نام و نام خانوادگی : مرضیه نجفی _هدا مهرانی پور

نام استاد : محمد احمدزاده

موضوع : بخش ۳ Data structures and Algorithms

تاریخ : ۱۴۰۴/۰۱/۲۰

list و array چه تفاوتی دارند ؟

Arrayها در سی شارپ نسبت به List ها کمتر مورد استفاده قرار می گیرند، اما بسیار سریع تر هستند. ما می توانیم عناصر یک آرایه را با استفاده از index ها جستجو کنیم. اما لیست نیاز به(iteration تکرار (دارد، این به معنای کندی زمان جستجو برای لیست است. بر خلاف لیست ها، آرایه ها به حافظه کمتری نیاز دارند

dictionary در python چگونه کار می کند؟

در پایتون، دیکشنری (dictionary) یک ساختار داده است که به شما امکان می‌دهد تا داده‌ها را به صورت جفت‌های "کلید–مقدار" (key-value pairs) ذخیره کنید. این ساختار داده، مشابه یک لغت‌نامه است که در آن هر کلمه (کلید) با یک معنی (مقدار) مرتبط است.

list و tuple چه تفاوتی دارند؟

تفاوت‌های اصلی بین لیست (List) و تاپل (Tuple) در پایتون

- لیست: برای ذخیره مجموعه‌ای از داده‌ها که نیاز به تغییر دارند (مثل اضافه یا حذف عنصر) مناسب است.
- تاپل: برای ذخیره مجموعه‌ای از داده‌ها که نباید تغییر کنند (مثل مختصات یک نقطه) مناسب است. همچنین، به دلیل سرعت و مصرف حافظه کمتر، در مواردی که کارایی مهم است (مثل استفاده به عنوان کلید در دیکشنری‌ها) استفاده می‌شود.

set در python چرا برای حذف داده های تکراری استفاده می شود؟

`set` در پایتون به دلیل داشتن دو ویژگی اصلی برای حذف داده‌های تکراری استفاده می‌شود:

۱.عناصر یکتا (Unique elements**) :** set **فقط می‌تواند عناصر یکتا را در خود نگه دارد. وقتی شما یک لیست یا هر دنباله دیگری را به یک** `set` **تبدیل می‌کنید،** `set` **به طور خودکار عناصر تکراری را حذف می‌کند و فقط یک نسخه از هر عنصر را نگه می‌دارد.**

۲.Unordered **بودن: ترتیب عناصر در** `set` **مهم نیست. این ویژگی به** `set` **اجازه می‌دهد تا عملیات مربوط به حذف تکراری‌ها را با سرعت بالاتری انجام دهد.**

stack و queue چه تفاوتی دارند؟

تفاوت اصلی بین Stack **(پشته) و** Queue **(صف) در نحوه اضافه و حذف عناصر است. هر دو یک نوع ساختمان داده هستند که برای ذخیره و بازیابی اطلاعات استفاده می‌شوند، اما روش عملکردشان متفاوت است:**

۱. Stack **(پشته):**

اصل عملکرد : (LIFO (Last In, First Out**) – آخرین ورودی، اولین خروجی. مثل یک پشته بشقاب که آخرین بشقابی که روی پشته قرار می‌گیرد، اولین بشقابی است که برداشته می‌شود.**

عملیات اصلی:

- Push **: اضافه کردن یک عنصر به بالای پشته.**

- Pop **: حذف کردن عنصری از بالای پشته.**
- Peek **: دیدن عنصر بالای پشته بدون حذف آن.**

۲. Queue **(صف):**

اصل عملکرد: (FIFO (First In, First Out **– اولین ورودی، اولین خروجی. مثل یک صف نانوایی که اولین نفری که در صف قرار می‌گیرد، اولین نفری است که نان را دریافت می‌کند.**

عملیات اصلی:

- Enqueue **: اضافه کردن یک عنصر به انتهای صف.**
- Dequeue **: حذف کردن عنصری از ابتدای صف.**
- peek`/`front **: دیدن عنصر ابتدای صف بدون حذف آن.**

hash table چیست و چرا کاربرد دارد؟

Hash Table **(جدول درهم‌سازی) چیست؟**

Hash Table **یک ساختمان داده است که برای ذخیره و بازیابی داده‌ها به صورت جفت‌های "کلید–مقدار"** (key-value pairs) **استفاده می‌شود. این ساختمان داده امکان دسترسی بسیار سریع به داده‌ها را فراهم می‌کند، به طور معمول در زمان** O(1) **(میانگین زمان ثابت).**

چرا Hash Table کاربرد دارد؟

Hash Table **به دلیل ویژگی‌های زیر بسیار پرکاربرد است:**

۱. سرعت دسترسی

۲. کارایی در جستجو، درج و حذف

۳. انعطاف‌پذیری

۴. کاربردهای گسترده

binary tree و b-tree چه تفاوتی دارند؟

تفاوت بین درخت دودویی (Binary Tree) و درخت بی–تری (B-tree)

درخت دودویی (Binary Tree) و درخت بی–تری (B-tree) دو نوع مختلف از درخت‌ها در علوم کامپیوتر هستند که در ساختارهای داده‌ای مختلف کاربرد دارند.

تفاوت‌های کلیدی آن‌ها

۱.ساختار و تعداد فرزندان هر گره :

- **درخت دودویی (Binary Tree):**

– هر گره در یک درخت دودویی می‌تواند حداکثر دو فرزند داشته باشد.

– این فرزندان به‌طور معمول با نام‌های «چپ» (left) و «راست» (right) مشخص می‌شوند.

