


ساختمان داده‌ها

ساختمان داده صف:

صف ساختمان داده‌ای است که پروتکل FIFO را بر روی داده‌های ورودی اعمال می‌کند.

یکی از مهم‌ترین کاربردهای صف در زمان‌بندی برنامه‌ها در سیستم عامل است.

ساده‌ترین پیاده‌سازی صف با استفاده از آرایه است. (صف خطی) در این پیاده‌سازی تعداد عناصر محدود است و برای نگه‌داشتن ابتدا و انتهای صف به دو اشاره‌گر نیاز است.

در حالتی که در ساختمان داده صف تنها از اشاره‌گر انتها استفاده شود می‌توان درج را با مرتبه ۱ و حذف را با مرتبه n انجام داد. هنگامی که از اشاره‌گرهای front و rear استفاده می‌کنیم ٤ حالت پیش می‌آید:

۱) front به عنصر ابتدای صف و rear به عنصر انتهای صف اشاره می‌کند.

در این حالت در صف خالی front= rear+ 1 و در صف‌پر rear= max است.

۲) front به عنصر ابتدای صف و rear به اوّلین عنصر خالی اشاره می‌کند.

در این حالت در صف خالی front= rear و در صف‌پر rear= max+1 است.

۳) front به عنصر ماقبل ابتدا و rear به عنصر انتهای صف اشاره می‌کند.

در این حالت در صف خالی front= rear و در صف‌پر rear= max است.

٤) front به عنصر ماقبل ابتدا و rear به اوّلین عنصر خالی اشاره می‌کند.

در این حالت در صف خالی rear= front+ 1 و درصف‌پر rear= max+1 است.

مشکل اصلی صف خطی این است که هر خانه فقط یک بار قابل استفاده است و هنگامی که rear به انتها می‌رسد دیگر نمی‌توان در صف چیزی را ذخیره نمود. بنابراین صف دوار تعریف می‌شود. صف دوار از نظر ساختمان داده‌ها هیچ تفاوتی با صف خطی ندارد. تنها تفاوت در نحوه حرکت اشاره‌گرهاست. چهار حالت مطرح شده برای صف خطی را برای صف حلقوی نیز بررسی می‌کنید.

نکته مهم: در صف حلقوی تعریف صف تهی و صف‌پر بر هم منطبق است. بنابراین همواره یک عنصر خالی می‌ماند. به عبارت بهتر اگر طول آرایه مورد استفاده n باشد ماکزیمم تعداد عناصری که در یک صف دوار می‌توانند قرار بگیرند n-1 است. اگر R به عنصر انتهایی و F به عنصر ماقبل ابتدا اشاره کند داریم:

اگر F>R =R-F= تعداد خانه‌های‌پر

=n-(R-F) تعداد خانه‌های خالی

اگر F>R =F-R = تعداد خانه‌های خالی

=n-(F-R) تعداد خانه‌های‌پر

به عبارت بهتر همواره F پشت سر خود را خالی و R پشت سر خود را پر می‌کند.

صف پیوندی در حقیقت یک لیست پیوندی یک طرفه است که همواره درج در انتهای آن و حذف از ابتدای آن صورت می‌گیرد. برای کاهش مرتبه درج در انتهای لیست در ساختار صف پیوندی همواره آدرس آخرین عنصر لیست را نگه‌داری می‌کنیم. در صف پیوندی همواره F به عنصر ابتدای صف و R به عنصر انتهای صف اشاره می‌کند.

به دلیل خاصیت پویای لیست پیوندی تنها شرایط خالی بودن صف بررسی می‌شود. در صف تهی هر دو اشاره‌گر null هستند.

صف دو سره یک لیست خطی است که داده‌های آن را می‌توان از هر دو سر اضافه یا حذف کرد ولی درج و حذف داده از وسط آن امکان‌پذیر نیست.

