به نام خدا

آموزشکده ملی مهارت میناب

تهرینات بخش سه ۳

نام و نام خانوادگی : فردین صداقت

واحد درسی : مباحث ویژه

رشته : مهندسی کامپیوتر

مدرس : مهندس محمد احمدزاده

فروردین : ۱۴۰۴

Data structures and ALGorithms بخش سوم

A - List و Array چه تفاوتی با هم دارند؟

استفاده از لیست ها ساده تر است. لیست ها به سادگی با قرار دادن یک دنباله از عناصر در براکت ایجاد می شوند. از طرف دیگر، ایجاد آرایه به یک تابع خاص از ماژول آرایه (یعنی array.array) یا پکیج NumPy (یعنی numy.array) نیاز دارد. به همین دلیل، لیست ها بیشتر از آرایه استفاده می شوند.

آرایه ها می توانند برای داده های باینری، رابط فشرده و ذخیره کش و برای ذخیره مقدار زیاد داده کارآمدتر هستند.

آرایه ها برای عملیات عددی عالی هستند. لیست ها نمی توانند مستقیما عملیات ریاضی انجام دهند.

B - Dictionary در Python چگونه کار می کند؟

دیکشنری در پایتون یک مجموعه نامرتب از عناصر است که برخلاف سایر داده ساختارها که دنبالگی مقدار را به عنوان عنصر نگه می دارد، برای ذخیره جفت های کلید - مقدار استفاده می شود. کلید-مقدار در دیکشنری برای بهینه سازی بیشتر اجرا می گردد. در دیکشنری کلید ها باید یکتا باشند. برای تغییر مقدار دیکشنری، باید این کار را از طریق کلید انجام دهید. بنابراین ابتدا به کلید دسترسی پیدا کرده و مقدار آن را تغییر دهید.

C - List و Tuple چه تفاوتی با هم دارند؟

در زبان برنامه نویسی پایتون لیست (List) و تاپل (Tuple) دو نوع داده ساختاری هستند که در ادامه تفاوت های آنها را توضیح می دهیم:

1- تغییر پذیری:

لیست ها در پایتون تغییر پذیر هستند به این معنی که می توان در آنها عناصر را اضافه، حذف یا تغییر داد. اما تاپل ها تغییر ناپذیر هستند و بعد از ایجاد نمی توان عناصر آن را به طور مستقیم تغییر داد.

2- پرفرمانس:

با توجه به اینکه تاپل ها تغییر ناپذیر هستند، به طور کلی در پرفامنس سریع تر از لیست ها هستند. به عبارت دیگر تاپل ها از نظر سرعت دسترسی به عناصر، از حافظه بهینه تری استفاده می کنند.

3- نحوه دسترسی:

تاپل ها با استفاده از پرانتز تعریف می شوند، اما لیست ها با استفاده از کروشه [] تعریف می شوند.

4- استفاده:

از لیست ها در مواردی که نیاز به تغییر پذیری داده ها داریم، مثل لیست کردن داده ها، اضافه کردن و حذف عناصر داده، و غیره استفاده می شود.

اما تاپل ها معمولاً در مواردی که نیاز به دسترسی سریع داده ها و عدم تغییر پذیری داده ها داریم، مانند انتقال داده ها به توابع یا بازگشت مقادیر از توابع استفاده می شوند.

5- Set در پایتون چرا برای حذف داده های تکراری استفاده می شود؟

مجموعه به عنوان کلکسیونی از عناصر یکتا استفاده می شود که از ترتیب خاصی پیروی نمی کند. مجموعه ها زمانی مورد استفاده قرار می گیرند که وجود یک شی در مجموعه از اشیا مهمتر از تعداد دفعات ظاهر شدن یا ترتیب اشیاء باشد. مجموعه ها اگر داده ها بیش از یکبار تکرار شوند، فقط یکبار در مجموعه وارد می شوند برخلاف تاپل ها و مجموعه ها قابل دستیر هستند؛ یعنی می توان آنها اصلاح، اضافه، جایگزین و یا حذف کرد.

برای افزودن یک عنصر می توان از تابع ()add استفاده کرد.

مجموعه ها از عملیات پرکاربرد مجموعه های ریاضی مانند اجتماع، اشتراک و غیره پشتیبانی می کنند.

Stack و Queue چه تفاوتی با هم دارند؟

صف (Queue) و پشته (Stack) دو نوع ساختار داده انتزاعی هستند که برای ذخیره و مدیریت داده ها در برنامه نویسی استفاده می شوند.

تفاوت اصلی بین این دو داده ساختار در رویه وارد شونده و خارج شونده از داده ساختار است:

پشته (Stack): رویه اضافه کردن

Push: در پشته اطلاعات یا المان ها تنها از یک طرف اضافه می شوند (معمولا بالا یا پایین) اضافه می شوند این عمل با نام (Push) شناخته می شود.

رویه خارج کردن (Pop): عناصر همان طرفی که اضافه شده اند حذف می شوند این عمل با نام Pop شناخته می شود.

این داده ساختار از اصطلاح (Last in, First out) یا به اختصار Lifo استفاده می شود.

صف (Queue): رویه اضافه کردن

Enqueue :

در صف معمولاً اطلاعات یا المان ها از یک طرف (معمولاً پشته ای که به عنوان "پشته پایین" نامیده می شود) اضافه می شوند.