– درخت دودویی می‌تواند گاهی اوقات به شکل‌های مختلفی باشد: درخت دودویی جستجو، درخت دودویی متوازن، درخت دودویی کامل، و غیره.

- **درخت بی–تری (B-tree) :**

– درخت بی–تری یک درخت چندراهی است که به‌طور معمول در پایگاه‌داده‌ها و سیستم‌های فایل برای جستجو، درج و حذف سریع داده‌ها استفاده می‌شود.

– در این درخت، هر گره می‌تواند بیش از دو فرزند داشته باشد. تعداد فرزندان هر گره بستگی به تعداد کلیدهای موجود در گره دارد.

– به طور کلی، درخت بی–تری به‌گونه‌ای طراحی شده است که تعداد زیادی کلید و گره‌ها در هر سطح از درخت قرار گیرد تا عملکرد جستجو در حجم‌های بالای داده بهینه شود.

چرا graph data structure برای شبکه های اجتماعی استفاده می شود؟

ساختار داده‌ی گراف (Graph) به دلایل متعددی برای مدل‌سازی و مدیریت شبکه‌های اجتماعی مناسب است:

۱. مدل‌سازی روابط : شبکه‌های اجتماعی اساساً مجموعه‌ای از افراد (یا نهادها) و روابط بین آن‌ها هستند. گراف یک مدل طبیعی برای نمایش این روابط است. در یک گراف:

- **افراد (یا نهادها) به عنوان گره (Node) یا راس (Vertex) نمایش داده می‌شوند.**
- **روابط بین افراد (مانند دوستی، دنبال کردن، همکار بودن) به عنوان یال (Edge) نمایش داده می‌شوند.**

۲. انعطاف پذیری: گراف‌ها می‌توانند انواع مختلفی از روابط را مدل کنند. یال‌ها می‌توانند:

- **جهت‌دار** (Directed)**: مثلاً "الف، ب را دنبال می‌کند"** (Follow)
- **بدون جهت** (Undirected)**: مثلاً "الف و ب دوست هستند"** (Friend)
- **وزن‌دار** (Weighted)**: برای نشان دادن میزان نزدیکی یا قدرت رابطه (** Strength of relationship)

۳. تحلیل شبکه: گراف‌ها ابزارهای قدرتمندی برای تحلیل شبکه‌های اجتماعی فراهم می‌کنند:

- **یافتن دوستان مشترک: شناسایی افرادی که با دو نفر دوست هستند.**
- **پیشنهاد دوست: پیشنهاد افراد جدید برای دوستی بر اساس ارتباطات موجود.**
- **شناسایی افراد تاثیرگذار: پیدا کردن گره‌هایی که بیشترین ارتباط را دارند** (Centrality)**.**
- **تحلیل جوامع** (Community Detection)**: شناسایی گروه‌هایی از افراد که ارتباط نزدیکی با هم دارند.**
- **مسیریابی: یافتن کوتاه‌ترین مسیر بین دو نفر در شبکه.**

۴. مقیاس‌پذیری: با وجود چالش‌هایی که در مدیریت گراف‌های بزرگ وجود دارد، الگوریتم‌ها و تکنیک‌های مختلفی برای پردازش گراف‌های بسیار بزرگ توسعه یافته‌اند.

۵.نمایش بصری: گراف‌ها به راحتی قابل نمایش و تجسم هستند، که به درک بهتر ساختار و روابط شبکه کمک می‌کند.

dynamic programming چرا در حل مسائل پیچیده کاربرد دارد؟

برنامه‌نویسی پویا (Dynamic Programming - DP**) یک روش طراحی الگوریتم است که برای حل مسائل پیچیده با شکستن آن‌ها به زیرمسئله‌های همپوشان (** overlapping subproblems**) و ذخیره‌سازی نتایج این زیرمسئله‌ها برای استفاده مجدد استفاده می‌شود.**

برنامه‌نویسی پویا به دلیل جلوگیری از محاسبات تکراری، تقسیم مسئله به زیرمسئله‌های ساده‌تر، استفاده از اصل بهینگی، قابلیت پیاده‌سازی آسان و کاربرد گسترده، یک روش بسیار قدرتمند برای حل مسائل پیچیده است.

recursion چیست و چرا در الگوریتم های پیشرفته استفاده می شود؟

بازگشت (Recursion**)**

بازگشت یک تکنیک برنامه‌نویسی است که در آن یک تابع خودش را برای حل یک مسئله فراخوانی می‌کند. این رویکرد معمولاً شامل دو بخش است:

۱.حالت پایه (Base Case**): شرایطی که تابع بدون فراخوانی مجدد خودش پاسخ می‌دهد.**

۲. حالت بازگشتی (Recursive Case**): جایی که تابع خودش را با ورودی‌های کوچکتر فراخوانی می‌کند.**

چرا بازگشت در الگوریتم‌های پیشرفته استفاده می‌شود؟

۱. سادگی و خوانایی : کد بازگشتی معمولاً ساده‌تر و قابل درک‌تر است.

۲. مدل‌سازی مسائل به طور طبیعی : مسائل با ساختار درختی یا گرافی به راحتی با بازگشت حل می‌شوند (مانند جستجوی عمق اول).

۳. تقسیم و حل: می‌تواند مسائل را به زیرمسئله‌های کوچکتر تقسیم کند و راه‌حل‌های آن‌ها را ترکیب کند (مانند الگوریتم‌های مرتب‌سازی).

۴. مدیریت حافظه : به طور خودکار از پشته فراخوانی برای مدیریت وضعیت توابع استفاده می‌کند.

۵. سیستم‌های پیچیده : در حل مسائل پیچیده (مانند مسائل ترکیبیاتی یا برنامه‌نویسی پویا) کاربرد دارد.