=(V,E)$:



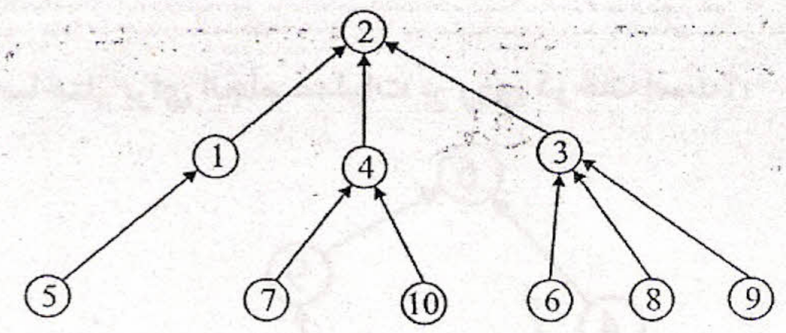
۱) بین دو راس 5 و 2 می‌تواند یال وجود داشته باشد.

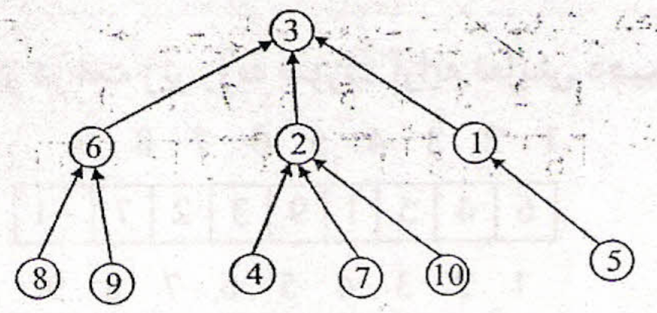
۲) بین دو راس 6 و 2 و دو راس 7 و 6 می‌تواند یال وجود داشته باشد.

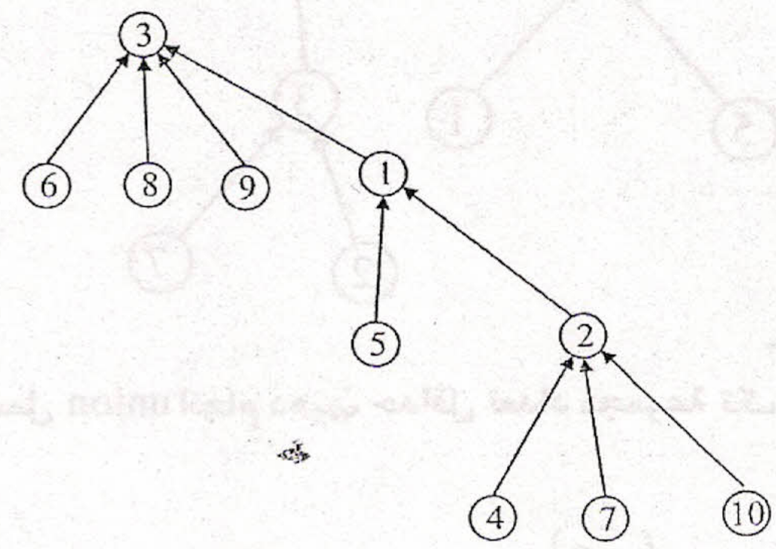
۳) بین دو راس 1 و 4 می‌تواند یال وجود داشته باشد.

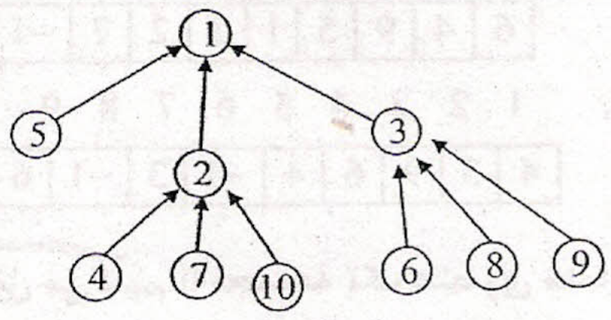
۴) بین دو راس 7 و 6 می‌تواند یال وجود داشته باشد.

۱۰ـ درخت حاصل از اجتماع وزن‌دار سه مجموعه (دو به دو از چپ به راست) $(3,6,8,9)$ و $(2,4,7,10)$ و $(1,5)$ ،کدام درخت می‌تواند باشد؟

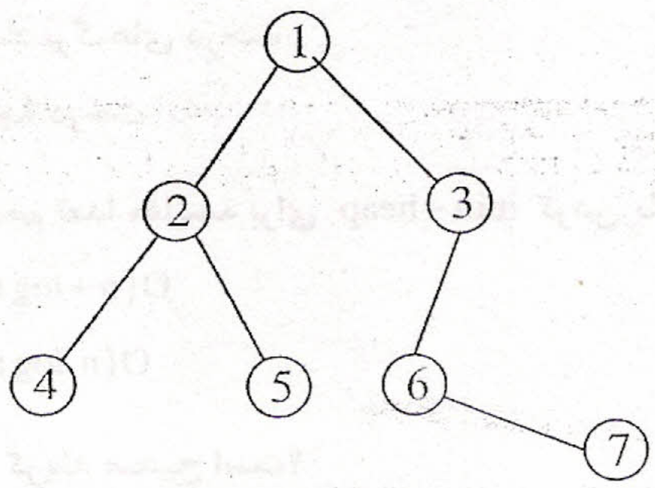
۱)


۲)


۳)


۴)


۱۱ـ اگر در پیمایش BFS یک گراف غیر جهت‌دار $G=(V,F)$ درخت فراگیر حاصل برابر با درخت زیر باشد در آن صورت در گراف $G=(V,E)$ کدام گزینه صحیح است؟



۱) بین دو راس 5 و 4 می‌تواند یال وجود داشته باشد.

۲) بین دو راس 5 و 1 می‌تواند یال وجود داشته باشد.

۳) بین دو راس 7 و 3 و دو راس 4 و 1 می‌تواند یال وجود داشته باشد.

۴) بین دو راس 7 و 1 و دو راس 5 و 1 می‌تواند یال وجود داشته باشد.

۱۲ـ برای یک درخت جستجوی باینری که در گره‌های آن تعدادی عدد ذخیره شده است. با داشتن پیمایش Pre – order آن، کدام جمله صحیح است؟

۱) فقط با یک پیمایش نمی‌توان درخت را ساخت.

۲) می‌توان درخت را در $O(n^2)$ ساخت.

۳) می‌توان درخت را در $O(n \log n)$ ساخت .

۴) می‌توان درخت را در $O(n)$ ساخت.

۱۳ـ شرط اصلی برای اینکه بتوان از روی دو پیمایش داده شدهٔ یک درخت باینری، درخت اصلی را ساخت، چیست؟

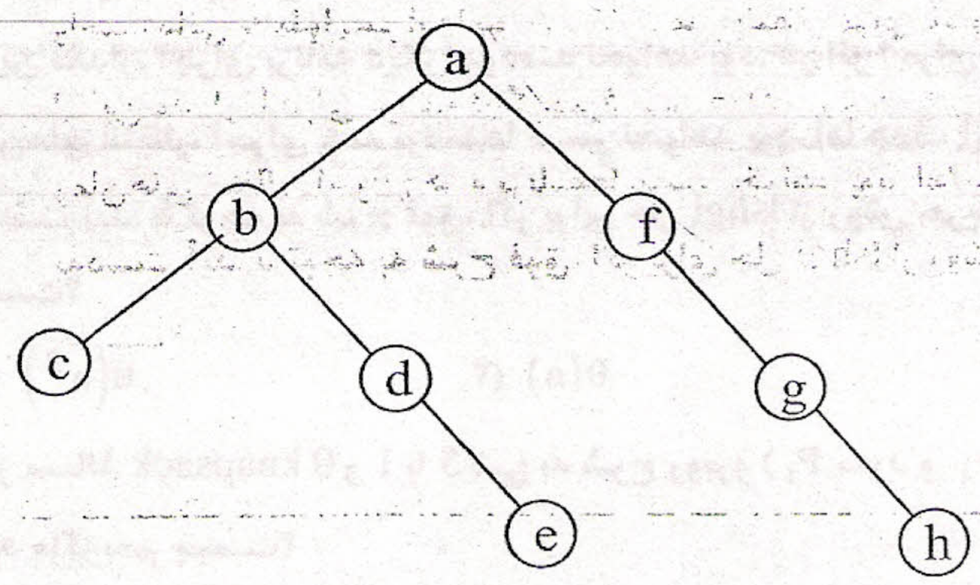
۱) تمام داده‌های درخت متمایز باشند.

۲) فقط ریشه از بقیه متمایز باشد.

۳) داده‌های زیر درختِ چپ ریشه از داده‌های زیر درخت راست متمایز باشند، ولی در خود زیر درخت‌های چپ و راست می‌تواند داده تکراری وجود داشته باشد.

۴) داده‌های موجود در برگ‌های درخت متمایز باشند.

۱۴ـ در درخت جستجوی دودویی زیر با حذف ریشه کدام یک از عناصر زیر جایگزین می‌شود؟



۱) b یا f

۲) c یا h

۳) e یا f

۴) h یا e

۱۵ـ الگوریتم Bucketsort می‌تواند n داده را که در محدوده 0 تا n^2-1 هستند در زمان $O(n)$ مرتب کند. چرا این با قضیه‌ای که حد پایین برای مرتب‌کردن $O(n \log n)$ است تناقض ندارد؟

۱) حد پایین فقط برای الگوریتم‌هایی که روی داده‌هایی با تعداد کمتر از 100 کار می‌کنند، صادق است.

۲) حد پایین فقط برای الگوریتم‌های که بر اساس درخت تصمیم‌گیری عمل می‌کنند، صادق است.

۳) حد پایین فقط برای الگوریتم‌های که عناصر کنار هم را با یکدیگر عوض می‌کنند، صادق است.

۴) حد پایین فقط برای الگوریتم‌های که روی داده‌هایی اعداد حقیقی کار می‌کنند صادق است.

۱ـ اگر در ضرب ماتریس‌ها به روش استراسن (Strassen)، مسالهٔ کوچک، ضرب ماتریس‌های 2*2 باشد، برای ضرب دو ماتریس 8*8 چند ضرب عددی صورت می‌پذیرد؟

۱) 57 ۲) 343 ۳) 392 ۴) 512

۲ـ در پیاده سازی الگوریتم Branch &Bound Best First Search برای پیمایش درخت فضای حالت از چه ساختمان داده‌ای استفاده می‌شود؟

۱) BST ۲) Heap ۳) Stack ۴) Queue

۳ـ برای اجرای n برنامه آماده به اجرا روی یک کامپیوتر، lshgi بهینه‌سازی به شکل زیر تعریف شده است. اجرای هر برنامه یک واحد زمانی به طول می‌انجامد. حداکثر زمان انتظار برای شروع به اجرای هر برنامه، D_i واحد زمان است و در صورت انتظار بیش از این مقدار، اجرای برنامه دیگر سودمند نخواهد بود. در اثر اجرای هر برنامه، سود p_i بدست می‌آید. بدیهی است که با توجه به زمان‌های انتظار، اجرای همه برنامه‌ها میسر نخواهد بود. اما هدف آن است که برنامه‌ها به گونه‌ای اجرا شوند که بیشترین سود به‌دست آید. با توجه به شرح فوق، اگر برای حل lshgi از روش حریصانه (Greedy) استفاده شود، پیچیدگی زمانی الگوریتم چیست؟

۱) $\theta(n^2)$ ۲) $\theta(n)$ ۳) $\theta(n^3)$ ۴) $\theta(n \log n)$

۴ـ در مسالهٔ knapsack 0 و 1 با 5 شئ به شرح روبرو (P_i سود و w_i وزن شئ i ام) ظرفیت کوله‌پشتی برابر 13kg است. مقدار سود ماکزیمم چیست؟

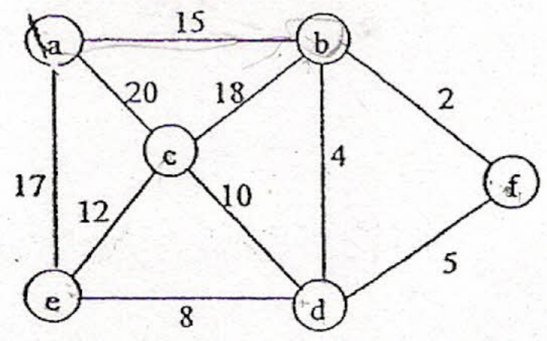
i	1	2	3	4	5
P_i	35	30	20	12	3
W_i	7	5	2	3	1

۱) 65 ۲) 68 ۳) 70 ۴) 80

۵ـ اگر در الگوریتم Merge Sort برای مرتب‌سازی لیست‌های دارای کمتر از 20 عنصر از الگوریتم Insertion Sort استفاده شود، پیچیدگی زمانی الگوریتم چه خواهد شد؟

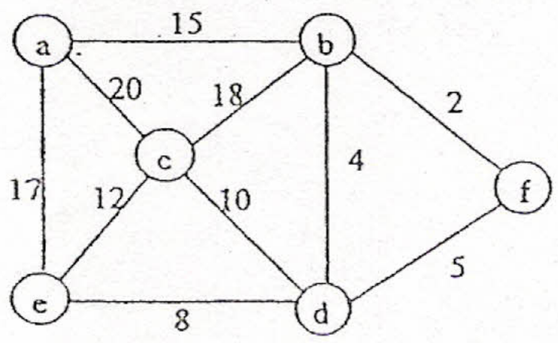
۱) $\theta(n^2)$ ۲) $\theta(n)$ ۳) $\theta(n^2 \log n)$ ۴) $\theta(n \log n)$

۶ـ ترتیب انتخاب یال‌ها در الگوریتم Kruskal بر روی گراف زیر کدام مورد است(از چپ به راست)؟



۱) (b,f) , (b,d) , (e,d) , (c,d) , (a,b)
۲) (a,b) , (b,f) , (b,d) , (f,d) , (c,d)
۳) (b,f) , (b,d) , (e,d) , (e,c) , (a,e)
۴) هیچ کدام

۷ـ ترتیب انتخاب یال‌ها در الگوریتم Prim بر روی گراف زیر با شروع از راس a کدام مورد است (از چپ به راست)؟



۱) (b,f) , (b,d) , (e,d) , (c,d) , (a,b)
۲) (a,b) , (b,f) , (f,d) , (d,e) , (e,c)
۳) (a,b) , (b,f) , (b,d) , (d,e) , (d,c)
۴) هیچ کدام

۸ـ اگر قرار باشد 6 سخنرانی زیر را با شروع و پایان مشخص طوری برنامه‌ریزی کنیم که بیشترین تعداد سخنرانی در یک سالن قابل ارایه باشد، بیشترین تعداد سخنرانی‌های ممکن کدام است؟

شماره سخنرانی	1	2	3	4	5	6
شروع	2	1	3	3	4	6
پایان	9	2	4	5	7	8

۱) 2 ۲) 5 ۳) 4 ۴) 3

۹ـ رشته abcabbaccaabdffe را با روش کدگذاری هافمن کدگذاری کنیم. طول کد چند بیت خواهد شد؟

۱) 38 ۲) 36 ۳) 34 ۴) هیچ‌کدام

۱۰ـ کمترین مرتبهٔ زمانی الگوریتم پیدا کردن i امین کوچک‌ترین عنصر از میان n عنصر کدام است؟

۱) $O(n^3)$ ۲) $O(n)$ ۳) $O(n \log n)$ ۴) هیچ‌کدام

۱ـ کدام‌یک از عبارات زیر جواب رابطهٔ بازگشتی $T(n)\le T\left(\frac{n}{5}\right)+T\left(\frac{7n}{10}\right)+n$ است؟

۱) $T(n)\le \sum_{i=0}^{\log_{10}^{n}} \left(\frac{7}{10}\right)^{i} n$

۲) $T(n)\le \sum_{i=0}^{\log_{5}^{n}} \left(\frac{7}{10}\right)^{i} n$

۳) $T(n)\le \sum_{i=0}^{\log_{10}^{n}} \left(\frac{9}{10}\right)^{i} n$

۴) $T(n)\le \sum_{i=0}^{\log_{\frac{2}{5}}^{n}} \left(\frac{9}{10}\right)^{i} n$

۲ـ دو آرایهٔ $X[1..n]$ و $Y[1..n]$ مرتب شده‌اند. سریع‌ترین الگوریتم برای یافتن میانهٔ $2n$ عضو آرایه‌های X و Y کدام است؟

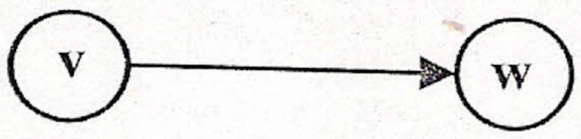
۱) روش استفاده از ادغام است که $\theta(n)$ زمان نیاز دارد.

۲) روش استفاده از ادغام است که $\theta(\log n)$ زمان نیاز دارد.

۳) یک الگوریتم تقسیم و غلبه است که نمونه‌های n تایی را به زیر نمونه‌های $\frac{n}{2}$ تایی تبدیل می‌کند و $\theta(\sqrt{n})$ زمان نیاز دارد.

۴) یک الگوریتم تقسیم و غلبه است که نمونه‌های n تایی را به زیر نمونه‌های $\frac{n}{2}$ تایی تبدیل می‌کند و $\theta(\log n)$ زمان نیاز دارد.

۳ـ در یک گراف جهت‌دار در هر گره عددی صحیح قرار دارد. یک یال جهت‌دار رابطهٔ کوچک‌تری را بیان می‌کند. تعداد سیکل‌های سادهٔ گرافی با مشخصات بالا با n گره برابر است با:



۱) $2n-n+1$

۲) n^2-3n+2

۳) $n^2-\frac{3}{2}n+1$

۴) $\frac{n^2-3n+2}{2}$

۴ـ T درخت پوشای مینیمم برای یک گراف $G=(V,E)$ است. وزن یک یال $e\in E$ را کاهش می‌دهیم. با چه مرتبه‌ای می‌توان درخت مینیمم جدید را به دست آورد.

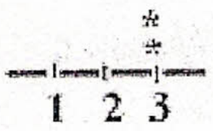
۱) $O(|V|)$

۲) $O(|V|+|E|)$

۳) $O(|E|\log|V|)$

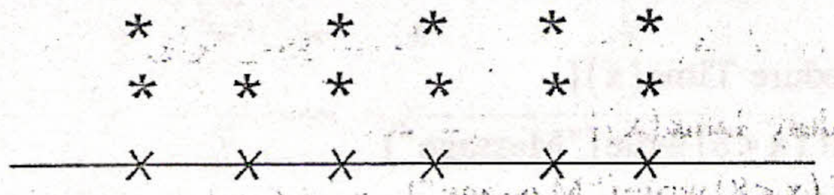
۴) $O(|V|\log|E|)$

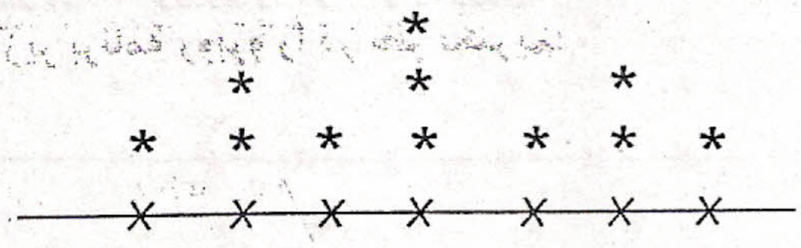
۵ـ الگوریتم روبرو را در نظر بگیرید: فرض کنید الگوریتم g، با تعریف $g(x,y)$، در محل x ، y تا علامت ستاره می‌نویسد. برای مثال

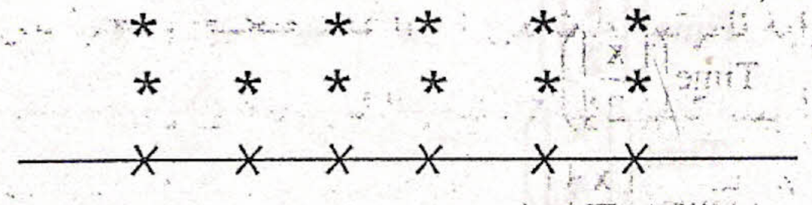
$g(3,2)=$ 

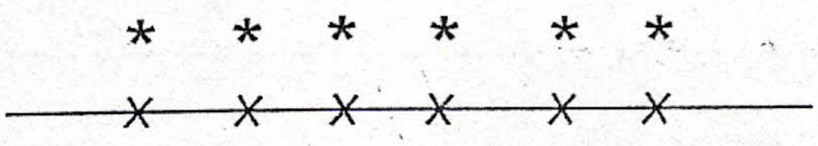
خروجی الگوریتم $f(0,8,3)$ کدام یک از موارد ذیل است؟

```
f(int a,int b,int c)
{
int m=(a+b)/2;
if (c>0)
{
g(m,c);
f(a,m,c-1);
f(m,b,c-1);
}
```

۱)


۲)


۳)


۴)


۶ـ تابع $T(n)=2T(\lfloor\sqrt{n}\rfloor)+\log n$ را در نظر بگیرید. کدام‌یک از روابط زیر برای $T(n)$ درست است؟

۱) $T(n)=O(\log\log n)$

۲) $T(n)=O(\log n\log\log n)$

۳) $T(n)=O(n\log\log n)$

۴) $T(n)=O(n.\log n\log n)$

۷ـ الگوریتم زیر هزینهٔ ضرب بهینه n تا ماتریس $M_1\times\ldots M_2\times\ldots\times M_n$ را مشخص می‌کند که در آن ابعاد M_i برابر $d_{i-1}\times d_i$ است. مجموع تعداد خواندن درایه‌های c در این الگوریتم چندتاست؟

```
for i:=to n do
    c[i,i]=0
for j:=2 to n do
    for i=1 to n-1+1 do
        { j:=i+1
            c[i,j]=min (c[i,k]+c[k+1,j]+d_{i-1}*d_k*d_j)
                    i<k≤j
 }
```

۱) $\sum_{l=1}^{n}(2l-1)(n-1)$

۲) $\sum_{l=1}^{n}(2l-1)(n-l)$

۳) $\sum_{l=1}^{n}2(l-1)(n-l+1)$

۴) $\sum_{l=1}^{n}2(l-1)(n-l)$

۱ـ زیر برنامهٔ روبرو را در نظر بگیریم:

```
Procedure Time(x){
    if (x<8) write("Massage")
    else{
        Time(⌊x/2⌋)
        Time(⌊x/4⌋)
        Time(⌊x/8⌋)
    }
}
```

اگر $t(x)$ تعداد دفعات چاپ پیغام Message روی صفحه برای فراخوانی $time(x)$ باشد کدام یک از موارد زیر صحیح است؟

۱) $t(9)=4;t(16)=5;t(36)=8$

۲) $t(9)=3;t(16)=5;t(36)=9$

۳) $t(9)=3;t(16)=6;t(36)=8$

۴) $t(9)=4;t(16)=6;t(36)=9$

۲ـ یک شمارندهٔ k بیتی در ابتدا مقدار صفر دارد. این شمارنده را 2^k بار و هر بار به مقدار 1 واحد افزایش می‌دهیم. مجموع تعداد تغییرات بیت‌های این شمارنده از 0 به 1 و برعکس چندتاست؟

۱) $\sum_{i=1}^{k} \frac{n}{2^i}$

۲) $\theta(n^2)$

۳) $\sum_{i=1}^{k} \frac{n}{2^{i-1}}$

۴) $\theta(n\log n)$

۳ـ کدام‌یک از گزینه‌های زیر غلط است؟

۱) وجود یک یال cross در پیمایش عمق اول معرف عدم وجود سیکل هامیلتونی است.

۲) وجود یک یال cross در پیمایش عمق اول نشانگر وجود گره مفصلی در گراف است.

۳) وجود یک یال cross در پیمایش عمق اول را نمی‌توان دلالت بر عدم وجود مسیر هامیلتونی دانست.

۴) وجود یک یال cross در پیمایش عمق اول معرف عدم وجود مسیر اولری در گراف است.

۴ـ فرض کنیم که T یک درخت پوشای مینیمم برای گراف G باشد. اگر مقادیر برخی از یال‌های G کاهش پیدا کنند و گراف حاصل را 'G بنامیم در این صورت :

۱) T در هر حال یک درخت پوشای مینیمم از 'G خواهد بود.

۲) اگر یال‌هایی که مقدارشان کاهش پیدا کرده همگی متعلق به T باشند، آن‌گاه T یک درخت پوشای مینیمم برای 'G خواهد بود.

۳) حتی اگر فقط بخشی از یال‌هایی که مقدارشان کاهش یافته متعلق به T باشند، T یک درخت پوشای مینیمم از 'G خواهد بود.

۴) T یک درخت پوشای مینیمم از 'G خواهد بود، اگر یال‌هایی که مقدارشان کاهش پیدا کرده همگی متعلق به 'T باشند و نیز کلیهٔ یال‌های متعلق به T کاهش یافته باشد.

۵ـ توابع $f(n)=4^{\log n}$, $g(n)=(\log n)^{\log n}$ و $h(n)=\log^2 n$ را در نظر می‌گیریم. کدام یک از گزاره‌های زیر صحیح است؟

۱) $f(n)\in O(g(n)), f(n)\in \Omega(h(n))$

۲) $g(n)\in \Omega(h(n)), h(n)\in \Omega(f(n))$

۳) $f(n)\in \theta(h(n)), g(n)\in \Omega(f(n))$

۴) $h(n)\in O(g(n)), f(n)\in \theta(g(n))$

۶ـ آرایهٔ A تقریباً مرتب شده است، یعنی برای $i=1,2,....n-k$ داریم $A[i]\le A[i+k]$. برای مرتب نمودن تمام n عضو، حداقل چه مقدار زمان لازم است؟

۱) $\theta(n)$ ۲) $\theta(nk)$ ۳) $O(n\log k)$ ۴) $O(n\log n)$

۷ـ آرایهٔ نامرتب A شامل n-1 عدد صحیح و مجزا بین 1 تا n است به جز یک عدد x. می‌خواهیم x را با یک الگوریتم کارآ پیدا کنیم. کدام یک از الگوریتم‌های زیر درست و نسبت به بقیه سریع‌تر است؟

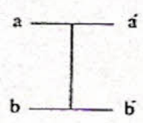
۱) A را مرتب می‌کنیم و با یک پیمایش A, x را به‌دست می‌آوریم.

۲) یک min –heap بر روی A می‌سازیم. از ریشه شروع می‌کنیم و برای پیدا کردن x به یکی از زیردرخت‌های آن می‌رویم و کار را به صورت بازگشتی دنبال می‌کنیم.

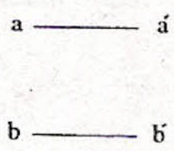
۳) یک درخت دودویی جستجوی متوازن بر روی عناصر A می‌سازیم و کار جستجو را برای پیدا کردن x از ریشه و سپس از یکی زیردرخت‌های آن دنبال می‌کنیم.

۴) میانهٔ A را پیدا می‌کنیم و آن را برای عمل افراز (Partition)، محور قرار می‌دهیم. بر این اساس، کار جستجو را دنبال می‌کنیم تا x به دست آید.

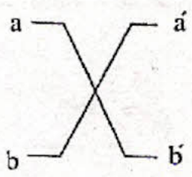
۸ـ مدار زیر را در نظر بگیرید. هر المان



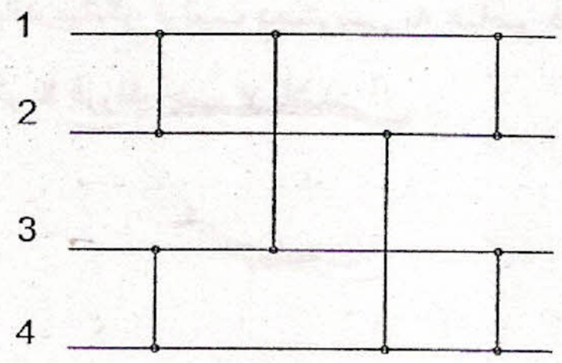
این مدار می‌تواند در دو حالت مستقیم



یا زیگزاک



قرار گیرد اگر ورودی 1 تا 4 باشد، با تعیین حالت‌های همهٔ المان‌ها کدام یک از گزینه‌های زیر را می‌توان تولید کرد؟



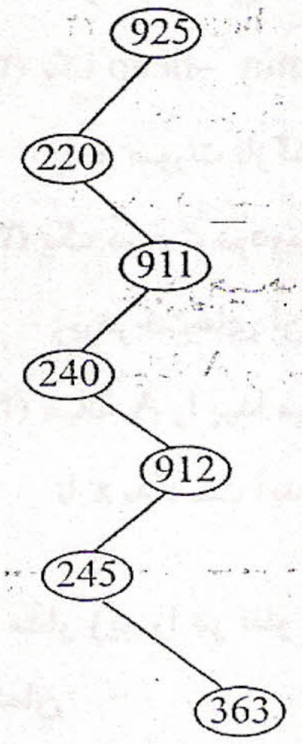
۱) 2 – 4 – 1 – 3 ۲) فقط 3 – 1 – 2 – 4

۳) 4 – 1 – 3 – 2 ۴) همه موارد فوق

جواب سوال‌های کارشناسی ارشد کنکور سراسری

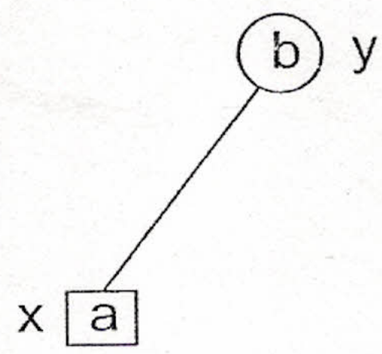
۱ـ الگوریتم ارایه شده ،سه پرانتز اول را در گزینه ۱ وارد پشته می‌کند. سپس برای دو پرانتز بستهٔ آخر، ۲ تا از پرانتزهای باز از پشته برداشته می‌شود. زمانی که دومین پرانتز باز از پشته برداشته می‌شود پشته خالی نیست، لذا رشته پرانتز مزبور توسط الگوریتم فوق متوازن در نظر گرفته می‌شود.

۲- هر کدام از ترتیب‌های ارایه شده در این چهار گزینه در صورت درست بودن باید قسمتی از ساختار درخت دودویی جستجو باشد، اما برای گزینهٔ ۱ قسمتی از ساختار درخت دودویی جستجو به صورت زیر خواهد بود.

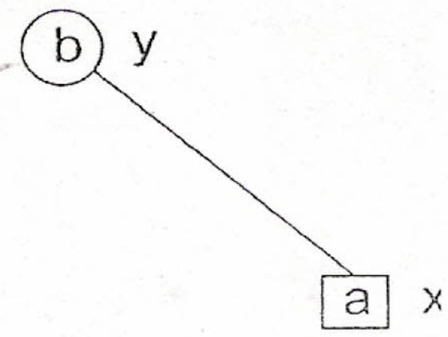


چون گره حاوی ۹۱۲ در زیر درخت سمت چپ گره ۹۱۱ واقع است مقدار آن باید کوچک‌تر از ۹۱۱ باشد. بنابراین ترتیب داده شده نمی‌تواند بیانگر ترتیب دسترسی به عناصر درخت در این جستجو باشد.