رو به خارج کردن : Dequeue

حذف عناصر از طرف مقابل از آنجایی که عناصر از یک طرف اضافه می شوند. این عمل با نام "Dequeue" شناخته می شود

این داده ساختار از اصطلاح "First in, First out" (اولین وارد شده اولین خارج می شود) یا به اختصار Fifo استفاده می کند

به طور کلی، اصلی ترین تفاوت بین پشته و صف در روبه اضافه و حذف المان هاست.

که در پشته این دو عملیات از یک طرف انجام می شوند و در صف از دو طرف مختلف.

Hash table - F چیست و چرا کاربرد دارد؟

مش دیبل یکی از ساختارهای داده های اساسی است که استفاده های زیادی در ترسیم نرم افزار دارد. یکی از دلایل اصلی محبوبیت مش دیبل سرعت بالای آن در ثبت و خواندن داده هاست. پیچیدگی زمانی مش دیبل برای ثبت و خواندن داده به صورت O(1) است و این نشان می دهد که مش دیبل یکی از بهترین گزینه ها برای زمان هایست که با حجم زیادی از داده ها سروکار داشته باشیم. مش دیبل از ترکیب آرایه ها و تابع مش برای مدیریت داده ها استفاده می کند.

علل استفاده

1- سرعت بالا در درج و حذف داده

2- جست و جوی سریع بین داده ها

3- استفاده در دیکشنری ها و دیتا بیس ها

4- استفاده برای رمزگذاری کردن داده ها و بالا بردن امنیت

clips...

Binary Tree و B-Tree چه ساختمان داده‌ای هستند؟

Binary Tree (درخت دودویی):

درخت دودویی یک ساختار داده‌ای است که در آن هر گره حداکثر دو فرزند دارد. این دو فرزند معمولاً به عنوان فرزند چپ و فرزند راست شناخته می‌شوند.

درخت جستجوی دودویی (Binary Search Tree (BST)) یکی از پرکاربردترین الگوریتم‌ها و ساختمان داده‌ها در حافظه اصلی (RAM) است.

B-Tree (درخت B):

درخت B یک ساختار داده خود-متعادل (self-balancing) است که برای نگهداری داده‌های مرتب شده استفاده می‌شود و امکان جست‌وجو، درج و حذف بهینه را در زمان لگاریتمی فراهم می‌کند. این درخت معمولاً در سیستم‌های پایگاه داده و سیستم‌فایل‌ها کاربرد دارد.

- هر گره می‌تواند بیش از دو فرزند داشته باشد.
- برای عملکرد بالا در فضای دیسک طراحی شده است.

Graph Data Structure چیست و برای شبکه‌های اجتماعی استفاده می‌شود؟

گراف داده‌ساختار کلی است که به منظور مدل‌سازی مجموعه‌ای از اشیاء و ارتباطات مابین آن‌ها مورد استفاده قرار می‌گیرد و به‌راستی شاید ریاضیاتی $G(V, E)$ نمایش داده شده است.

یک مثال کاربردی از گراف می‌توان مدلسازی روابط مابین افراد در شبکه‌های اجتماعی را نام برد که تک‌تک افراد در آن به منزله رئوس گراف در نظر گرفته شده و ارتباطات مابین ایشان نیز توسط خطوطی نمایش داده می‌شوند که دو یا چند رأس از گراف مزبور را به هم وصل کرده و تحت عنوان یال شناخته می‌شود.

۱- Dynamic Programming چرا برای حل مسائل پیچیده کلا بهره داد؟

برنامه نویسی پویا که به آن DP نیز گفته می‌شود، تکنیک الگوریتمی ای برای حل مسائل بهینه سازی است و اینکار را با شکستن یک مسئله‌ی بزرگ به مسائل کوچکتر و ساده‌تر انجام می دهد. در اصل DP یک بهینه سازی روش بازگشت (Recursion) است.

برای حل اینگونه مسائل، راه حل بهینه مسئله‌ای اصلی به راه‌حل های بهینه برای مسائل کوچکتر بستگی دارد.

ایده پشت این الگوریتم که باعث بهینه‌سازی تو در تو بازگشتی می‌شود خیلی ساده است. در واقع این الگوریتم، ما را به نتایج بهتر آمده از مسائل کوچکتر رهنمون کرد تا مجبور نباشیم آنها را چندین بار حساب بکنیم.

۲- Recursion چیست و چرا در الگوریتم های پیشرفته استفاده می‌شود؟

الگوریتم بازگشتی در زبان برنامه نویسی است که در آن یک تابع برای حل مسائل، خودش را با ورودی های کوچکتر فراخوانی می کند. این فرایند تا زمانی ادامه می یابد که مسئله به یک نقطه پایه (Base case) برسد. جایی که دیگر نیازی به بازگشت نباشد. نتیجه مستقیما قابل محاسبه می باشد. در هر مرحله مسئله اصلی به یک یا چند زیر مسئله‌ی کوچکتر شکسته می شود، و پس از حل آنها، پاسخ ها ترکیب داده شده و نتیجه نهایی ساخته می شود.

الگوریتم های بازگشتی به طور گسترده در مسائل ترکیباتی، پردازش داده های سلسله مراتبی و الگوریتم های جست و جو، مرتب سازی استفاده می شوند. از آنجایی که این روش باعث ساده‌تر شدن کد و افزایش خوانایی آن می شود، در بسیاری از موارد ترجیح داده می شود. اما در حین حال، بازگشت می تواند منجر به افزایش مصرف حافظه و کاهش کارایی شود، به ویژه اگر بدون بهینه سازی مناسب اجرا گردد.