ساختمان داده‌ها

ساختمان داده گراف:

در گراف G(V,E)، V مجموعه رئوس است و E مجموعه یال‌ها. V نمی‌تواند تهی باشد ولی E می‌تواند تهی باشد. بنابراین گراف حداقل یک رأس دارد و نمی‌تواند کاملاً تهی باشد.

گراف یک ساختمان داده غیر خطی است.

برای یک گراف بدون جهت با n رأس، حداکثر تعداد یال‌ها برابر $\frac{n(n-1)}{2}$ است.

برای یک گراف جهت‌دار با n رأس، حداکثر تعداد یال‌ها برابر $n(n-1)$ است.

یک گراف بدون جهت را همبند (متصل) می‌گوییم اگر بین هر دو رأس آن مسیری وجود داشته باشد.

یک گراف جهت‌دار را همبند قوی (کاملاً متصل) می‌گوییم اگر بین هر دو رأس آن مسیری جهت‌دار وجود داشته باشد.

یک مؤلفه همبندی (کاملاً متصل) بزرگ‌ترین زیرگرافی است که همبند قوی باشد.

سؤال: گراف همبند ضعیف چه تفاوتی با گراف همبند قوی دارد؟

تعداد گراف‌های متمایز بدون جهت با n گره و e یال برابر $\binom{\binom{n}{2}}{e}$ است.

برای نمایش گراف دو روش وجود دارد: ماتریس مجاورت و لیست مجاورت.

۱- ماتریس مجاورت

اگر G(V,E) گرافی با n رأس باشد، ماتریس مجاورت آن یک ماتریس n×n است و درایه سطر iام و ستون jام آن در صورتی برابر ۱ است که از رأس i گراف G به رأس j آن یالی وجود داشته باشد. در غیر این صورت این درایه برابر ۰ است.

ماتریس مجاورت گراف‌های بدون جهت همواره متقارن است.

در ماتریس مجاورت گراف‌های بدون جهت، مجموع درایه‌های سطر iام برابر درجه رأس iام است. در ماتریس مجاورت گراف‌های جهت‌دار، مجموع درایه‌های سطر iام برابر درجه خارجه رأس iام و مجموع درایه‌های ستون jام برابر درجه وارده رأس jام است.

در ماتریس مجاورت، تعیین مجاورت دو گره از O(۱) است.

۲- لیست مجاورت

در این نمایش، n سطر ماتریس مجاورت در n لیست پیوندی قرار می‌گیرد. به عبارت بهتر برداری با n عنصر وجود دارد که هر عنصر آن لیست پیوندی متناظر با رئوسی است که در گراف متناظر به آن رأس وصل هستند.

یکی از اشکالات بارز استفاده از ماتریس مجاورت، دشوار بودن اضافه کردن یک رأس به گراف است.

هنگام استفاده از لیست مجاورت برای یک گراف بدون جهت با n رأس و e یال، نیاز به n گره head و ۲e گره لیست است.

در یک گراف جهت‌دار با n رأس و e یال در این روش نیاز به n گره head و e گره لیست داریم.

برای تعیین درجه خارجه هر رأس در یک گراف جهت‌دار با این نمایش کافی است تعداد گره‌های آن را در لیست مجاورت بشماریم. بنابراین می‌توان تعداد کل خطوط یک گراف جهت‌دار را در زمان O(n+e) تعیین نمود.

تعیین درجه وارده یک رأس با استفاده از لیست مجاورت بسیار مشکل است و به همین دلیل به این منظور از لیست مجاورت معکوس

استفاده می‌شود. لیست مجاورت معکوس مشابه لیست مجاورت معمولی است با این تفاوت که در لیست سطر i، رئوسی قرار می‌گیرند که از آنها به این گره یالی وجود داشته باشد. در این صورت تعداد گره‌ها در لیست i درجه ورودی این گره را نشان می‌دهد. لیست مجاورت و لیست مجاورت معکوس برای گراف‌های بدون جهت یکی هستند.