۳- اگر x فرزند چپ y باشد:



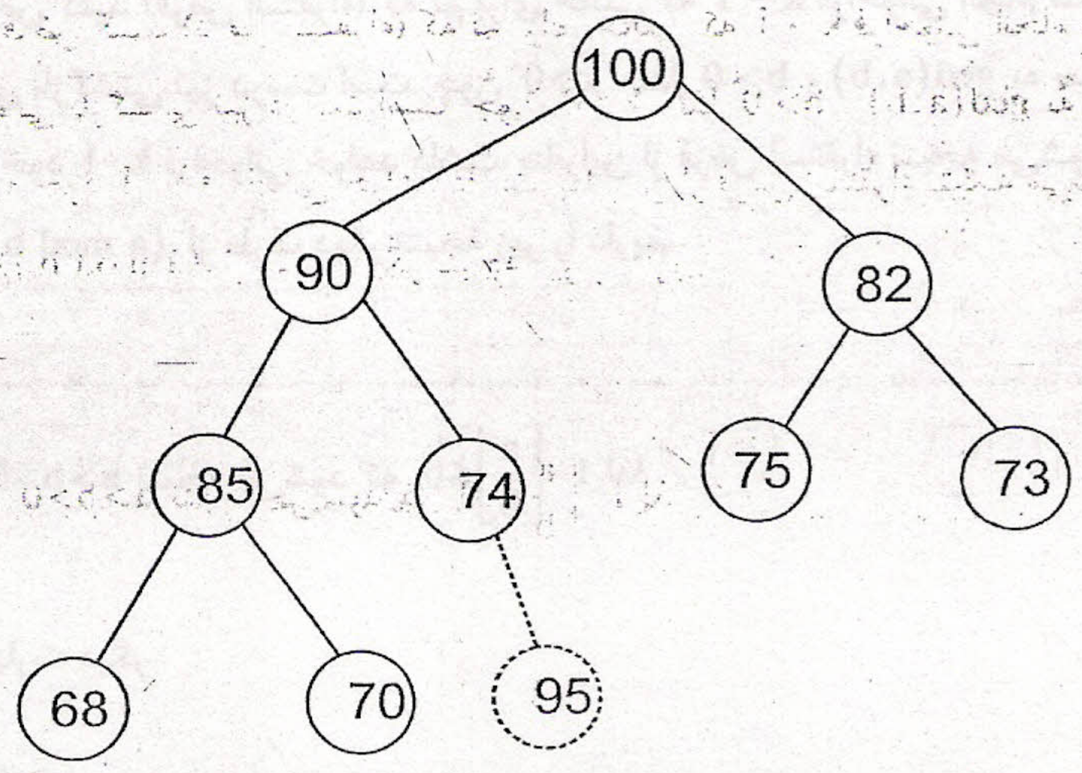
در این صورت گزینه‌های ۱و۲ درست نیست، زیرا b کوچک‌تر a و a بزرگ‌تر از b نیست. اگر x فرزند راست y باشد در این صورت گزینهٔ ۳ هم درست نخواهد بود.



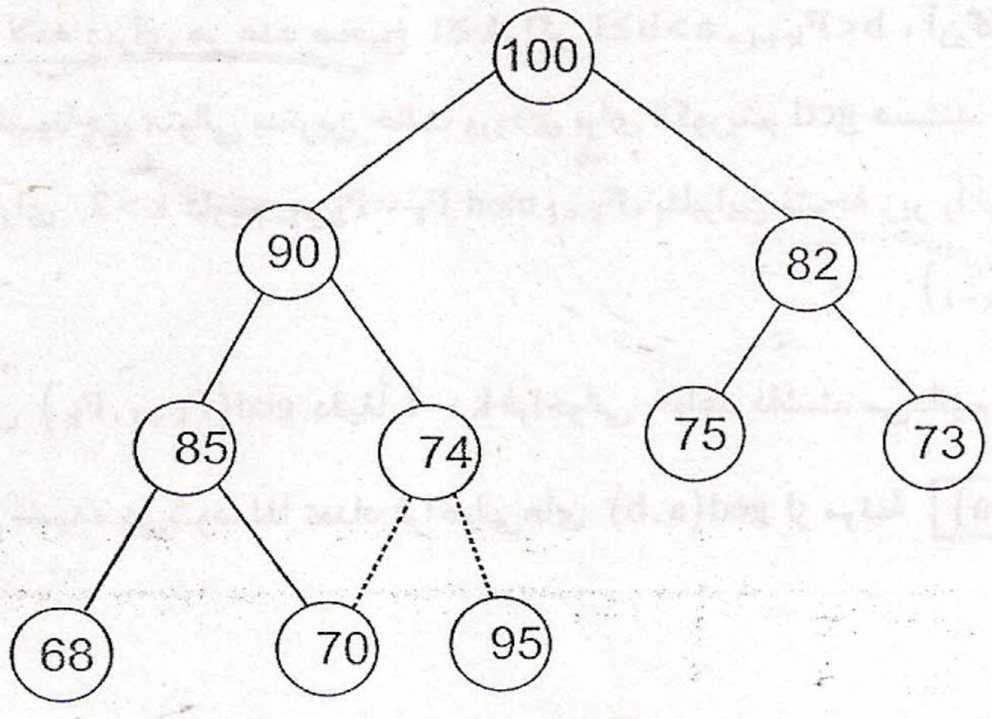
زیرا b کوچک‌تر از a نیست.

۴- جواب این سوال قبلاً در سوال ۶ فناوری اطلاعات سال ۱۳۸۳ داده شده است. با وجود این یادآوری می‌شود که بدترین حالت زمان اجرا برای الگوریتم مرتب‌سازی سریع زمانی است که آرایه از قبل مرتب شدهٔ صعودی (نزولی) باشد و بخواهیم آن را به صورت صعودی (نزولی) مرتب کنیم. در چنین حالتی مرتبهٔ زمانی الگوریتم $O(n^2)$ است.

۵- max – heap داده شده را می‌توان به صورت ساختار درختی زیر نیز نمایش داد.



برای درج عدد 95 آن را در گره نقطه چین قرار می‌دهیم. تعداد تعویض‌های لازم برابر تعداد عناصری است که در مسیر از گره 95 به ریشه قرار دارند و کوچک‌تر یا مساوی آن هستند. تعداد این عناصر در درخت $\boxed{2}$ است (اعداد 90,74) .بعد از این دو تعویض، ساختار آرایه‌ای تبدیل به صورت زیر خواهد بود:



۶- مرتبهٔ بزرگی این الگوریتم معادل تعداد، فراخوانی‌های آن است. بدون اینکه عمومیت مساله از بین برود فرض می‌کنیم که $a>b\geq 0$. این فرض را می‌توان با مشاهدهٔ این‌که اگر $b>a\geq 0$، آن‌گاه $gcd(a,b)$ بلافاصله $gcd(b,a)$ را فرا می‌خواند، توجیه کرد. به عبارت دیگر اگر آرگومان اول کوچک‌تر از آرگومان دوم باشد، آن‌گاه الگوریتم gcd یک فراخوانی بیشتر انجام می‌دهد تا آرگومان‌های خود را تعویض کرده سپس به پردازش ادامه می‌دهد. به طریق مشابه اگر $a=b\geq 0$، الگوریتم بعد از یک فراخوانی خاتمه می‌پذیرد، زیرا $a \bmod b=0$ می‌شود.

تحلیلی که در زیر برای الگوریتم gcd می‌آوریم از عدد فیبوناچی استفاده می‌کند که توسط رابطهٔ بازگشتی زیر تعریف می‌شود.

$$\begin{cases} F_k = F_{k-1} + F_{k-2} & k\geq 2 \\ F_0 = 0\ ,\ F_1 = 1 \end{cases}$$

لم: اگر $a>b\geq 1$ و اجرای $\gcd(a,b)$ منجر به $k\geq 1$ فراخوانی شود، در این صورت $a\geq F_{k+2}$ ، $b\geq F_{k+1}$ اثبات را به طریقهٔ استقراء روی k انجام می‌دهیم. برای پایهٔ استقراء فرض کنید که $k=1$ است. در این صورت $b\geq 1=F_2$ و چون $a>b$ لذا باید داشته باشیم $a\geq 2=F_3$. چون $b>(a \bmod b)$ لذا در هر فراخوانی متوالی آرگومان اول بزرگ‌تر از آرگومان دوم است، پس فرض $a>b$ برای هر فراخوانی متوالی ثابت می‌ماند.

حال فرض کنید (فرض استقراء) که لم برای حالتی که $k-1$ فراخوانی انجام شده باشد درست است. می‌خواهیم ثابت کنیم که برای k فراخوانی بازگشتی نیز درست است. چون $k>0$ ، لذا $b>0$ و $\gcd(a,b)$ به صورت بازگشتی $\gcd(b, a \bmod b)$ را فرا می‌خواند که به نوبهٔ خود $k-1$ فراخوانی خواهد داشت. بنابراین از فرض استقراء نتیجه می‌شود که $b\geq F_{k+1}$ (که قسمتی از لم ثابت می‌شود) و $(a \bmod b)\geq F_k$. از طرف دیگر نتیجهٔ زیر را داریم.

$$b+(a \bmod b)=b+\left(a-\left\lfloor\frac{a}{b}\right\rfloor b\right)=a+b\left(1-\left\lfloor\frac{a}{b}\right\rfloor\right)$$

اما از $a>b>0$ نتیجه می‌شود که $1-\left\lfloor\frac{a}{b}\right\rfloor\leq 0$ لذا

$$b+(a \bmod b)\leq a$$

یا به عبارت دیگر

$$\begin{aligned} a &\geq b+(a \bmod b) \\ &\geq F_{k+1}+F_{k+2} \\ &\geq F_{k+2} \end{aligned}$$

از لم مزبور قضیهٔ معروف لامه (Lame') نتیجه می‌شود.

قضیهٔ لامه : برای هر عدد صحیح $k\geq 1$، اگر $a>b\geq 1$ ، $b<F_{k+1}$ ، آن‌گاه $\gcd(a,b)$، کمتر از k فراخوانی بازگشتی دارد.

اعداد فیبوناچی متوالی بدترین حالت ورودی برای الگوریتم gcd هستند. چون $\gcd(F_3,F_2)$ دقیقاً یک فراخوانی بازگشتی دارد و چون برای $k>2$ داریم $F_{k+1} \bmod F_k=F_{k-1}$، بنابراین نتیجهٔ زیر را نیز خواهیم داشت.

$$\gcd(F_{k+1},F_k)=\gcd(F_k, F_{k+1} \bmod F_k)=\gcd(F_k,F_{k-1})$$

بنابراین $\gcd(F_{k+1},F_k)$ دقیقاً $k-1$ فراخوانی خواهد داشت. می‌دانیم که F_k تقریباً برابر $\frac{\phi^k}{\sqrt{5}}$ است که در آن $\phi=\frac{1+\sqrt{5}}{2}$ عدد طلایی نامیده می‌شود. لذا تعداد فراخوانی‌های $\gcd(a,b)$ از مرتبهٔ $\boxed{O(\log a)}$ یا $\boxed{O(\log b)}$ است.

۱- بهترین الگوریتمی که برای پیدا کردن بزرگ‌ترین و کوچک‌ترین عنصر آرایه‌ای نامرتب می‌توان ارایه داد الگوریتمی است تقسیم و غلبه که خلاصهٔ آن به‌صورت زیر است (برای جزییات بیشتر به کتاب طراحی و تحلیل الگوریتم‌ها قلی‌زاده صفحات ۸۷- ۸۲ مراجعه کنید). با دید استقرایی به مساله می‌توان آن را به‌صورت زیر حل کرد.

الف) اگر آرایه تنها یک عضو داشته باشد نیازی به مقایسه نیست کوچک‌ترین و بزرگ‌ترین عنصر آرایه همان تک عنصر است (پایهٔ استقراء)

ب) اگر آرایه دو عضو داشته باشد می‌توان با یک مقایسه، هم بزرگ‌ترین عضو و هم کوچک‌ترین عضو را به دست آورد. (پایهٔ استقراء البته پایه استقرای الف کافی بود، ولی کار از محکم‌کاری عیب نمی‌کند!)

ج) حل فرض کنید آرایه، $n>2$ (برای سهولت فرض می‌کنیم $n=2^m$) عضو دارد. حال فرض کنید (فرض استقراء) که آرایه را به دو زیر آرایه با تعداد اعضای $\frac{n}{2}$ تقسیم کرده برای این دو زیر آرایه بزرگ‌ترین و کوچک‌ترین اعضا را پیدا کرده‌ایم. بدیهی است که بزرگ‌ترین عضو آرایه با یک مقایسه بین بزرگ‌ترین اعضای زیر آرایه‌های چپ و راست حاصل می‌شود. کوچک‌ترین عضو آرایه نیز با مقایسه‌ای دیگر بین کوچک‌ترین اعضای زیر آرایه‌های چپ و راست به دست می‌آید. خلاصه این‌که رابطهٔ تقسیم و غلبهٔ زیر را برای به دست آوردن تعداد مقایسه‌ها داریم.

$$\begin{cases} t(n)=2t\left(\frac{n}{2}\right)+2 & n>2 \\ t(1)=0, t(2)=1 \end{cases}$$

این رابطه را با استفاده از روش جایگذاری حل می‌کنیم.

$$t(n)=2+2t\left(\frac{n}{2}\right)=2+2\left[2+2t\left(\frac{n}{2^2}\right)\right]=2+2^2+2^2t\left(\frac{n}{2^2}\right)$$
$$=2+2^2+2^2\left[2+2t\left(\frac{n}{2^3}\right)\right]=2+2^2+2^3+2^3t\left(\frac{n}{2^3}\right)$$
$$=....=2+2^2+2^3+...+2^{m-1}+2^{m-1}t(2)$$
$$=2+2^2+2^3+...+2^{m-1}+2^{m-1}$$

اما چون فرض کرده بودیم $n=2^m$ لذا $2^{m-1}=\frac{n}{2}$، بنابراین

$$t(n)=2\left(\frac{n}{2}-1\right)+\frac{n}{2}=\boxed{\frac{3n}{2}-2}$$

۲- می‌دانیم که ضرب اعداد صحیح خیلی بزرگ کاربردی از ضرب دو چند جمله‌ای است و برای ضرب دو چند جمله‌ای هم روش های مختلفی وجود دارد (مراجعه کنید به کتاب طراحی الگوریتم‌ها یا جزوه) که سریع‌ترین آن‌ها روش تقسیم و غلبهٔ سریع است. برای ضرب دو عدد داده شده، روش مزبور را به صورت زیر اِعمال می‌کنیم:

$$1234\times5618=\left(12\times10^2+34\right)\left(56\times10^2+18\right)=12\times56\times10^4+\left(46\times74-12\times56-34\times18\right)\times10^2+34\times18$$

بنابراین ضرب دو عدد چهار رقمی فعلاً به ضرب اعداد دو رقمی منجر شده است که باید آن‌ها را به صورت بازگشتی ضرب کرد تا به اعداد یک رقمی برسیم.

$$46\times74=(4\times10+6)(7\times10+4)=4\times7\times10^2+\left(10\times11-\boxed{4\times7}-6\times4\right)\times10^1+6\times4$$
$$12\times56=\left(1\times10^1+2\right)\left(5\times10^1+6\right)=1\times5\times10^2+\left(3\times11-\boxed{1\times5}-\boxed{2\times6}\right)\times10^1+2\times6$$

$$34\times18=(3\times10^1+4)(1\times10^1+8)=1\times3\times10^2+(\boxed{7\times9}-\boxed{3\times1}-\boxed{4\times8})\times10^1+4\times8$$

مشاهده می‌شود که از سه ضرب مربوط به اعداد دو رقمی هنوز هم دو ضرب دو رقمی یعنی 3×11 , 10×11 حاصل شده است که باید آن‌ها را نیز به طریق مشابه ضرب کرد. فراموش نکنید که عدد 10 در ضرب 10×11 را دیگر نمی‌توان به صورت شیفت انجام داد، زیرا نتیجه‌ای است که در حین محاسبات حاصل شده است و به صورت 10^x توضیح داده شده در صورت مساله نیست.

$$10\times11=(1\times10^1+0)(1\times10^1+1)=1\times1\times10^2+(\boxed{1\times2}-\boxed{1\times1}-\boxed{0\times1})\times10^1+0\times1$$

$$3\times11=(0\times10^1+3)(1\times10^1+1)=0\times1\times10^2+(\boxed{3\times1}-\boxed{0\times1}-\boxed{3\times1})\times10^1+1\times3$$

با شمارش نتایجی که داخل مستطیل‌ها آمده‌اند به این نتیجه می‌رسیم که برای ضرب دو عدد 1234 و 5618 به $\boxed{13}$ ضرب اعداد یک رقمی نیاز داریم.

۳- سریع‌ترین روش برای ارزیابی چند جمله‌ای در یک نقطهٔ عددی روش هرنر (Horner) است که می‌توان آن را به‌صورت فاکتورگیری خلاصه کرد (برای جزییات بیشتر به کتاب روش‌های محاسبات عددی قلی‌زاده فصل اول مراجعه کنید):

$$p_n(x)=a_0+a_1x+a_2x^2+...+a_nx^n$$
$$=a_0+x(a_1+x(a_2+x(a_3+....+xa_n))...)$$

برای مثال اگر

$$p_3(x)=3+5x+7x^2+8x^3$$

آنگاه $p(x)$ را می‌توان به‌صورت زیر نوشت:

$$p_3(x)=3+x(5+x(7+8x))$$

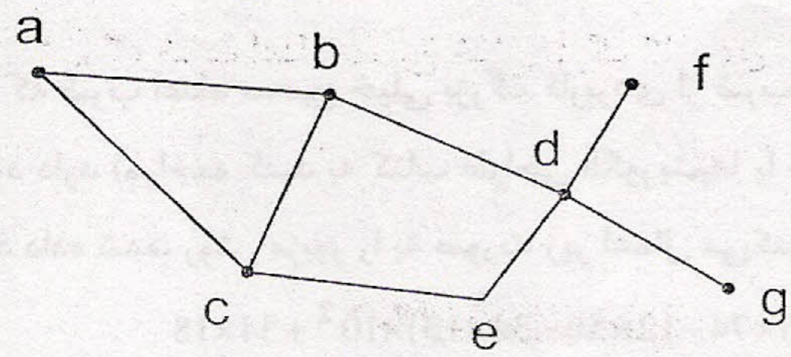
مشاهده می‌شود که با این عمل فاکتورگیری برای محاسبهٔ $p_n(x)$ به $\boxed{n}$ عمل ضرب عددی نیاز است.

۴- طبق قضیهٔ اولر رابطهٔ زیر بین تعداد یالها (e)، تعداد رئوس (v) و تعداد نواحی (r) گرافی مسطح وجود دارد:

$r=e-v+2$ یا $e+2=v+r$

یکی از این نواحی نامتناهی و بقیه، دورهای ساده هستند بنابراین تعداد دوره‌های ساده (r') گرافی مسطح را می‌توان به‌صورت $\boxed{r'=r-1}$ به‌دست آورد.

برای مثال در یک درخت که حالت خاصی از یک گراف مسطح است یک ناحیهٔ نامتناهی وجود دارد $(r=1)$ ولی دوری وجود ندارد $(r'=0)$. از طرف دیگر تعداد رئوس درخت یک واحد بیشتر از تعداد یال‌های آن است: $e+1=v \Leftrightarrow e+2=v+1$



$v=7$
$e=8$
$r=3$
$8+2=7+3$

مشاهده می‌شود که تعداد نواحی آن سه تاست، ولی تعداد دورهای آن 2=1-3 است. خلاصه این‌که از روی فرمول مزبور با مرتبهٔ زمانی $\boxed{O(1)}$ می‌توان تعداد دوره‌های گرافی مسطح را به دست آورد. (برای جزییات بیشتر در مورد گراف‌های مسطح به کتاب ساختمان‌های گسسته قلی‌زاده فصل گراف‌ها مراجعه کنید.)

۵- چون $10^n+n^{20}=O(n^n)$ ولی $10^n+n^{20}\neq\Omega(n^n)$ لذا $\boxed{10^n+n^{20}\neq\theta(n^n)}$، در نمادهای مجانبی O، $\Omega,\theta,0,\omega$، مبنای الگوریتم‌ها تأثیری در مرتبهٔ بزرگی ندارد. برای مثال :

$$\log_2^n=\theta(\log_{10} n)$$

قضیه : اگر

$$A(n)=a_m n^m+a_{m-1}n^{m-1}+..+a_1 n+a_0,\quad a_m>0$$

آنگاه $\boxed{A(n)=\theta(n^m)}$

مرتبه‌های بزرگی متداول عبارت‌اند از:

$$\theta(1) < \theta(\log n) < \theta(n) < \theta(n\log n) < \theta(n^2) < ... < \theta(2^n)$$

لذا

$$4n^3+5n^2+7n=\Omega(n^2\log n)$$

$$4n^3+5n^2+7n=\Omega(n^2)$$

$$4n^3+5n^2+7n=\Omega(n\log n)$$

از جمله

$$\boxed{4n^3+5n^2+7n=\Omega(\log_2^n)}$$

$(\log_2^n)!=O(n!)$ ولی $(\log_2^n)!\neq\Omega(n!)$

مشاهده می‌شود که حتی برای n های کوچک نیز رابطهٔ

$$(\log_2^n)!>n!$$

برقرار نیست. برای مثال برای $n=8$ داریم

$$(\log_2^8)!=6$$

در صورتی که $8!=5780$

۶-

```
f(n)
   if n<1 return 1
   else
        return f(n-1)+g(n)
g(n)
        if n<2 return 1
        else return f(n-1)+g(n/2)
```

با یک نگاه ساده به تعاریف $g(n),f(n)$ مشاهده می‌شود که هر دوی آن‌ها توابعی مثبت و صعودی هستند: برای $n_1\leq n_2$ داریم

$$0\leq f(n_1)\leq f(n_2),g(n_1)\leq g(n_2)$$

توابع بالا را به صورت روابط بازگشتی زیر نیز می‌توان بیان کرد:

$$\begin{cases} f(n)=f(n-1)+g(n) & n\geq 1 \\ f(0)=1 \end{cases}$$

$$\begin{cases} g(n)=f(n-1)+g\left(\frac{n}{2}\right) & n\geq 2 \\ g(0)=g(1)=1 \end{cases}$$

لذا $g(n)=f(n-1)+g\left(\frac{n}{2}\right)\geq f(n-1)$ یا $\boxed{g(n)\geq f(n-1)}$

نامساوی بالا را در رابطهٔ $f(n)=f(n-1)+g(n)$ اعمال می‌کنیم. نتیجه به صورت زیر خواهد بود.

$$f(n)=f(n-1)+g(n) \geq 2f(n-1)$$

پس $f(n)\geq 2f(n-1)$، با استفاده از روش جایگذاری با تکرار داریم:

$$f(n)\geq 2f(n-1)\geq 2^2 f(n-2)\geq 2^n f(0)=2^n$$

$$f(n)\geq 2^n \Rightarrow \boxed{f(n)=\Omega\left(2^n\right)}$$

لذا تابع رشد $f(n)$ نمایی است.

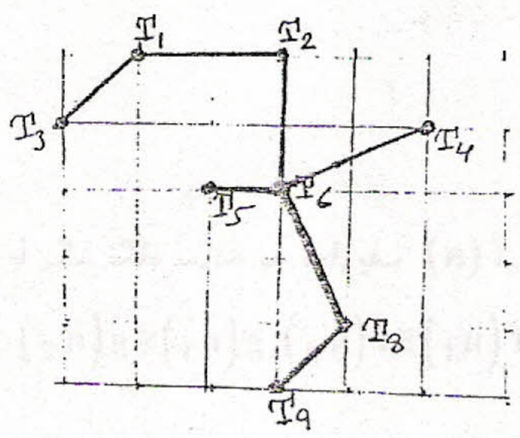
۷- برای تعیین این‌که عددی صحیح و مثبت مثل n اول است یا نه، می‌توان از الگوریتم سادهٔ زیر استفاده کرد:

```
IsPrime (n)
  prime=True
  i=2
   while (prime) and (i<n) do
        if (n mod i=0) then prime=False
          else
               i=i+1
```

بدیهی است که مرتبهٔ زمانی الگوریتم مزبور برحسب بزرگی n، خطی است $\theta(n)$ ،اما نکته ای که در این مساله وجود دارد این است که مرتبهٔ بزرگی برحسب تعداد بیت‌های n خواسته شده است. می‌دانیم که اگر n یک عدد N بیتی است داریم $n=2^N$. لذا مرتبهٔ بزرگی الگوریتم مزبور برحسب تعداد ورودی‌ها یعنی تعداد بیت‌ها نمایی خواهد بود.

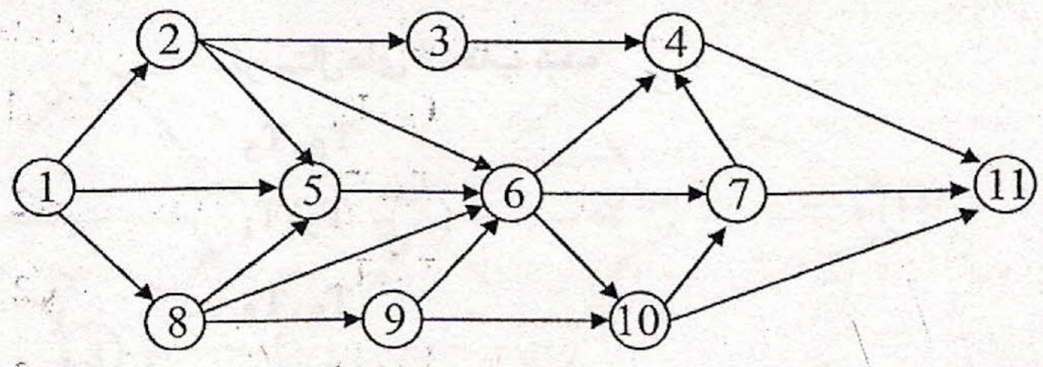
$$\theta(n)=\theta\left(2^N\right)$$

۸- برای حل این مساله می‌توان از الگوریتم کراسکال استفاده کرد. گرافی را در نظر بگیرید که رئوس آن 9 ترمینال داده شده است و هزینهٔ هر یال هم برابر با فاصلهٔ اقلیدسی بین دو ترمینال در صفحه است. در این صورت کوچک‌ترین فاصله بین ترمینال‌ها ۱ و مربوط به فاصلهٔ بین T_5 و T_6 است. مراحل مختلف انتخاب یال‌ها (فاصلهٔ بین ترمینال‌ها در زیر خلاصه شده است. در کل باید 8 یال به شرط تشکیل ندادن دور انتخاب شوند.



ترمینال‌های انتخاب شده	فاصله
T_6, T_5	1
T_3, T_1	$\sqrt{2}$
T_9, T_8	$\sqrt{2}$
T_2, T_1	2
T_6, T_2	2
T_7, T_3	2
T_6, T_4	$\sqrt{5}$
T_8, T_6	$\sqrt{5}$
جمع	$7+2\sqrt{2}+2\sqrt{5}$

۱-



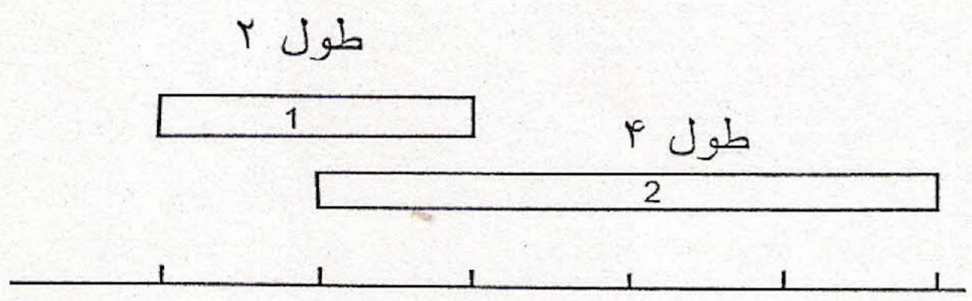
اگر گراف بالا را به روش DFS و با شروع از راس 1 پیمایش کنیم ترتیب ملاقات شدن رئوس به‌صورت زیر خواهد بود.

1,2,3,4,11,5,6,7,10,8,9

درخت پوشای DFS به مبدا راس 1 نیز به صورت پررنگ در خود گراف نشان داده شده است.

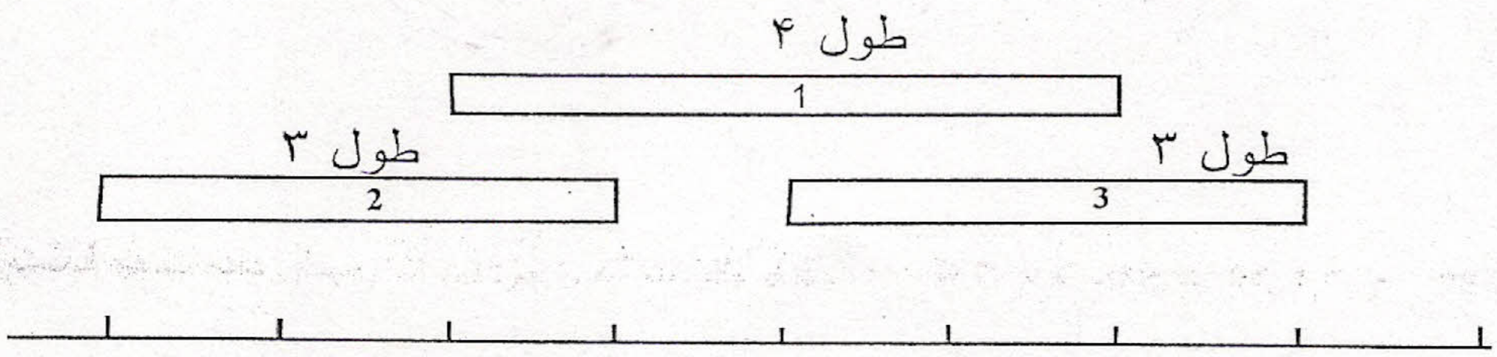
۲- این مساله مشابه مسالهٔ انتخاب فعالیت‌هاست با این نکته انحرافی که در این‌جا مجموع زمان فعالیت‌های انتخاب شده (مجموع طول بازه‌های انتخاب شده) خواسته شده است و نه تعداد آنها هیچ کدام از روش‌های حریصانه داده شده در گزینه‌ها ترتیبی درست برای این الگوریتم حریصانه نیست. برای هر کدام مثالی می‌آوریم. فرض شده است که ترتیب مورد نظر صعودی است. برای حالت نزولی نیز می‌توان مثال‌های مشابهی آورد.

ترتیب (صعودی) برحسب طول بازه‌ها جواب درست نمی‌دهد:



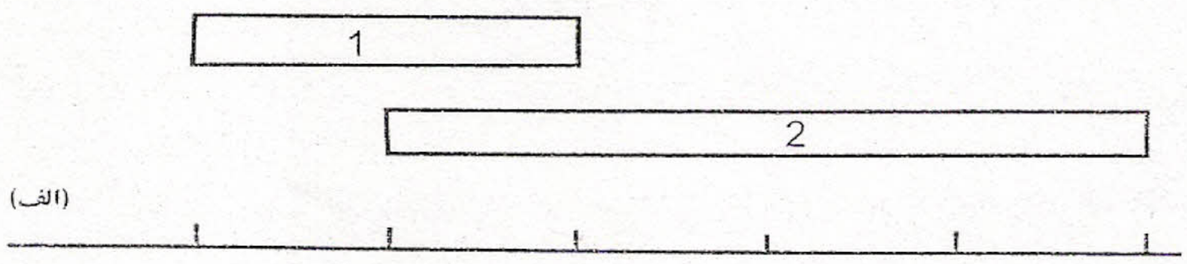
چون اولویت با فعالیت 1 به طول 2 است انتخاب می‌شود، که هم‌پوشان با فعالیت 2 است. در صورتی که با انتخاب فعالیت شماره 2، طول بازهٔ بیشتری حاصل می‌شود.

