درسنامه جامع
ساختمان داده‌ها

گردآورنده

مریم نجاتی‌گرائی

سرشناسه | نجاتی‌گرائی، مریم، ۱۳۶۶ -
عنوان و نام پدیدآور | درسنامه جامع ساختمان داده‌ها/ نویسنده مریم نجاتی‌گرائی
مشخصات نشر | تهران: انتشارات علمی سنا، ۱۳۹۷.
مشخصات ظاهری | ۱۵۰ ص.: مصور، جدول، نمودار
شابک | ۹۷۸-۶۰۰-۸۴۵۶-۶۳-۶
وضعیت فهرست‌نویسی | فیپا
موضوع | ساختار داده‌ها -- راهنمای آموزشی (عالی)
موضوع | Data structures (Computer science) -- Study and (teaching Higher)
موضوع | ساختار داده‌ها -- مسائل، تمرین‌ها و غیره (عالی)
موضوع | Data structures (Computer science) -- Problems, exercises, etc. (Higher)
رده‌بندی کنگره | QA۷۶/۹ ۱۳۹۵ ۳ن ۲س/
رده‌بندی دیویی | ۷۳۰۷/۰۰۵
شماره کتابشناسی ملی | ۴۵۶۰۴۸۰

مؤسسه علمی انتشاراتی سنا

نام کتاب | درسنامه جامع ساختمان داده‌ها
نویسنده | مریم نجاتی‌گرائی
شابک | ۹۷۸ - ۶۰۰ - ۸۴۵۶ - ۶۳ - ۶
نوبت چاپ | دوم - ۱۴۰۲
صفحه‌آرایی | سیده آمنه ابوالحسنی
طراح جلد | لیلا انصاری
پست الکترونیک | elmisana@gmail.com
سایت انتشارات | sanabook.com
تیراژ | ۱۰۰ نسخه
قیمت | برای مشاهده قیمت اسکن کنید. ☜

مقدمه مولف

کتابی که در دست دارید به‌منظور استفاده دانشجویان دوره‌های کارشناسی ارشد انفورماتیک پزشکی تدوین یافته است. با توجه به اینکه درس ساختمان داده‌ها یکی از دروس امتحانی دوره‌ی کارشناسی ارشد انفورماتیک پزشکی است این کتاب تألیف شده است.

ساختار کتاب

این کتاب در هفت فصل تدوین شده است. سعی بر آن شده است که مطالب کلیدی و اصلی ساختمان داده‌ها مطرح شود. فصل اول به معرفی الگوریتم و نحوه‌ی تحلیل پیچیدگی زمانی الگوریتم‌ها پرداخته شده است. در فصل دوم مفهوم آرایه مورد بررسی قرار گرفته است و نحوه‌ی ذخیره‌سازی آرایه‌ها در حافظه شرح داده شده است. در فصل سوم ابتدا پشته و مفهوم آن و چگونگی انجام عملیات مختلف بر روی پشته توضیح داده شده است و سپس ساختمان داده صف مورد بررسی و تحلیل قرار گرفته است. فصل چهارم تحت عنوان لیست‌های پیوندی است در ابتدا مفهوم لیست پیوندی و نحوه‌ی نمایش آن شرح داده شده و سپس عملیات مختلفی که با استفاده از لیست‌های پیوندی می‌توان انجام داد بررسی شده است.

خوانندگان این کتاب در فصل پنجم با مفهوم درخت آشنا خواهند شد. نحوه‌ی نمایش درخت‌ها و نحوه‌ی نمایش درخت‌ها به همراه معرفی چندین درخت معروف در این فصل شرح داده شده است.

در فصل ششم گراف‌ها معرفی و نحوه‌ی پیمایش آن‌ها شرح داده شده است و همچنین چندین الگوریتم با حداقل هزینه معرفی شده‌اند.

فصل آخر در رابطه با الگوریتم‌های مرتب‌سازی است. در این فصل چندین الگوریتم مرتب‌سازی معرفی شده و نحوه‌ی عملکرد آن‌ها به‌صورت کاملاً واضح شرح داده شده است.