هنگامی که تعداد یال‌های گراف زیاد باشد استفاده از ماتریس مجاورت مناسب‌تر است و هنگامی که تعداد یال‌های گراف کم باشد، استفاده از لیست مجاورت مناسب‌تر به نظر می‌رسد.

اگر A ماتریس مجاورت گرافی باشد درایه واقع در سطر i ام و ستون jام $A^k,\cdots,A^3,A^2$ به ترتیب نشان دهنده تعداد مسیرهای به طول $k,\cdots,3,2$ از رأس iام به رأس jام هستند.

اگر B_k ماتریس مجموع توانی باشد، هر درایه $B_k[i,j]$ تعداد مسیرهای به طول حداکثر k از رأس i به رأس j را نشان می‌دهد.

$$B_k=\sum_{i=1}^{k}A^i=A^1+A^2+\cdots+A^k$$

تست: یک گراف جهت‌دار (Digraph) را در نظر بگیرید. فرض کنید این گراف را با ماتریس هم‌جواری نشان می‌دهیم. کدام یک از گزاره‌های ذیل درست است؟ (مهندسی IT، سراسری ۸۴)

۱) شرط کافی برای آنکه این ماتریس از نوع بالا مثلثی باشد، آن است که Acyclic نباشد.

۲) شرط لازم برای آنکه این ماتریس از نوع پایین مثلثی باشد، آن است که دو نقطه انفصالی داشته باشد.

۳) شرط لازم و کافی برای آن که این ماتریس از نوع پایین مثلثی باشد، آن است که گراف فوق از نوع Acyclic باشد.

۴) شرط لازم برای آنکه این ماتریس از نوع بالا مثلثی باشد، آن است که گراف فوق از نوع Acyclic باشد.

حل ✓ گزینه ۳.

تست: فضای مورد نیاز برای نمایش یک گراف G(V,E) به روش لیست همسایگی (adjacency list) کدام است؟ (سراسری ۷۸)

۱) $O(|E|+|V|)$ ۲) $O(|E|)$ ۳) $O(|V|)$ ۴) $O(|E|.|V|)$

حل ✓ گزینه ۱. هر چند که در گراف غیر جهت‌دار تعداد عناصر لیست‌ها |2|E| است، مرتبه حافظه مصرفی $O(|E|+|V|)$ است.

پیمایش عمقی درخت مشابه پیمایش preorder یک درخت است. در پیمایش عمقی (dfs) از پشته و در روش سطحی (bfs) از صف استفاده می‌شود.

در dfs اگر برای نمایش گراف از لیست مجاورت استفاده کنیم زمان جستجو O(e) است و اگر از ماتریس مجاورت استفاده کنیم $O(n^2)$ است.

نکته مهم: در هر گراف ممکن است گره‌هایی وجود داشته باشند که اگر جستجوی عمقی را از آن‌ها آغاز کنیم نتوانیم همه رئوس را پیمایش کنیم.

در هر گراف غیر جهت‌دار اگر dfs یک گره دلخواه، تمام رئوس گراف را بدهد، آن گراف حتماً همبند است.

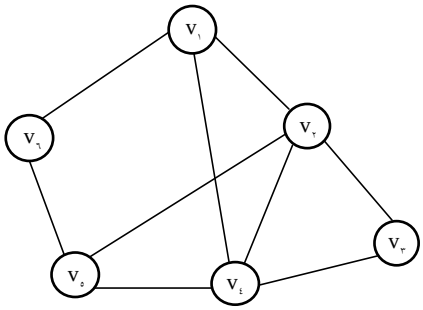
در bfs نیز مرتبه اجرایی با استفاده از لیست مجاورت O(e) و با استفاده از ماتریس مجاورت $O(n^2)$ است.

