به نام خالق یکتا و به یاد یوسف زهرا

گزارش تمرین برنامه‌سازی پیشرفته

تمرین سری سوم

سیدرضا هاشمی‌راد

۲۳ فروردین ۱۳۹۸

فهرست مطالب

لینک دستیابی به فایل ها در گیت هاب

۱ سوال یک

کلید اصلی حل این سوال فهم الگوریتم و منطق مساله بود که به کمک منبع کمکی که در کنار این سوال در اختیار داشتیم، به سادگی با تبدیل شبه کد داده شده توابع مورد نیاز پیاده سازی شدند و مورد استفاده قرار گرفتند.

با توجه به اینکه الگوریتم ها خیلی مفصل و با مثال های متعدد در این سورس توضیح داده شده بودند، به جهت اختصار این گزارش به توضیح آنها نمی پردازیم و صرفا در اینجا نکاتی را بیان خواهیم نمود که ارزش مفهومی از نظر پیاده سازی کد زبان C++ داشته باشند.

برای پیاده سازی آرایه ی اصلی این درخت از یک کلاس وکتور استفاده نمودیم که به سادگی امکانات خوبی را در اختیار ما قرار می دهد.

تابع `delete()` را با تغییر نام به `deleteMax()` ساختیم زیرا نام قبلی با پاک کردن یک پوینتر داینامیک مشابهت داشت و نمی توان از آن استفاده نمود.

ارتفاع درخت در واقع برابر است با تعداد سطر گره منهای یک که می توان با استفاده از تابع $log_2()$ آن را به سادگی محاسبه نمود.

نکته برنامه نویسی دیگری در این سوال استفاده نشد؛ تنها اینکه اسم گذاری ها به جهت حفظ قاعده‌ی camelCase تماما به این صورت نوشته شدند که کمی با آنچا در تابع `main()` داده شده با این سوال متفاوت است. همچنین برای اسم کلی کلاس برای زیبایی بیشتر از `MaxHeap` استفاده شده است.

نکته‌ی خیلی مهم در حل سوال این است که در تابع اصلی `main()` اندیس های درخت از شماره یک (۱) در نظر گرفته شده اند. (مشابه نرم افزار متلب!) علت این قضیه هم این بود که در راهنمایی داده شده، اینطور فرض شده بود لذا برای اینکه مشابه همان منبع سوال حل شده باشد، به این صورت درنظر گرفته شد.

۲ سوال دو

برای حل سوال رعایت نکات ذیل ضروری بود که در ادامه به توضیح روند حل سوال و سپس بیان نکات می پردازیم:

دو متغیر سایز[۱] و کاپاسیتی[۲] را به این صورت قرار می دهیم که یکی تعداد خانه های رزرو شده برای وکتور را می دهد و دیگری طول فعلی آن. در اینجا اگر خانه ای با تابع پوش بک مقدار بگیرد جزو وکتور محسوب می شود ولی اگر

[۱] `size`.
[۲] `capacity`.

هنوز مقداری گرفته نشده باشد، صرفا یک خانه رزرو خواهد بود.

بعد از ساخته شدن آبجکت و در کانستراکتور آن، مقدار سایز را برابر صفر و مقدار کاپاسیتی را برابر یک (یک خانه رزرو) قرار داده و arr موجود را به تعداد کاپاسیتی (یک) به صورت داینامیک و پوینتری ایجاد می نماییم.

قبل توضیح در مورد تابع پوش بک، به این نکته می پردازیم که کاپاسیتی به این صورت تغییر می کند که همواره مقدار آن بر اساس توان دوم بعدی سایز وکتور است. به این معنا که برای وکتورهایی با سایز ۶، ۷، ۸ و ۹ به ترتیب کاپاسیتی یا طول آرایه موجود در کلاس برابر است با ۸، ۸، ۱۶ و ۱۶.

با توجه به نکته فوق، تابع push_back را به این صورت می نویسیم که در ابتدا یکی به سایز وکتور اضافه می نماییم. سپس بررسی می نماییم که آیا برابر توان دومی هست یا نه، اگر بود لازم است تا کاپاسیتی و طول آرایه موجود را دو برابر نماییم و مقدار آرایه قبلی را در آن کپی می نماییم با این فرق که این بار به تعداد دو برابر قبل، خانه خالی برای مقداردهی جدید وجود دارد. پس بررسی این شرط و انجام اقدامات لازم، مقدار ورودی تابع را به خانه arr[size-1] وارد می نماییم.

عملکرد تابع pop_back نیز مشابه قبل است فقط با این تفاوت که اولا هر بار سایز یکی کم شده و موقع رسیدن سایز به توان های دوم، کاپاسیتی و طول آرایه نصف می شود.

تابع show به سادگی نوشته می شود و به طول سایز، آرایه arr را چاپ می نماییم.

همچنین برای مقایسه از نظر طول عملگرهای مختلف را نیز به سادگی می توان نوشت. اما همانطور که در کلاس نیز تدریس شد، در اینجا به جای اینکه همه ی مقایسه گرهای ممکن را بنویسیم، تنها > (کوچکتر) و == (مساوی) را نوشته با استفاده از کتابخانه utility و اضافه کردن خط using namespace std::rel_ops قبل از تابع main() در فایل main.cpp سایر عملگرها نیز کار می کنند.