صمیمانه آرزو داریم صاحب‌نظران و دانشجویان گرامی که این کتاب را مطالعه می‌کنند نقطه نظرات و نقد خود را بر ساختار و محتوای این مجموعه به‌صورت کتبی از طریق ایمیل خاطر نشان سازند. بی‌تردید پیشنهادات سازنده‌ی شما می‌تواند نقش مؤثری در ارتقای کیفیت کتاب در چاپ‌های آتی ایفا نماید.

مریم نجاتی‌گرائی
Nejatim.6687@yahoo.com

فهرست مطالب

فصل اول: الگوریتم ۷

فصل دوم: آرایه ۲۷

فصل سوم: پشته وصف ۴۹

فصل چهارم: لیست‌های پیوندی ۷۷

فصل پنجم: درخت ۹۱

فصل ششم: گراف‌ها ۱۲۳

فصل هفتم: مرتب‌سازی ۱۳۹

فصل اول

الگوریتم

داده‌ها به صورت‌های مختلفی سازمان‌دهی می‌شوند. مدل منطقی یا ریاضی خاصی را که به‌منظور سازمان‌دهی داده‌ها به کار می‌رود، ساختمان داده‌ها نامیده می‌شود. ساختمان داده‌ها در واقع مبحثی است که به بررسی نحوه‌ی ذخیره‌سازی و آدرس‌دهی داده‌ها در حافظه و زمان مورد نیاز برای دسترسی به این داده‌ها می‌پردازد. با این توصیف، ساختمان داده‌ها مبحثی مستقل از نوع کامپیوتر و زبان‌های برنامه‌نویسی است. بر این اساس الگوریتم‌ها و برنامه‌های ارائه شده در این بخش هرچند که در قالب یک زبان برنامه‌نویسی خاص ارائه شده باشد، قابل تعمیم مانند C بوده و با سایر زبان‌های برنامه‌نویسی نیز می‌توان این الگوریتم‌ها و برنامه‌ها را پیاده سازی کرد. لذا در مبحث ساختمان داده‌ها نمادها و دستورات گرامری مورداستفاده در یک زبان خاص مانند C مورد توجه نمی‌باشند. برای فهم بهتر، می‌توان الگوریتم را شبیه به یک برنامه‌ی کامپیوتری نوشته شده دانست که این برنامه می‌تواند پایان‌پذیر نباشد و از زمان روشن شدن کامپیوتر تا زمان خاموش شدن آن در یک سیکل قرار گیرد و مدام تکرار شود اما یک الگوریتم حتماً پایان‌پذیر است. ویژگی‌هایی که یک الگوریتم می‌تواند داشته باشد به شرح زیر است:

داشتن ورودی: یک الگوریتم می‌تواند ورودی نداشته باشد یا چندین ورودی داشته باشد.

داشتن خروجی: یک الگوریتم باید حداقل یک کمیت به‌عنوان خروجی داشته باشد.

داشتن قطعیت: هر دستور باید کاملاً واضح باشد.

داشتن محدودیت: یعنی حتماً خاتمه پذیر باشد.

داشتن کارایی: باید بتوان کارایی و نحوه‌ی کارکرد الگوریتم را با استفاده از کاغذ و قلم امتحان کرد یعنی هر دستورالعمل انجام‌پذیر باشد.