ترتیب (نزولی) برحسب طول بازه‌ها



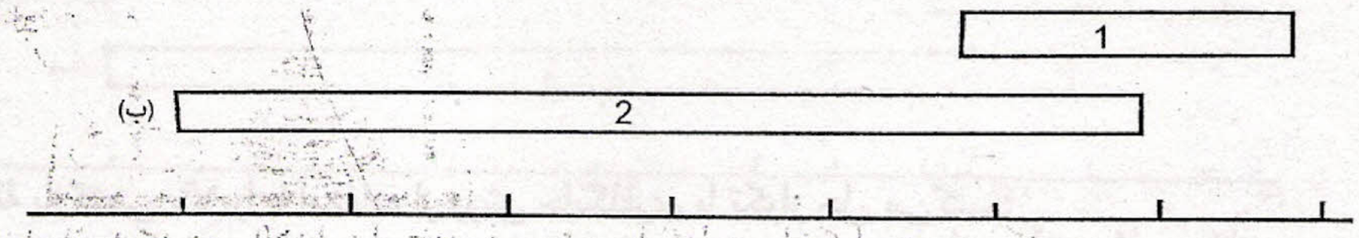
با این اولویت فعالیت 1 به طول 4 انتخاب می‌شود در صورتی که می‌توانیم فعالیت‌ها ۲و۳ را با مجموع طولهای ۶ انتخاب کنیم.

ترتیب (صعودی) U_i بازه‌ها



با این اولویت تنها فعالیت 1 به طول 2 انتخاب می‌شود، در صورتی که می‌توانستیم فعالیت شماره ۲ را به طول 5 انتخاب کنیم.

ترتیب (نزولی) U_i بازه ها



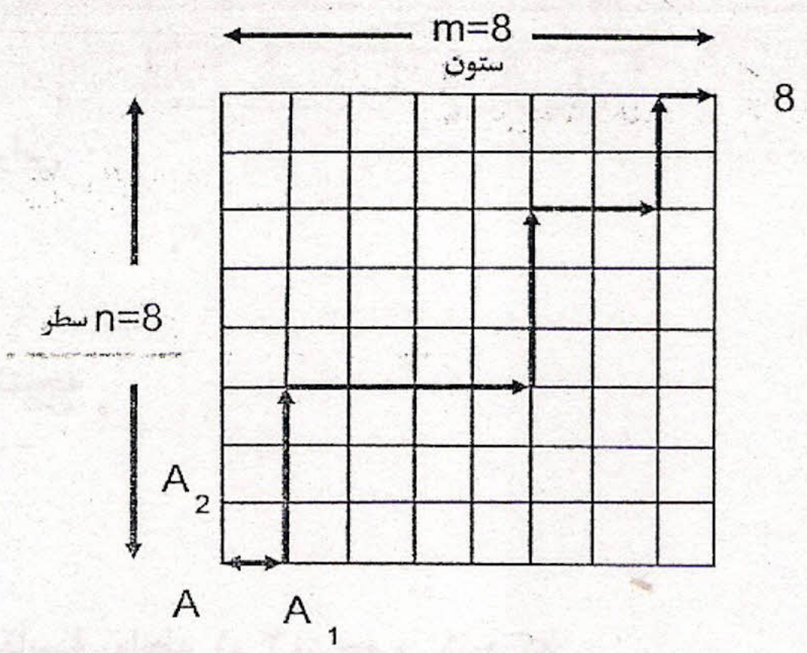
فعالیت ۱ به طول ۲ انتخاب می‌شود که می‌شد فعالیت ۲ به طول 6 را انتخاب کرد.

ترتیب (صعودی) L_i بازه‌ها

با این اولویت نیز برای شکل الف، فعالیت ۱ به طول 2 انتخاب می‌شود، در صورتی که طول فعالیت ۲ برابر 5 است.

ترتیب (نزولی) l_i بازه‌ها

برای شکل ب، فعالیت ۱ به طول 2 انتخاب می‌شود، در صورتی که می‌توانستیم فعالیت 2 را به طول 6 انتخاب کنیم.



۳- برای شروع از نقطهٔ A دو حالت امکان دارد یا می‌توان به نقطه‌ی A_1 رفت که در این صورت تعداد مسیرهای ممکن از A_1 به B با $P(n,m-1)$ تعریف می‌شود یا می‌توان به A_2 رفت که در این صورت تعداد مسیرهای ممکن از A_2 به B از رابطه‌ی $P(n-1,m)$ حاصل خواهد شد. از طرف دیگر اگر $n=0$ باشد، وضعیتی مشابه A—•—•—•...—• را خواهیم داشت که تنها یک مسیر از A به B موجود است، به عبارت دیگر $P(0,m)=1$ به طریق مشابه اگر $m=0$ باشد، خواهیم داشت $P(n,0)=1$. بنابراین تعداد مسیرهای ممکن از A به B با شرایط داده شده به وسیلهٔ رابطهٔ بازگشتی زیر تعریف می‌شود:

$$\begin{cases} P(n,m)=p(n-1,m)+p(n,m-1), & n\geq 1,\ m\geq 1 \\ P(0,m)=P(n,0)=1 \end{cases}$$

ب: این رابطه‌ی بازگشتی را می‌توان با استفاده از سری مولد حل کرد داریم:

$$G_n(x)=\sum_{m=0}^{\infty} P(n,m)x^m \qquad (1)$$

یا

$$G_n(x)=\sum_{m=0}^{\infty} P(n,m)x^m=P(n,0)+\sum_{m=1}^{\infty} P(n,m)x^m=1+\sum_{m=1}^{\infty}\left[P(n-1,m)+P(n,m-1)\right]x^m$$

$$=1+\sum_{m=1}^{\infty} P(n-1,m)x^m+x\sum_{m=1}^{\infty} P(n,m-1)x^{m-1}=G_{n-1}(x)+xG_n(x)$$

لذا

$$G_n(x)=\frac{G_{n-1}(x)}{1-x}$$

رابطهٔ بازگشتی بالا را با استفاده از روش جایگذاری با تکرار حل می‌کنیم:

$$G_n(x)=\frac{G_{n-2}(x)}{(1-x)^2}=\frac{G_{n-3}(x)}{(1-x)^3}=\ldots$$

و با ادامهٔ این روند:

$$G_n(x)=\frac{G_0(x)}{(1-x)^n}$$

$$G_0(x)=\sum_{m=0}^{\infty} t(0,m)x^m=\sum_{m=0}^{\infty} x^m=\frac{1}{1-x}$$

بنابراین

$$G_n(x)=\frac{1}{(1-x)^{n+1}}$$

در نتیجه

$$G_n(x)=\sum_{m=0}^{\infty} C_{n+m}^{m} x^m \qquad (2)$$

از مقایسهٔ روابطه ۱و ۲ نتیجه می‌شود که:

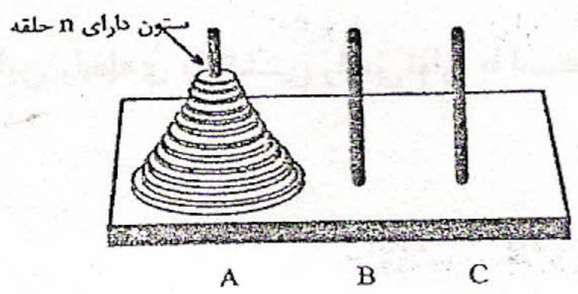
$$P(n,m)=C_{n+m}^{m}$$

اگر $n=m=8$ باشد، خواهیم داشت

$$P(8,8)=C_{16}^{8}$$

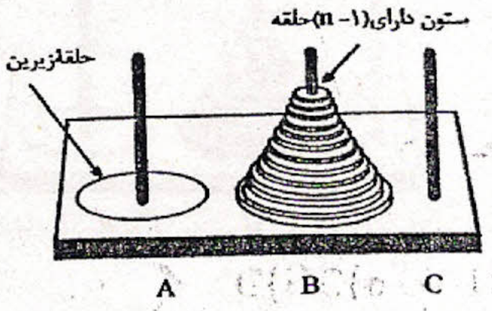
۴- بدیهی است که برای $n=1$ داریم : $t(1)=2$. برای $n\geq 2$ مراحل مختلف انجام کار (با استفاده از استقرای ریاضی) در زیر به تصویر کشیده شده‌اند.

الف) شروع



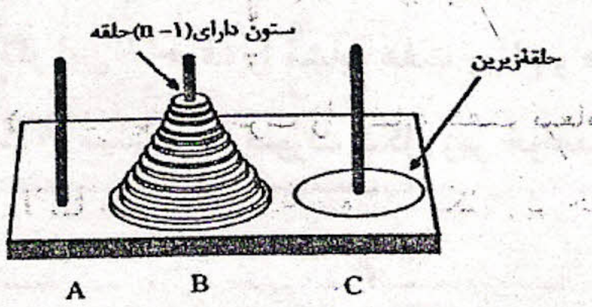
(الف)

ب) فرض کنید (فرض استقراء)که n-1 حلقه بالایی را با $t(n-1)$ جابه‌جایی توسط میلهٔ C به میلهٔ B برده باشیم.



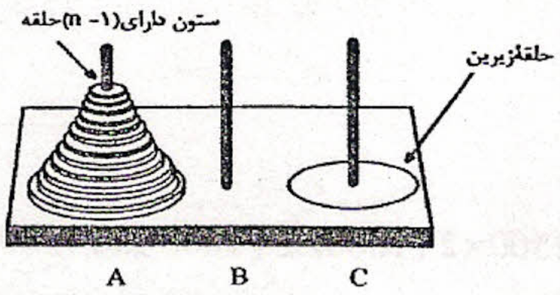
(ب)

ج) با یک انتقال حلقهٔ زیرین را از میلهٔ A به میلهٔ C می‌بریم.



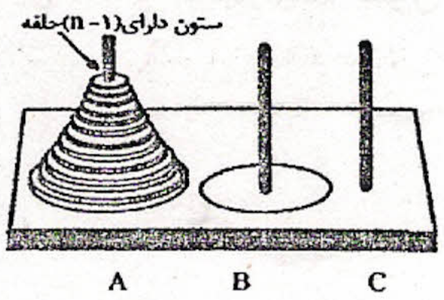
(ج)

د) باز بنا به فرض استقراء $(n-1)$ حلقهٔ واقع در میلهٔ B را توسط C با $t(n-1)$ انتقال به میلهٔ A می‌بریم.



(د)

ر) حلقه‌ی زیرین را با یک انتقال از میلهٔ C به میلهٔ B می‌بریم.



(ر)

ز) مجدداً $n-1$ حلقهٔ واقع در ستون A را با $t(n-1)$ جابه‌جایی دیگر توسط میلهٔ کمکی C به میلهٔ B می‌بریم.

بنابراین تعداد کل جابه‌جایی‌ها از رابطهٔ زیر حاصل خواهد شد:

$$\boxed{\begin{cases} t(n)=3t(n-1)+2 \\ t(1)=2 \end{cases}}$$

ب: رابطهٔ ۱ بازگشتی مزبور را می توان با استفاده از روش جایگذاری با تکرار حل کرد.

$$t(n)=2+3t(n-1)=2+3\left[2+3t(n-2)\right]$$

$$=2+2\times3+3^2\,t(n-2)$$

$$=2+2\times3+3^2\left[2+3t(n-3)\right]$$

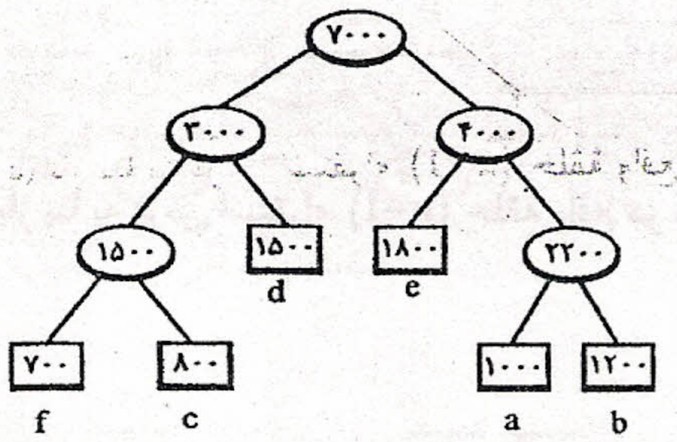
$$=2+2\times3+2\times3^2+3^3t(n-3)$$

$$=2+2\times3+2\times3^2+2\times3^2+..+2\times3^{n-2}+3^{n-1}t(1)$$

$$=2+2\times3+2\times3^2+2\times3^3+...+2\times3^{n-2}+2\times3^{n-1}$$

$$=2\frac{3^n-1}{3-1}=\boxed{3^n-1}$$

۵- اگر این 7 حرف را مشابه هفت پیغام و دفعات تکرار آن‌ها را مشابه فرکانس نسبی مورد اشاره در بالا تعبیر کنیم، درخت دودویی با WEPL مینیمم به صورت شکل زیر خواهد بود.



$$WEPL=700\times3+800\times3+1500\times2+1800\times2+1000\times3+1200\times3$$

$$\Rightarrow \boxed{WEPL=17700}$$

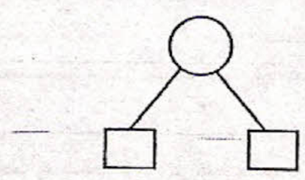
۶- این مساله مشابه مسالهٔ سؤال ۶ هوش مصنوعی سال ۱۳۸۳ حل می‌شود.

۱- چند نکته در مورد عدد کاتالان

الف)عدد کاتالان به صورت $C(n)=\frac{1}{n+1}C_{2n}^{n}$ تعریف می‌شود که از حل رابطهٔ بازگشتی زیر حاصل می‌شود.

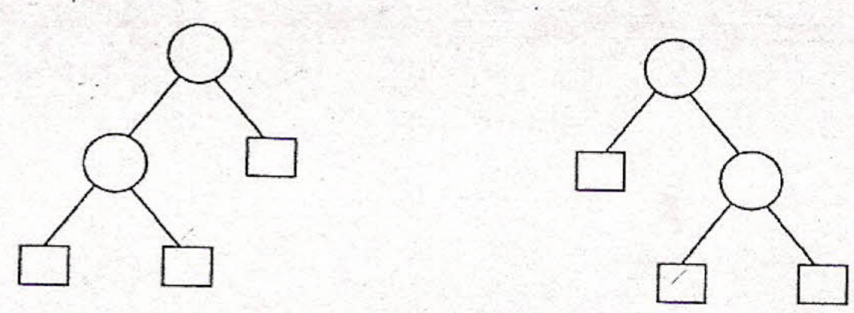
$$C(n)=\sum_{i=0}^{n-1} C(i)C(n-i-1)$$

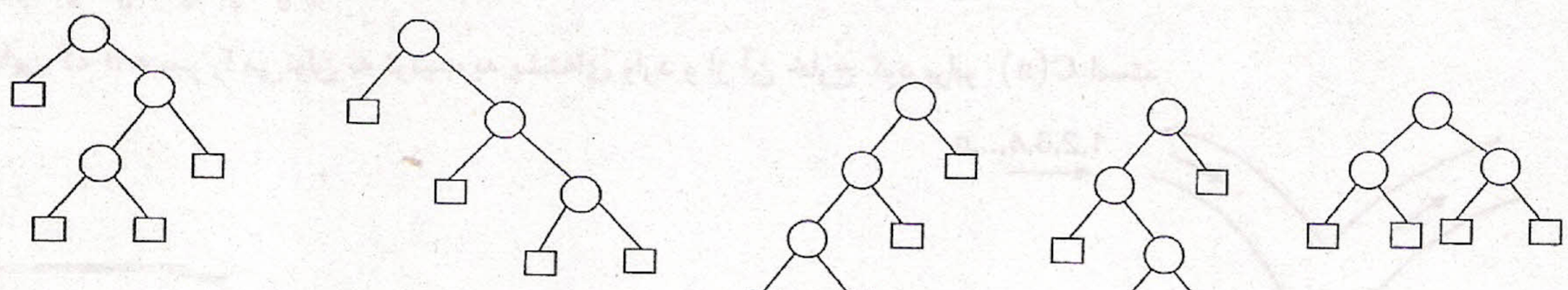
ب) تعداد درخت‌های دودویی (جستجو) که با n گره داخلی می‌توان تشکیل داد برابر $C(n)$ است. اگر فرض کنیم که $T(n)$ تعداد درخت‌های دودویی است که با n گره داخلی می‌توان تشکیل داد داریم:



$T(1)=1$

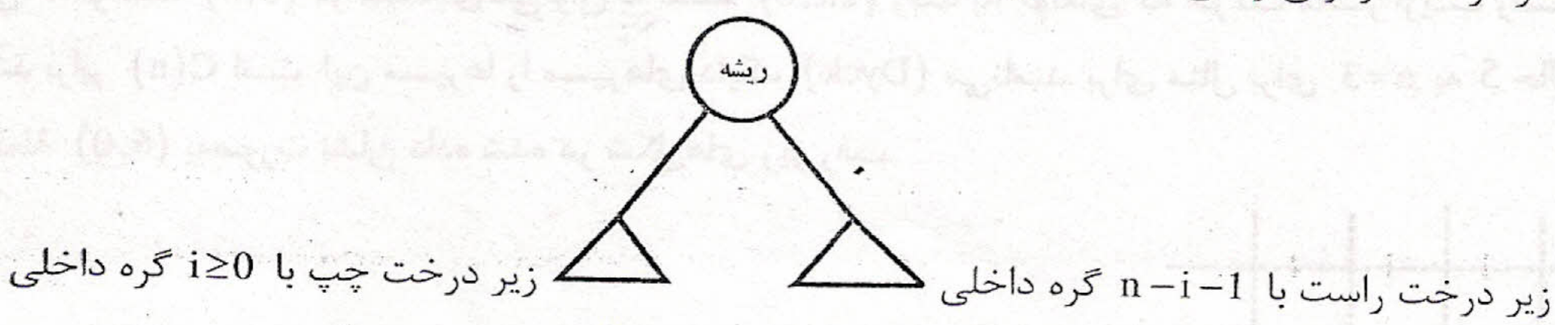
$T(2)=2$





$T(3)=5$

برای $n\geq 1$ ساختار درخت دودویی را می‌توان به صورت زیر در نظر گرفت.



لذا تعداد درخت‌های دودویی را که با n گره داخلی می‌توان تشکیل داد از رابطهٔ زیر به دست می‌آید:

$$t(n)=\sum_{i=0}^{n-1} t(n)t(n-i-1)$$

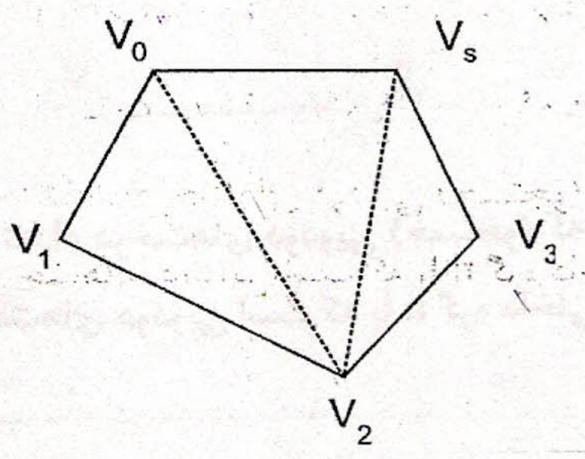
برای مثال برای $n=4$ داریم

$$t(4)=t(0)t(3)+t(1)t(2)+t(2)t(1)+t(3)t(0)=5+2+2+5=14$$

ج) تعداد حالت‌هایی که $n+1$ ماتریس را می‌شود در هم ضرب کرد معادل با تعداد درخت‌های دودویی است که می‌توان با $n+1$ برگ یا با n گره داخلی تشکیل داد و برابر $C(n)$ است.

د) تعداد حالت‌هایی که یک $n+2$ ضلعی محدب را می‌توان مثلث بندی کرد برابر $C(n)$ است. شکل مقابل یکی از 5 حالت مثلث‌بندی را برای یک پنج ـ ضلعی محدب نشان می‌دهد:

۱- برای ۴ حالت دیگر مثلث‌های مثلث‌بندی عبارتند از :



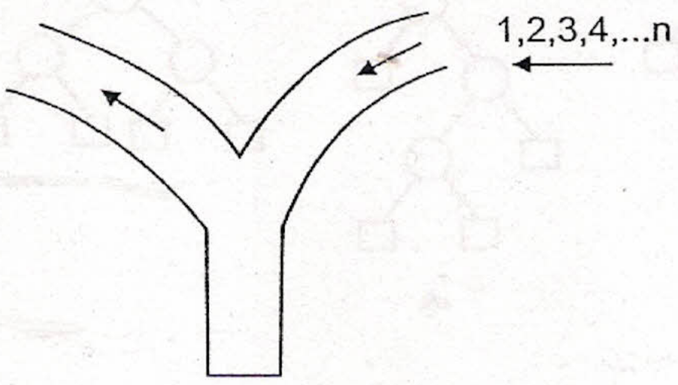
۲- $V_0 \overset{\Delta}{V_1} V_2 , V_0 \overset{\Delta}{V_3} V_5 , V_0 \overset{\Delta}{V_2} V_3$

۳- $V_2 \overset{\Delta}{V_3} V_5 , V_1 \overset{\Delta}{V_5} V_2 , V_0 \overset{\Delta}{V_1} V_5$

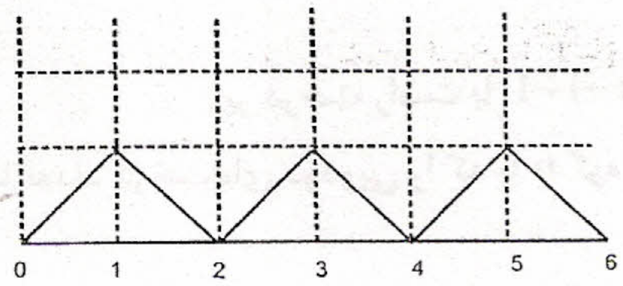
۴- $V_1 \overset{\Delta}{V_5} V_3 , V_0 \overset{\Delta}{V_1} V_5 , V_1 \overset{\Delta}{V_3} V_2$

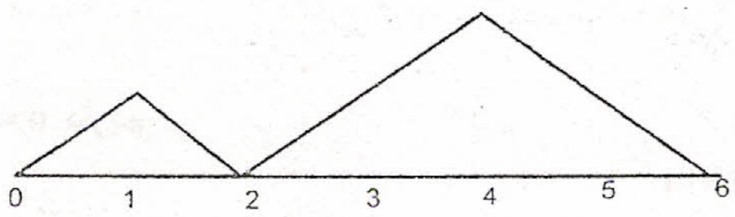
۵- $V_0 \overset{\Delta}{V_1} V_3 , V_0 \overset{\Delta}{V_3} V_5 , V_1 \overset{\Delta}{V_3} V_2$

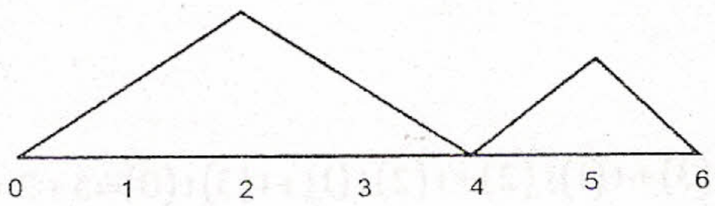
ر) تعداد حالت‌هایی که n عنصر را می‌توان به ترتیب به پشته‌ای وارد و از آن خارج کرد برابر $C(n)$ است.

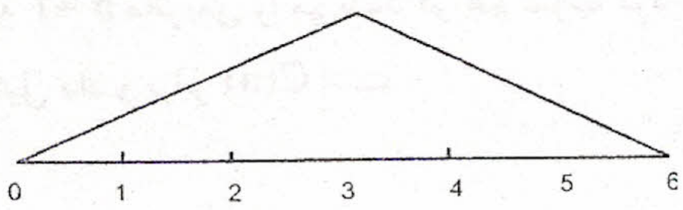


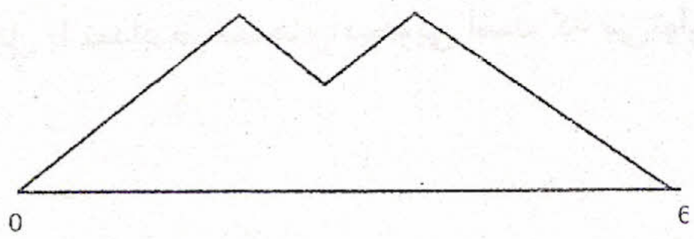
ز) تعداد مسیرهایی که از مبدا $(0,0)$ در شبکه‌ای می‌توان به نقطهٔ $(2n,0)$ رفت به گونه‌ای که حرکت مجاز اریب راست ـ بالا یا راست ـ پایین باشد برابر $C(n)$ است. این مسیرها را مسیرهای دایک (Dyck) می‌نامند. برای مثال برای $n=3$ به 5 حالت می‌توان از نقطهٔ $(0,0)$ به نقطهٔ $(6,0)$ به‌صورت نشان داده شده در شکل‌های زیر رفت.



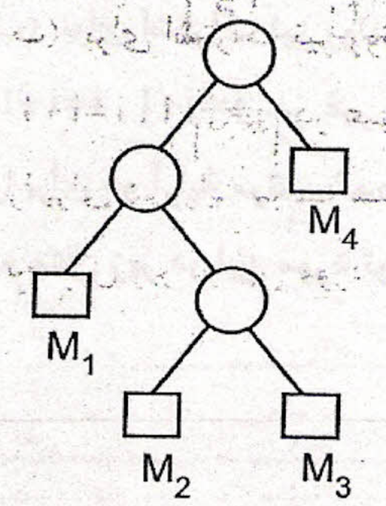




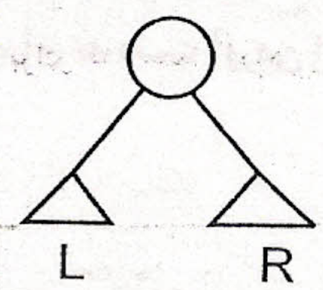




حال به حل مساله می‌پردازیم. تعداد پرانتز بندی‌های مختلف برای ضرب n ماتریس معادل باتعداد درخت‌های دودویی است که با n برگ می‌توان تشکیل داد، زیرا برای هر پرانتزبندی می‌توان درخت دودویی تشکیل داد که برگ‌های آن معرف n ماتریس داده شده هستند. برای مثال برای ضرب زنجیر ماتریس $M_1M_2M_3M_4$ که به صورت $(M_1(M_2M_3)M_4)$ پرانتزبندی شده باشد می‌توان درخت دودویی زیر را تشکیل داد.



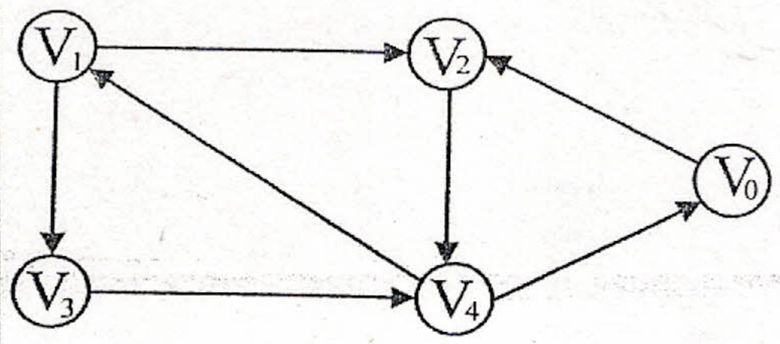
ساختار یک درخت دودویی با $n>1$ برگ به صورت زیر است.



که در آن هر کدام از زیر درخت‌های L و R حداقل یک برگ دارند و مجموع برگ‌های L , R نیز برابر n است. بنابراین تعداد پرانتز بندی‌های مختلف برای ضرب n ماتریس یا تعداد درخت‌های دودویی با n برگ را می‌توان از روی رابطهٔ بازگشتی زیر به دست آورد.

$$\boxed{t(n)=\sum_{i=1}^{n-1} t(i)\,t(n-i)}$$

۲- چون خود گراف قویاً همبند است، لذا تعداد مؤلفه‌های قویاً همبند آن یکی بیش‌تر نیست و آن‌هم خود گراف است: $\{V_0, V_1, V_2, V_3, V_4\}$



یادآوری: گراف جهت دارای را قویاً همبند گوییم هر گاه از هر راس آن به هر أس دیگر مسیری جهت‌دار موجود باشد.

۳- این سوال مشابه سوال ۱ فنآوری اطلاعات سال ۱۳۸۴ است.

۴- اجتماع و اشتراک دو آرایهٔ مرتب شده مشابه ادغام دو آرایه مرتب شده است:

الف) اجتماع دو آرایهٔ مرتب شدهٔ A, B را می‌توان به صورت زیر به‌دست آورد. فرض کنید $C=A\cup B$ در شروع کار داریم.

$=1\ ,\ j=1, k=1 \qquad C[k]=\text{minimum}\ \{A[i], B[j]\}$

$\text{if } A[i] \le A[j] \text{ then } i=i+1$

$\text{else} \qquad j=j+1$

سپس در مراحل بعد اگر $C[k]\neq Min\{A[i],B[j]\}$ بود، آن‌گاه $k=k+1$ و $C[k]=Min\{A[i],B[j]\}$ و روند را از دستور * سر می‌گیریم تا به انتهای یکی از آرایه‌ها برسم. عناصر باقی‌مانده از یکی از آرایه ها را یکی یکی با آخرین عنصر C یعنی $C[k]$ مقایسه می‌کنیم. هر کدام که مخالف $C[k]$ بود آن را در مکان $k+1$ قرار داده و به k یک واحد اضافه می‌کنیم. مشاهده می‌شود که این کار را می‌شود در زمان $\theta(n+m)$ انجام داد که با توجه به شرط $m\leq n$ نتیجه می‌شود که اجتماع در زمان $\theta(n)$ شدنی است.

ب) برای اشتراک نیز روند مشابهی را می‌توان ارایه داد با این تفاوت که اگر $A[i]=B[j]\neq C[k]$، $k=k+1$ و $C[k]=A[i]$ ، $i=i+1$, $j=j+1$. در غیر این‌صورت اگر $A[i]\leq B[j]$ آن‌گاه $i=i+1$ و در غیر این‌صورت $j=j+1$. مشاهده می‌شود که عمل اشتراک را برای دو آرایهٔ مرتب شده نیز می توان در زمان $\theta(n+m)$ انجام داد که اگر $m\leq n$ آن‌گاه $\theta(n+m)=\theta(n)$.

به مثال زیر در این مورد توجه کنید. فرض کنید $m=4, n=5$ و

	1	2	3	4	5
A=	1	2	2	4	4

	1	2	3	4
B=	2	2	2	4

برای به دست آوردن اجتماع A و B به‌صورت زیر عمل می‌کنیم

$i=1$, $j=1$, $k=1$

$A[i]\leq B[j] \Rightarrow C[k]=A[i] \Rightarrow C=$

1	
1	

$i=i+1 \quad (i=2)$

$A[i]\leq B[j]$, $A[i]\neq C[k]$

$k=k+1 \qquad C[k]=A[i]$, $i=i+1 \Rightarrow C$

	1	2	
	1	2	

$A[i=3]=B[j=1]=C[k=2] \Rightarrow (i=3)$

$i=4$, $j=2$

$B[2]\leq A[4]$, $B[3]=C[k=2] \Rightarrow$

$j=j+1 \Rightarrow (j=3)$

$B[3]\leq A[4]$, $B[3]=C[k=2]$, $j=j+1$, $(j=4)$

$A[i=3]\leq B[j=4]$, $A[i=4]\neq C[k=2] \Rightarrow$

$k=k+1 \qquad C[3]=A[4] \qquad C$

	1	2	3	
	1	2	4	

$i=i+1 \quad (i=5)$

$A[i]\leq B[j]$, $C[3]=A[i] \Rightarrow$

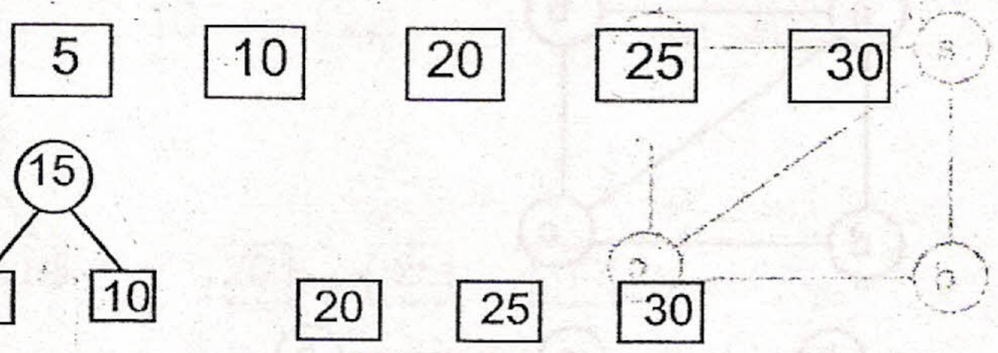
$i=i+1 \quad (i=6)$

اکنون فقط یک عنصر از B مانده $(B[4])$ و چون قبلاً در C آمده است، لذا آنچه در C آمده است اجتماع B و A است.