تست: برای حذف یک عنصر از اوّل یک صف حلقوی که عناصر آن درون یک آرایه به نام queue از نوع Element ذخیره شده، برنامه زیر ارائه شده است ولی یک سطر از برنامه حذف شده است. کدام گزینه باید به جای سطر حذف شده قرار گیرد تا برنامه کامل شود؟ (MAX QSIZE تعداد خانه‌های آرایه queue است) (مهندسی IT سراسری ۸۳)

```
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

حل ✓ گزینه ٤ صحیح است.

ساختمان داده پشته:

پشته ساختمان داده‌ای است که برای اعمال پروتکل Lifo بر روی مجموعه داده‌های ورودی استفاده می‌شود. پشته می‌تواند با استفاده از آرایه پیاده‌سازی شود. در این حالت با تعریف یک اشاره‌گر اندیسی، اندیس آخرین عنصر پشته نگه‌داری می‌شود. برای این اشاره‌گر نیز دو تعریف وجود دارد.(آخرین عنصر یا اوّلین سلول خالی)

اگر بخواهیم دو پشته را با یک آرایه پیاده‌سازی کنیم، کافیست ابتدای آرایه را ابتدای یک پشته و انتهای آرایه را ابتدای پشته دیگر در نظر بگیریم. در این صورت با افزایش عناصر، پشته‌ها به سمت یکدیگر رشد می‌کنند. این روش استفاده از حافظه را بهینه می‌کند.

در صورتی که بخواهیم بیش از دو پشته را با یک آرایه پیاده‌سازی کنیم، لازم است برای هر کدام دو اشاره‌گر ابتدایی و انتهایی پشته را تعریف کنیم.

نکته: یکی از را ه‌های کم هزینه برای معکوس کردن محتویات یک صف، استفاده از یک پشته کمکی است. (و بر عکس).

پشته پیوندی یک لیست پیوندی یک طرفه است که درج و حذف فقط در ابتدای آن انجام می‌شود. پشته پیوندی هنگامی خالی است که اشاره‌گر آن Null باشد.

تست: بر روی پشته S اعمال زیر تعریف شده‌اند:

Push(s,x): درج x در بالای پشته با هزینه O(1). pop(s,x): حذف عنصر بالای پشته با هزینه O(1).

Multipop(s,k): حذف k عنصر بالای پشته (با فرض آن که k حداکثر برابر تعداد عناصر موجود S است) با هزینه O(k)

اگر N عمل از اعمال فوق به ترتیب دلخواه بر روی پشته s که در ابتدا تهی است انجام شود مجموع هزینه این N عمل در بدترین حالت چه قدر است؟ (سراسری- مهندسی کامپیوتر ۷۸)

۱) O(NlogN)　　۲) $O(N^2)$　　۳) O(NK)　　۴) O(N)

حل ✓ گزینه ٤؛ بیشترین هزینه مربوط به دستور Multipop(s, k) است که هزینه O(k) دارد و در بدترین شرایط همواره با O(N) اجرا می‌شود. ولی تا هنگامی که N بار با دستور push (O(1)) N عدد در S ذخیره نشده باشد، دستور Multipop با هزینه O(N) اجرا نخواهد شد. پس بالاترین هزینه اجرایی N دستور برابر O(N) است.

تست: یک پشته خالی با اعداد از ۱ تا ٦ در ورودی داده شده است. اعمال زیر بر روی پشته قابل انجام هستند:

push: کوچک‌ترین عدد ورودی را برداشته و وارد پشته می‌کنیم.

pop: عنصر بالای پشته را در خروجی نوشته و سپس آن را حذف می‌کنیم.

کدام یک از گزینه‌های زیر را نمی‌توان با هیچ ترتیبی از اعمال فوق بدست آورد. (اعداد را از چپ به راست بخوانید) (سراسری- مهندسی کامپیوتر ۷٤)

۱) ۱ ۲ ۳ ٥ ٦ ٤ ۲) ۳ ۲ ٤ ٦ ٥ ۱ ۳) ٤ ۳ ۲ ۱ ٦ ٥ ٤) ۲ ۱ ٥ ۳ ٤ ٦

حل ✓ گزینه ٤.

نکته: چنانچه n عنصر در ورودی یک پشته موجود باشد، تعداد حالت‌های خروجی که با ترکیب توابع push و pop استاندارد می‌توان ایجاد کرد برابر است با عدد کاتالان: $\frac{\binom{2n}{n}}{(n+1)}$

عبارات prefix- infix- postfix

در تبدیل این عبارات به یکدیگر ترتیب حروف (عملوندها) تغییر نمی‌کند.

در postfix سمت راست و در prefix در سمت چپ حتماً عملگر داریم.

روش دستی تبدیل عبارات infix به postfix:

۱- ابتدا عبارت را به طور کامل پرانتز گذاری می‌کنیم. (با توجه به اولویت عملگرها)

۲- هر عملگر را به سمت راست پرانتز بسته خودش منتقل می‌کنیم.

۳- تمام پرانتزها را حذف می‌کنیم.