در این بخش مشخصات الگوریتم‌ها را از دیدگاه زمانی مورد تجزیه‌وتحلیل قرار می‌دهیم. برای حل مسائل مختلف، الگوریتم‌هایی متفاوت وجود دارند. طراحی یک الگوریتم کارآمد در استفاده بهینه و توسعه سیستم‌های کامپیوتری، نقش مهمی ایفا می‌کند. الگوریتم، در واقع مجموعه‌ای از دستورالعمل‌هاست که اگر دنبال یا اجرا شود، هدف خاصی را ایفا می‌کند. در بررسی یک الگوریتم به لحاظ ساختمان داده‌ای به دو فاکتور بسیار مهم توجه می‌شود، یکی میزان حافظه‌ای است که الگوریتم موردنظر اشغال می‌کند و دیگری میزان زمان مصرفی آن الگوریتم است. بدیهی است

الگوریتمی بهتر است که فضا و زمان کمتری نیاز داشته باشد. در اینجا بیشتر بازدهی الگوریتم‌ها برحسب زمان مورد بررسی قرار می‌گیرد. در تحلیل میزان زمان مصرفی یک الگوریتم روش‌های مختلفی وجود دارد که در ذیل به دو نوع از مهم‌ترین آن‌ها یعنی تعیین تعداد گام‌ها و پیچیدگی زمانی الگوریتم اشاره شده است.

مرتبه و پیچیدگی زمانی

در میان الگوریتم‌ها، الگوریتم‌هایی از کارایی بهتری برخوردار هستند که از ۲ نظر کارا باشند:
الف) حافظه کمتری نیاز داشته باشند.
ب) زمان اجرای کمتری داشته باشند، یعنی در زمان کمتری نتیجه دلخواه را به ما ارائه دهند.
محاسبه‌ی تعداد گام‌ها و تعداد مراحل یک الگوریتم مفهومی مشابه با پیچیدگی زمانی دارد اما کمی متفاوت‌تر از آن است. در محاسبه‌ی تعداد گام‌های یک الگوریتم به هرکدام از دستورات برنامه که توسط CPU مورد محاسبه قرار می‌گیرد، ۱ واحد زمانی اختصاص می‌یابد. هرکدام از این واحدهای زمانی با گام‌های اجرای تمام دستورات یک برنامه، تعداد گام‌های یک برنامه را می‌سازند. در تعیین این گام‌ها به نکات زیر باید توجه شود:

۱- عبارت توضیحی (comments): در هنگام برنامه‌نویسی یا نوشتن الگوریتم گاهی اوقات یک برنامه‌نویس جهت فهم بهتر یک خط برنامه، یک عبارت توضیحی را در مقابل خط موردنظر می‌نویسد که به آن comment می‌گویند Commentها ارزش اجرایی ندارند و تعداد گام‌های آن‌ها صفر است.

۲- عبارت تعیین نوع (Declarative statements): در برنامه‌نویسی تمامی متغیرهایی که در طول برنامه استفاده می شوند، باید در ابتدای برنامه تعیین شوند. برای مثال باید نوع متغیر نوشته شود که نشان می‌دهد این متغیر از نوع اعداد صحیح است یا کاراکتر و یا... است. این عبارت نیز چون ارزش اجرایی ندارند تعداد گام‌های آن صفر است.

۳- عبارت و دستورات انتساب (Expressions & assignment statements): منظور از این بخش هنگامی است که مقدار یک متغیر تعیین می‌شود برای مثال $x = ۵$ ، تعداد گام این دستور برابر با یک است.

٤- عبارت تکرار (Iteration statement): دستوراتی که باعث تکرار یک عمل می‌شوند مانند for و while که در درس برنامه‌نویسی با آن آشنا شدید در این قسمت قرار می‌گیرند. در این دستورات شمارش گام فقط برای قسمت کنترل در نظر گرفته می‌شود.

انواع حلقه‌های تکرار

الف) حلقه‌هایی که ابتدا شرط بررسی می‌شود و سپس دستورات اجرا می‌شوند. تعداد تکرار قسمت کنترل $k+1$ ، است و تعداد تکرار دستورات k مرتبه است (مانند حلقه‌ی For یاWhile)
ب) حلقه‌هایی که ابتدا دستورات اجرا می‌شوند و سپس شرط حلقه چک می‌شود. تعداد تکرار قسمت کنترل و دستورات هر دو، k مرتبه است. (مانند حلقه‌ی Do While)

۵- عبارت نام تابع: تعداد گام صفر است.
٦- عبارت { }: تعداد گام صفر است.
۷- عبارت return : تعداد گام یک است.