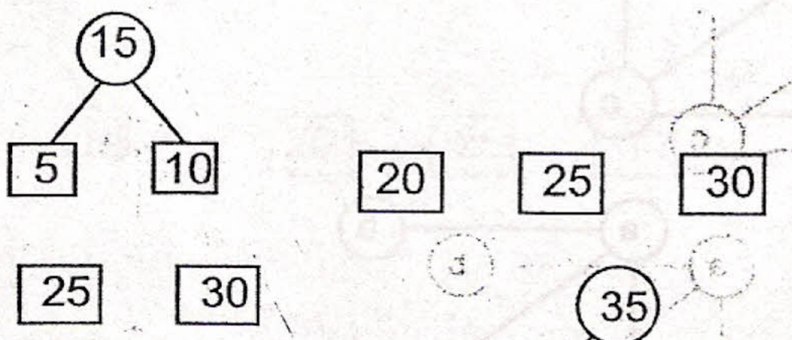
اشتراک A و B برای مثال داده شده نیز آرایه‌ای مثل D است که فقط شامل عنصر 2 است $(D[1])=2$

۵- برای حل این مساله باید درخت ادغام دودویی بهینه را تشکیل داده برای آن WEPL (Weighted External Path Length) را حساب کنیم: مراحل مختلف تشکیل درخت ادغام دودویی با شروع از دو فایل با کمترین اندازه به صورت زیر است:

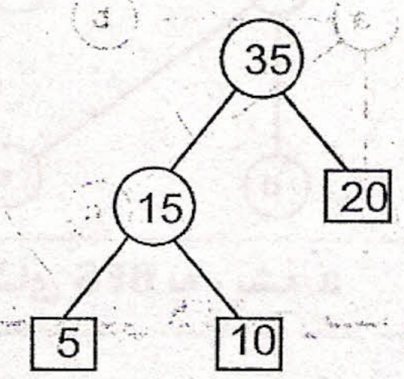
مرحله شروع



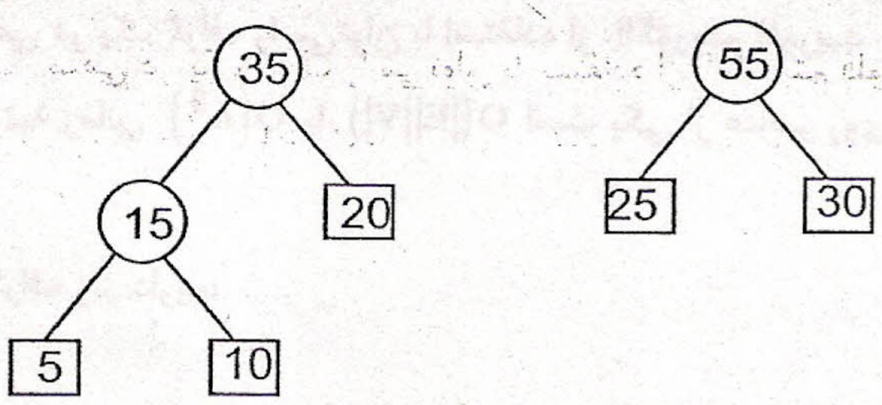
مرحله ۱



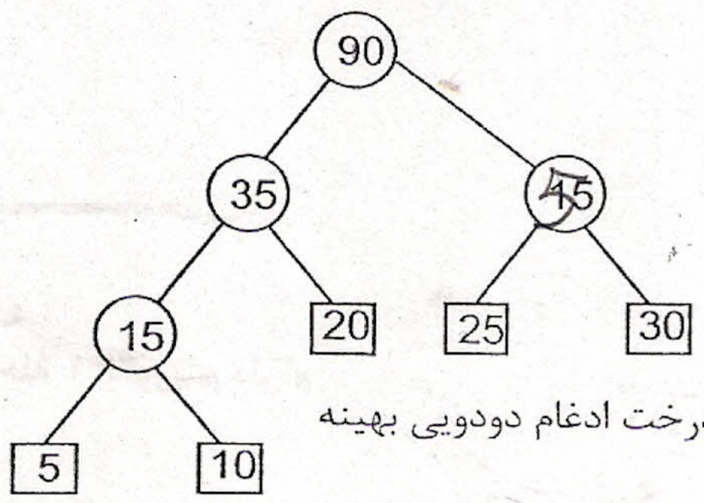
مرحله ۲



مرحله ۳



مرحله ۴



درخت ادغام دودویی بهینه

برای این درخت، WEPL به صورت زیر محاسبه می‌شود.

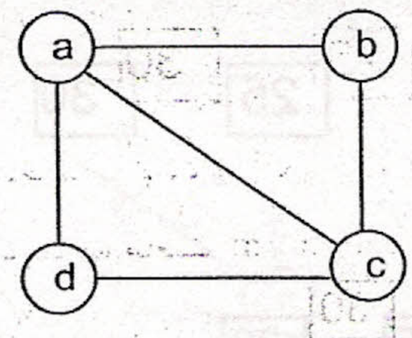
$$WEPL=\sum_{\text{گره } x} d(x).w(x)=5\times3+10\times3+20\times2+25\times2+30\times2=195$$

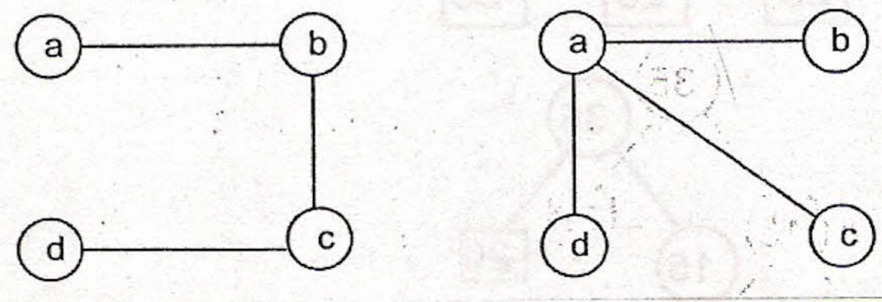
خارجی

۶-

۷- این مساله دقیقاً مسالهٔ شماره ۱ فنآوری اطلاعات سال ۱۳۸۴ است.

۸- الف) درخت‌های پوشای DFS , BFS برای یک گراف الزاماً برابر نیستند. برای مثال برای گراف زیر اگر درخت‌های پوشای DFSو BFS را با شروع از راس a به ترتیب اولویت c,b,a و d تشکیل دهیم نتایج به ترتیب به صورت شکل‌های T_1 و T_2 خواهند بود.

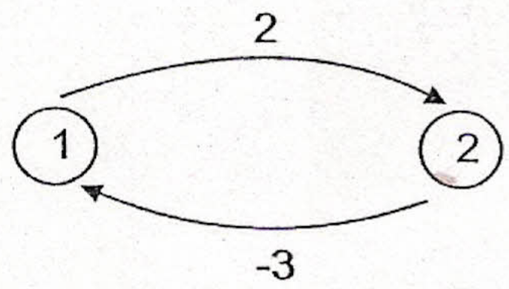




درخت پوشای BFS به ریشه a درخت پوشای DFS به ریشه a

ب) وجود دور منفی در یک گراف را می‌توان با استفاده از الگوریتم فلوریـد ـ وارشـال تشـخیص داد. اگـر در مرحلـه‌ای از اِعمـال ایـن الگوریتم که از مرتبهٔ زمانی $O(n^3)$ یا $O(|E||V|)$ است یکی از عناصر روی قطر اصلی منفی شـود، در گـراف دوری بـا هزینـهٔ منفـی وجود دارد.

برای مثال برای گراف زیر داریم:



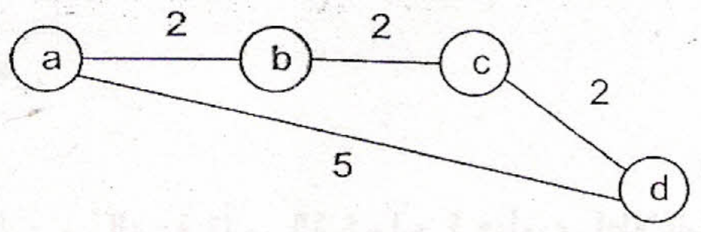
$$A = A^{\circ} = \begin{bmatrix} 0 & 2 \\ -3 & 0 \end{bmatrix}$$

بعد از اجرای مرحلهٔ ۱ الگوریتم داریم

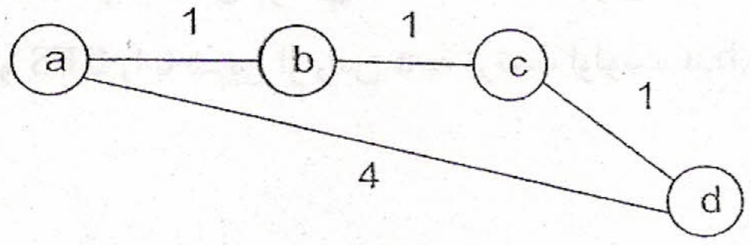
$$A^{1} = \begin{bmatrix} 0 & 2 \\ -3 & -1 \end{bmatrix}$$

و بعد از اتمام الگوریتم $A^{2} = \begin{bmatrix} -1 & 2 \\ -3 & -1 \end{bmatrix}$

ج) در گراف زیر طول کوتاه‌ترین مسیر بین d,a برابر 5 است که وزن یال (a,d) است.

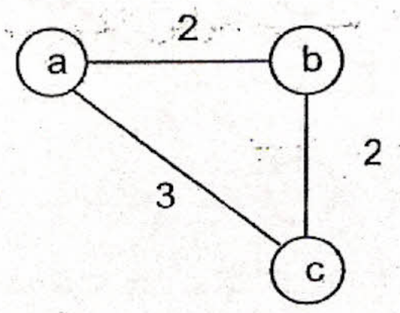


حال وزن تک تک یال‌ها را به اندازهٔ یک واحد کم می‌کنیم. گراف حاصل به صورت زیر است.

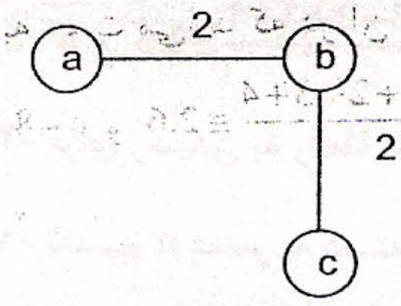


مشاهده می‌شود که کوتاه‌ترین مسیر بین d,a دیگر یال (a,d) نیست زیرا الآن طول مسیر d,c,b,a برابر ۳ است که کمتر از وزن یال (a,d) است.

د) گراف زیر را در نظر بگیرید:



برای این گراف کوتاه‌ترین مسیر بین a, c همان (a,c) است که هزینهٔ آن 3 است. درخت پوشای مینیمم برای این گراف به صورت زیر است.

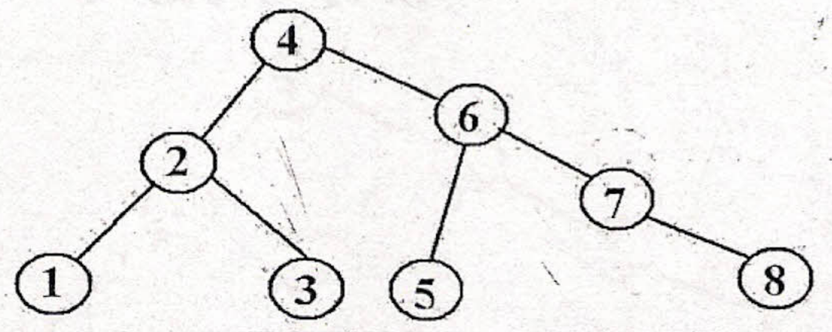


مشاهده می‌شود که طول تنها مسیر بین a, c در این درخت پوشای مینیمم برابر 4 است. بنابراین مسیری که در درخت پوشای مینیمم بین دو راس v,u وجود دارد، الزاماً کوتاه‌ترین مسیر بین دور راس در خود گراف نیست.

۱- درخت جستجوی دودویی برای آرایه مرتب شدهٔ

$A[1..8]=[5,10,15,20,25,30,35,40]$

به صورت زیر است:



میانگین تعداد مقایسه‌ها برای جستجوی مؤفق از رابطهٔ $\frac{\sum[d(x)+1]}{n}$ به دست می‌آید که در آن $d(x)$ مسافت گره x از ریشه است. برای مثال، مسافت گره 1 از درخت بالا از ریشه برابر 2 است. لذا داریم :

$n=8$ و $\frac{3+2+1+3+3+2+3+4}{8}\cong 2.6$

۲- برای رسیدن به رابطهٔ بازگشتی الگوریتم ارایه شده، مراحل این الگوریتم را به صورت خلاصه مرور می‌کنیم.

۱- تقسیم n عنصر به دسته‌های 5 تایی (به جزء احتمالاً یک دسته) مستلزم زمان $O(n)$ است. تعداد این دسته‌ها هم $\left\lceil \frac{n}{5} \right\rceil$ است.

۲) پیدا کردن میانه برای هر دسته. $O(n)$

۳) میانهٔ این $\left\lceil \frac{n}{5} \right\rceil$ دسته را برداشته در آرایه‌ای دیگر می‌گذاریم، سپس میانهٔ میانه‌ها (mm) را به‌صورت بازگشتی به دست می‌آوریم $O\left(\left\lceil \frac{n}{5} \right\rceil\right)$.

۴) با استفاده از mm به عنوان عنصر افراز و الگوریتم Partition ، آرایهٔ اولیه را افراز می‌کنیم. فرض کنید j اندیس عنصر mm بعد از افراز باشد:

الف) اگر $k=j$، مساله حل شده است.

ب) اگر $k<j$ باید مساله را به صورت بازگشتی برای آرایه $A[1..j-1]$ حل کنیم.

ج) اگر $k>j$ باید مساله را به‌صورت بازگشتی برای آرایهٔ $A[j+1..n]$ حل کنیم.

حال می‌خواهیم نشان دهیم حداکثر تعداد عناصری که در زیر آرایهٔ $A[1..j-1]$ یا $A[j+1..n]$ می‌توانند وجود داشته باشد $\frac{7n}{10}+6$ است. برای درک بهتر مطلب مثالی را ارایه می‌دهیم. فرض کنید $n=42$ و مراحل 3,2,1 را انجام داده‌ایم

2	17	1	15	10	11	55	12	
3	18	2	20	20	12	60	15	6
10	20	30	40	50=mm	60	70	80	90
12	25	31	100	60	65	75	90	
21	40	32	200	61	66	80	95	

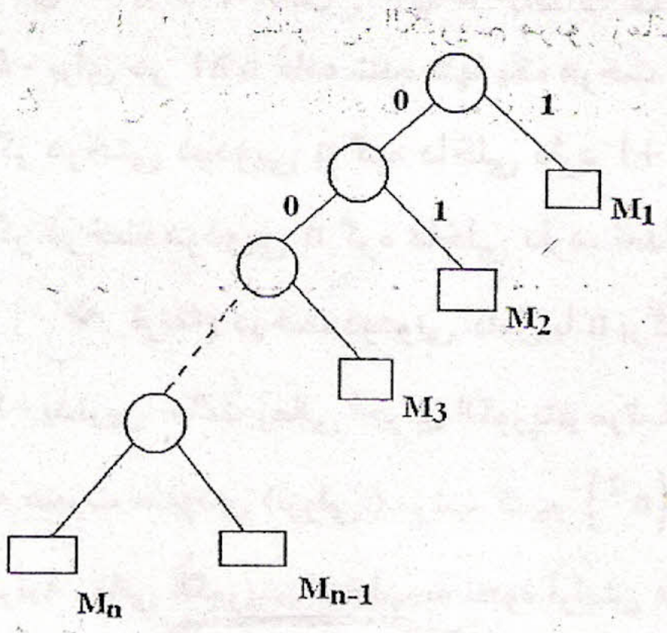
مشاهده می‌شود که حداقل نصف میانه‌های پیدا شده در مرحلهٔ ۲ بزرگ‌تر یا مساوی mm هستند. بنابراین حداقل نصف $\left\lceil \frac{n}{5} \right\rceil$ دستهٔ مورد اشاره، 3 عنصر بزرگ‌تر یا مساوی mm دارند. البته به استثنای دسته آخر که احتمالاً تعداد اعضای آن کمتر از 5 است و دسته‌ای

که میانهٔ میانه‌ها در آن قرار دارد. اگر این دو دسته را کنار گذاریم تعداد عناصری که بزرگ‌تر یا مساوی mm هستند، حداقل برابر $3\left(\left\lceil\frac{1}{2}\right\rceil\left\lceil\frac{n}{5}\right\rceil-2\right)\geq\frac{3n}{10}-6$ است.

به طریق مشابه می‌توان نشان داد که تعداد اعضایی که کوچک‌تر یا مساوی mm هستند، حداقل برابر $\frac{3n}{10}-6$ است. بنابراین الگوریتم انتخاب را باید حداکثر بر روی $n-\left(\frac{3n}{10}-6\right)=\frac{7n}{10}+6$ عنصر اعمال کنیم. در نتیجه رابطهٔ بازگشتی به صورت زیر است.

$$\boxed{t(n)=t\left(\left\lceil\frac{n}{5}\right\rceil\right)+t\left(\frac{7n}{10}+6\right)+O(n)}$$

۳- درخت کشف رمز برای کد هافمن می‌تواند درختی دودویی و اریب به شکل زیر باشد که طولانی‌ترین کد را برای پیغامی مثل M_n تولید می‌کند. ارتفاع این درخت $n-1$ است. بنابراین حداکثر بیت‌های اختصاص داده شده برای یک کد در کدگذاری به روش هافمن برابر $n-1$ است.



۴-

```
i=1 , j=n
while i≤j do
{
        if S[i]+S[j]=k then return i, j
        if S[i]+S[j]<k then i=i+1
        if S[i]+S[j]>k then j=j-1
}
return not possible
```

آرایه نزولی زیر را در نظر بگیرید

	1	2	3	4	5
S	5	4	3	2	1

فرض کنید $k=7$ در شروع $i=1$, $j=5$ داریم:

$S[i]+S[j]=S[1]+S[5]=6<7$

لذا $i=i+1 \Rightarrow i=2$

$S[i]+S[j]=S[2]+S[5]=5<7$

لذا $i=i+1 \Rightarrow i=3$

$S[i]+S[j]=S[3]+S[5]=4<7$

پس $i=4 \Leftarrow i=i+1$

$S[i]+S[j]=3<k \Rightarrow i=5$

$S[5]+S[5]=2<k \Rightarrow i=i+1 \Rightarrow i=6$

مشاهده می‌شود که برای مثال $S[1]+S[4]=7=k$، ولی الگوریتم ارایه شده نتوانسته آن را پیدا کند. نتیجه این‌که الگوریتم برای همهٔ حالت‌ها از جمله برای حالتی که آرایه به صورتی نزولی مرتب است ممکن است نتواند جواب درست دهد.

حال فرض کنید آرایهٔ S به صورت صعودی مرتب شده است در این صورت:

الف اگر $A[i]+A[j]=k$ که الگوریتم جواب درست داده است.

ب) اگر $A[i]+A[j]<k$ آن‌گاه بدیهی است که $\forall l<i$ داریم $A[l]+A[j]<k$ بنابراین کافی است عناصر $l>i$ را با $S[j]$ بررسی کنیم. لذا $i=i+1$

ج) اگر $S[i]+S[j]>k$ آن‌گاه $\forall l>j$ نیز رابطهٔ $S[i]+S[l]>k$ برقرار است بنابراین کافی است $S[i]$ را با عناصر $l<j$ مقایسه کنیم، یعنی $j=j-1$. بنابراین، الگوریتم مزبور زمانی که آرایه به صورت صعودی مرتب شده باشد، درست عمل می‌کند.

۵- برای هر $n\geq 1$ داده شده تنها یک درخت دویی می‌توان ساخت.

اگر درختی دودویی n گره داخلی دارد $n+1$ برگ خواهد داشت.

اگر درخت دودویی n گره داخلی دارد، تعداد کل گره‌های آن (برگ‌ها + گره‌های داخلی) $2n+1$ است.

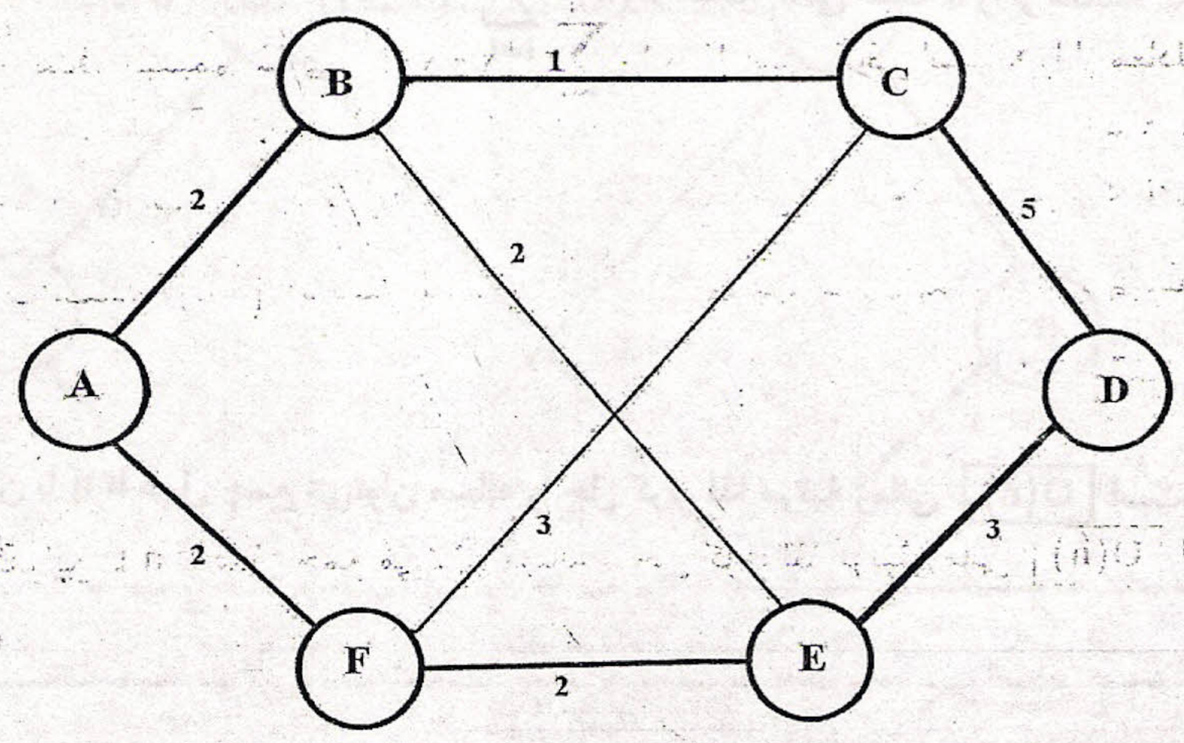
❖ ارتفاع درخت دودویی کامل با n برگ یا $n-1$ گره داخلی از رابطهٔ $\lfloor \log_2(n-1) \rfloor$ به دست می‌آید.

۶- بدترین حالت زمانی اجرای الگوریتم مرتب سازی سریع زمانی است که خود آرایهٔ داده شده صعودی (نزولی) باشد و بخواهیم آن را به صورت صعودی (نزولی) مرتب کنیم $O(n^2)$.

مرتبهٔ زمانی الگوریتم انتخاب به نحوهٔ آرایش عناصر آرایه بستگی ندارد و در بدترین حالت، بهترین حالت و حالت متوسط برابر $O(n^2)$ است. اما برای مرتب‌سازی درجی این مطلب متفاوت است. اگر آرایه قبلاً به صورت صعودی مرتب شده باشد و بخواهیم آن را به‌صورت صعودی مرتب کنیم، زمان اجرا $O(n)$ است. مرتبهٔ زمانی مرتب‌سازی هرمی نیز در بهترین، بدترین و حالت متوسط $O(n\log n)$ است. لذا برای آرایهٔ A که صعودی است و می‌خواهیم آن را به‌صورت صعودی مرتب کنیم مرتبهٔ زمانی $O(n)$ است که سریع‌تر از الگوریتم‌های مرتب‌سازی انتخابی از مرتبهٔ $O(n^2)$ و مرتب‌سازی سریع از مرتبهٔ $O(n^2)$ است.

۷- ترتیب انتخاب یال‌ها در الگوریتم کرسکال عبارت‌اند از: BC که هزینهٔ آن 1 است. بیشترین هزینه بعد از 1 مربوط به یال‌های AB ، BE، FE , AF و برابر 2 است. اضافه کردن هیچ کدام از آن‌ها با یال BC دور تشکیل نمی‌دهد. با وجود این، خود این، یال‌ها دور تشکیل داده‌اند. لذا در مراحل ۲ ،۳، ۴ الگوریتم کراسکال می‌توان هر کدام از این ۴ یال را انتخاب کرد (در مرحلهٔ ۴ تعداد انتخاب‌ها ۴ است، در مرحلهٔ ۳ این تعداد 3 است و در مرحلهٔ ۴ هم یکی از یال دو یال باقی مانده را می‌توان انتخاب کرد) در مرحلهٔ ۵ (مرحلهٔ آخر) نیز با توجه به این‌که یال AB انتخاب نشده است یا انتخاب شده است به ترتیب می‌توان FC یا ED را انتخاب کرد.

اما در الگوریتم پریم اگر از راس A شروع کنیم می‌توانیم AB یا AF را انتخاب کنیم. در مرحلهٔ بعد بسته به این‌که در مرحلهٔ نخست AB یا AF را انتخاب کرده‌ایم به ترتیب BC یا یکی از دو یال AB یا FE را انتخاب خواهیم کرد و الی آخر.

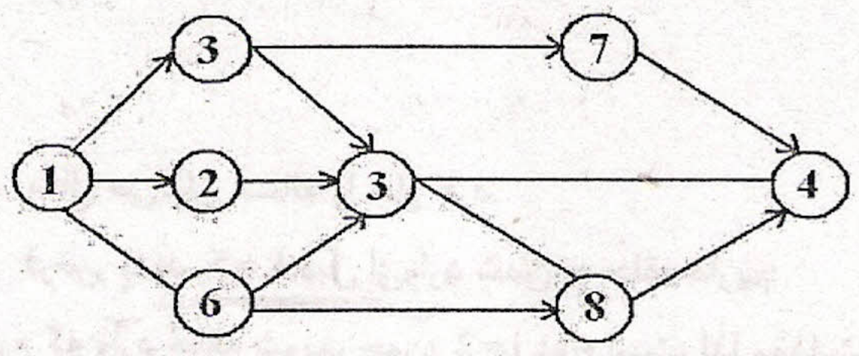


بنابراین در کراسکال ترتیب انتخاب یال‌ها می‌تواند به صورت زیر باشد (از چپ به راست).

BC, BE, FE, AF, ED

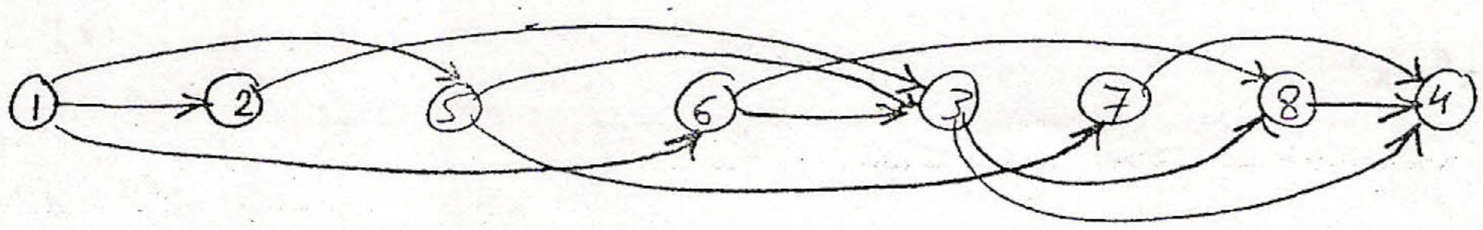
و در الگوریتم پریم نیز این ترتیب از چپ به راست عبارت است از :

AF, FE, BE, BC, ED



۸- ترتیب توپولوژیکی برای گراف جهت‌دار $G=(V,E)$ ترتیبی خطی از رئوس (چیدن رئوس بر روی خط راست از چپ به راست) است به گونه‌ای که اگر $(u,v)\in E$ آن گاه u سمت چپ v است. برای به دست آوردن ترتیب توپولوژیکی می‌توان از الگوریتم DFS استفاده کرد.

۷ نمونه از ترتیب‌های همه توپولوژیکی برای گراف مزبور به‌صورت زیر است:



2 ① → ② ⑥ ⑤ ③ ⑦

3 ① → ⑤ ② ⑥ ③ ⑦

4 ① → ⑤ ⑥ ② ③ ⑦

5 ① → ⑥ ② ⑤ ③ ⑦

6 ① → ⑥ ⑤ ② ③ ⑦

7 ① → ⑥ ⑤ ② ③ ⑧

بنابراین آنچه مسلم است این است که گراف مزبور بیش‌تر از شش ترتیب توپولوژیکی دارد.

۱- هدف مینیمم سازی $E(x)=\sum_{i=1}^{n}(x-x_i)^2$ است. کافی است x را از معادلهٔ $E'(x)=0$ به دست آوریم.

$$E'(x)=2\sum_{i=1}^{n}(x-x_i)=0$$

$$nx-\sum_{i=1}^{n}x_i=0 \Rightarrow \boxed{x=\frac{1}{n}\sum_{i=1}^{n}x_i}$$

بنابراین با n تا عمل جمع می‌توان مساله را حل کرد، لذا مرتبهٔ زمانی $\boxed{O(n)}$ است.

۲-

```
Function power(x,k)
  y=x , i=k , a=1
  while i>0 do
     if i is odd then
        a=a*y
     y=y*y
     i=⌊i/2⌋
  }
  Return a
```

با یک مثال می‌توان مساله را حل کرد.

مثال فرض کنید k=5.قبل از وارد شدن به حلقه داریم: $a=1, i=5, y=x$

بار دوم که وارد حلقه شویم، چون i=2 زوج است، لذا مقدارش عوض نمی‌شود. الآن در شروع حلقه هستیم و چهار رابطهٔ ارائه شده را بررسی می‌کنیم

$$\begin{cases} y(ay)^i=x^2(xx^2)^2=x^8\neq x^5 \\ a(ay)^i=x(xx^2)^2=x^7\neq x^5 \\ y^i a^i=(x^2)^2(x)^2=x^6\neq x^5 \\ y^i a=(x^2)^2 x=x^5 \end{cases}$$

۳- مشابه سوال ۲ سال ۱۳۸۴ هوش مصنوعی است.