در bfs یک گراف بدون جهت نیز اگر همه گره‌ها را ببینیم، حتماً گراف همبند است.

پیمایش‌های bfs و dfs تنها در صورتی یک جواب یکتا دارند که ملاقات همسایه‌های یک گره براساس معیار مشخصی تعریف شده باشد.

برای تعیین همبند بودن یک گراف می‌توان از dfs یا bfs استفاده نمود. در این صورت اگر از لیست مجاورتی استفاده شود در زمان O(n+e) و اگر از ماتریس مجاورت استفاده شود در زمان $O(n^2)$ می‌توان به همبند بودن یا نبودن گراف پی برد.

تست: جستجوی BFS از رأس V_1 منجر می‌شود به ... (علوم کامپیوتر ۷۹)



۱) $v_1, v_2, v_3, v_4, v_5, v_6$
۲) $v_1, v_6, v_4, v_2, v_5, v_3$
۳) $v_1, v_6, v_5, v_4, v_2, v_3$
۴) $v_1, v_6, v_4, v_2, v_3, v_5$

حل ✓ گزینه ۲

تست: فرض کنید یک نوع الگوریتم پیمایش گراف بر روی دو گراف ساده همبند G_1 و G_2 اجرا شده است و $T_1 = a_1 \cdots a_n$ و $T_2 = b_1 \cdots b_m$ نشان دهنده دنباله رأس‌های ملاقات شده به ترتیب در G_1 و G_2 هستند. کدام گزاره نادرست است؟ (علوم کامپیوتر ۸۳)

۱) اگر T_1 و T_2 از پیمایش BFS به دست آمده باشند، آنگاه $T=T_1T_2$ نشان دهنده یک پیمایش BFS در گرافی است که از اتصال رأس‌های a_n و b_1 به دست آمده است.

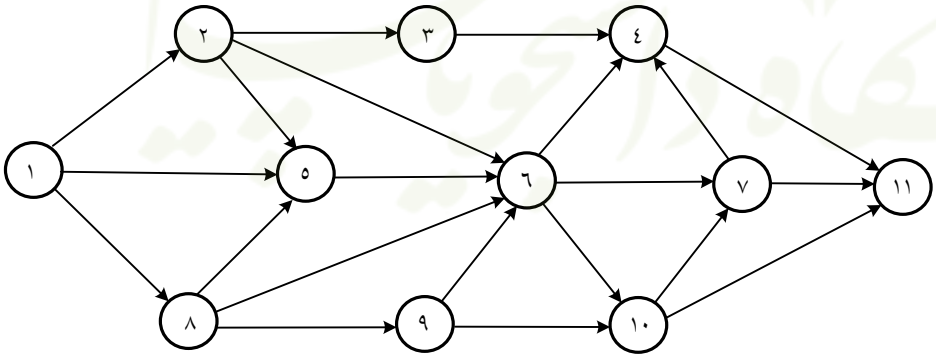
۲) اگر T_1 و T_2 از پیمایش BFS به دست آمده باشند، آنگاه $T=T_1T_2$ نشان دهنده یک پیمایش BFS در گرافی است که از اتصال رأس‌های a_1 و b_1 به دست آمده است.

۳) اگر T_1 و T_2 از پیمایش DFs به دست آمده باشند، آنگاه $T=T_1T_2$ نشان دهنده یک پیمایش DFS در گرافی است که از اتصال رأس‌های a_1 و b_1 به دست آمده است.

۴) اگر T_1 و T_2 از پیمایش DFS به دست آمده باشند، آنگاه $T=T_1T_2$ نشان دهنده یک پیمایش DFS در گرافی است که از اتصال رأس‌های a_n و b_1 به دست آمده است.

حل ✓ گزینه ۲. با چند مثال نیز جواب کاملاً مشخص می‌شود.