مثال ۱: تعداد گام‌های برنامه‌ی زیر را بنویسید.

در بدترین حالت به ازای هر حرکت شمارنده‌ی حلقه، $x == S[i]$ می‌باشد و خط شماره ۷، n، مرتبه اجرا می‌شود بنابراین در بدترین حالت $T(n) = (n)$ است و $T(n) \in \theta(n)$ است.

در حالت متوسط فرض می‌شود خط شماره ۷، $\frac{n}{۲}$ بار اجرا می‌شود بنابراین در حالت متوسط $T(n) \in \theta(n)$ است.

نکته ۴

اگر شمارنده‌ی یک حلقه ضرب متوالی در یک عدد مثل m یا تقسیم متوالی در یک عدد مثل m را داشته باشد پیچیدگی زمانی حلقه تابعی از $o\left(\log_m^n\right)$ خواهد بود.

مثال ۳: پیچیدگی زمانی این الگوریتم از چه مرتبه‌ای برخوردار است؟

```
Int      i, j = ۰
For (i = ۱;<= n;i* = ۲)
            j++;
```

الف) $T(n) \in \theta(\log_n^۲)$ ب) $T(n) \in \theta(\log_۲^n)$

ج) $T(n) \in \theta(n)$ د) $T(n) \in \theta(n^۲)$

پاسخ: گزینه ب

تحلیل گزینه:

الف) طبق نکته گفته شده چون شمارنده‌ی حلقه در عدد ۲ ضرب می‌شود، $T(n) \in \theta(\log n_۲^n)$ می‌باشد نه $\theta(\log_n^۲)$

ب) طبق توضیحات گفته شده در (الف) این گزینه صحیح می‌باشد.

ج) اگر به جای i* = 2 شمارنده‌ی حلقه به‌صورت i-- یا i-=1 یا i = i-1 یا i++ یا i=i+1 یا i+=1 باشد $T(n) \in \theta(n)$ می‌شود.

د) اگر ۲ حلقه‌ی for تودرتو وجود داشت که شمارنده‌های حلقه‌ی آن‌ها به اندازه‌ی ۱ واحد تغییر می‌کردند، این گزینه به‌عنوان گزینه صحیح در نظر گرفته می‌شد.

الگوریتم‌های بازگشتی

در اینجا به بررسی زمان لازم جهت اجرای الگوریتم‌های بازگشتی می‌پردازیم. در هر الگوریتم بازگشتی یک شرط پایان و یک بخش فراخوانی مجدد تابع وجود دارد. با توجه به این دو بخش می‌توان به راحتی فرمول $T(n)$ را محاسبه کرد. بعد از محاسبه $T(n)$ به جدول موجود در صفحه‌ی بعد مراجعه کنید و $T(n)$ را با آن جدول تطبیق دهید.

حال به راحتی می‌توانید مرتبه الگوریتم بازگشتی را به دست آورید. راه حل دوم برای الگوریتم‌های بازگشتی مقداردهی به تابع است، یعنی شما n را برابر با چند عدد پشت سر هم قرار دهید و مقدار دقیق $T(n)$ را به ازای آن اعداد محاسبه می‌کنید حال رابطه‌ی بین n و جواب به دست آمده را حدس بزنید. سپس به راحتی پیچیدگی زمانی این الگوریتم محاسبه می‌شود.