۴- می‌دانیم که برای ضرب اعداد صحیح خیلی بزرگ نمی‌توان از روش معمولی و کلاسیک استفاده کرد برای این منظور از روش تقسیم و غلبه استفاده می‌شود. از طرف دیگر این مساله حالت خاصی از ضرب دو چند جمله‌ای مثل $A(x),B(x)$ است. اگر $A(x)$ و $B(x)$ هم درجه باشند، از روش تقسیم و غلبه معمولی می‌توان برای ضرب $A(x)$ در $B(x)$ استفاده کرد. حال فرض کنید $A(x)$ از درجه $(n-1)$ و $B(x)$ از درجهٔ $4(n-1)$ است. برای درک بهتر مطلب فرض کنید $A(x), n=4$ از درجه $B(x),3$ از درجه 15 است:

$$A(x)=a_0+a_1x+a_2x^2+a_3x^3$$

$$B(x)=b_0+b_1x+b_2x^2+b_3x^3+b_4x^4+...+b_{15}x^{15}$$

چون $A(x), B(x)$ هم درجه نیستند به دو صورت می‌توان از روش تقسیم و غلبه برای ضرب $A(x)$ و $B(x)$ استفاده کرد. می‌توان $A(x)$ را تا درجه 15 با ضرایب صفر $b_4=b_5=..=b_{15}$ گسترش داد که در این حالت دو چند جمله‌ای، هر کدام با درجهٔ $4(n-1)$ را باید در هم ضرب کرد. چون مرتبهٔ بزرگی الگوریتم تقسیم و غلبهٔ معمولی(تقسیم و غلبهٔ سریع مورد نظر نیست) $\theta(n^2)$ است، لذا ضرب این دو چند جمله‌ای 16 برابر کندتر از ضرب دو چند جمله‌ای از درجهٔ n-1 خواهد بود.

حال فرض کنید $B(x)$ را به صورت زیر بنویسیم:

$$B(x)=(b_0+b_1x+b_2x^2+b_3x^3)+x^4(b_4+b_5x+b_6x^2+b_7x^3)$$
$$+x^8(b_8+b_9x+b_{10}x^2+b_{11}x^3)+x^{12}(b_{12}+b_{13}x+b_{14}x^2+b_{15}x^3)$$
$$=B_1(x)+x^4B_2(x)+x^8B_3(x)+x^{12}B_4(x)$$

اکنون ضرب $A(x)$ و $B(x)$ را می‌توان به صورت زیر بیان کرد.

$$A(x)B(x)=A(x)B_1(x)+x^4A(x)B_2(x)+x^8A(x)B_3(x)+x^{12}A(x)B_4(x)$$

به عبارت دیگر مسالهٔ ضرب $A(x)$ از درجهٔ $n-1$ به $B(x)$ از درجهٔ $4(n-1)$، به چهار ضرب چندجمله‌ای از درجه‌های $n-1$ تبدیل شده است. فراموش نکنید که برای مثال ضرب x^8 در $A(x)B_3(x)$ را می‌توان به صورت شیفت انجام داد. بنابراین در این حالت حل، 4 برابر کندتر از ضرب دو چند جمله‌ای از درجه‌های n-1 است و در عوض 4 برابر سریع‌تر از ضرب دو چند جمله‌ای از درجه‌های 4(n-1) است.

```
function con cow (n)
    if  n≤1 then    write ln ('stop') Re turn 1
      else
        for  i=1  to n do
            write ln (con cow (n div2))
        Re turn n
```

۵-

این تابع، تابعی تقسیم و غلبه است. اگر حلقهٔ for را نداشتیم رابطهٔ تقسیم و غلبه برای شمارش دستور writeln به صورت زیر بود:

$$t(n)=1+t\left(\left\lfloor\frac{n}{2}\right\rfloor\right)$$

اما با توجه به این‌که هر بار، n فراخوانی به con cow وجود دارد، لذا داریم

$$\boxed{t(n)=n+n\,t\left(\left\lfloor\frac{n}{2}\right\rfloor\right)}$$

۱- چون $a_1, a_2, \ldots$ و a_k اعدادی صحیح و مثبت هستند:

$$a_i \geq 1 \;,\; i=1,2,\ldots,k$$

لذا مجموع آن‌ها باید ناکمتر از k باشد:

$$\sum_{i=1}^{k} a_i \geq k$$

برای حالت تساوی هم باید داشته باشیم $a_i=1$، $\forall_i=1,2,\ldots,k$

در نتیجه $t(n)=n\,t(n)+\theta(n)$ که امکان ندارد. پس تنها امکان موجود $\boxed{\sum_{i=1}^{k} a_i > k}$ است.

۲-

```
Function g(n)
    if n≤1  then g=n
      else
            g=5*g(n-1)-6*g(n-2)
end
```

این تابع معادل رابطهٔ بازگشتی زیر است:

$$\begin{cases} g_n = 5g_{n-1} - 6g_{n-2} \\ g_0 = 0,\; g_1 = 1 \end{cases}$$

که رابطه‌ای همگن و درجهٔ 2 است. ساده‌ترین روش برای حل آن استفاده از معادلهٔ مشخصه است:

$$r^2 - 5r + 6 = 0$$

ریشه‌های آن عبارت‌اند از $r_1, r_2 = 2,3$. پس جواب عمومی به صورت زیر است.

$$g_n = C_1 2^n + C_2 3^n$$

با توجه به شرط مرزی $g_0=0$ داریم $0=C_1+C_2$. بنابراین مجموع ضرایب صفر است و جواب $3^n - 2^n$ درست است.

۳- چند نکته در مورد نمادهای $\omega, 0, \theta, \Omega, O$

تشابه بین نمادهای مجانبی $\omega, 0, \theta, \Omega, O$ در مقایسهٔ دو تابع عددی $f(n)$ و $g(n)$ و نمادها (روابط) $\leq, \geq, =, <$ و $>$ در مقایسه دو عدد حقیقی a و b

1) $f(n)=O(g(n)) \approx a \leq b$

2) $f(n)=\Omega(g(n)) \approx a \geq b$

3) $f(n)=\theta(g(n)) \approx a = b$

4) $f(n)=0(g(n)) \approx a < b$

5) $f(n)=w(g(n)) \approx a > b$

بنابراین هر خاصیت شناخته شده‌ای را در مورد روابط $\leq, \geq, =, <$ و $>$ در مقایسهٔ بین دو عدد حقیقی a و b می‌توان به‌ترتیب به نمادهای O و $\omega, 0, \theta, \Omega$ تعمیم داد. برای مثال همهٔ این روابط خاصیت تعدی دارند، لذا همهٔ این نمادهای مجانبی نیز

خاصیت تعدی دارند. برای مثال اگر $f(n)=O(g(n)), g(n)=O(h(n))$ داریم $f(n)=O(h(n))$. رابطهٔ تساوی = خاصیت تقارنی دارد $a=b \Leftarrow b=a$. لذا اگر $f(n)=\theta(g(n))$ داریم $g(n)=\theta(f(n))$

(برای توضیحات بیشتر به کتاب طراحی و تحلیل الگوریتم‌ها: قلی‌زاده صفحات ۱۷و۱۸ مراجعه کنید)

حال با یادآوری خاصیتی دیگر به حل مساله می‌پردازیم. می‌دانیم که اگر $a \le b$ باشد الزاماً $b \le a$ نیست.

چون $f(n)=O(g(n))$ لذا الزاماً $g(n)=O(f(n))$ نیست. اما $2^{f(n)}=O\left(2^{g(n)}\right)$ و نیز $2^{g(n)}=\Omega\left(2^{f(n)}\right)$ درست است.

هم‌چنین اگر $0 \le a \le b$ آن‌گاه $\log a \le \log b$ ولی رابطهٔ $\log b \le \log a$ برقرار نیست. لذا $\log g(n)=O(\log f(n))$ درست نیست.

۴- برای پیدا کردن کوتاه‌ترین مسیر بین دو راس در گرافی وزن‌دار، الگوریتم‌های شناخته شده‌ای مثل دایکسترا ارایه شده است که مرتبهٔ زمانی پایین‌تری هم دارد ($O\left(n^2\right)$، که در آن n تعداد رئوس گراف است).

اما برای پیدا کردن بلندترین مسیر ساده بین دو راس در گرافی وزن‌دار، الگوریتمی از مرتبهٔ چند جمله‌ای ارایه نشده است. این مساله، یک مسالهٔ NP کامل است حتی اگر وزن همهٔ یال‌ها مساوی و مثلاً برابر 1 باشد (کتاب Cormen صفحه ۹۶۶)

۵-

$$P=\begin{bmatrix} 0 & 0 & 4 & 2 \\ 4 & 0 & 4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

چون $P[1,3]=4$ لذا مسیر از راس 1 به راس3 حتماً از راس ۴ عبور می‌کند:

① → ....④ → ...②

حال برای مشخص کردن اینکه آیا یا راس 2 در مسیر از 1 به 3 هست یا نه و اگر هست در زیر مسیر از راس 1 به 4 است یا در زیر مسیر از 4 به 3 قرار دارد به $P[4,2]$, $P[1,4]$ مراجعه می‌کنیم. $P[1,4]=2$, $P[4,2]=0$

پس راس 2 در زیر مسیر از راس 1 به 4 قرار دارد . پس مسیر به صورت زیر است.

① → ② → ④ → ③

۶- جواب این سؤال در سؤال۶ سال ۱۳۸۴ مهندسی کامپیوتر داده شده است.با این توضیح اضافی که مرتبهٔ زمانی الگوریتم دایکسترا در صورت استفاده از درخت سیاه ـ سفید می‌تواند به صورت $O(|E|\log|V|)$ باشد.

۷- بیشترین تعداد زوج ـ معکوس‌ها در یک آرایهٔ n عضوی زمانی است که آرایه به صورت نزولی مرتب شده باشد. در این حالت داریم:

n-1 عضو بزرگ‌تر از $A[1]$ هستند $\Rightarrow A[1] > A[j], j=2,3,...,n$

n-2 عضو بزرگ‌تر از $A[2]$ هستند $\Rightarrow A[2] > A[j], j=3,4,...,n$

$\vdots$

1 عضو بزرگ‌تر از $A[n-1]$ است. $\Rightarrow A[j-1] > A[j], j=n$

بنابراین در جمع

$$1+2+..+(n-2)+(n-1)=\frac{n(n-1)}{2}$$

زوج معکوس وجود دارد.

۸- مساله ،معادل با استفاده از الگوریتم مرتب‌سازی سریع است که خود از زیر الگوریتم Partition استفاده می‌کند. فرق در این است که در این‌جا نیازی به مرتب‌سازی کامل آرایه نیست (نیازی به تقسیم آرایه تا حد یک عنصر نیست). اگر هر زیر آرایهٔ حاصل از تقسیم حداکثر $\left\lceil \frac{n}{k} \right\rceil$ عنصر داشته باشد نیازی به تقسیم مجدد آن نیست. به عبارت دیگر اگر زیر آرایه ای 2^r عضو داشته باشد الگوریتم متوقف می‌شود. اگر برای سهولت فرض کنیم که $n=2^m$ است، جواب مساله از حل رابطهٔ تقسیم و غلبهٔ زیر حاصل می‌شود:

$$\begin{cases} t(n)=2t\left(\frac{n}{2}\right)+cn & n>2^r \\ t(n)=cte & n\le 2^r \end{cases}$$

که در آن cn مرتبهٔ زمانی برای تعیین عنصر افراز است. اگر این رابطهٔ تقسیم و غلبه را با استفاده از روش جایگذاری حل کنیم خواهیم داشت:

$$t(n)=cn+2t\left(\frac{n}{2}\right)$$

$$=cn+cn+2^2t\left(\frac{n}{2^2}\right)$$

$$=cn+cn+cn+2^3t\left(\frac{n}{2^3}\right)$$

$$=\underbrace{cn+cn+.....+cn}_{r}+2^r t\left(\frac{n}{2^r}\right)=crn+c_1k=\theta(rn)=\theta(n\log k)$$

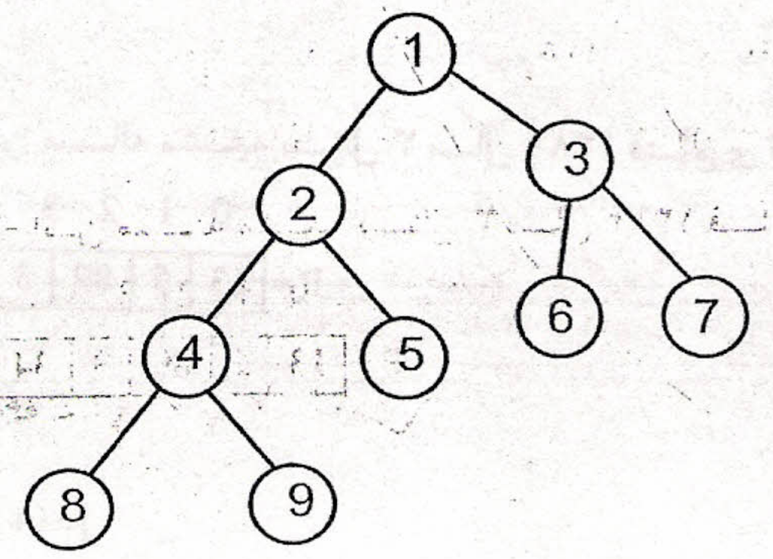
۱-

هیپ درختی است دودویی و کامل که عمق ریشهٔ آن (گره شماره 1) 0 است.

عمق گره‌های شماره 3,2 (در صورت وجود) 1 است.

عمق گره‌های شمارهٔ 7,6,5,4, (در صورت وجود) 2 است.

بنابراین عمق گره شماره i برابر $\boxed{\lfloor \log_2 i \rfloor}$ خواهد بود.



۲-

```
1)      for i=1 to n
2)          for j=n to i
3)              for k=1 to n^2 do
4)                  sum=sum+A
```

کافی است تعداد دفعات اجرای دستور 4 را بشماریم:

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=1}^{n^2} 1 = n^2\sum_{i=1}^{n}\sum_{j=i}^{n} 1 = n^2\sum_{i=1}^{n}(n-i+1)$$

$$= n^2\left[n(n+1)-\frac{n(n+1)}{2}\right] = \frac{n^3(n+1)}{2} = \theta\left(n^4\right)$$

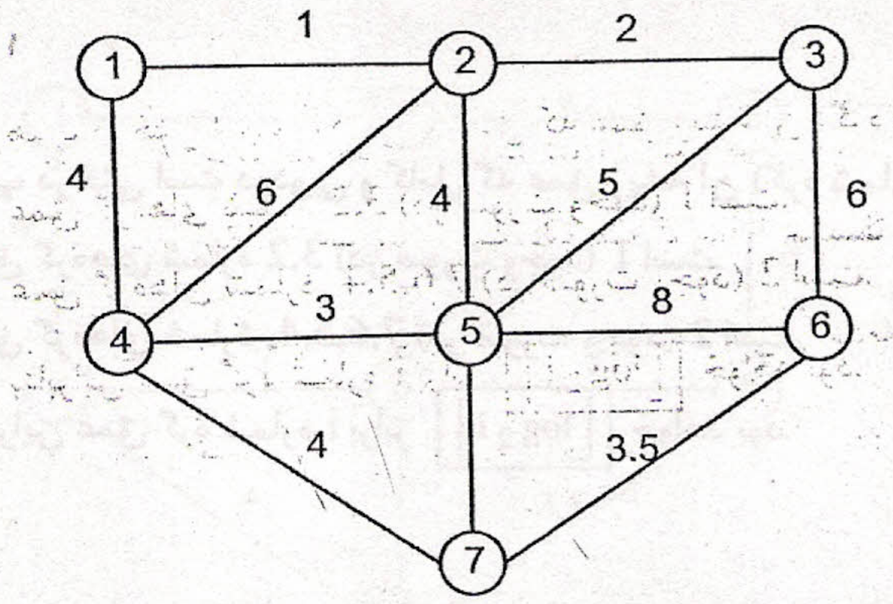
۳- الگوریتم کراسکال یال‌ها را به‌ترتیب صعودی هزینهٔ آنها انتخاب می‌کند، مشروط بر این‌که هر یال مورد بررسی با یال‌های قبلاً انتخاب شده دور تشکیل ندهد. بنابراین اگر الگوریتم کراسکال را بر روی گراف بالا اعمال کنیم یال‌های انتخاب شده تا مرحلهٔ چهارم عبارت‌اند از :

۱) یال (1,2)

۲) یال (2,3)

۳) یال (4,5)

۴) یال $\boxed{(6,7)}$



۴- این مساله مشابه سؤال ۳ سال ۱۳۸۴ فناوری اطلاعات است و به روش پویا می‌توان آن را حل کرد. داریم n=4

	0	1	2	3	4
P=	13	5	89	3	34

جواب مساله $m[1,4]$, $m[i,i]=0$ شرط مرزی

$$m[i,j]=\underset{i\le k<j}{\mathrm{Min}}\{m[i,k]+m[k+1,j]+P[i-1]P[k]P[j]\}$$

برای $j=i+1$

$$m[i,i+1]=p[i-1]p[i]p[i+1]$$

$$m[1,3]=\mathrm{Min}\left\{\overset{0}{m[1,1]}+\overset{1335}{m[2,3]}+P[0]P[1]P[3]\ ,\ \overset{5785}{m[1,2]}+\overset{0}{m[3,3]}+P[0]P[2]P[3]\right\}$$

$$m[2,4]=\left\{\overset{0}{m[2,2]}+\overset{9078}{m[3,4]}+P[1]P[2]P[4],m[2,3]+\overset{0}{m[4,4]}+P[1]P[3]P[4]\right\}$$

$$m[1,4]=\mathrm{Min}\left\{\overset{0}{m[1,1]}+\overset{1845}{m[2,4]}+\overset{2210}{P[0]P[1]P[4]},m[1,2]+m[3,4]+P[0]p[2]P[4],\underset{1530}{m[1,3]}+\underset{0}{m[4,4]}+P[0]p[3]P[4]\right\}$$

i \ j	1	2	3	4
1	0	5785 (k=1)	1530 (k=2)	2856 (k=3)
2		0	1335 (k=2)	1845 (k=3)
3			0	9078 (k=3)
4				0

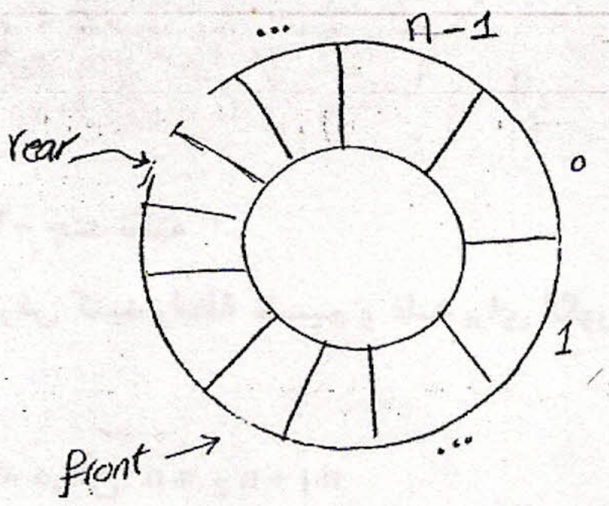
پس حداقل تعداد ضرب‌های لازم برابر $\boxed{2856}$ و نحوهٔ ضریب ماتریس‌ها هم به‌صورت زیر است.

$$((A(BC))D)$$

البته با توجه به این‌که 4 ماتریس را می‌شود به 5 حالت در هم ضرب کرد، می‌توانیم مساله را به روش سعی و خطا نیز حل کنیم:

نحوهٔ ضرب	تعداد ضرب‌ها
$((A(BC))D)$	$1335+195+1326=\boxed{2856}$
$(A((BC)D))$	$1335+510+2210=4055$
$((AB)(CD))$	$5785+9078+39338=54201$
$(((AB)C)D)$	$5785+3471+1326=10582$
$(A(B(CD)))$	$9078+15130+2210=16418$

۵- با فرض $n=MAXQSIZE$ ، داریم



اگر $front=rear$ آن‌گاه صف خالی است در غیر این‌صورت برای حذف عنصری که در مکان front است کافی است اشاره‌گر font را به اندازهٔ یک واحد در جهت چرخش عقربه‌های ساعت جلو ببریم. برای این منظور کافی است به front یک واحد اضافه کنیم با وجود این اگر $front+1=n$ در این صورت باید front را مساوی صفر قرار دهیم. عبارت مزبور معادل دستور زیر است.

$$\boxed{front=(front+1)\bmod n}$$

۶- مرتبهٔ زمانی چند الگوریتم مرتب‌سازی معروف در بدترین حالت، بهترین حالت، و حالت متوسط:

نام الگوریتم	مرتبهٔ زمانی		
مرتب‌سازی درجی	$O(n^2)$	$\Omega(n)$	$\theta(n^2)$
مرتب‌سازی ادغامی	$O(n\log n)$	$\Omega(n\log n)$	$\theta(n\log n)$
مرتب‌سازی سریع	$O(n^2)$	$\Omega(n\log n)$	$\theta(n\log n)$
مرتب‌سازی انتخابی	$O(n^2)$	$\Omega(n^2)$	$\theta(n^2)$
مرتب‌سازی حبابی	$O(n^2)$	$\Omega(n^2)$	$\theta(n^2)$

بنابراین الگوریتم مرتب‌سازی درجی در بدترین حالت و در حالت متوسط از مرتبهٔ زمانی $O(n^2)$ است.

۱- در الگوریتم زیر در هر تعویض دقیقاً یک عنصر سرجای خودش قرار می‌گیرد، بنابراین حداکثر n-1 تعویض لازم است تا همهٔ عناصر به صورت صعودی مرتب شوند.

```
for i=1 to n-1 do
    while A[i]<> i do
        swap(A[i],A[A[i]])
```

به مثال زیر توجه کنید:

	1	2	3	4	5
A	4	3	1	5	2
i=1	5	3	1	④	2
i=2	2	3	1	④	⑤
i=3	3	②	1	④	⑤
i=4	1	②	③	④	⑤

۲- چند نکته

فرض کنید رابطهٔ تقسیم و غلبه برای الگوریتم تقسیم و غلبه‌ای بر روی آرایه‌ای n عنصری مثل $A[1...n]$ به‌صورت زیر است.

$$t(n)=t(n_1)+t(n_2)+f(n)$$

که در آن $n_1+n_2=n$

❖ بدترین حالت برای این رابطهٔ تقسیم و غلبه زمانی است که n_1 یا n_2 حداقل ممکن باشد. برای مثال در الگوریتم مرتب‌سازی سریع که این رابطه به صورت

$$t(n)=t(n_1)+t(n_2)+n+1$$

است. بدترین حالت زمانی است که $n_1=1$ و $n_2=n-1$ و آن زمانی است که آرایه‌ای صعودی (نزولی) را بخواهیم به صورت صعودی (نزولی) مرتب کنیم:

$$t(n)=t(n-1)+n+1$$

که اگر با روش جایگذاری با تکرار آن را حل کنیم خواهیم داشت $t(n)=O(n^2)$

❖ بهترین حالت زمانی اتفاق می‌افتد که $n_1=n_2=\frac{n}{2}$. برای مثال در الگوریتم مرتب‌سازی سریع اگر برای سهولت فرض کنیم $n=2^m$ داریم

$$t(n)=2t\left(\frac{n}{2}\right)+n+1$$

که اگر از قضیه اصلی استفاده شود، نتیجه به صورت $t(n)=\theta(n\log n)$ است.

با این توضیحات تکمیلی به حل مسأله برمی‌گردیم.

❖ مرتبهٔ زمانی الگوریتم مرتب سازی درجی برای مرتب کردن $2\sqrt{n}+1$ عنصر برابر $O(n)$ است. اگر عنصر میانهٔ این تعداد عناصر را به‌عنوان عنصر افراز قرار دهیم بعد از فراخوانی الگوریتم Partition حداقل $\sqrt{n}$ عنصر قبل از عنصر افراز و حداقل $\sqrt{n}$ عنصر بعد از عنصر افراز قرار می‌گیرند. بنابراین اگر حداقل $\sqrt{n}$ عنصر در یک طرف عنصر افراز باشد، در طرف دیگر آن، $n-\sqrt{n}$ عنصر وجود خواهد داشت، لذا رابطهٔ تقسیم و غلبه به صورت زیر خواهد بود.

$$t(n)=t(\sqrt{n})+t(n-\sqrt{n})+O(n)$$

که در آن $O(n)$ زمان اجرای مرتب‌سازی درجی برای مرتب کردن $2\sqrt{n}+1$ عنصر اول آرایه است.

۳- جواب این سوال در سوال ۲ مهندسی کامپیوتر سال ۱۳۸۴ داده شده است.

۴- $t(n)=8t\left(\frac{n}{9}\right)+n\log n$

با فرض آنکه $f(n)=n\log n$ داریم $n<f(n)<n^2$

$$a=8 \qquad b=9 \qquad k=1+\varepsilon \qquad 0<\varepsilon<1$$

$$\left(b^k=9^{1+\varepsilon}\right)>a=8$$

در نتیجه $\boxed{t(n)=\theta(n\log n)}$

۵- $\begin{cases} f(n)=bn^2+nf(n-1) & n>1 \\ f(1)=a \end{cases}$

برای سهولت اگر فرض کنیم $b=1$ و از روش جایگذاری با تکرار استفاده کنیم داریم:

$$f(n)=n^2+nf(n-1)=n^2+n\left[(n-1)^2+(n-1)f(n-2)\right]=$$

$$n^2+n(n-1)^2+n(n-1)f(n-2)=n^2+n(n-1)^2+n(n-1)\left[(n-2)^2+(n-2)f(n-3)\right]$$

$$=n^2+n(n-1)^2+n(n-1)(n-2)^2+n(n-1)(n-2)f(n-3)$$

$$=n^2+n(n-1)^2+n(n-1)(n-2)^2+\ldots+an!=O(n!)$$

۶- جواب این سوال در سوال ۱ فنآوری اطلاعات سال ۱۳۸۴داده شده است.

۱- حداکثر زمان اتمام همهٔ کارها برابر با مجموع زمان پردازش کارها در پردازنده‌ای است که آخرین کار را اجرا کند. برای مثال اگر آخرین کار در پردازندهٔ $1 \le k \le m$ اجرا شود، زمان اتمام کارها برابر $p_k + p_{k+m} + p_{k+2m} + \ldots + p_{k+rm}$ است که در آن $r = \left[\frac{n}{m}\right]$ است. برای مثال اگر $m=3 , n=14$ آن‌گاه چون آخرین کار، یعنی کار چهاردهم در پردازندهٔ ۲ اجرا می‌شود. لذا زمان اجرا برای همهٔ کارها برابر $p_2 + p_5 + p_8 + p_{11} + p_{14} = 4+8+15+22+35 = 74$ خواهد بود.

	پردازندهٔ ۳	پردازندهٔ ۲	پردازندهٔ ۱
کارهای 3,2,1	6	4	2
کارهای 6,5,4	9	8	7
کارهای 9,8,7	20	15	12
کارهای 12,11,10	25	22	21
کارهای 14,13		35	30

اما در این مثال

$$p_2 = 4 \le \frac{p_2 + p_3 + p_4}{3} = \frac{17}{3}$$

$$p_5 = 8 \le \frac{p_5 + p_6 + p_7}{3} = \frac{29}{3}$$

$$p_8 = 15 \le \frac{p_8 + p_9 + p_7}{3} = \frac{56}{3}$$

$$p_{11} = 20 \le \frac{p_{11} + p_{12} + p_{13}}{3} = \frac{77}{3}$$

لذا $p_2 + p_5 + p_8 + p_{11} \le \frac{p_2 + p_3 + \ldots p_{13}}{3 = m}$

چون $m=3$ داریم: $\le \frac{p_1 + p_2 + \ldots p_{13}}{3} = \frac{A - p_{14}}{m}$

و لذا زمان اجرای کارها برابر $\frac{A - p_n}{m} + p_n$ است.

با تعمیم مساله برای n و m در حالت کلی، داریم:

$$\begin{cases} p_k \le \dfrac{p_k + p_{k+1} + \dots + p_{k+m-1}}{m} \\ p_{k+m} \le \dfrac{p_{k+m} + \dots + p_{k+2m-1}}{m} \\ \vdots \\ p_{k+(r-1)m} \le \dfrac{p_{k+(r-1)m} + \dots + p_{k+rm-1}}{m} \end{cases}$$

با جمعَ طرفین نامساوی‌ها

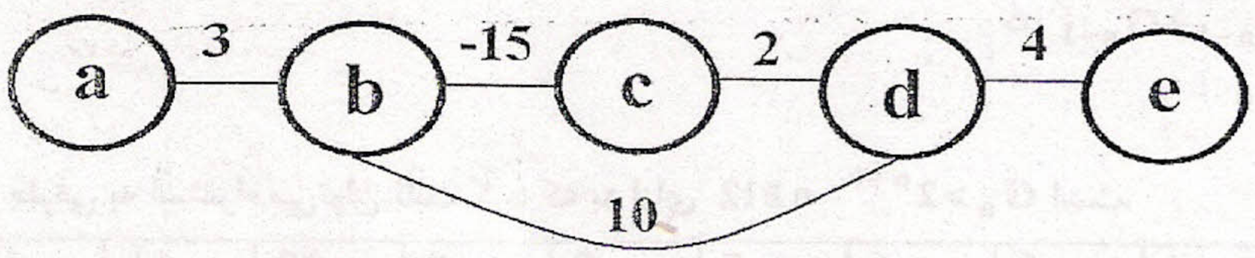
$$p_k + p_{k+m} + \dots + p_{k+(r-1)m} \le \frac{p_k + p_{k+1} \dots + p_{n-1}}{m} \le \frac{A - p_n}{m}$$

که اگر زمان اجرای کار n ام را نیز در نظر بگیریم داریم

$$p_k + p_{k+m} + \dots + p_{k+(r-1)m} + p_{k+rm} \le \frac{A - p_n}{m} + p_n$$

۲- چند نکته

❖ در یک گراف وزن‌دار اگر دوری با هزینهٔ منفی داشته باشیم هیچ الگوریتمی قادر به ارائهٔ کوتاه‌ترین مسیر بین دو رأس نخواهد بود. برای مثال برای گراف زیر :



کوتاه‌ترین مسیر بین a و e به‌صورت

a,b,c,d,b,c,d,b,c,d,....,e

است که هیچ الگوریتمی قادر به پیدا کردن آن نیست، زیرا یکی از ویژگی‌های الگوریتم خاتمه پذیر بودن آن است.

❖ با وجود این، فرض کنید گرافی داریم که یال‌های با وزن‌های منفی دارد، ولی دور با هزینهٔ منفی ندارد. در این صورت:

❖ الگوریتم حریصانهٔ دایکسترا برای چنین گرافی جواب درستی نخواهد داد.

❖ الگوریتم پویای فلوید ـ وارشال برای چنین گرافی جواب درست می‌دهد.

❖ الگوریتم پویای بلمن ـ فورد برای چنین گرافی جوابی درست می‌دهد.

❖ هر چند هر الگوریتم از جمله الگوریتم فلوید نمی‌تواند کوتاه‌ترین مسیر را برای گرافی که دور منفی دارد حل کند با وجود این فرض کنید که گرافی داریم که دور با وزن منفی هم دارد. چون الگوریتم فلوید ـ وارشال n مرحله دارد، بعد از اتمام آن، حداقل یکی از عناصر قطر اصلی ماتریسِ نتیجه منفی خواهد بود. بنابراین با این الگوریتم می‌توان وجود یا عدم وجود دور با هزینهٔ منفی را در یک گراف تشخیص دارد.

۳- مشابه سوال ۱ مهندسی کامپیوتر ۱۳۸۵ است.

❖ ۴- مرتبهٔ زمانی الگوریتم مرتب‌سازی درجی در بهترین حالت $O(n)$، در بدترین حالت $O(n^2)$ و در حالت میانگین هم $O(n^2)$ است. بهترین حالت زمانی اتفاق می‌افتد که آرایه‌ای را که می‌خواهیم به صورت نزولی مرتب کنیم، نزولی مرتب شده باشد و بدترین حالت هم زمانی است که آرایه به صورت نزولی مرتب شده باشد.

❖ برای الگوریتم مرتب‌سازی سریع، مرتبهٔ زمانی الگوریتم در بدترین حالت $O(n^2)$ است و این زمانی اتفاق می‌افتد که آرایه به صورت نزولی مرتب باشد و بخواهیم آن را به‌صورت نزولی مرتب کنیم.