روش دستی تبدیل عبارات infix به prefix نیز به همین صورت است تنها در مرحله دوم هر عملگر را به سمت چپ پرانتز باز خودش منتقل می‌کنیم.

الگوریتمی از مرتبه O(N) برای ارزیابی مستقیم عبارات infix وجود ندارد.

ارزیابی عبارات prefix با الگوریتمی از O(n) و با کمک پشته امکان‌پذیر است. برای ارزیابی عبارات Prefix باید تمام اعضای عبارت حاضر باشند. به عبارت بهتر پردازش عبارت از سمت راست آن انجام می‌شود.

برای محاسبه عبارات postfix از این الگوریتم استفاده می‌شود: عبارت از چپ به راست پویش می‌شود و عملوندها تا مشاهده یک عملگر در داخل پشته قرار می‌گیرند. سپس تعداد لازم از عملوندها را از پشته خارج می‌کنیم و پس از انجام عملیات مربوطه نتیجه را دوباره به داخل پشته منتقل می‌کنیم. این کار را آن قدر ادامه می‌دهیم تا به انتهای عبارت برسیم. در این حالت تنها یک عنصر در پشته وجود دارد که همان جواب نهایی عبارت است.

نکته: از این الگوریتم می‌توان برای تعیین صحت یک عبارت postfix نیز استفاده نمود.

نکته: مرتبه این الگوریتم O(n) است که n طول رشته یا تعداد عملگرها و عملوندهاست.

از الگوریتم فوق می‌توان برای ارزیابی یک عبارت پیشوندی نیز استفاده کرد. در این حالت کافیست رشته ورودی از راست به چپ بررسی شود.

نکته: حداقل طول stack برای پیاده‌سازی این الگوریتم و یا ارزیابی یک عبارت postfix همواره برابر است با حداکثر طول آن در روند اعمال الگوریتم.

برای محاسبه حداقل طول پشته کافیست توجه کنید که هر عملگر دودویی یک واحد از طول پشته کم می‌کند امّا عملگر یکتایی طول پشته را تغییر نمی‌دهد.

الگوریتم تبدیل عبارات infix به postfix به کمک پشته را بررسی می‌کنید.

نکته: در این الگوریتم تعداد pushها برابر است با مجموع تعداد عملگرها و تعداد پرانتزهای باز.

تست: مینیمم تعداد متغیرهای میانی در محاسبه عبارت جبری ab+ cd*/ a+ که به صورت postfix است برابر است با: (سراسری- علوم کامپیوتر ۷۹)

۱) ۱ ۲) ۲ ۳) ۳ ٤) ٤

حل گزینه ۳

تست: عبارت postfix معادل عبارت (A+B)* D+E/ (F+A*D) برابر است با: (سراسری- علوم کامپیوتر ۸۰)

۱) AB+ D*E+F/ A+D* ۲) AB+D* EFAD*+/+

۳) ABDEFAD+*+/+* ٤) AB+ DE+ FAD*+/

حل گزینه ۲

تست: به چند حالت می‌توان عبارت زیر را به صورت کامل پرانتزگذاری کرد تا عبارت حاصل (با توجه به اولویت عملگرها) به ازای همه مقادیر متغیرها برابر شوند؟ a- b* c- d* e/f

توضیح: در پرانتزگذاری کامل هر زیر عبارت به صورت $(E_1\ OP\ E_2)$ است که E_1 و E_2 هم به همین صورت پرانتزگذاری شده و OP یک عملگر است. (سراسری- علوم کامپیوتر ۸۲)

۱) ۱ ۲) ۲ ۳) ۵! ٤) $\frac{۱}{۵}\binom{۱۰}{۵}$

حل گزینه ۱

ساختمان داده درخت:

از ساختمان داده درخت برای نمایش رابطه سلسله مراتبی بین مجموعه‌ای از عناصر استفاده می‌شود.

درخت را می‌توان به صورت استقرایی نیز تعریف نمود.

نتیجه: چنانچه یک درخت موجود باشد هر فرزند از ریشه درخت خود یک درخت کوچک‌تر است.

در یک درخت ریشه عنصری است که پدر ندارد. در حالی که برگ عنصری است که فرزند ندارد. درجه هر برگ در درخت همواره ۱ است. ارتفاع یک گره فاصله آن تا دورترین برگ متصل به آن است. به فاصله یک گره تا ریشه درخت عمق آن گره گفته می‌شود. ارتفاع درخت، ارتفاع ریشه درخت است.

درخت k تایی درختی است که هر گره آن دارای حداکثر k فرزند است. درخت k تایی کامل درختی است که هر گره آن دقیقاً k فرزند دارد. عناصری از درخت که دارای عمق یکسان باشند اجزای یک سطح درخت را تشکیل می‌دهند. درخت متوازن درختی است که تفاوت سطح برگ‌های آن حداکثر ۱ باشد. اگر اختلاف سطح برگ‌های یک درخت ۰ باشد (همه برگ‌ها در یک سطح باشند) آن درخت کاملاً متوازن است.