تست‌های طبقه‌بندی شده فصل اول

۱- در تکه برنامه زیر خط process a چند بار اجرا می‌شود؟ *(کارشناسی ارشد انفورماتیک پزشکی سال ۹۰)*

```
For (int k = o; k < n − k; ++k)
    For (int k = o; k < n − k; ++k)
        {
            /*process a*/
        }
```

الف) $n/2$ ب) $n^2/4$ ج) $n^2(n+1)^2/4$ د) $n(n+1)/2$

۲- قطعه برنامه زیر را در نظر بگیرید (فرض کنید A و B متغیرهای صحیح یا مقادیر $A = 2^{21}$ و $B = 2^6$ هستند) تعداد دفعات اجرای حلقه‌ی زیر کدام است؟ *(کارشناسی ارشد انفورماتیک پزشکی سال ۹۰)*

```
1) repeat
2)      A: =A/2;
3)      B: B-2;
4) until (A=B)
```

الف) ۱۵ ب) ۱۶ ج) ۱۴ د) ۱۷

۳- درجه‌ی الگوریتم زیر چیست؟ *(کارشناسی ارشد انفورماتیک پزشکی سال ۹۰)*

```
1) for k←o to  n  do
2) {
3)     m←k
4)     while m≠o  do
5)           m←m   div 2
6) }
```

الف) n^2 ب) $n\log n$ ج) $n+\log n$ د) n

٤- تابع برگشتی زیر را در نظر بگیرید، زمان اجرای تابع برابر است با: *(کارشناسی ارشد انفورماتیک پزشکی سال ۹۰)*

```
1) int fun (int n)
2) {
3)       if(n< =1)
4)              return 1;
5)       else
6)              return fun(n-1)+(n-1);
7) }
```

الف) $O(n)+(n-1)$ ب) $O\left(\frac{n}{2}\right)$ ج) $O(n-1)$ د) $O(n)$

۵- مرتبه بزرگی (order of magnitude) قطعه برنامه زیر چیست؟ *(کارشناسی ارشد انفورماتیک پزشکی ۸۸)*

```
k: o;
for i:=1 to n do begin
      for j:=1 to m do
            k:=k+1
      J:=1;
      while y<n do begin
```

تست‌های طبقه‌بندی شده فصل دوم

۱- آرایه a(۱۵×۳۰×۴۰) در اختیار داریم که هر عنصر آن ۲ بایت حافظه نیاز دارد. اگر آن را به‌صورت ستونی از آدرس ۱۰۰۰۰ حافظه ذخیره کنیم (یعنی ۱۵ ماتریس ۳۰×۴۰ که هر یک نیز به‌صورت ستونی ذخیره شده‌اند) در این صورت آدرس عنصر a(۲۵,۱۷,۱۰) چیست؟ *(کنکور کارشناسی ارشد انفورماتیک پزشکی ۹۴)*

الف) ۲۲۰۹۸ ب) ۲۲۹۲۸ ج) ۳۲۰۹۸ د) ۳۲۲۹۲۸

۲- قسمت برنامه زیر برای ضرب دو ماتریس C = A[n×m]×B[m×p] می‌باشد. دستورات صحیح جایگزین x, y, z,w کدام گزینه است؟ *(کنکور کارشناسی ارشد انفورماتیک پزشکی ۹۴)*

الف) x: n, y: p, z:m,w: c[i][j] = A[i][k]*B[k][j]

ب) x: n, y:m, z: p,w: c[i][j] = A[i][k]*B[k][j]

ج) x: n, y: p, z:m,w: c[i][j]+ = A[i][k]*B[k][j]

د) x: n, y:m, z: p,w: c[i][j]+ = A[i][k]*B[k][j]

۳- تعداد عناصر غیر صفر در ماتریس پایین مثلثی کدام است؟ *(کنکور کارشناسی ارشد انفورماتیک پزشکی ۹۴)*

الف) $\frac{n(n+1)}{2}$ ب) $\frac{n(n-1)}{2}$ ج) $\frac{n^2}{2}$ د) n

۴- حاصل‌ضرب دو ماتریس اسپارس (پراکنده) *(کنکور کارشناسی ارشد انفورماتیک پزشکی ۹۴)*

الف) پراکنده نخواهد بود. ب) همواره پراکنده است.

ج) لزوماً پراکنده نیست. د) امکان‌پذیر نیست.

۵- کدام گزینه در مورد ماتریس اسپارس (پراکنده) صحیح نیست؟ *(کنکور کارشناسی ارشد انفورماتیک پزشکی ۹۴)*

الف) دارای عناصر صفر زیادی است.