بنابراین با الگوریتم داده شده، زمانی که عنصر افراز $Pivot(T,p,k)$ در مکان k ام قرار گرفت، بهترین حالت زمان اجرا برای مرتب‌سازی زیرآرایه‌های چپ و راست k، یعنی زیرآرایه‌های $T[k+1..n], T[1..k-1]$ وقتی است که این زیرآرایه‌ها به صورت نزولی مرتب باشند و این پدیده هم زمانی اتفاق می‌افتد که آرایهٔ اولیه خود به‌صورت نزولی مرتب باشد، زیرا در این حالت پردازهٔ $Pivot(T,T[1],k)$ با صرف زمانی $k=1, O(n)$ را به عنوان خروجی برمی‌گرداند $Insertion\ sort(T[1,0])$ کاری انجام نمی‌دهد و $Insertionsort(2,n)$ نیز با صرف زمان $O(n)$، عناصر $T[2,n]$ را به‌صورت نزولی مرتب می‌کند. پس اگر آرایهٔ اولیهٔ به صورت نزولی مرتب شده باشد، مرتبهٔ زمانی الگوریتم New Quick Sort برابر $O(n)$ است.

۵- مساله مشابه مسالهٔ ۷ سال ۱۳۸۵ هوش است.

۶-

$$\begin{cases} G_n = G_{n-1} + 2G_{n-2} + G_{n-3} & n \geq 3 \\ G_0 = 1,\ G_1 = 2,\ G_2 = 4 \end{cases}$$

بدیهی است که این دنباله یک دنبالهٔ صعودی است:

$$G_0 < G_2 < G_3 < \ldots$$

لذا

$$G_n = G_{n-1} + 2G_{n-2} + G_{n-3} \leq G_{n-1} + 2G_{n-1} + G_{n-1} \Rightarrow$$

$$G_n \leq 4G_{n-1} \Rightarrow G_n \leq 4^n G_0 \Rightarrow G_n \leq 4^n$$

از طرفی به استقراء می‌توان ثابت کرد که به ازای $n \geq 12$ $G_n > 2^{n+1}$ است.

n	0	1	2	3	4	5	6	7	8	9	10	11	12
G_n	1	2	4	9	19	41	88	189	406	827	1873	3933	8506

پایهٔ استقراء :

$$G_{12} = 8506 > 2^{13} = 8192$$

حال فرض کنید (فرض استقراء) که $\forall n \geq 12$ داریم:

$$G_{n-1} > 2^n$$

می‌خواهیم ثابت کنیم که $G_n > 2^{n+1}$

$$G_n = G_{n-1} + 2G_{n-2} + G_{n-3} > 2^n + 2\times 2^{n-1} + 2^{n-2} > 2^n + 2^n = 2^{n+1}$$

۷- زمانی که در یک گراف همبند مثل $G=(V,E)$ هزینهٔ همهٔ یال‌ها با هم مساوی و مثلاً برابر 1 باشد، مسالهٔ پیدا کردن درخت پوشای مینیمم معادل با پیدا کردن یک درخت پوشاست که این کار را می‌توان با یکی از الگوریتم‌های DFS یا BFS انجام داد. که مرتبهٔ زمانی هر کدام از آن‌ها بسته به این‌که برای نمایش گراف از لیست همسایگی یا ماتریس هزینه‌ها استفاده شود به ترتیب برابر $O(|E|)$ یا $O(n^2)$ است (کتاب طراحی و تحلیل الگوریتم‌ها صفحات ۲۶۵ تا ۲۶۹)

۱- نکاتی در مورد الگوریتم‌های پریم و کراسکال برای به‌دست آورده درخت پوشای مینیمم برای گراف همبند $D=(V,E)$ که در آن $e=|E|, n=|V|$

❖ مرتبهٔ زمانی الگوریتم پریم $O(n^2)$ است (کتاب طراحی و تحلیل الگوریتم‌ها قلی‌زاده صفحه ۱۶۶)

❖ مرتبهٔ زمانی الگوریتم کراسکال $O(e\log e)$ یا $O(n^2\log n)$ است (کتاب طراحی و تحلیل الگوریتم‌ها قلی‌زاده صفحه ۱۷۱)

پس مرتبهٔ زمانی الگوریتم پریم کمتر از کراسکال است.

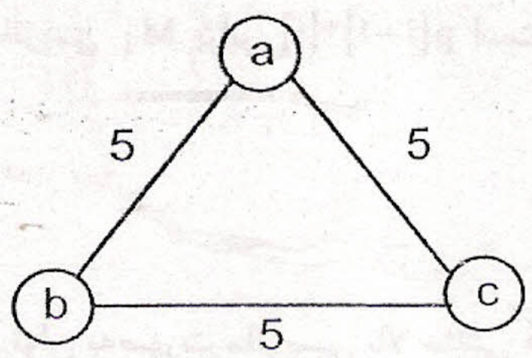
❖ در الگوریتم پریم چون از یک راس دلخواه شروع و در هر مرحله یالی به یکی از رئوس قبلاً انتخاب شده وصل می‌کنیم، همبندی درخت رعایت می‌شود.

❖ در الگوریتم کراسکال، چون یال‌ها را به صورت صعودی هزینه‌شان مرتب کرده سپس آن‌ها را به ترتیب و تنها با این شرط انتخاب می‌کنیم که با مجموعه یال‌های قبلاً انتخاب شده تشکیل دور ندهد، لذا امکان دارد همبندی درخت در هر مرحله رعایت نشود. همبندی در مرحلهٔ آخر حتماً رعایت می‌شود. (برای توضیحات بیشتر به کتاب مراجعه کنید.

اگر هزینهٔ یال‌ها متمایز باشد درخت پوشای حاصل از هر کدام از الگوریتم‌های کراسکال و پریم منحصر به فرد است.

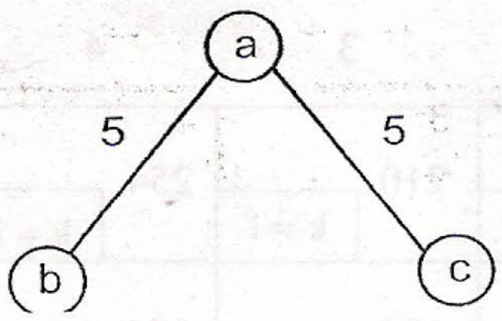
❖ هزینهٔ درخت پوشای مینیمم در هر صورت منحصر به فرد است.

اگر هزینهٔ یال‌ها متمایز نباشد درخت پوشای مینیمم حاصل از هر کدام از این دو الگوریتم می‌تواند منحصر به فرد نباشد حتی درخت پوشای حاصل از یکی از این الگوریتم‌ها، برای مثال الگوریتم پریم بسته به این‌که از کدام راس شروع کنیم می‌تواند متفاوت باشد. برای مثال سادهٔ زیر

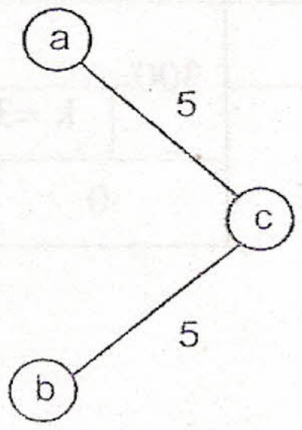


سه درخت پوشای مینیمم وجود دارد.

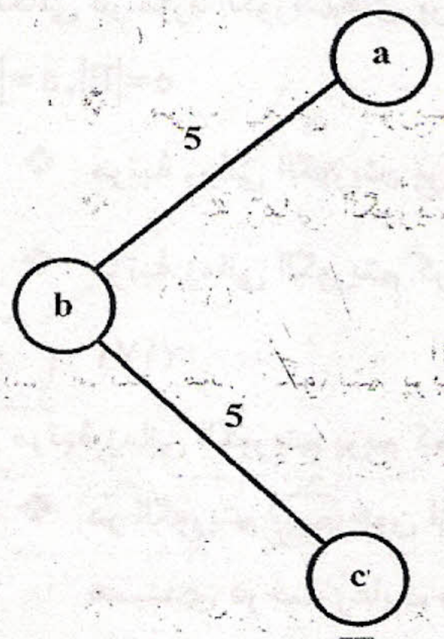
اگر برای مثال در الگوریتم پریم از راس a شروع کنیم نیتجه به‌صورت زیر است:



اگر از راس b شروع کنیم داریم :



و اگر از راس c شروع کنیم درخت پوشای مینیمم به صورت زیر است:



که در هر سه حالت، هزینه درخت پوشای مینیمم 10 است.

۲- هر الگوریتم بازگشتی الزاماً الگوریتمی از بالا به پایین نیست و ممکن است نیاز به حل زیر مساله‌ای نباشد برای مثال در الگوریتم جستجوی دودویی، زیر مسالهٔ مربوط به یکی از نیم آرایه‌ها حل نمی‌شود.

۳- حل مساله به روش پویا. برای ضرب بهینهٔ زنجیر ماتریس $M_1 M_2 ... M_n$ داریم:

شرط مرزی

$$\begin{cases} \{m[i,i]=0 \\ m[i,j]=\min_{i\le k<j}\{m[i,l]+m[k+1,j]+P[i-1]*P[k]*P[j]\} \\ m[1,n] \text{ جواب مساله } \Rightarrow m[1,4] \end{cases}$$

اندازهٔ ماتریس M_i برابر $p[i-1]*[i]$ است.که در آن $P[0..n]$ آرایه‌ای $n+1$ عنصری است. برای این مثال آرایهٔ p به‌صورت زیر است:

	0	1	2	3	4
P	10	2	25	3	4

m را می‌توان به‌صورت ماتریسی بالا مثلثی در نظر گرفت که عناصر روی قطر آن با توجه به شرط مرزی، صفر هستند. این ماتریس را به صورت قطری و با شروع از قطر اصلی پر می‌کنیم:

$j-i=1$ (عناصر بالای قطر اصلی)

i \ J	1	2	3	4
1	0	500 (k=1)	210 (k=1)	254 (k=1)
2		0	150 (k=2)	174 (k=3)
3			0	300 (k=3)
4				0

$$m[i,i+1]=m[i,i]+m[i+1,i+1]+P[i-1]P[i]P[i+1]=P[i-1]P[i]P[i+1]$$

۱- نکاتی در مورد الگوریتم‌های پریم و کراسکال برای به‌دست آورده درخت پوشای مینیمم برای گراف همبند $D=(V,E)$ که در آن $e=|E|, n=|V|$

- مرتبهٔ زمانی الگوریتم پریم $O(n^2)$ است (کتاب طراحی و تحلیل الگوریتم‌ها قلی‌زاده صفحه ۱۶۶)
- مرتبهٔ زمانی الگوریتم کراسکال $O(e\log e)$ یا $O(n^2\log n)$ است (کتاب طراحی و تحلیل الگوریتم‌ها قلی‌زاده صفحه ۱۷۱)

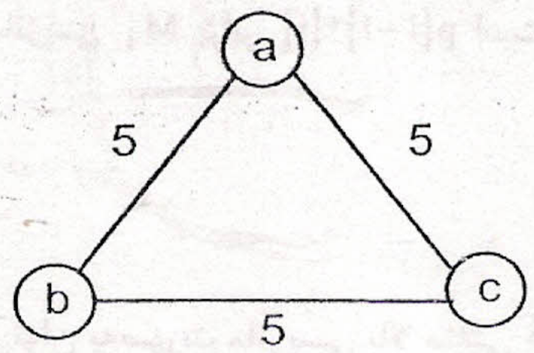
پس مرتبهٔ زمانی الگوریتم پریم کمتر از کراسکال است.

- در الگوریتم پریم چون از یک راس دلخواه شروع و در هر مرحله یالی به یکی از رئوس قبلاً انتخاب شده وصل می‌کنیم، همبندی درخت رعایت می‌شود.
- در الگوریتم کراسکال، چون یال‌ها را به صورت صعودی هزینه‌شان مرتب کرده سپس آن‌ها را به ترتیب و تنها با این شرط انتخاب می‌کنیم که با مجموعه یال‌های قبلاً انتخاب شده تشکیل دور ندهد، لذا امکان دارد همبندی درخت در هر مرحله رعایت نشود. همبندی در مرحلهٔ آخر حتماً رعایت می‌شود. (برای توضیحات بیشتر به کتاب مراجعه کنید.

اگر هزینهٔ یال‌ها متمایز باشد درخت پوشای حاصل از هر کدام از الگوریتم‌های کراسکال و پریم منحصر به فرد است.

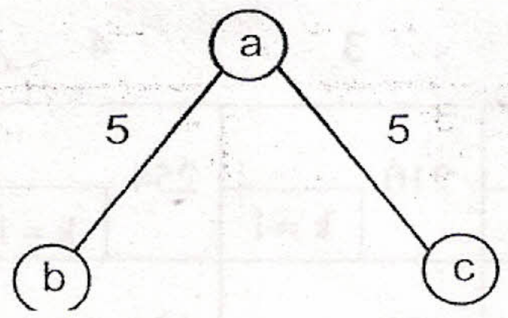
- هزینهٔ درخت پوشای مینیمم در هر صورت منحصر به فرد است.

اگر هزینهٔ یال‌ها متمایز نباشد درخت پوشای مینیمم حاصل از هر کدام از این دو الگوریتم می‌تواند منحصر به فرد نباشد حتی درخت پوشای حاصل از یکی از این الگوریتم‌ها، برای مثال الگوریتم پریم بسته به این‌که از کدام راس شروع کنیم می‌تواند متفاوت باشد. برای مثال سادهٔ زیر

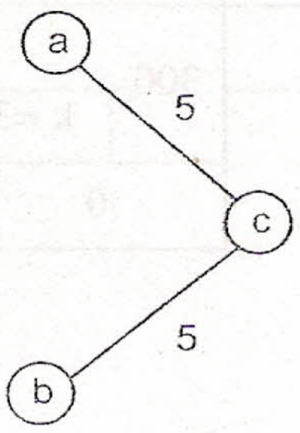


سه درخت پوشای مینیمم وجود دارد.

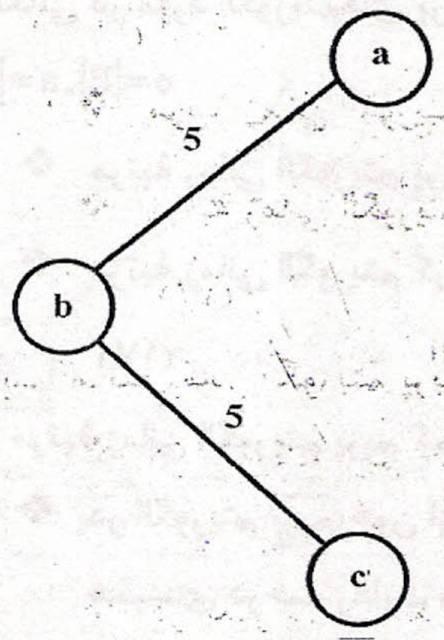
اگر برای مثال در الگوریتم پریم از راس a شروع کنیم نیتجه به‌صورت زیر است:



اگر از راس b شروع کنیم داریم :



و اگر از راس c شروع کنیم درخت پوشای مینیمم به صورت زیر است:



که در هر سه حالت، هزینه درخت پوشای مینیمم 10 است.

۲- هر الگوریتم بازگشتی الزاماً الگوریتمی از بالا به پایین نیست و ممکن است نیاز به حل زیر مساله‌ای نباشد برای مثال در الگوریتم جستجوی دودویی، زیر مسالهٔ مربوط به یکی از نیم آرایه‌ها حل نمی‌شود.

۳- حل مساله به روش پویا. برای ضرب بهینهٔ زنجیر ماتریس $M_1 M_2 ... M_n$ داریم:

شرط مرزی

$$\begin{cases} \{m[i,i]=0 \\ m[i,j]=\min_{i\le k<j}\{m[i,l]+m[k+1,j]+P[i-1]*P[k]*P[j]\} \\ m[1,n] \text{ جواب مساله } \Rightarrow m[1,4] \end{cases}$$

اندازهٔ ماتریس M_i برابر $p[i-1]*[i]$ است.که در آن $P[0..n]$ آرایه‌ای $n+1$ عنصری است. برای این مثال آرایهٔ p به‌صورت زیر است:

	0	1	2	3	4
P	10	2	25	3	4

m را می‌توان به‌صورت ماتریسی بالا مثلثی در نظر گرفت که عناصر روی قطر آن با توجه به شرط مرزی، صفر هستند. این ماتریس را به صورت قطری و با شروع از قطر اصلی پر می‌کنیم:

$j-i=1$ (عناصر بالای قطر اصلی)

i \ J	1	2	3	4
1	0	500 (k=1)	210 (k=1)	254 (k=1)
2		0	150 (k=2)	174 (k=3)
3			0	300 (k=3)
4				0

$$m[i,i+1]=m[i,i]+m[i+1,i+1]+P[i-1]P[i]P[i+1]=P[i-1]P[i]P[i+1]$$

$m[1,2]=10\times2\times25=500,\ k=1$

$m[2,3]=2\times25\times3=150,\ k=2$

$m[3,4]=25\times3\times4=300,\ k=3$

$j-i=2$

$$m[1,3]=\min\left\{\overset{0}{m[1,1]}+\overset{150}{m[2,3]}+\overset{60}{P[0]P[1]p[3]},\ \underset{500}{m[1,2]}+\underset{0}{m[3,3]}+P[0]P[2]P[3]\right\}=210,k=1$$

$$m[2,4]=\min\left\{\cancel{m[2,2]}^{0}+\overset{300}{m[3,4]}+p[1]p[2]p[4],\ \underset{150}{m[2,3]}+\cancel{m[4,4]}^{0}+p[1]p[3]p[4]\right\}=174,\ k=3$$

$j-i=3$

$$m[1,4]=\min\left\{\begin{array}{l}\overset{0}{\cancel{m[1,1]}}+\overset{174}{m[1,4]}+\overset{80}{P[0]P[1]P[4]},\ \overset{500}{m[1,2]}+\overset{300}{m[3,4]}+P[0]P[2]P[4],\\ \underset{210}{m[1,3]}+\underset{120}{m[4,4]}+P[0]P[3]P[4]\end{array}\right\}=\boxed{254}$$

لذا حداقل تعداد ضرب‌های به کار رفته برای ضرب این چهار ماتریس 254 است اما ترتیب ضرب آنها: چون برای $m[1,4]$ مقدار $k=1$ است،

چون $n=4$ (کوچک است) مساله را به روش سعی و خطا می‌شود سریع‌تر از روش پویا حل کرد:

تعداد ضرب‌های لازم برای $(A\times B)\times(C\times D)$ $10\times2\times25+25\times3\times4+10\times25\times4=1800$

تعداد ضرب‌های لازم برای $(A\times(B\times C))\times D$ $2\times25\times3+10\times2\times3+10\times3\times4=330$

تعداد ضربهای لازم برای $\boxed{A\times((B\times C)\times D)\quad 2\times25\times3+2\times3\times4+10\times2\times4=254}$

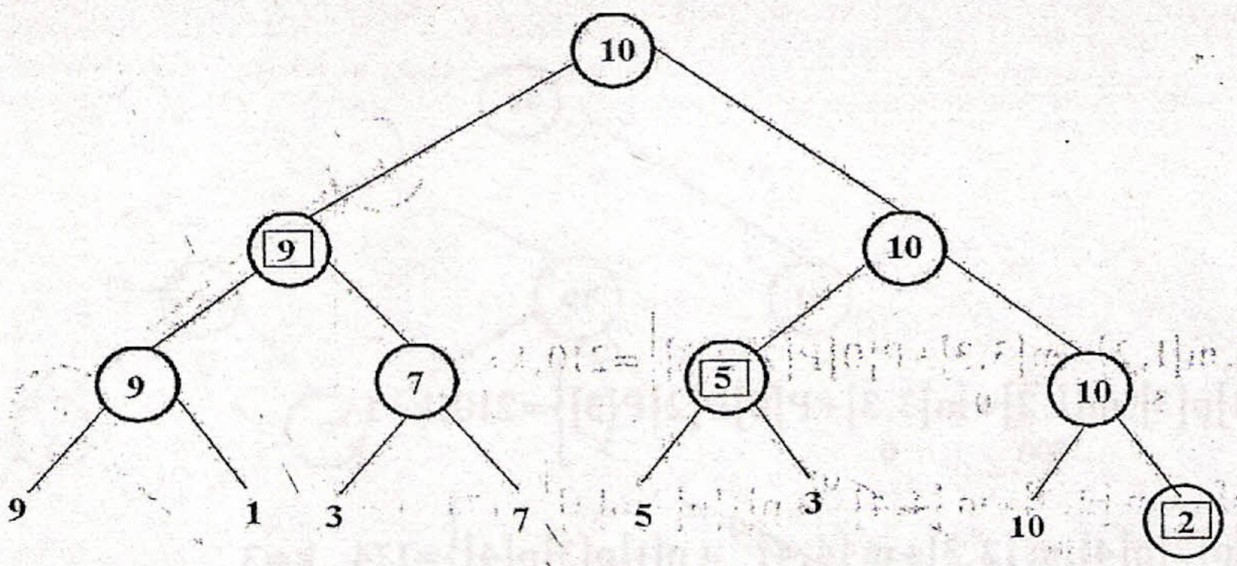
تعداد ضرب‌های لازم برای $((A\times B)\times C)\times D$ $10\times2\times25+10\times25\times3+10\times3\times4=1370$

که مینیمم برای حالت $\boxed{A\times((B\times C)\times D)}$ حاصل می‌شود.

<u>چند نکته</u>

- تعداد حالت‌هایی که $n+1$ ماتریس را می‌شود در هم ضرب کرد برابر عدد کاتالان $\frac{1}{n+1}C_{2n}^{n}$ است که برای حالت $n=3$، $\frac{1}{4}C_{6}^{3}=5$ است.
- $\frac{1}{n+1}C_{2n}^{n}=\Omega\left(\frac{4^n}{n^{\frac{3}{2}}}\right)$ (کتاب طراحی الگوریتم‌های قلی‌زاده صفحهٔ ۲۳۳)
- حل این مساله به روش پویا با استفاده از فرمول بالا مستلزم مرتبهٔ زمانی $O(n^3)$ است. بنابراین برای n های بزرگ نمی‌توان مساله را به روش سعی و خطا حل کرد.

۴- این مساله معادل پیدا کردن برنده‌های اول و دوم یک مسابقهٔ حذفی بین n بازیکن (تیم) است. بدیهی است که برندهٔ اول را می‌شود با $\boxed{n-1}$ مقایسه (مسابقه) تعیین کرد. برندهٔ دوم مسلماً از بین بازیکن‌هایی خواهد بود که به برندهٔ اول باخته‌اند. حداکثر تعداد بازیکن‌هایی که به برندهٔ اول باخته‌اند برابر $\lceil \log_2^n \rceil$ است و پیدا کردن برنده از میان این تعداد بازیکن مستلزم $\lceil \log_2^n \rceil-1$ مقایسه است. در نتیجه در کل به $\boxed{n+\lceil \log_2^n \rceil-2}$ مقایسه (مسابقه) برای تعیین(اولین و) دومین عنصر یک آرایهٔ n عنصری نیاز است. به مثالی در این رابطه توجه کنید: فرض $n=8$



مشاهده می‌شود که بزرگ‌ترین عنصر با 7 مقایسه (1 – 8) پیدا شده است. دومین عدد بزرگ‌تر، بزرگ‌ترین عنصر از میان عناصر 9,5,2 خواهد بود که با 2 مقایسه یعنی $\lceil \log_2 8 \rceil - 1$ مقایسه حاصل می‌شود.

۵- چون $|\sin n| \le 1$ لذا می‌شود آن را برای تعیین نمادها ندیده گرفت. لذا $\boxed{n^2 \sin n = \Omega(n)}$

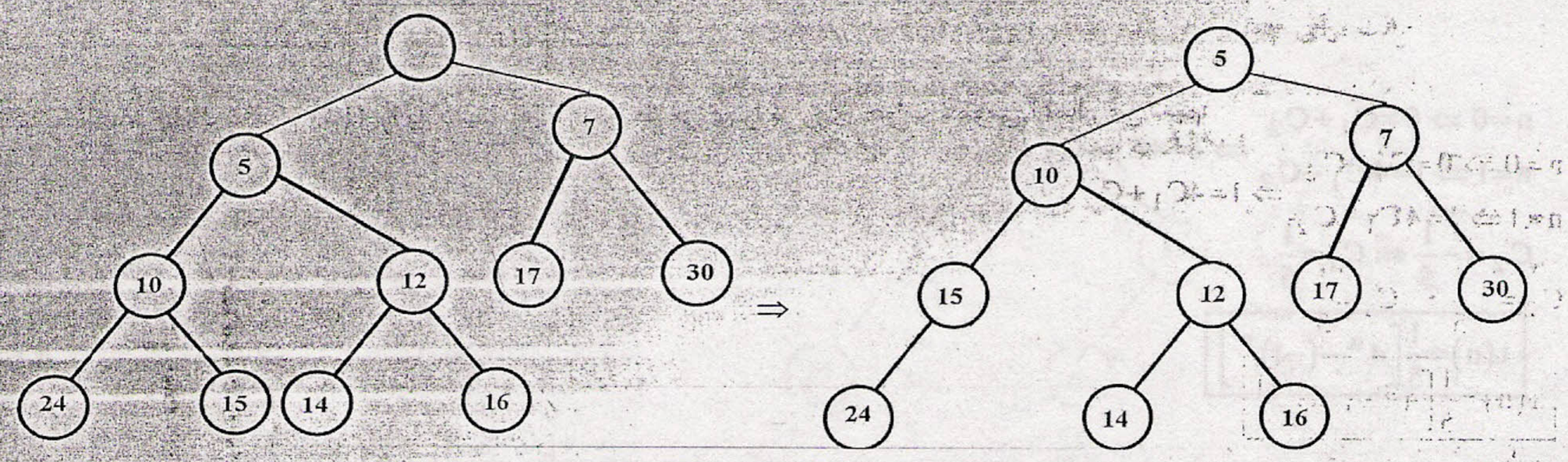
۶-

```
min max (A)
1)      min=1
2)      max=1
3)      for i=2 to n do
4)      if A[i]<A[min] then
5)          min=i
6)          else if A[i]>A[max] then
7)          max=i
```

بدترین حالت زمانی است که دستور6 همیشه اجرا شود و این زمانی اتفاق می‌افتد که دستور 4 همیشه false باشد. این پدیده برای آرایه‌ای صعودی روی می‌دهد، زیرا در این حالت همیشه $A[i] \not< A[min]$ است. بنابراین حداکثر تعداد مقایسه‌های دو عنصر از آرایه $\boxed{2(n-1)}$ است.

بهترین حالت هم زمانی است که آرایه به صورت نزولی مرتب شده باشد که دستور 4 همیشه true است، لذا دستور 6 هرگز اجرا نمی‌شود و تعداد دفعاتی که دستور 6 اجرا می‌شود برابر $\boxed{n-1}$ است.

۷- برای ادغام دو هیپ با m,n عنصر $(n>m)$ نخست گرهی جدید (ریشه) ایجاد کرده و دو هیپ داده شده را زیر درخت‌های چپ و راست آن قرار می‌دهیم. بعد، حداکثر با $\log n$ مقایسه می‌توان عنصر مینیمم دو هیپ را به ریشه انتقال داد. بدیهی است که این دو هیپ ادغام شده را الزاماً نمی‌توان به‌صورت آرایه نمایش داد. به مثالی در این مورد توجه کنید؟



۸- اصل بهینه سازی ممکن است بدیهی به نظر رسد، ولی در عمل قبل از این‌که فرض کنیم جوابی بهینه را می‌توان با استفاده از برنامه‌نویسی پویا به دست آورد لازم است نشان دهیم که اصل بهینه‌سازی برقرار است. مثال بعد نشان می‌دهد که اصل بهینه‌سازی برای مسالهٔ طولانی‌ترین مسیر ساده از یک راس به راس دیگر برقرار نیست. در شکل مقابل بدیهی است که طولانی‌ترین مسیر از a به d مسیر [a,c,b,d] به طول 7 است. (البته اگر منظور گراف وزن‌دار نبوده و تعداد یال‌ها مورد نظر است، این طول برابر ۳ است.) بنابراین در صورت برقرار بودن اصل بهینه‌سازی مسیر از aبه c یعنی مسیر [a,c] که طول آن 1 است(برای حالتی که منظور تعداد یال‌ها باشد، این تعداد نیز برابر 1 است) باید طولانی‌ترین مسیر باشد در صورتی که طولانی‌ترین مسیر ساده از a به c مسیر [a,b,c] به طول3 (با تعداد یال‌های 2) 3 است. بنابراین اصل بهینه‌سازی در مورد طولانی‌ترین مسیر ساده از یک راس به راس دیگر برقرار نیست.این مطلب هم برای گراف‌های جهت‌دار و هم برای گرافهای بدون جهت درست است.

۹- چند نکته در مورد مسألهٔ کوله‌پشتی

- ❖ برای مسألهٔ کوله‌پشتی که در آن هر درصد از جسمی را بتوان انتخاب کرد، بهترین روش، حریصانه است که اگر اجسام به صورت نزولی ارزش بر واحد وزن مرتب شده باشند از مرتبهٔ زمانی $O(n)$ است در غیر این صورت باید اجسام را مرتب کرد که مرتبهٔ زمانی آن $O(n\log n)$ خواهد بود.
- ❖ اگر اجسام به صورت بسته‌بندی شده انتخاب شوند(مسألهٔ کوله‌پشتی 1,0)، در این صورت الگوریتمی از مرتبهٔ زمانی چند جلمه‌ای وجود ندارد مگر این‌که وزن بسته‌ها اعدادی طبیعی باشد که با نگرش ماتریسی به مساله، با روش برنامه‌نویسی پویا (صفحه ۲۰۸ کتاب طراحی الگوریتم‌ها) می‌توان آن را در مرتبهٔ زمانی $\theta(nm)$ حل کرد که در آن، n ،تعداد بسته‌ها و m ،ظرفیت کوله‌پشتی است. در غیر این صورت مرتبهٔ زمانی الگوریتم $O(2^n)$ خواهد بود.
- ❖ با روش پسگرد مساله کوله‌پشتی 1,0 را می‌توان در مرتبهٔ زمانی $O(2^n)$ حل کرد.
- ❖ به روش انشعاب و تحدید نیز حل مساله از مرتبهٔ زمانی $O(2^n)$است.

۱۰-

$$\begin{cases} t(n)=3t(n-1)+4t(n-2) \\ t(0)=0, t(1)=1 \end{cases}$$

این، یک رابطهٔ بازگشتی است که چون همگن است، لذا ساده‌ترین راه برای آن، استفاده از معادلهٔ مشخصه است:

$$r^2-3r-4=0$$

که ریشه‌های آن عبارت‌اند از $r_1, r_2=4,-1$. لذا جواب عمومی به صورت زیر است

$$t(n)=C_1 4^n+C_2(-1)^n$$

که $C_2 \neq C_1$ هر چه باشند $\boxed{t(n)=\theta(4^n)}$ است.

اگر بخواهیم جواب دقیق $t(n)$ را به دست آوریم شرایط مرزی را بر روی جواب عمومی اعمال می‌کنیم:

$$\begin{aligned} n=0 &\Rightarrow 0=C_1+C_2 \\ n=1 &\Rightarrow 1=4C_1-C_2 \end{aligned} \Rightarrow 1=4C_1+C_2$$

$$C_2=-\frac{1}{5} \Leftarrow C_1=\frac{1}{5}$$

$$\boxed{t(n)=\frac{1}{5}\left[4^n-(-1)^n\right]}$$

۱- بدترین حالت برای جستجوی یک عنصر در یک درخت جستجوی دودویی چه برای جستجوی موفق و چه برای جستجوی ناموفق برابر $O(\log n)$ است (کتاب طراحی و تحلیل الگوریتم‌ها قلی‌زاده صفحهٔ ۸۱)

۲- عناصر آرایه از A[1] تا A[5] به صورت محلی مرتب شده‌اند تنها الگوریتمی از میان الگوریتم‌های مزبور که چنین رفتاری را در ۴ مرحله از خود نشان می‌دهد، الگوریتم مرتب سازی درجی است.