تست: در جستجوی اول عمق $(Depth-First-Search)$ گراف جهت‌دار زیر، فرض کنید که گره ۱، گره شروع باشد و گره‌های مجاور یک گره به ترتیب مقدار عددی‌شان ملاقات می‌شوند. کدام گزینه از چپ به راست ترتیب گره‌های ملاقات شده را نشان می‌دهد؟ (سراسری ۸۱)



۱) ۱/۲/۳/۴/۱۱/۵/۶/۷/۱۰/۸/۹
۲) ۱/۲/۸/۵/۳/۹/۶/۴/۱۱/۱۰/۷
۳) ۱/۲/۵/۸/۳/۹/۶/۴/۱۰/۷/۱۱
۴) ۱/۲/۳/۴/۵/۶/۷/۸/۹/۱۰/۱۱

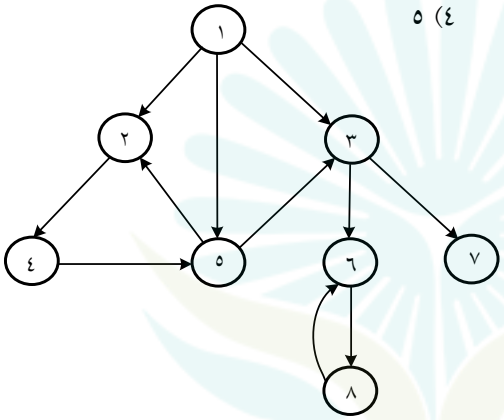
حل ✓ گزینه ۱.

تست: الگوریتم یک گراف G را به صورت اول عمق $(Depth-First)$ جستجو می‌کند و به رئوس شماره‌های $DFN(Depth\ First\ Number)$ می‌دهد.

```
Procedure dfs(v: vertex)
w: vertex;
begin
    mark v as "visited";
    for all vertices w adjacent to v do
        if w is not "visited" then dfs(w);
    Count := Count + ۱;
    DFN[v] := Count
end;
```

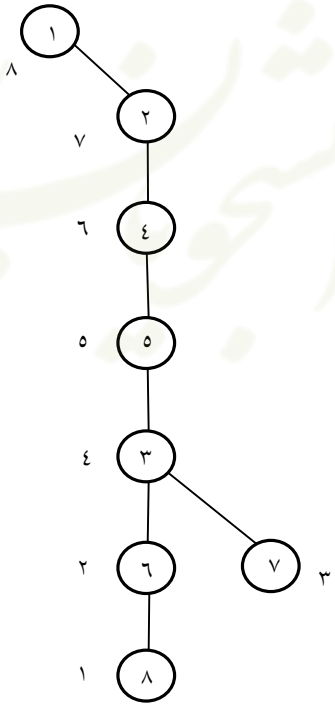
می‌خواهیم گراف زیر را با استفاده از الگوریتم فوق جستجو کنیم. فرض کنید که با dfs(۱) آغاز می‌کنیم و رئوس مجاور یک رأس، به ترتیب شماره‌های آنها مورد بررسی قرار می‌گیرند. همچنین مقدار اولیه Count را صفر فرض کنید. DFN[4] برابر چه عددی است؟

۱) ٤ ۲) ۳ ۳) ٦ ٤) ٥



حل ✓ گزینه ۳. در این الگوریتم مقدار Count بعد از حلقه به گره‌ها منتسب می‌شود. بنابراین هنگامی Count=۱ می‌شود که به اولین بن‌بست برسیم:

اولین جایی که به بن‌بست می‌رسیم در گره ۸ است.

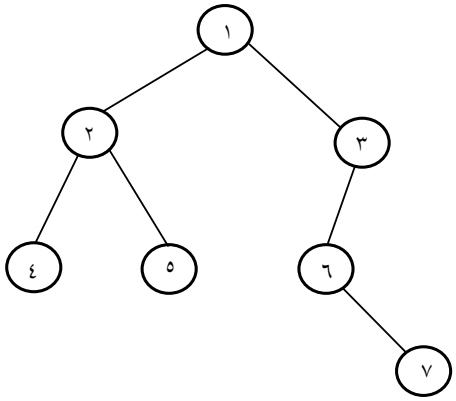


درخت پوشا با هزینه حداقل، درختی است که همه گره‌ها و n-۱ یال از گراف را طوری در بردارد که هزینه یال‌ها مینیمم باشد.