ب) می‌توان به‌صورت خطی آن را ذخیره کرد.

ج) زمان مورد نیاز برای بازیابی یک مقدار غیر صفر در ماتریس اسپارس که به کمک آرایه سه ستونی نمایش داده شده کمتر از حالت ماتریسی است.

د) می‌توان به‌صورت لیست پیوندی ذخیره شود.

۶- قطعه برنامه زیر برای ضرب در ماتریس مربع A,B با مرتبه n موردنظر است. برای تکمیل این قطعه برنامه دستور حذف شده به‌صورت..... است. *(کنکور کارشناسی ارشد انفورماتیک پزشکی ۸۸)*

```
for   i:۱ to n do
  for   j:=۱ to n do
  C[i.j]:= o;
  for   k:=۱ to n do

_____

end;
```

الف) c[i, j]:= A[i, k]*B[k, j]+ c[i, j] ب) c[i, j]:= A[i, j]*B[k, j]

ج) c[i, j]:= A[i, k]*B[k, j] د) c[i, j]:= A[i, j]*B[k, j]+ c[i, j]

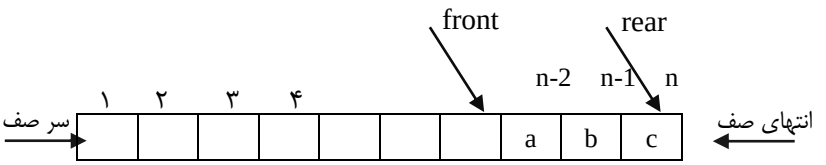
یعنی ابتدای آرایه خالی است و rear می‌تواند دور بزند و دوباره کار خود را از ابتدای آرایه شروع کند. به این نوع صف، صف حلقوی می‌گویند.

اگر فرض شود که آرایه به‌صورت یک حلقه است و صف می‌تواند در این آرایه همیشه بچرخد این صف، یک صف حلقوی است.

در این صف شرط پر بودن صف، دیگر rear == n نیست بلکه front = (Rear +1)%n است.

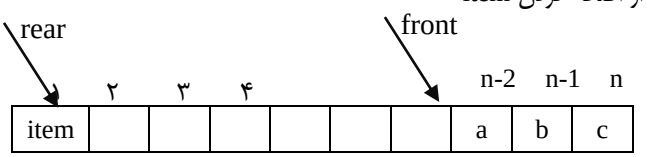
مثال ٤: صف حلقوی زیر را در نظر بگیرید و عنصر item را به صف اضافه کنید.

صف حلقوی Q قبل از اضافه کردنitem



با توجه به اینکه front ≠ (rear +۱)%n است، بنابراین صف پر نیست. پر نبودن صف از تصویر صف نیز کاملاً مشهود است.

صف حلقوی Q بعد از اضافه کردن item



نکته ۸-۹

۸- از مثال بالا نتیجه گرفته می‌شود که در صف حلقوی در زمان درج یک عنصر برای پیدا کردن مکان صحیح rear می‌توان از فرمول زیر استفاده کرد:

rear = (rear + ۱)%n

۹- برای حذف از یک صف حلقوی و پیدا کردن مکان مناسب برای front از فرمول زیر استفاده می‌شود:

front = (front + ۱)%n

عملیات درج در یک صف حلقوی

```
void  insert(items item)
{
    if (front ==(rear + 1- cout << "foll";
    else
    {   rear = (rear + 1)%n;
        Q [rear]= item;}
  }
```

عبارت پس از انتقال عملگرها به سمت راست

((A(BC+)*) D*))

حال با حذف پرانتزها، عبارت موردنظر به فرم پسوندی تبدیل می‌شود.

A BC + *D *

۱۰- پاسخ: گزینه (ب)

الگوریتم موجود در این سؤال می‌خواهد تا زمانی که صف Q۱ و Q۲ عضو دارند به مقدار متغیر I یک واحد اضافه کند و سپس مقدار I را با Q1 مقایسه کند اگر برابر بودند مقدار Q۳ را برابر با Q۲ قرار بدهد.