۳- می‌دانیم در یک گراف بدون جهت مجموع درجه‌های رئوس دو برابر تعداد یال‌هاست. پس

$3+3+3+3+2+2=16=2|E| \Rightarrow |E|=8$

اما تعداد یال‌های یک درخت پوشا برای گرافی همبند و بدون جهت برابر $n-1$ است که در آن، n برابر تعداد رئوس گراف است. در این مثال، n=6. پس تعداد یال‌های برای هر درخت پوشایی برای گراف این مساله برابر 3 است. پس تعداد یال‌های backedge یا غیردرختی برابر (کتاب ساختمان‌های گسسته قلی‌زاده صفحه ۳۲۳) $|E|-(n-1)=|E|-n+1$

که در این مثال $8-6+1=3$ است.

۴ – چون ریشهٔ این درخت حاوی عدد 6 است، لذا اسم مجموعه هم در آرایهٔ با اندیس ۵ مشخص می‌شود و در این مکان هم باید 1- داشته باشیم. تنها آرایه‌ای که این خاصیت را دارد آرایهٔ زیر است

1	2	3	4	5	6	7	8	9
4	3	9	6	4	−1	3	−1	6

البته در این آرایه باید داشته باشیم (از روی درخت ارائه شده) :

مسیر از گره 5 به ریشه ، $A[6]=-1$, $A[4]=6$, $A[5]=4$

مسیر از گره 1 به ریشه ، $A[1]=4$

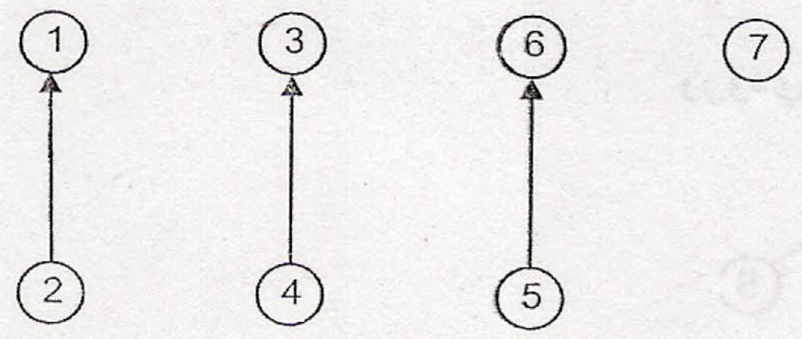
مسیر از گره 2 به ریشه، $A[6]=-1$ $A[9]=6$ $A[3]=9$ $A[2]=3$

مسیر از گره 7 به ریشه ، $A[7]=3$

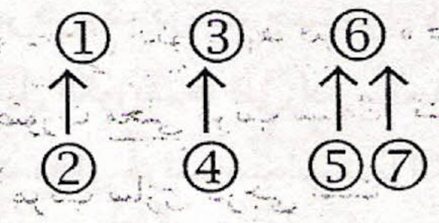
۵- اگر n مجموعهٔ مجزای تک عضوی داشته باشیم و u مرحله عمل اجتماع انجام دهیم، حداقل تعداد، مجموعه‌های تک عضوی باقی مانده زمانی اتفاق می‌افتد که در هر مرحله از u مرحله، عمل اجتماع را بر روی دو مجموعه تک عضوی انجام دهیم، مگر این‌که دیگر مجموعهٔ تک عضوی نداشته باشیم $2u>n$ باشد. اما هر عمل اجتماع بر روی مجموعه‌های تک عضوی، 2 واحد از تعداد آنها می‌کاهد، لذا اِعمال u مرحله عمل اجتماع بر روی مجموعه‌های تک عضوی باقی‌مانده (در صورت وجود) 2u واحد از تعداد آن‌ها می‌کاهد، لذا حداقل تعداد مجموعه‌های تک عضوی باقی‌مانده $\boxed{\max\{n-2u,0\}}$ است. به مثالی در این رابطه توجه کنید. اگر n=7 باشد داریم:

① ② ③ ④ ⑤ ⑥ ⑦

بعد از اعمال ۳ مرحله اجتماع بر روی مجموعه‌های تک عضوی داریم:



اما الآن فقط یک مجموعهٔ تک عضوی داریم که اگر اجتماع آن را با یکی از مجموعه‌های دو عضوی دیگر بگیریم خواهیم داشت:



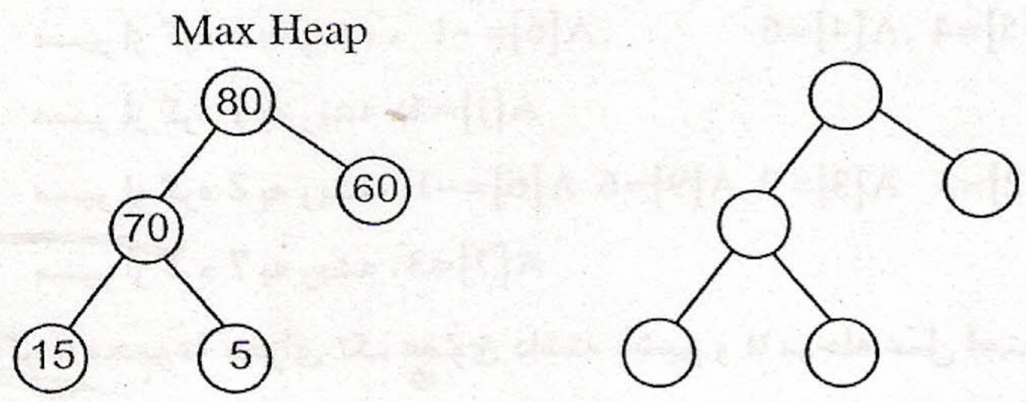
$n=7\ ,\ u=4$

الآن دیگر مجموعه‌ای تک عضوی نداریم. در نتیجه

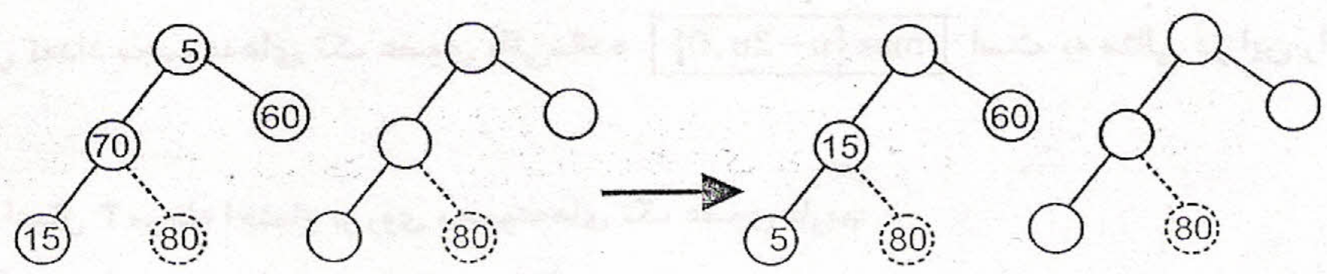
$n-2u=-1 \Rightarrow \max\{-1,0\}=0$

۶- چون در پیمایش یک درخت، همهٔ گره‌های آن باید ملاقات شود، لذا مرتبهٔ زمانی هر الگوریتم پیمایشی از مرتبهٔ تعداد گره‌های آن خواهد بود.

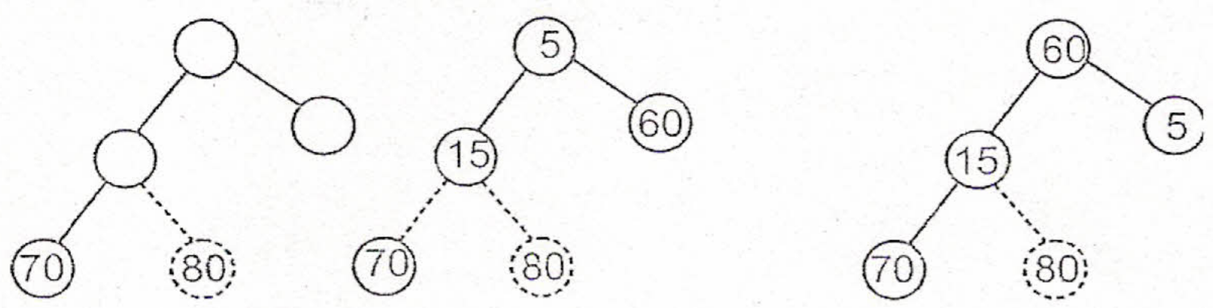
۷- می‌دانیم که عمق یک heap برابر $O(\log n)$ است. برای تبدیل یک max-heap به یک min-heap، $(n-1)$ مرحله (تعداد عناصر heap) خواهیم داشت و روش به صورت زیر است: در مرحلهٔ نخست، عنصر ماکزیمم max-heap یعنی عنصر اول آن را در مکان n ام آرایه‌ای که می‌خواهیم min-heap شود قرار می‌دهیم. سپس عنصر n ام max-haep را در مکان اول (ریشه) آن قرار می‌دهیم. حال زیر درخت‌های چپ و راست درخت حاصل، max- heap هستند و فقط عنصر ریشه است که خاصیت max-heap بودن را نقض کرده است که آن را می‌شود در $O(\log n)$ مقایسه به یک max-heap تبدیل کرد. همین روند را برای n-1 عنصر باقی مانده انجام می‌دهیم. لذا مرتبهٔ زمانی این تبدیل $\boxed{O(n\log n)}$ خواهد بود. به مثالی در این رابطه توجه کنید



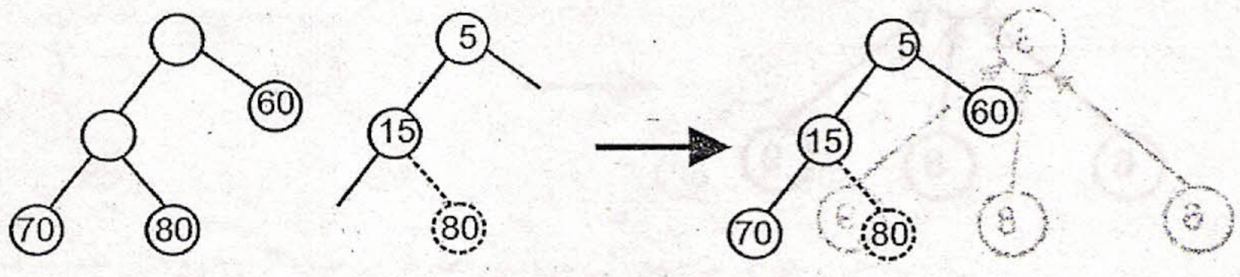
ساختار ورودی ساختار خروجی



ورودي بعد از يك مرحله



خروجی بعد از مرحله ۲ ورودی بعد از مرحله ۲



خروجی بعد از مرحله ۳ ورودی بعد از مرحله ۳ خروجی بعد از مرحله ۴

۸- نشان می‌دهیم n_0 ای وجود دارد که به ازای هر $n \geq n_0$ داریم: $n^3 \log n \leq n^{\varepsilon+3}$

زیرا:

$$n^3 \log n \leq n^{3+\varepsilon} \Leftrightarrow \log n \leq n^{\varepsilon}$$

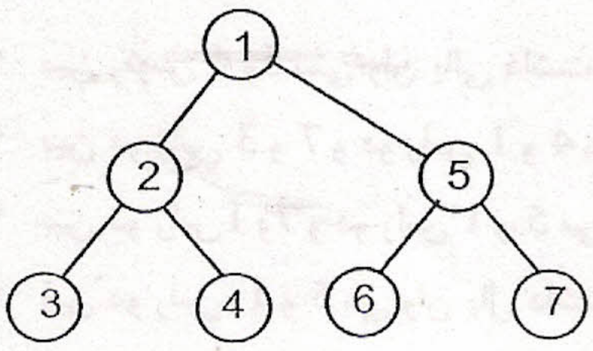
با فرض آن‌که $n=2^m \Leftrightarrow m=\log n$ داریم:

$$\log n \leq n^{\varepsilon} \Leftrightarrow m \leq \left(2^m\right)^{\varepsilon} \Leftrightarrow m \leq 2^{\varepsilon m} \quad (1)$$

ε هر اندازه کوچک باشد می‌توان $m_0=\log n_0 \Leftrightarrow n_0=2^{m_0}$ ای پیدا کرد که به ازای $m \geq m_0$ رابطهٔ (1) برقرار باشد. برای مثال m_0 را می‌توان $\left(\frac{1}{\varepsilon}\right)^2$ انتخاب کرد که در آن $k=\frac{1}{\varepsilon}>1$ لذا $k^2 \leq 2^k$ که برای $k \leq 2$ برقرار است.

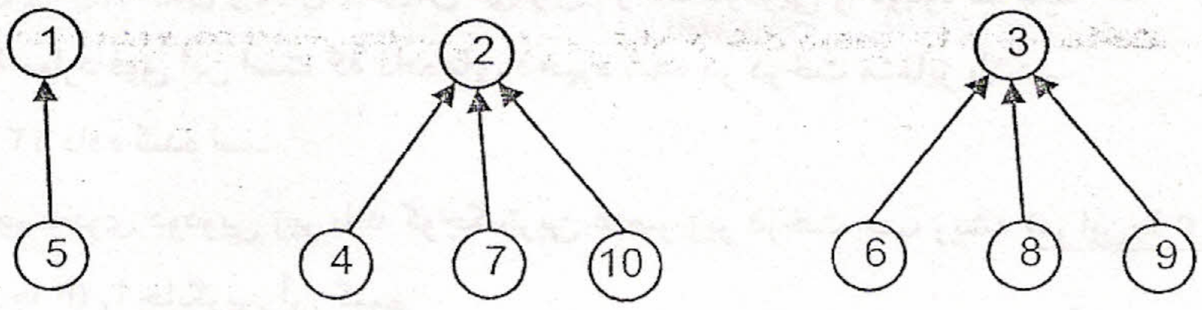
۹- در پیمایش یک گراف غیرجهت‌دار $G=(V,E)$ به روش DFS یال cross – edge (یال صلیبی) وجود ندارد.

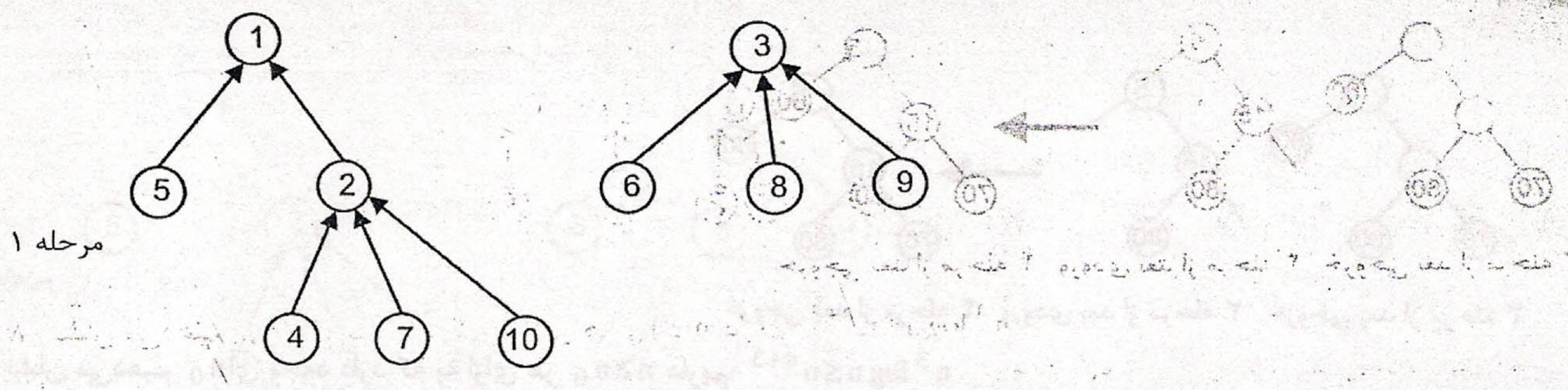
- یالی بین رئوس 2 و 5 نمی‌تواند وجود داشته باشد، چون یک یال صلیبی خواهد بود.
- یال بین 2 و 6 هم یک یال صلیبی خواهد بود.
- یال بین رئوس 6 و 7 هم یک یال صلیبی خواهد بود.



۱۰- اگر عمل اجتماع وزن‌دار را بر روی سه مجموعهٔ $\{3,6,8,9\}$, $\{2,4,7,10\}$, $\{1,5\}$ انجام دهیم، مراحل و نتیجه به صورت زیر خواهد بود:

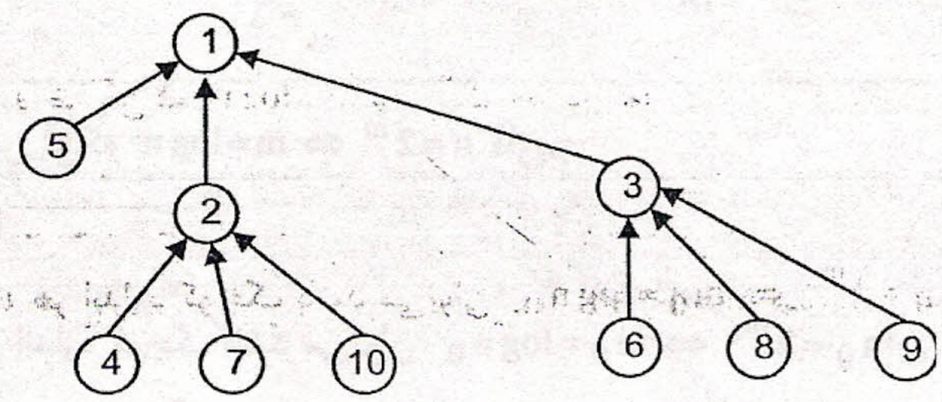
مرحله صفر (ورودی‌ها)



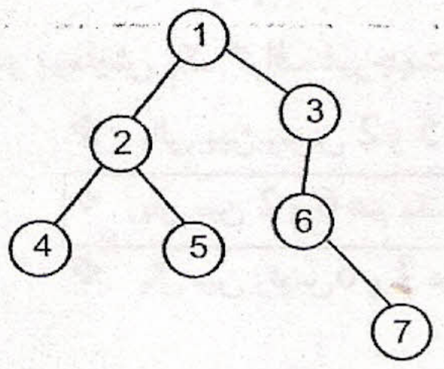


مرحله ۱

حال با توجه به این‌که مجموعهٔ 1 را زیر درخت 3 قرار دهیم یا برعکس گزینهٔ ۳ یا ۴ را خواهیم داشت. برای مثال :



۱۱- این سوال مشابه سوال ۹ است فقط بین دو راسی که یال بین آنها صلیبی است، نمی‌توانیم یالی در گراف داشته باشیم در نتیجه:



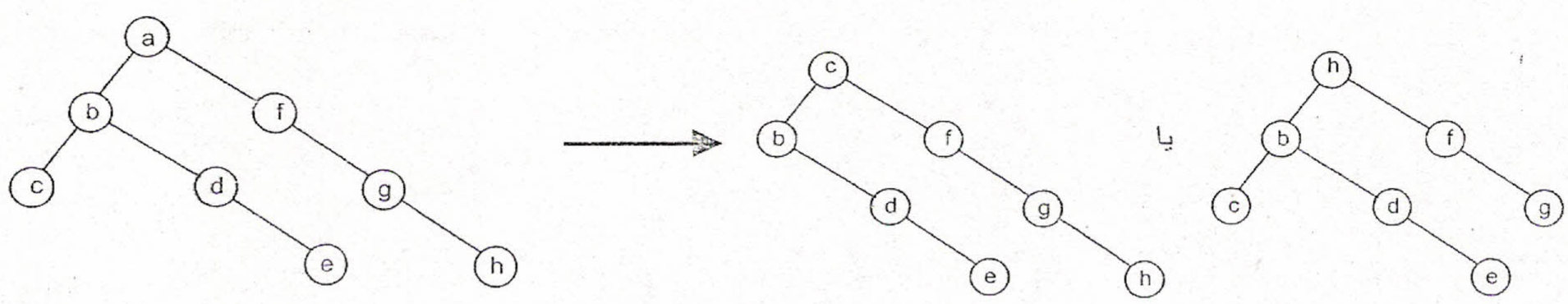
- بین رئوس 4 و5 نمی‌توان یالی داشت.
- بین دو راس 3 و 7 و دو راس 1 و 4 می‌توان یال داشت.
- بین دو راس1 و7 و دو راس 1 و 5 می‌توان یال داشت.
- بین دو راس 1 و 5 می‌توان یال داشت

۱۲- نکته : با داشتن فقط یکی از پیمایش‌های پیش پیمایش، پس پیمایش یا میان پیمایش برای درختی دودویی نمی‌توان درخت را باز ساخت.

- با داشتن پیمایش‌های پیش‌پیمایش و میان پیمایش می‌توان درخت دودویی را دوباره ساخت.
- شرط اصلی برای همهٔ موارد فوق این است که داده‌های ذخیره شده در درخت متمایز باشند.

۱۳- جواب این سوال در سوال ۱۲ داده شده است.

۱۴- با حذف ریشه در درخت جستجوی دودویی زیر باید کوچک‌ترین عنصر زیر درخت چپ ریشه (در این‌جا c) یا بزرگ‌ترین عنصر زیر درخت راست ریشه (در این‌جا h) را جایگزین آن کنیم



۱- فرمول تعداد ضرب‌های عددی به کار رفته در روش استراسن به صورت زیر است $\begin{cases} t(2)=8 \\ t(n)=7\left(\frac{n}{2}\right) \end{cases}$

که اگر به جای $n=8$ قرار دهیم و رابطهٔ تقسیم و غلبه را با روش جایگذاری با تکرار حل کنیم، خواهیم داشت :

$$t(8)=7t(4)=7^2t(2)=7^2\times 8=392$$

۲- در روش پسگرد درخت فضای حالت (ضمنی) به صورت DFS پیمایش می‌شود، لذا از ساختمان داده‌های پشته استفاده می‌شود.

در روش انشعاب و تحدید اگر درخت فضای حالت به روش Breadth First Search پیمایش شود از ساختمان داده‌های صف استفاده می‌شود.

در روش انشعاب و تحدید اگر درخت فضای حالت را به روش Best First Search پیمایش کنیم باید از ساختمان داده‌های صف اولویت (Heap) استفاده کنیم.

۳- این مساله، مسالهٔ زمان‌بندی فعالیت‌ها با مهلت است که مرتبهٔ زمانی آن در بدترین حالت $O(n^2)$ است (کتاب طراحی و تحلیل الگوریتم‌های قلی‌زاده صفحهٔ ۱۹۳).

۴- داریم:

$$w=(7,5,2,3,1) \quad n=5$$

$$P=(35,30,20,12,3) \qquad M=13$$

با استفاده از نگرش مجموعه‌ای مساله را حل می‌کنیم:

$$S^\circ=\{(0,0)\} \quad , \quad S_1^1=\{(7,35)\}$$

$$S^1=\{(0,0),(7,35)\}$$

$$(5,30) \qquad S_1^2=\{(5,30),(12,65)\}$$

$$S^2=\{(0,0),(5,30),(7,35),(12,65)\}$$

$$(2,20) \qquad S_1^3=\{(2,20),(7,50),(9,55)\}$$

$$S^3=\{(0,0),(2,20),(5,30),(7,50),(9,55),(12,65)\}$$

$$(3,12) \qquad S_1^4=\{(3,12),(5,32),(8,42),(10,62),(12,67)\}$$

$$S^4=\{(0,0),(2,20),(5,30),(7,50),(9,55),(12,67)\}$$

$$(1,3) \qquad s_1^5=\{(1,3),(3,23),(6,33),(8,53),(10,58),(13,70)\}$$

$$S^5=\{(0,0),(1,3),(2,20),(3,23),(5,30),(6,33),(7,50),(8,53),(9,55),(10,58),(12,67),(13,70)\}$$

آخرین زوج در مجموعهٔ S^5 $(13,70)$ لذا جواب 70 است و مساله حل شده است. اگر بردار جواب مورد نظر بود می‌شد به صورت زیر عمل کرد:

$$(13,70)\in S^5,(13,70)\notin S^4 \Rightarrow \boxed{x_5=1} \Rightarrow (13,70)-(1,3)=(12,67)$$

$$(12,67)\in S^4,(12,67)\notin S^3 \Rightarrow \boxed{x_4=1} \Rightarrow (12,67)-(3,12)=(9,55)$$

$$(9,55)\in S^3,(9,55)\notin S^2 \Rightarrow \boxed{x_3=1} \Rightarrow (9,55)-(2,20)=(7,35)$$

$$(7,35)\in S^2,(7,35)\in S^1 \Rightarrow \boxed{x_3=0}$$

$$(7,35)\in S^1,(7,35)\notin S^0 \Rightarrow \boxed{x_1=1}$$

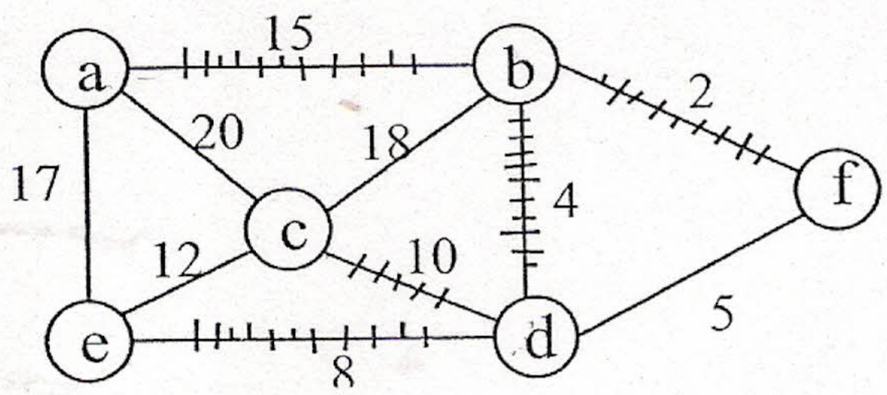
۵- رابطهٔ تقسیم و غلبه به صورت زیر خواهد بود:

$$\begin{cases} t(n)=2t\left(\frac{n}{2}\right)+cn & , \quad n\geq 20 \\ t(n)=cte & \quad n<20 \end{cases}$$

که با توجه به قضیه اصلی و این‌که a=2 , b=2 , k=1 جواب به صورت زیر است.

$$t(n)=O(n\log n)$$

۶- در الگوریتم کراسکال ترتیب انتخاب یال‌ها به صورت زیر است:



(b,f)

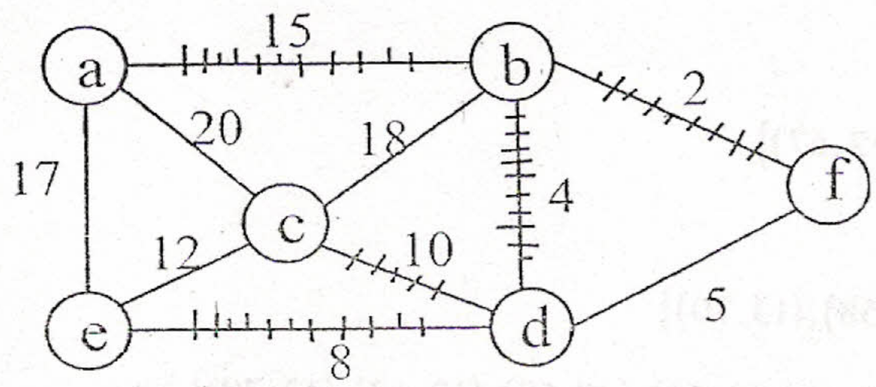
(b,d)

(e,d)

(c,d)

(a,b)

۷- ترتیب انتخاب یال‌ها در الگوریتم پریم با شروع از راس a به ترتیب زیر است:



(a,b)

(b,f)

(b,d)

(d,e)

(d,c)

۸- سخنرانی‌ها را به ترتیب صعودی زمان خاتمه آن‌ها مرتب می‌کنیم. فعالیت اول انتخاب می‌شود. سپس بقیهٔ فعالیت‌ها را به ترتیب بررسی می‌کنیم. اگر فعالیتِ در حال بررسی با آخرین فعالیت انتخاب شده سازگار باشد آن را انتخاب می‌کنیم.

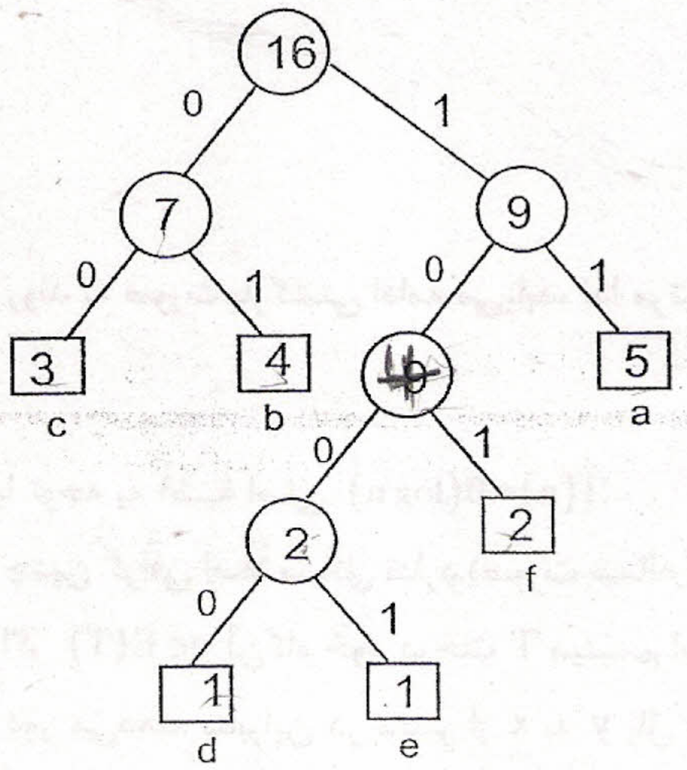
شمارهٔ فعالیت	زمان شروع	زمان خاتمه	
2	1	2	✓
3	3	4	✓
4	3	5	–
5	4	7	✓
6	6	8	–
1	2	9	–

فعالیت‌های انتخاب شده عبارتند از فعالیت‌های شمارهٔ 2 , 3 و 5 که در کل، 3 فعالیت انتخاب می‌شود.