الگوریتم‌های زیادی برای تعیین درخت پوشای مینی‌مال وجود دارد که از مهم‌ترین آنها الگوریتم پریم و کروسکال است. در الگوریتم پریم، زمان اجرا کاملاً بستگی به ساختمان داده انتخاب شده برای پیاده‌سازی آن دارد. اگر از ساختمان داده ماتریسی استفاده شود زمان اجرا $\theta(n^2)$ است و اگر از HEAP استفاده شود $O((a+n)\lg n)$ است. الگوریتم کروسکال از $O(a \lg n)$ است که a تعداد یال‌ها و n تعداد

گره‌هاست.

تست: اگر در پیمایش BFS یک گراف غیر جهت‌دار $G=(V,E)$ درخت فراگیر حاصل برابر با درخت زیر باشد، در آن صورت در گراف $G=(V,E)$ کدام گزینه صحیح است؟ (علوم کامپیوتر ۸۴)



۱) بین دو رأس ٥ و ٤ می‌تواند یال وجود داشته باشد.

۲) بین دو رأس ٥ و ۱ می‌تواند یال وجود داشته باشد.

۳) بین دو رأس ۷ و ۳ و دو رأس ٤ و ۱ می‌تواندیال وجود داشته باشد.

٤) بین دو رأس ۷ و ۱ و دو رأس ٥ و ۱ می‌تواندیال وجود داشته باشد.

حل ✓ گزینه ۱.

الگوریتم‌های مرتب‌سازی:

الگوریتم‌های مرتب سازی برای مرتب نمودن داده‌ها یا آنها را با هم مقایسه می‌کنند و یا غیر مقایسه‌ای هستند (مثل radix sort و counting sort). الگوریتم‌های مقایسه‌ای از $\Omega(n\lg n)$ و الگوریتم‌های غیر مقایسه‌ای از $\Omega(n)$ هستند.

در گروه بندی دیگری می‌توان الگوریتم‌های مرتب سازی را به دو دسته داخلی (internal) و خارجی (external) تقسیم نمود. در الگوریتم‌های داخلی لازم است تمامی داده‌ها همزمان در حافظه اصلی حضور داشته باشند در حالی که الگوریتم‌های خارجی می‌توانند با قسمتی از داده‌های ورودی نیز کار خود را انجام دهند.

از نظر وابستگی حافظه مصرفی به اندازه ورودی، الگوریتم‌های مرتب‌سازی به دودسته درجا (in place) و غیر درجا (non-inplace) تقسیم می‌شوند. در الگوریتم‌های درجا، حافظه کمکی مصرفی ارتباطی به اندازه ورودی ندارد ($O(1)$) در حالی که مقدار حافظه کمکی الگوریتم‌های غیر درجا (مثل merge sort) متناسب با اندازه ورودی است.

بعضی از الگوریتم‌های مرتب‌سازی، ترتیب نسبی عناصر با کلید یکسان راتغییر نمی‌دهند. در حالی که در بعضی الگوریتم‌های دیگر این ترتیب عوض می‌شود.

در selection sort در بهترین حالت، بدترین حالت و حالت متوسط تعداد مقایسه از $O(n^2)$ است. اما تعداد تعویض در بهترین حالت $O(1)$ (ورودی به صورت مرتب باشد)، در حالت متوسط $O(n)$ و در بدترین حالت $O(n)$ (در هر مرحله یک تعویض و نهایتاً n-۱ تعویض) است.