Q۱ : ۱, ۷, ۵, ۴, ۶, ۲, ۷, ۸

Q۲ : ۵, ۲, ۳, ۶, ۷, ۱, ۹

مرحله اول: حال الگوریتم اجرا می‌شود. مقدار I = ۰ است و هر دو صف Q۱ و Q۲ عضو دارند بنابراین مقدار I یک واحد افزایش می‌یابد حال مقدار I را با مقدار Q۱ بررسی می‌شود، مقدار هر دو برابر با ۱ است بنابراین مقدار Q۳ را مساوی با مقدار Q۲ قرار می‌گیرد، Q۳ = ۵.

مرحله دوم: مقدار I = ۱ است و هر دو صف Q۱ و Q۲ عضو دارند بنابراین مقدار I یک واحد افزایش میابد و i = ۲ می‌باشد. حال مقدار I را با مقدار Q۱ بررسی می‌شود، مقدار Q۱ = ۷ است.

مرحله سوم: مقدار I = ۲ است و هر دو صف Q۱ و Q۲ عضو دارند بنابراین مقدار I یک واحد افزایش میابد و i = ۳ می‌باشد. حال مقدار I را با مقدار Q۱ بررسی می‌شود، مقدار Q۱ = ۵ است.

مرحله چهارم: مقدار I = ۳ است و هر دو صف Q۱ و Q۲ عضو دارند بنابراین مقدار I یک واحد افزایش میابد و i = ۴ می‌شود. حال مقدار I را با مقدار Q۱ بررسی می‌شود، مقدار هر دو برابر با ۴ است بنابراین مقدار Q۳ را مساوی با مقدار Q۲ قرار می‌گیرد، Q۳ = ۵, ۶ .

مرحله پنجم: مقدار I = ۴ است و هر دو صف Q۱ و Q۲ عضو دارند بنابراین مقدار I یک واحد افزایش میابد و i = ۵ می‌باشد. حال مقدار I را با مقدار Q۱ بررسی می‌شود، مقدار Q۱ = ۶ است.

مرحله ششم: مقدار I = ۵ است و هر دو صف Q۱ و Q۲ عضو دارند بنابراین مقدار I یک واحد افزایش می‌یابد و i = ۶ می‌باشد. حال مقدار I را با مقدار Q۱ بررسی می‌شود، مقدار Q۱ = ۲ است.

مرحله هفتم: مقدار I = ۶ است و هر دو صف Q۱ و Q۲ عضو دارند بنابراین مقدار I یک واحد افزایش میابد و i = ۷ می‌شود. حال مقدار I را با مقدار Q۱ بررسی می‌شود، مقدار هر دو برابر با ۷ است بنابراین Q۳ را مساوی با مقدار Q۲ قرار می‌گیرد Q۳ = ۵, ۶, ۹

حال صف Q۲ به انتها رسیده است بنابراین برنامه خاتمه می‌یابد.

تحلیل گزینه‌ها:

ب) با توجه به تحلیل بالا گزینه ب تنها گزینه صحیح می‌باشد.

الف) این گزینه کاملاً اشتباه می‌باشد.

ج) در صورتی صحیح می‌باشد که قرار باشد مقدار Q۳ برابر با Q۱ قرار گیرد.

د) این گزینه کاملاً اشتباه می‌باشد.

۱۱- پاسخ: گزینه (د)

تحلیل گزینه‌ها:

د) LIFO یعنی آخرین ورودی اولین خروجی است که در پشته این قانون وجود دارد.