درخت kتایی کامل کاملاً متوازن را درخت پر می‌گوییم. در یک درخت پر به ارتفاع h همواره داریم:

k: درجه درخت h: ارتفاع V: تعداد رئوس $|V| = \sum_{i=0}^{h} k^i$

چنانچه تعداد برگ‌های یک درخت پر را b فرض کنیم آن گاه برای محاسبه تعداد گره‌های این درخت (n) با فرض این که k درجه درخت باشد داریم: $n = \frac{bk-1}{k-1}$

تعداد گره‌هایی که می‌توان به یک درخت دوتایی با n گره در یک لحظه اضافه کرد n+1 است.

حداکثر تعداد کل گره‌ها در یک درخت دودویی به عمق d برابر است با $(d \geq 1)\ 2^d - 1$

در یک درخت دودویی پر با n گره، عمق برابر $\log_2^{(n+1)}$ است.

در هر درخت تعداد یال‌ها برابر n-1 است. (n تعداد گره‌هاست)

تعدا کل یال‌ها در یک درخت دودویی برابر $n_1 + 2n_2$ است که n_1 تعداد گره‌های مرتبه ۱ و n_2 تعداد گره‌های مرتبه ۲ است.

در هر درخت دودویی اگر n_0 تعداد گره‌های پایانی (برگ‌ها) و n_2 تعداد گره‌های درجه ۲ باشد، آن گاه داریم: $n_0 = n_2 + 1$

در حالت کلی در یک درخت kتایی اگر n_k تعداد گره‌هایی باشد که k فرزند دارند، داریم:

تعداد برگ‌ها $= n_0 = (k-1)\, n_k + (k-2)\, n_{k-1} + \ldots + n_2 + 1$

یعنی تعداد برگ‌ها به تعداد گره‌های مرتبه ۱ بستگی ندارد.

در یک درخت k تایی با n گره nk اتصال وجود دارد که n-1 اتصال استفاده شده است (یال‌ها) بنابراین nk- (n-1) اتصال تهی و بدون استفاده است.

در یک درخت k تایی پر با n گره، تعداد برگ‌ها برابر است با: $n_0 = n - \frac{n-1}{k}$

و تعداد گره‌های غیر برگ برابر است با: $\frac{n-1}{k}$

در یک درخت k تایی کامل تعداد گره‌های غیر برگ برابر $\left\lfloor \frac{n-1}{k} \right\rfloor$ و تعداد گره‌های برگ $n - \left\lfloor \frac{n-1}{k} \right\rfloor$ است.

ساختمان داده درخت یک ساختمان داده غیر خطی است.

برای پیاده‌سازی درخت دودویی می‌توان از آرایه استفاده نمود. در این صورت اگر عنصری در خانه شماره i قرار بگیرد، فرزندان آن در 2i و 2i+1 قرار می‌گیرند و پدرش در $\left\lfloor \frac{i}{2} \right\rfloor$ قرار می‌گیرد.

تعداد درخت‌های دودویی که با n گره می‌توان ساخت برابر است با عدد کاتالان: $\frac{\binom{2n}{n}}{n+1}$

تست: درخت T با n رأس مفروض است. اگر این درخت دارای پنج رأس (node) از درجه (degree) چهار و شش رأس از درجه سه و پنج رأس از درجه دو باشد، تعداد برگ‌های آن چه قدر است؟ (سراسری ۷۱- آزاد ۸۱)

۱) n-1 ۲) 2n-2 ۳) ۱۸ ۴) هیچکدام

حل ✓ گزینه ۴

تست: یک درخت دودویی درختی است که هر گره آن صفر یا ۲ فرزند دارد. اگر n(h) و N(h) به ترتیب حداقل و حداکثر تعداد گره‌های یک درخت دودویی به ارتفاع h باشد، برای $h > 0$ کدام یک از گزینه‌های زیر درست است؟ (علوم کامپیوتر ۸۱)

۱) $N(h) = 2^{h+1} - 1 ,\ n(h) = 2h+1$ ۲) $N(h) = 2^{h-1} ,\ n(h) = h+1$

۳) $N(h) = 2^{h+1} ,\ n(h) = h+1$ ۴) $N(h) = 2^{h} - 1 ,\ n(h) = 2h+1$

حل ✓ گزینه ۱

تست: در یک درخت T با n عنصر، همه عناصر غیر برگ دارای دقیقاً ۲ فرزند هستند. E(T) و I(T) را به ترتیب مجموع عمق برگ‌ها و

مجموع عمق عناصر غیر برگ تعریف می‌کنیم. اگر $T(n)= E(T)- I(T)$ باشد، داریم: (سراسری ۸۲)

۱) $T(n)= T(n-2)+ 2$ ۲) $T(n)= T(n-1)+ 1$

۳) $T(n)= T(n-1)+ n-1$ ٤) $T(n)= T(n-2)+ n-2$

حل ✓ گزینه ۱

تست: کدام گزینه نادرست است؟ (سراسری ۸۲)

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

۲) ارتفاع درخت کامل دودویی که n برگ دارد برابر است با $\left\lfloor \log_2^n \right\rfloor$

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

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

حل ✓ گزینه ۲.