نکته: اگر در selection sort صعودی، داده‌های ورودی به ترتیب نزولی باشند تعداد مقایسه‌ها $\frac{n(n-1)}{2}$ و تعداد تعویض $\frac{n}{2}$ است.

در Bubble sort معمولی همواره $\frac{n(n-1)}{2}$ مقایسه صورت می‌گیرد و تعداد تعویض‌ها در بدترین حالت (ورودی مرتب شده برعکس) $\frac{n(n-1)}{2}$ و در بهترین حالت برابر صفر است.

Quick sort از نوع الگوریتم‌های تقسیم و غلبه است و در آن انتخاب pivot در زمان اجرای الگوریتم بسیار مؤثر است. به طوری که اگر pivot آرایه ورودی را نصف کند بهترین حالت روی داده و الگوریتم از $O(nlgn)$ است. می‌توان ثابت نمود که در حالت متوسط نیز الگوریتم از $O(nlgn)$ است. بدترین حالت هنگامی روی می‌دهد که آرایه ورودی به دو قسمت ۱ عنصری و n-۱ عنصری تقسیم شود. در این حالت الگوریتم از $O(n^2)$ است.

بدترین حالت: $T(n)=T(n-1)+O(n) \Rightarrow T(n)\in O(n^2)$

بهترین حالت و حالت متوسط: $T(n)=2T(n/2)+O(n) \Rightarrow T(n)\in O(n\lg n)$

در Insertion sort معمولی یا straight Insertion sort زمان اجرای الگوریتم در بهترین حالت از $O(n)$ و در حالت متوسط و بدترین حالت از $O(n^2)$ است.

در Binary Insertion sort برای پیدا کردن موقعیت یک عنصر در ناحیه مرتب از binary search استفاده می‌شود. بنابراین در همه حالات از $O(nlgn)$ است.

در heap sort پس از ساختن heap از داده‌های ورودی، هر بار ریشه حذف می‌شود و heap بازسازی می‌شود. برای ساختن Heap به $O(n)$ زمان نیاز است. این الگوریتم از $O(nlgn)$ است.

الگوریتم merge sort یک الگوریتم غیر درجاست و نیاز به حافظه کمکی با $O(n)$ دارد. این الگوریتم در همه حالات از $O(nlgn)$ است چرا که:

$$T(n)=2T\left(\frac{n}{2}\right)+O(n)\Rightarrow T(n)\in O(n\lg n)$$

اما تعداد مقایسه‌های آن در حالات مختلف متفاوت است. بهترین حالت هنگامی است که یکی از آرایه‌ها پیشوند آرایه دیگر باشد. در این حالت تعداد مقایسه‌ها برابر $\frac{n}{2}$ خواهد بود. بدترین حالت هنگامی اتفاق می‌افتد که عناصر به صورت بازگشتی در هر مرحله از ادغام به صورت یکی در میان چیده شده باشند. در این حالت تعداد مقایسه‌ها n-۱ خواهد بود.

الگوریتم Bucket sort یک الگوریتم غیر مقایسه‌ای است و نیاز به تعیین محدوده داده‌ها دارد که در $O(n)$ امکان پذیر است. اگر از هر داده چند نمونه داشته باشیم این الگوریتم به Counting sort تبدیل می‌شود استفاده از این الگوریتم‌ها هنگامی به صرفه است که محدوده عناصر زیاد نباشد.

مبنای الگوریتم radix sort نیز bucket sort است. در این الگوریتم داده‌های ورودی ابتدا هم طول می‌شوند. این الگوریتم از $O(n(k+d))$ است که n تعداد داده‌ها، k طول آنها، d تنوع ارقام یا کاراکترهای آنهاست.