توابع جمع و ضرب دکارتی نیز به سادگی نوشته شده که خروجی هر دو یک عدد صحیح از نوع unsigned long است. (توجه شود با توجه به این امکان که طول آرایه ها ممکن است خیلی زیاد باشد، متغیرهای سایز و کاپاسیتی از همین نوع انتخاب شدند. همچنین متغیرهای حلقه های فور نیز از همین نوع انتخاب شدند.)

همچنین به دلیل استفاده از آرایه دینامیک اپراتور مساوی نیز نوشته شده تا مشکل دوبار پاک شدن یک قسمت حافظه رخ ندهد. در ادامه برای اینکه همین مشکل در ساختن یک آبجکت از روی یک آبجکت دیگر پیش نیاید، باید کپی کانستراکتور نوشته می شد که در آن مقدار آرایه ها درایه به درایه کپی شد.

اما زمانی پیش می آید که مثلا ما می خواهیم از روی یک متغیر r-value یک آبجکت جدید بسازیم که در این موارد برای بهینه شدن کد و جلوگیری از عدم کپی شدن دوباره آرایه، صرفا پوینتر اشاره کننده به آن را برای آبجکت جدید قرار داده و برای اینکه مقدار موجود در آن پاک نشود، مقدار آن را پس از کپی کردن در متغیر مورد نظر (آرایه مقصد)، برابر nullptr قرار می دهیم. به این ترتیب از یک کپی شدن بیهوده جلوگیری شد. فقط این کار در مواردی استفاده می شود که دیگر نخواهیم از آرایه ی r-value که روی هواست استفاده نماییم. (کاربرد دیگر در تعویض دو

وکتور و استفاده از متغیر تمپ است.)

در ادامه به عنوان تمرین بیشتر برخی اپراتورهای دیگر از جمله + و » را نیز نوشته و برای اپراتور مساوی نیز نسخه move را نوشتیم.

۳ سوال سه

۴ سوال پایگاه داده

روند حل این سوال کاملا مشابه آنچه در متن صورت سوال آمده است انجام می شود و به راحتی با استفاده از کدهای آموخته شده در کارگاه اول و دوم می توان تمام پایگاه داده را پیاده نمود.

توجه شود که از بعضی اسامی به دلیل اینکه اسامی کلیدی خود پایگاه داده PostgreSQL هستند، نمی توان استفاده نمود. برای همین آنها به صورتی که نوشتیم که حرف اول آنها بزرگ باشد. برای مشخص نمودن این موضوع اسم را در کوتیشن مارک قرار دادیم.

همانطور که در کوئری درست کردن جدول ها هم آمده است کلید اولیه و خارجی را به گونه ای استفاده می نماییم که برای کلید اولیه، هیچ دو ستونی یکی نشود. مثلا برای بلاک، باید هر دو طرف جز این کلید باشند تا یک سطر متمایز ایجاد شود. یا مثلا برای یک کانال چون ممکن است اعضای دیگری هم داشته باشد یا یک فرد عضو چند کانال باشد باید هر دو را برای جدول اعضای هر کانال در نظر گرفت. برای گروه ها و سایر جداول نیز به همین ترتیب انجام می شود.

برای کوئری هم به گونه ای داده ها را اضافه نمودیم تا به نحو مطلوب همه ی کوئری ها حتما یک پاسخ داشته باشند. و این اتفاق برای همه افتاد. البته نوشتن کوئری ها کمی مشکل بود و نیاز به فکر بود.

۵ ارسال روی گیت

برای ارسال روی گیت، مطابق آنچه در کلاس گفته شد ابتدا یک ریپازیتوری[۳] به نام درس برنامه سازی پیشرفته درست کردیم و سپس در سیستم لوکال خود آن را دانلود کرده و فایل های مورد نظر را با پوشه بندی مناسب در دایرکتوری ایجاد شده قرار دادیم. با استفاده از دستور

```
git add DIRECTORYofFILES
```

فایل ها را اضافه کرده و سپس با دستور

```
git commit -m "MYCOMMENT"
```

[۳] Repositoriy.

آنها را روی سیستم لوکال کامیت یا ثبت می کردیم. حال با دستور

```
git push origin master
```

فایل های کامیت شده را روی سرور github.com و روی برنچ مستر بارگذاری نمودیم.

توجه نمایید که در ابتدا برای گرفتن ریپو از دستورهای ذیل به ترتیب استفاده نمودیم. (از طریق ssh)

```
git init
git clone git@github.com:SRHashemirad/AUT_AP_course.git
```

همچنین قابل ذکر است که فایل `.gitignore` را به به صورتی قرار می دهیم که فایل هایی که در حین کامپایل ایجاد می شوند[۴] را ثبت نکرده و همچنین نسخه هایی که نرم افزار emacs ایجاد می کند[۵] را نیز ثبت ننماید.

شما می توانید برای دستیابی به فایل ها در گیت هاب از این لینک اقدام نمایید.

با تشکر فراوان از حسن دقت نظر و توجه شما

[۴] *.o

[۵] *.*~