حذف از لیست

عملیات حذف در لیست به راحتی امکان‌پذیر است برای انجام این کار باید گره‌ای که قرار است حذف شود پیدا شود (برای مثال x) و سپس link مربوط به node قبل x (مثلاً Y) به‌گونه‌ای تنظیم شود که به node بعد از گره x اشاره کند. در این شرایط گره x از لیست حذف می‌شود.

```
Void delete(pointer x,pointer y,pointer first)
{
If(y ==null) first=first→Link;
Delete x;
}
```

نکته ۶-۷

۶- برای حذف یک گره خاص از لیست به ۲ اشاره‌گر کمکی نیاز است.
۷- در حذف از لیست چون ممکن است حذف از هرجایی از لیست صورت گیرد مرتبه زمانی این الگوریتم o(n) است.

اتصال دو لیست پیوندی

در این حالت یکی از لیست‌ها را تا انتها پیمایش کرده و توجه شود که اشاره‌گر گره آخر لیستی که تا آخر پیمایش شده، برابر با null است. حال باید آن را تغییر داد و به جای اینکه به null اشاره کند آن را به first لیست دوم یا همان اولین گره از لیست دوم اشاره داد در این حالت ۲ لیست را به هم پیوند داده شده است.

نکته ۸

مرتبه زمانی الگوریتم اتصال ۲ لیست برابر با o(max(m,n)) است که m و n تعداد گره‌های لیست اول و دوم هستند.

معکوس کردن یک لیست پیوندی یک طرفه

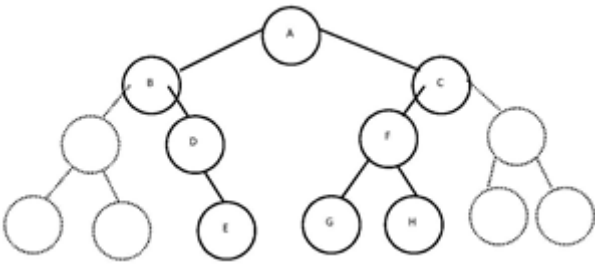
برای معکوس کردن یک لیست کافی است جهت تک‌تک اشاره‌گرها را در لیست جابه‌جا کرد.

```
P=first;
q=null;
while(P!=null){
        t=q; q=p;
        p=q→link;
        q→link=t;
}
First=q;
```

۱۶- پاسخ: گزینه (ب)

تحلیل گزینه:

ب) ابتدا درخت را رسم کرده:



حال درخت را به‌صورت inorder پیمایش کرده یعنی به‌صورت چپ، ریشه و سپس راست پیمایش می‌شود.

۱۷- پاسخ: گزینه (ب)

۱۸- پاسخ: گزینه (الف)

تعداد درختان دورویی که می‌توان با ۳ گره ساخت از فرمول $\frac{\binom{۶}{۳}}{۴}$ به دست می‌آید.

$$\frac{\binom{۶}{۳}}{۴} = \frac{۶!}{۳!\times ۳!} \times \frac{۱}{۴} = \frac{۶ \times ۵}{۶} = ۵$$

۱۹- پاسخ: گزینه (د)

در پیدا کردن کوچک‌ترین عدد باید تمام گره‌ها از کره ریشه شروع به پیمایش شود و چون تمامی گره‌ها بررسی می‌گردند کوچک‌ترین عدد در زمان $O(n)$ پیدا می‌شود.

۲۰- در درخت جست‌وجوی دورویی مقادیر گره‌های زیر درخت راست همیشه بزرگ‌تر از مقداری گره ریشه هستند و همیشه مقادیر گره‌های موجود در زیر درخت چپ، کوچک‌تر از مقدار ریشه هستند. طبق این تعریف اگر یک لیست مرتب وجود داشته باشد حالت مناسبی برای استفاده از الگوریتم جست‌وجوی دورویی است.

۲۱- پاسخ: گزینه (ب)

با trace کردن الگوریتم متوجه می‌شویم که هر درخت دورویی را که به‌عنوان ورودی به این الگوریتم بدهیم شماره‌ی آخرین سطح که عددی است با مقدار یکی کمتر از عمق خروجی می‌فرستد بنابراین برای بدست آوردن عمق صحیح درخت دورویی ورودی به این تابع باید مقدار بدست آمده با عدد ۱ جمع گردد.