تست: رویه زیر را در نظر بگیرید:

```
Procedure sort(A,n);
begin
   for i:=1 to n do
   begin
      j:=i;
      for k:= j+1 to n do
         if a[k]<a[j] then j:=k;
      t:=a[i];
      a[i]:=a[i];
      a[i]:=t;
   end;
end;
```

این رویه لیست n قلمی A را به کدام روش مرتب می‌کند؟ (آزاد ۸۰)

۱) Insertion sort ۲) selection sort

۳) Bubble sort ٤) این رویه اصولا برای مرتب کردن یک لیست نوشته نشده است.

٤) این رویه اصلاً برای مرتب کردن یک لیست نوشته نشده است.

حل ✓ گزینه ۲. در هر مرحله مینی‌مم پیدا می‌شود و در جای مناسب قرار می‌گیرد.

تست: رویه Partition در الگوریتم Quick sort به صورت زیر است: (سراسری ۸۱)

```
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  j := j+1  until  A[i]>=x;
      if i<j  then  swap(A[i],A[j])  else  return(i)
  end
end;
```

اگر تمام درایه‌های $A[p\cdots r]$ دارای مقدار یکسانی باشند، مقداری که رویه فوق برمی‌گرداند چه قدر است؟

۱) p ۲) r ۳) $\left\lceil \frac{p+r}{2} \right\rceil$ ٤) $\left\lfloor \frac{p+r}{2} \right\rfloor$

حل ✓ گزینه ۳. با حل یک مثال (مثلاً r=5 و p=2) به راحتی مشخص می‌شود.

تست: اگر در الگوریتم مرتب سازی سریع (Quick sort) به ترتیب صعودی، عنصر لولا (pivot) را همان عنصر اول لیست بگیریم و با استفاده از آن یک بار لیست مرتب نزولی و یک بار دیگر لیست مرتب صعودی را مرتب کنیم گزینه صحیح برای مرتبه تعداد عملیات اصلی (مقایسه و جابجایی) را در این دو حالت انتخاب کنید: (علوم کامپیوتر ۷۹)

۱) هر دو حالت از $O(nlgn)$ ۲) هر دو حالت از $O(n^2)$

۳) برای لیست صعودی $O(n)$ و برای لیست نزولی $O(n^2)$

٤) برای لیست صعودی $O(nlgn)$ و برای لیست نزولی $O(n^2)$

حل ✓ گزینه ۲. هر دو حالت برای quick sort بدترین حالت هستند.

تست: لیست زیر را در نظر بگیرید. اگر عنصر اول لیست یعنی عدد ۹ را به عنوان لولا (pivot) اختیار کنیم کدام یک از گزینه‌های زیر می‌توانند خروجی مرحله اول الگوریتم مرتب‌سازی سریع (Quick sort) باشد؟ (علوم کامپیوتر ۸۰) (9/10/8/7/6/15/3)

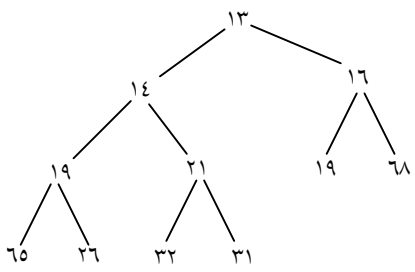
۱) (۷/۸/۹/۱۰/۳/۶/۱۵) ۲) (۷/۸/۹/۳/۶/۱۰/۱۵)

۳) (۶/۳/۸/۷/۹/۱۵/۱۰) ٤) (۶/۷/۸/۹/۳/۱۰/۱۵)

حل ✓ گزینه ۳. پس از اجرای مرحله اول الگوریتم، لازم است تمام عناصر سمت چپ ۹ از آن کوچک‌تر و عناصر سمت راست آن از آن بزرگ‌تر باشند.