۹- درخت کشف رمز می‌تواند به صورت مقابل باشد که wepl آن به صورت زیر است:

abcabbaccaabdffe

حرف	تعداد دفعات تکرار
a	5
b	4
c	3
d	1
e	1
f	2

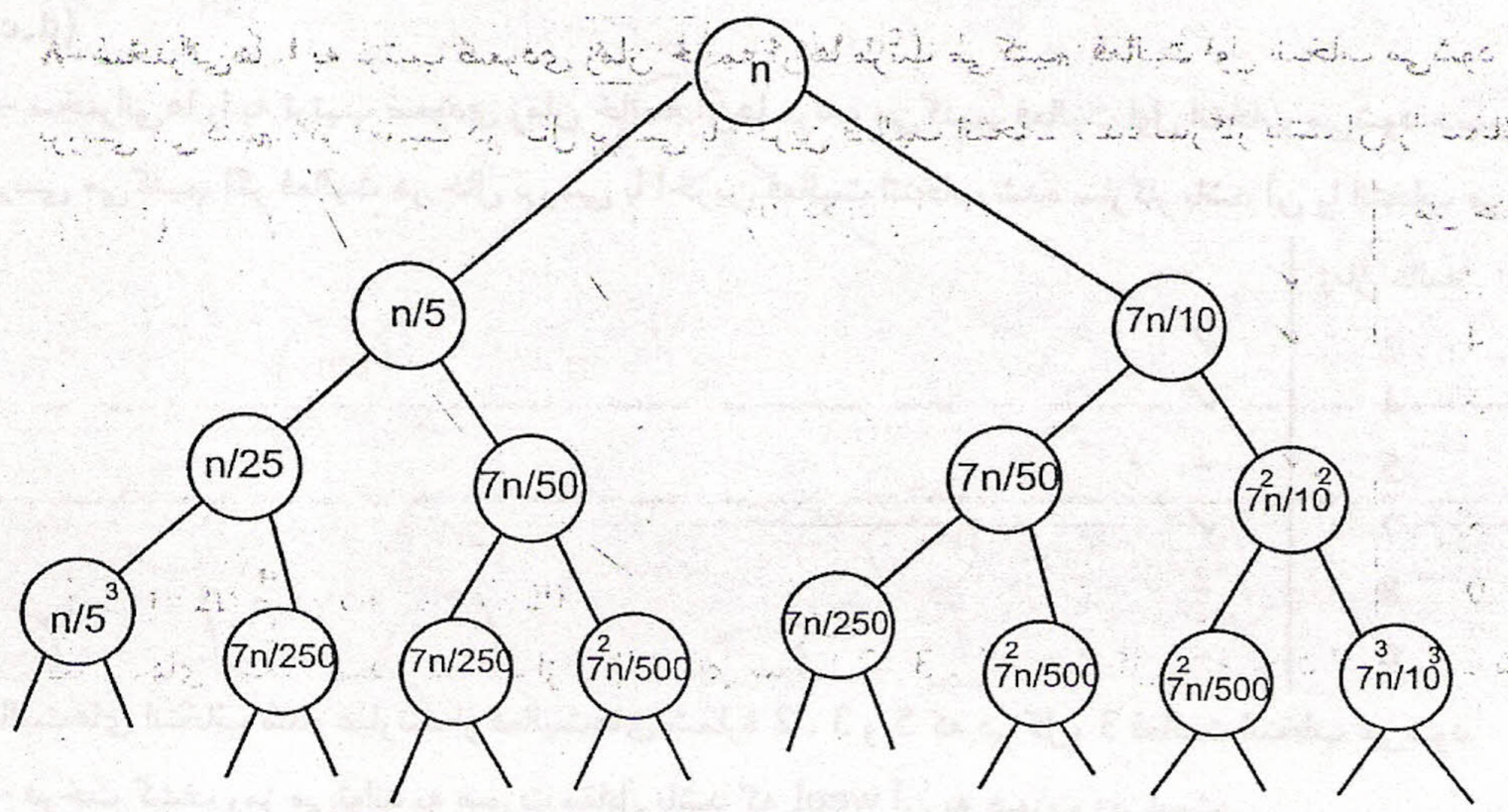


$3\times2+4\times2+1\times4+1\times4+2\times3+5\times2=6+8+4+4+6+10=38$

۱۰- از الگوریتم Selection استفاده می‌کنیم. همان‌گونه که قبلاً ذکر شد مرتبهٔ زمانی این الگوریتم برای پیدا کردن k امین عنصر کوچک‌تر (بزرگ‌تر) در بهترین حالت $O(n)$، در بدترین حالت $O(n^2)$ و در حالت متوسط نیز $O(n)$ است. (کتاب طراحی و تحلیل الگوریتم‌ها قلی‌زاده صفحهٔ ۱۱۲)

۱-

$$t(n)\le t\left(\frac{n}{5}\right)+t\left(\frac{7n}{10}\right)+n$$



$$\frac{n}{125}+\frac{7n}{500}+\frac{7n}{250}+\frac{49n}{500}+\frac{7n}{250}+\frac{49}{500}n+\frac{49n}{500}+\frac{7^3n}{10^3}=\frac{9^3}{10^3}n$$

$$t(n)\le\sum_{i=0}^{\log_{\frac{10}{7}}^{n}}\left(\frac{9}{10}\right)^i n$$

حداکثر ارتفاع درخت $\log_{\frac{10}{7}}^{n}$ است لذا

۲- فرض $x_m=x\left[\frac{n}{2}\right]$ و $y_m=y\left[\frac{n}{2}\right]$

if $x_m=y_m \Rightarrow$ میانه پیدا شده است

if $x_m<y_m \Rightarrow$ میانه باید در $y\left[1...\frac{n}{2}-1\right],X\left[\frac{n}{2}+1..n\right]$

if $x_m>y_m \Rightarrow$ جستجو $x\left[1..\frac{n}{2}-1\right],y\left[\frac{n}{2}+1..n\right]$ میانه باید در

این روند به صورت بازگشتی ادامه می‌یابد، لذا مرتبهٔ زمانی به‌صورت زیر است:

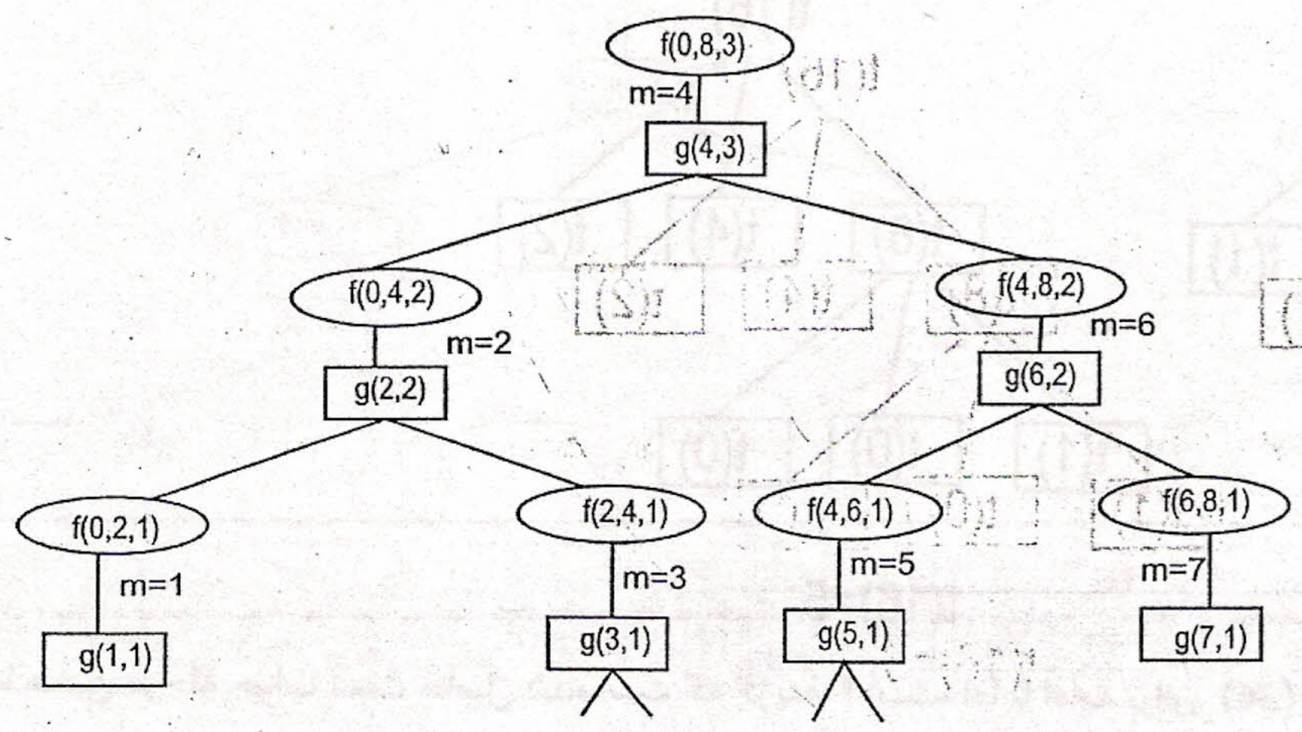
$$t(n)=t\left(\frac{n}{2}\right)+1$$

که با توجه به قضیهٔ اصلی $t(n)=\theta(\log n)$.

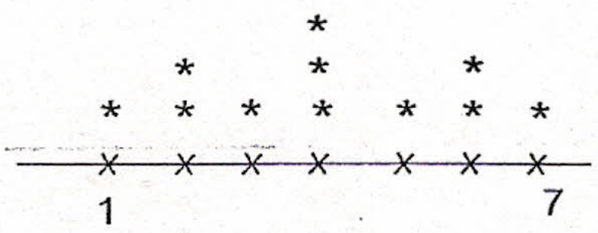
۳- چنین گرافی اصلاً سیکل ندارد.(صورت مساله اشتباه است.)

۴- اگر $e\in E(T)$ آن‌گاه خود درخت T مینیمم است. در غیر این صورت فرض کنید $e=\{x,y\}$. بدیهی است، اضافه کردن e به T تشکیل دور می‌دهد. بنابراین در مسیر از x به y یال با بیشترین هزینه را از T حذف کرده یال e را به آن اضافه می‌کنیم. پیدا کردن یال با هزینهٔ ماکزیمم در درخت پوشا از مرتبهٔ زمانی $\boxed{O(V)}$ است.

۵-



در نتیجه داریم :



۶-

$$t(n)=2t\left(\left\lfloor\sqrt{n}\right\rfloor\right)+\log n$$

با فرض $n=2^m$ و تعریف $S(m)=t(n)=t\left(2^m\right)$ داریم:

$$S(m)=2\,S\left(\frac{m}{2}\right)+m$$

که با توجه به قضیهٔ اصلی و این‌که $a=b=2\ ,\ k=1$ داریم:

$$s(m)=\theta\left(m^k\log m\right)\Rightarrow t(n)=\theta(\log n\log\log n)$$

۷-

$$\sum_{l=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i+1}^{i+l-1}2$$

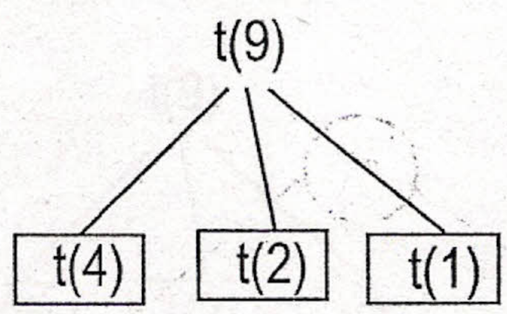
$$=2\sum_{l=2}^{n}\sum_{i=1}^{n-l+1}(i+l-1-i-1+1)$$

$$=\sum_{l=2}^{n}\sum_{i=1}^{n-e+1}2(l-1)$$

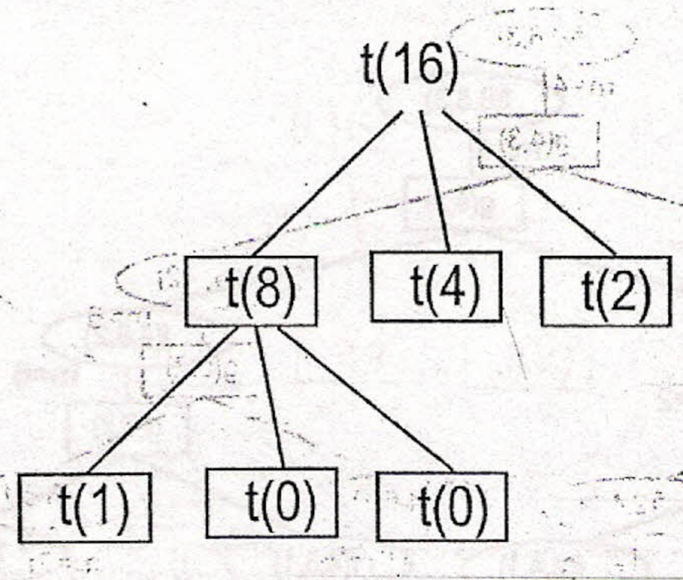
$$=\sum_{l=2}^{n}2(l-1)(n-l+1-l+1)$$

$$=\sum_{l=2}^{n}2(l-1)(n-l+1)$$

۱-

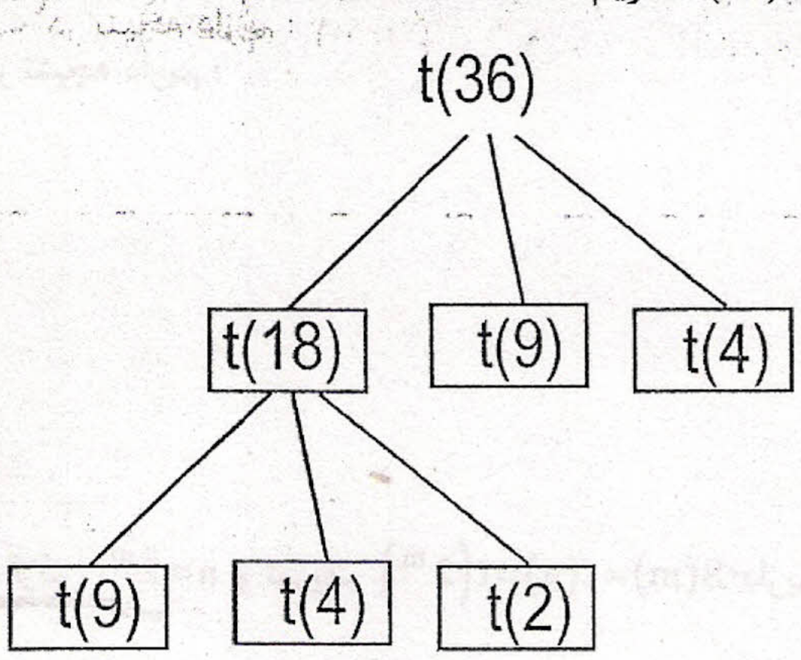


t(9)=3



t(16)=5

تا همین مرحله جواب تست حاصل شده است که گزینهٔ ۲ است، اما با ادامه برای t(36) داریم:



t(36)=3+1+1+3+1=9

۲- جواب از رابطهٔ بازگشتی $a_k=2a_{k-1}+k$ حاصل می‌شود. بدیهی است که تعداد تغییرات از 0 به 1 یا برعکس برای یک عدد k بیتی برابر مجموع سه کمیت زیر است:

الف) تعداد تغییرات از 0 به 1 یا برعکس برای اعدادی که با بیت 0 شروع می‌شوند که تعداد آن‌ها برابر a_{k-1} است.

ب) تبدیل $\underbrace{0111...1}_{k\ \text{بیت}}$ به $\underbrace{1000...0}_{k\ \text{بیت}}$ که تعداد تغییرات در آن برابر k است.

ج) تعداد تغییرات از 0 به 1 یا برعکس برای اعدادی که با بیت 1 شروع می‌شوند که تعداد آن‌ها نیز برابر a_{k-1} است.

$$a_k=k+2a_{k-1}=k+2\left[k-1+2a_{k-2}\right]=k+2(k-1)+2^2a_{k-2}$$

$$=k+2(k-1)+2^2(k-2)+...+2^{k-1}(1)+0$$

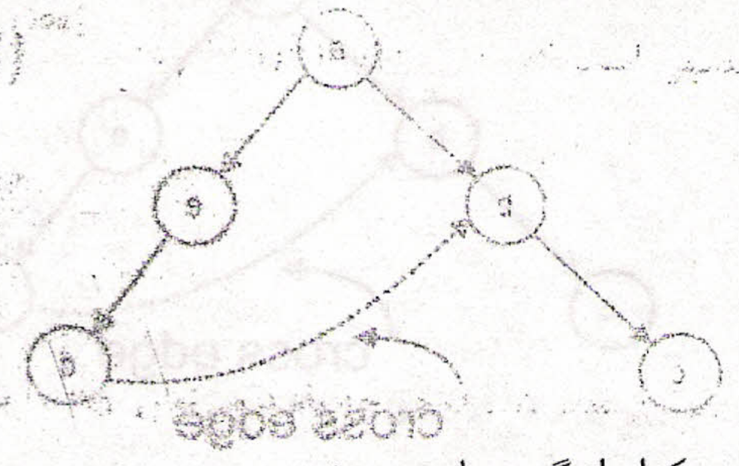
$$=n\left(\frac{k}{2^k}+\frac{k-1}{2^{k-1}}+\frac{k-2}{2^{k-2}}+...+\frac{1}{2}\right)\quad \boxed{=n\sum_{i=1}^{k}\frac{i}{2^i}}$$

با فرض اینکه $n=2^k$

برای مثال

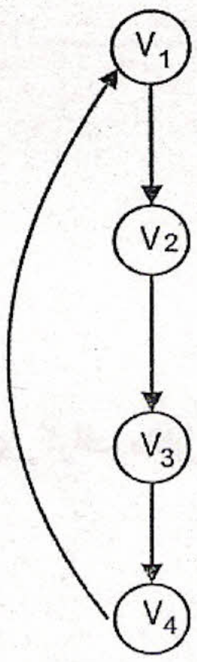
$a_3=2\times4+3=11$ از طرفی

$$\begin{cases} a_2=4 \\ a_3=8\left(\frac{1}{2}+\frac{2}{4}+\frac{3}{8}\right)=4+4+3=11 \\ a_4=16\left(\frac{1}{2}+\frac{2}{4}+\frac{3}{8}+\frac{4}{16}\right)=8+8+6+4=26 \end{cases}$$



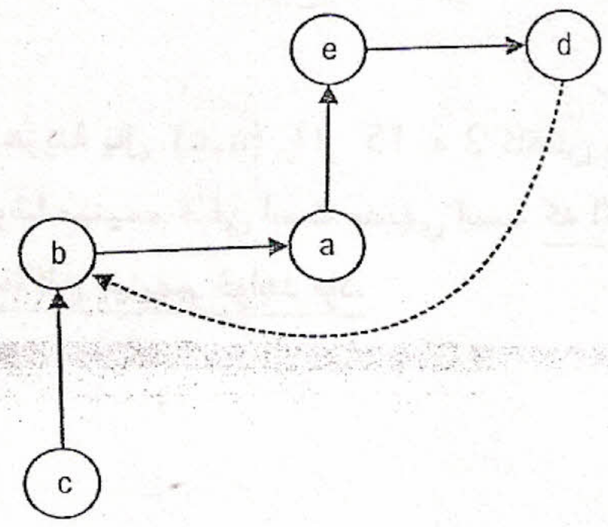
هیچ‌کدام از گزینه‌ها درست نیست.

۳ – الف) اگر گرافی دور هامیلتونی داشته باشد نمی‌تواند cross edge داشته باشد ⇔ اگر گرافی cross edge دارد نمی‌تواند دور هامیلتونی داشته باشد.

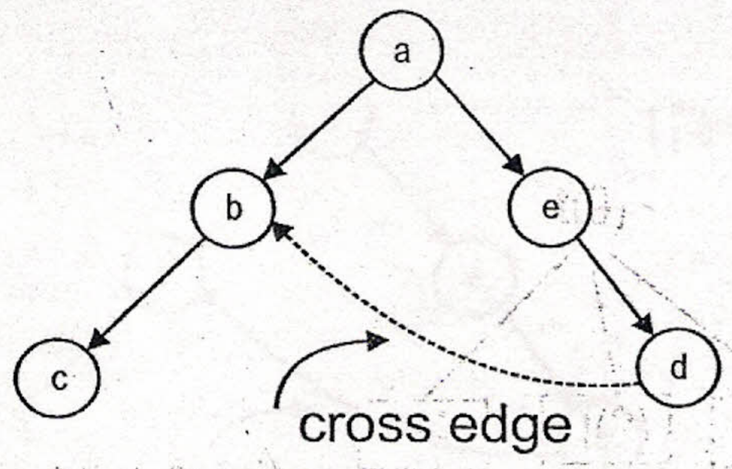


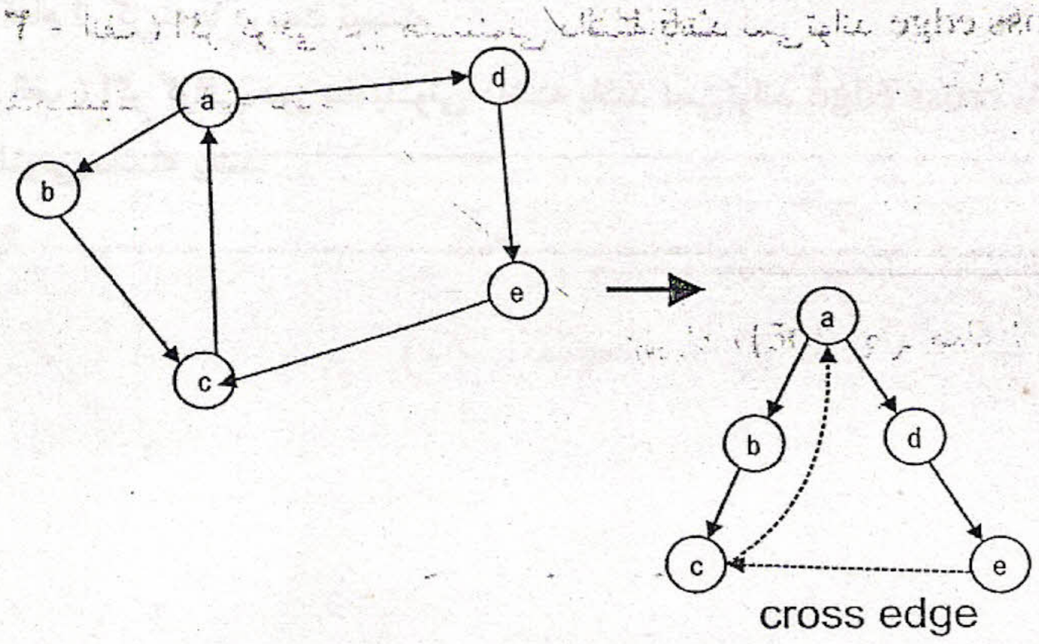
ب) در گراف G_4 یال (e,c)، یک cross edge است، با وجود این راس e نقطهٔ مفصل نیست.

ج) مثال از گرافی که هم مسیر هامیلتونی دارد و هم cross edge، مسیر هامیلتونی عبارت است از : c,b,d,e,a

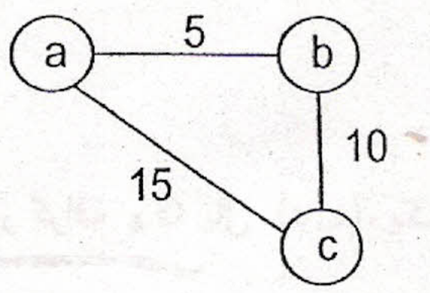


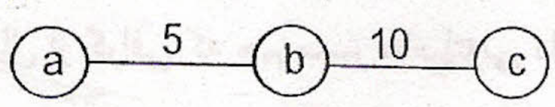
د- مثال از گرافی که هم cross edge و هم مسیر اولری دارد. مسیر اولری عبارت است از d,a,c,b,a و c.



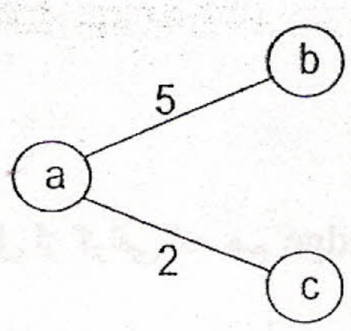


۴- برای گراف مقابل درخت پوشای مینیمم به صورت زیر است:





حال اگر هزینهٔ یال (a,c) را از 15 به 2 کاهش دهیم، بدیهی است که درخت پوشای مینیمم به صورت زیر خواهد بود که متفاوت از درخت پوشا مینیمم قبلی است. بدیهی است که اگر از هزینه یال‌هایی کم کنیم که به درخت پوشای مینیمم تعلق دارند، درخت پوشای حاصل کماکان مینیمم خواهد بود.



۵- $f(n)=4^{\log n}$, $g(n)=(\log n)^{\log n}$, $h(n)=\log^2 n$

برای سهولت فرض می‌کنیم که مبنای لگاریتم‌ها 2 است.

$f(n)=n^2$, $g(n)=(\log_2 n)^{\log_2^n}$ $h(n)=\log_2^2 n$

بدیهی است که الف $f(n)=Og(n)$ زیرا : $n^2 \le c\left(\log_2^n\right)^{\log_2^n}$ یا

$$2\log_2^n \le c\log_2^n \log_2^n \Leftrightarrow \frac{2}{c} \le \log_2^n \Leftrightarrow$$

$$n \ge 2^{\frac{2}{C}}$$

بنابراین عدد ثابت c هر چه باشد می‌توان n_0 را به صورت زیر انتخاب کرد که $n_0=2^{\frac{2}{C}}$ که به ازای $n \ge n_0$ رابطه $n^2 \le c\left(\log_2^n\right)^{\log_2^n}$ همواره برقرار است.

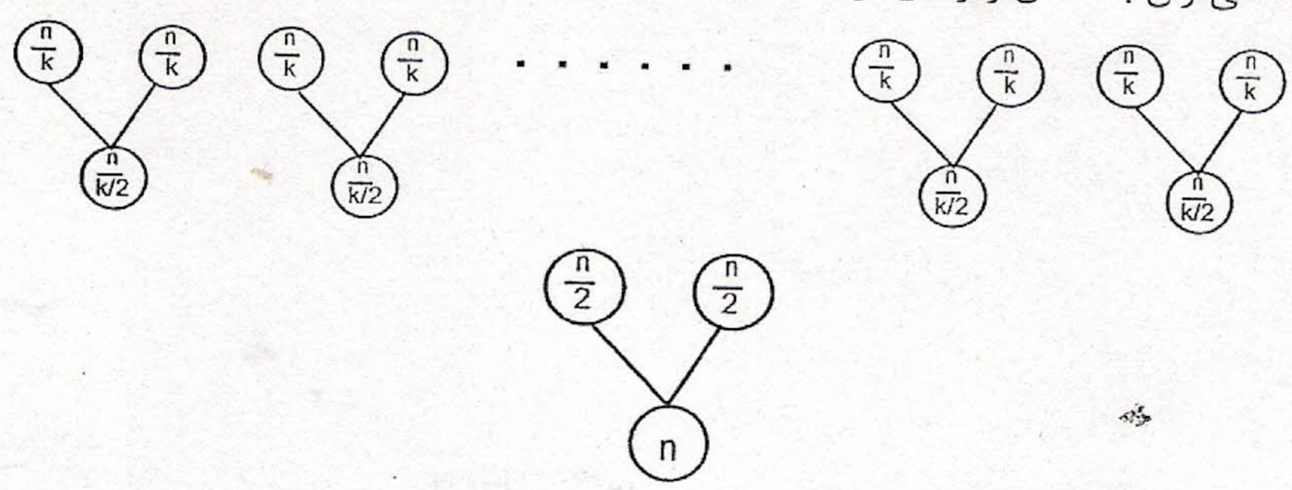
ب) $f(n)=\Omega(h(n))$ زیرا:

$$n^2 \ge c(\log n)^2 \Leftrightarrow n \ge c_1 \log n$$

با انتخاب $c_1=1$ به ازای $n \ge 1$ رابطهٔ مزبور برقرار است.

۶- برای $k=1$ داریم $A[1] \le A[2] \le A[3] \le A[n]$، لذا آرایه مرتب شده است و هیچ نیازی به جابه‌جایی نیست. تنها گزینه‌ای که می‌تواند درست باشد گزینهٔ ۳ است: $n\log 1=0$. اندکی جدی باشیم. اگر $k=2$ داریم $A[1] \le A[3] \le A[5] \le \ldots$ و $A[2] \le A[4] \le A[6] \le \ldots$ بنابراین آرایه از دو نیم آرایهٔ مرتب شده تشکیل شده است که نیاز به n جابه‌جایی $(n\log 2=n)$ دارد.

حال واقعاً جدی باشیم. برای سهولت فرض کنید که $k=2^m$. آرایه از k زیر آرایهٔ مرتب شده هر کدام با $\frac{n}{k}$ عنصر تشکیل شده است که در $m=\log_2 k$ مرحله می‌توان به شکل زیر کل آرایه را مرتب کرد. در هر مرحله نیز نیاز به n جابه‌جایی داریم:



برای مثال ، برای آرایهٔ زیر با $k=4, n=8$ داریم:

	1	2	3	4	5	6	7	8
A	5	3	12	1	9	4	18	2

↓

3	4	5	9	1	2	12	18

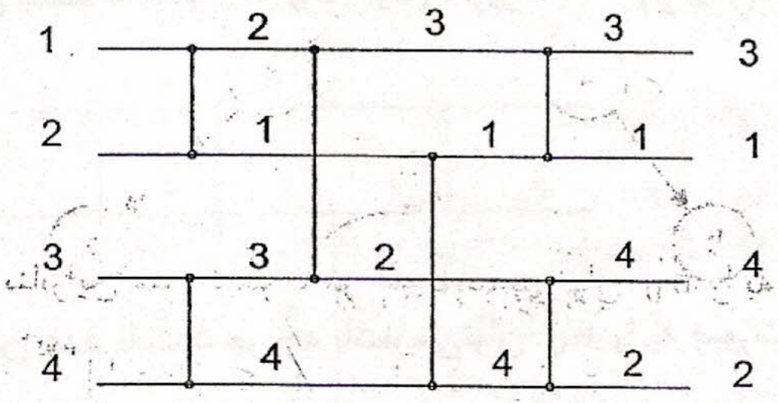
↓

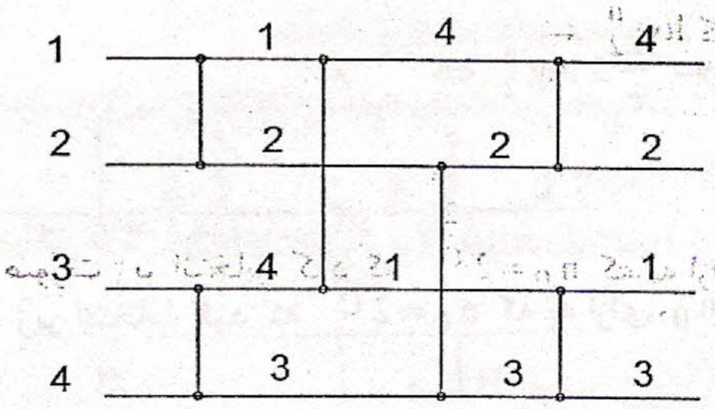
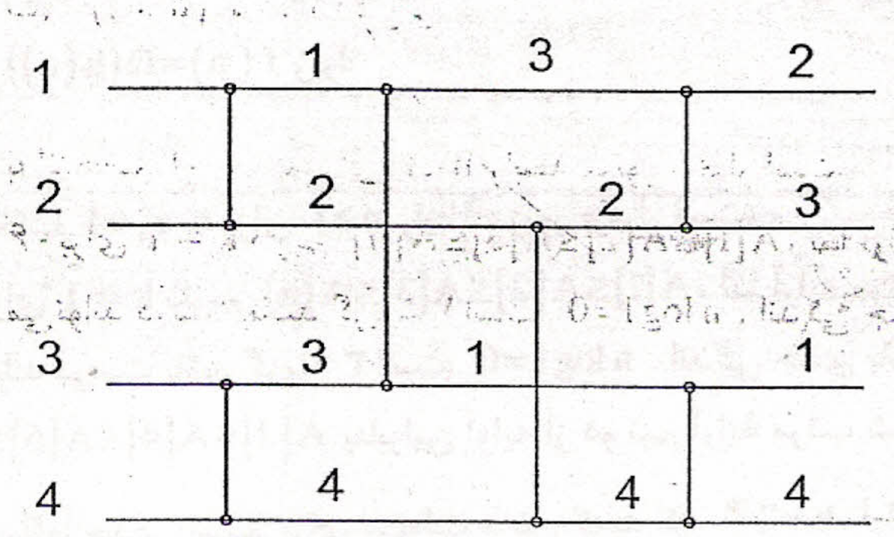
1	2	3	4	5	9	12	18

8 جابه‌جایی
+
8 جابه‌جایی
= $8\log 4=16$ جابه‌جایی

۷- مساله معادل با پیدا کردن k امین عنصر کوچک‌تر در آرایه‌ای نامرتب است که بهترین الگوریتم برای این منظور الگوریتم Selection است که خود از الگوریتم partition استفاده می‌کند. مرتبهٔ زمانی الگوریتم Selection در بدترین حالت $O\left(n^2\right)$، در بهترین حالت $O(n)$ و در حالت میانگین $O\left(n^2\right)$ است (کتاب طراحی و تحلیل الگوریتم‌های قلی زاده صفحه ۱۱۲)

۸-





مشاهده می‌شود که هر سه مورد درست است.