تست: درخت نیمه مرتب (Heap) زیر را به صورت آرایه پیاده‌سازی کرده‌ایم: (سراسری ۷۳)

۱۳	۱٤	۱٦	۱۹	۲۱	۱۹	٦۸	٦۵	۲٦	۳۲	۳۱



اگر D را عمل Deletemin (حذف کوچک‌ترین المان) و I(x) را عمل درج عنصر x در یک درخت نیمه مرتب بنامیم آرایه حاصل از اعمال زیر که به ترتیب از چپ به راست بر روی این درخت انجام می‌شوند کدام یک از گزینه‌های زیر است:

D/I(17)/I(15)/D/D

۱) ۱۶/۱۷/۱۹/۱۹/۳۱/۶۸/۲۱/۶۵/۲۶/۳۲
۲) ۱۶/۱۹/۱۷/۱۹/۳۱/۲۱/۶۸/۶۵/۲۶/۳۲
۳) ۱۶/۱۷/۱۹/۲۶/۱۹/۲۱/۶۸/۶۵/۳۱/۳۲
۴) ۱۶/۱۹/۱۷/۳۱/۱۹/۳۲/۶۸/۶۵/۲۶/۲۱

حل ✔ گزینه ۳.

تست: به منظور sort صعودی یک آرایه از heap sort استفاده نموده و max- heap می‌سازیم. در مورد زمان اجرای این الگوریتم کدام گزینه صحیح است؟ (سراسری ۸۴)

۱) اگر آرایه از ابتدا به صورت صعودی مرتبه 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 $\theta(n^2 \lg n)$ است.

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

حل ✔ گزینه ۴. Heap sort در همه حالات از O(nlgn) است.

تست: در الگوریتم merge sort اگر به جای آنکه هر بار لیست به دو قسمت مساوی تقسیم شود، به چهار قسمت مساوی تقسیم گردد و در هر مرحله ترکیب این چهار لیست در یکدیگر ادغام شوند، پیچیدگی زمانی الگوریتم چه خواهد شد؟ (آزاد ۸۰)

۱) $\theta\left(n^{3/4}\right)$
۲) $\theta(n \lg n)$
۳) $\theta\left(n^2\right)$
۴) $\theta\left(n^2 \log_4^n\right)$

حل ✔ گزینه ۲. با استفاده از تحلیل بازگشتی جواب به راحتی به دست می‌آید.

تست: آرایه n عنصری A را در نظر بگیرید، فرض کنید $n=2^k$ باشد الگوریتم Merge sort بر روی A در بدترین حالت چند مقایسه میان عناصر آرایه انجام می‌دهد؟ (سراسری ۸۲)

۱) $n \lg n - n + 1$
۲) $n \lg n + n - 1$
۳) $n \lg n - 2n - 1$
۴) $n \lg n - n - 1$

حل ✔ گزینه ۱.

تست: اگر بخواهیم یک لیست متصل (linked list) که آدرس اول آن first است را با کم‌ترین تعداد عملیات مرتب کنیم (sort) جای علامت ? در الگوریتم زیر چه مقادیری به ترتیب قرار دهیم؟ (علوم کامپیوتر ۸۰ و ۸۴)

```
for(p = first;?;p = p → link)
for(q = ?;q;q = q → link)
   if (p → Data > q → Data)
        swap(&p→Data, &q→Data);
```

۱) p ، $p \to link$
۲) $p \to link$ ، p
۳) p ، p
۴) $p \to link$ ، $p \to link$

حل ✔ گزینه ۴.

تست: یک الگوریتم مرتب سازی صعودی در زمان اجرا در چهارمین تکرار خود دارای ارزش‌های زیر است. این الگوریتم چه نوع مرتب سازی است؟ (علوم کامپیوتر ۸۴)

۱۵/۱۸/۳۰/۳۴/۴۱/۰/۹۱/۳۲/۴۹

۱) Bubble
۲) selection
۳) Insertion
۴) Quick Sort

حل ✔ گزینه ۳. ۴ عدد اول نسبت به هم مرتب شده‌اند. اگر گزینه ۱ یا ۲ بود این ۴ عنصر باید نسبت به کل آرایه مرتب می‌بودند.