هنگام نوشتن الگوریتم ها باید به دو موضع پیچیدگی زمان – پیچیدگی فضا توجه کرد.

هر دو به ترتیب با time and space شناخته می شوند( در دنیای الگوریتم ها )

(**time complexity** and **space complexity**)

برای محاسبه این پیچیدگی از مفهومی تحت عنوان **big O notation** استفاده خواهیم کرد که در ادامه به آن می پردازیم.

Constant time

پیچیدگی این مدل الگوریتم ها همیشه ثابت می باشند چرا که با افزایش داده ها زمان اجرای برنامه تغییری نخواهد کرد و همواره یک مقدار ثابت هست. این الگوریتم ها دارای پیچیدگی o(1) می باشند و از سریع ترین الگوریتم ها به حساب

می آیند.

O(1)یک نماد برای الگوریتم های ثابت می باشد که همواره به همین شکل مورد استفاده قرار می گیرد.

Logarithm Time O(log n )

در مفهوم هر بار اجرای الگوریتم مدت زمان اجرای آن نصف می شود ( به علت لگاریتمی بودن آن )

همواره در حال کاهش است ولی هرگز صفر نخواهد شد.

Linear Time O(n)

به الگوریتم هایی گفته می شود که مقدار operation با مقدار ورودی به شکل مستقیم ارتباط دارد، یعنی هرچقدر که میزان ورودی افزایش یابد به همان میزان پیچیدگی زمانی نیز افزایش پیدا خواهد کرد.

در عبارت بالا n نمادی از ورودی های شما می باشد و الزاما یک عدد ورودی نیست!!

شکل این پیچیدگی یک خط راست است که از مبدا محور مختصات عبور می کند.

Polynomial O(n \*\* n )

عبارت بالا نمادی از توان می باشد. که در واقع عبارت n به توان nدومی خواهد رسید. حالا این دومی می تواند هر عددی باشد که بستگی به اجرای الگوریتم دارد.( اولی ثابت دومی عدد )

مثلا حلقه های تو در تو می توانند مثالی برای این مدل پیچیدگی زمانی باشند.(bubble sort )

با افزایش توان همواره شیب این نمودار بیشتر خواهد شد( افزایش زمان )

Exponential O (n \*\* n )

نسبت به پیچیدگی قبلی تفاوتی که دارد این است که n اولی حاوی مقدار می باشد و n دومی همواره ثابت است.

یعنی یک عددی به توان n خواهد رسید.

نمایش این مدل تابع به شکل نمایی می باشد.

به رمز نگاری در دنیای برنامه نویسی cypher گفته می شود. در واقع یک متد رمز نگاری می باشد که قبلا مورد استفاده قرار می گرفته است.

یک مدل خاصی از رمز گذاری به رمز گذاری سزار معروف است : به این صورت است که همه حروف به مقدار مشخصی به سمت جلو shift پیدا می کنند. البته جمله ای که رمز گذاری شده اصلا قابل خواندن نیست.

نکته : اگر از کتاب خانه string پایتون ascii\_letters را وارد برنامه خود کنیم می توانیم تمامی حروف را مشاهده کنیم.

الگوریتم isomorphic مربوط به بررسی تقارن دو رشته می باشد. اگر تقارن برقرار بود True و اگر تقارن برقرار نبود False بر می گرداند.

باید توجه داشت که متقارن بودن به معنی برابر بودن نمی باشد بلکه باید یک نسبتی بین آنها رعایت شود.

هر کاراکتری در زبان های برنامه نویسی دارای یک unicode می باشد که به وسیله دستور ord() در پایتون می توان به آن یونیکد مخصوص دسترسی پیدا کرد.

برعکس این عمل نیز می توان انجام داد به این شکل که با گرفتن یونیکد، خود کاراکتر مربوطه را به ما نمایش دهد.

از یونیکد ها می توان برای الگوریتم های رمز نگاری استفاده کرد برای مثال مدل رمزگذاری a1z26 از یونیکدها استفاده می کند.

متد zip()

با استفاده از متد zip می توان دو لیست، تاپل یا هر چیز قابل تکراری را در کنار هم قرار داد.و در واقع آنها را zip کرد ولی باید توجه داشت که می توان تعداد بیشتری را نیز وارد کرد مثلا 3 یا 4 لیست و ...

نکته مهم این است که object های ورودی حتما باید iterable باشند.

خود متد zip قابل خواندن نیست و به صورت یک object در خروجی نمایش داده می شود، ولی می توان آنها را با تبدیل به انواع متد های قابل نمایش مانند لیست مشاهده کرد.

متد Enumerate()

Enumerate یک متد برای شمارش است که به طور پیش فرض از صفر شروع به شمارش شی های تکرار پذیر می کند ولی می توان با start = “” یک عددی را به عنوان شروع انتخاب کرد.

این متد مانند zip خود قابل خواندن نیست و باید به شکل یک لیست یا یک حلقه تکرار شونده بر روی آن خوانده شود.

Generators in python

نقطه مقابل فانکشن های معمولی می باشند.

یک روش ساده تر برای ساختن object های تکرار پذیر می باشند.

بر خلاف فانکشن ها در اینجا ما به جای کلمه return از کلمه **yield** استفاده می کنیم.

در واقع این یک کلمه کلیدی برای generator ها می باشد.

Generator ها در واقع iterable هستند و هنگامی که به شکل مستقیم صدا شوند یک object برگردانده می شود و نتیجه خاصی در خروجی نمایش داده نمی شود به همین دلیل می توانیم بر روی آنها حلقه بزنیم یا کارهایی که با شی های تکرار شونده انجام می شود همگی برای آنها قابل انجام است.

در این فانکشن ها yield ها چندین بار می توانند استفاده شوند. (بر خلاف return )

بر روی توابع ساخته شده توسط generator می توان حلقه زد و به محتوای آن دسترسی پیدا کرد.

با متد next() می توان در generator ها تک تک پیمایش انجام داد.

به طور کلی می توان generator ها را به مفهوم lazy evaluation نسبت داد به این معنی که تا زمانی مجبور نباشند به ما مقداری را بر نمی گردانند.

استفاده از generator ها می تواند منجر به صرفه جویی در منابع مصرفی در برنامه خود شود.( قابل استفاده در دیتاهای بزرگ )

نکته : با استفاده از متد extend() می توانیم به انتهای لیست خود هر مقداری را که میخواهیم اضافه کنیم.

روش جست و جوی sequential search همان linear search می باشد که به صورت خطی در یک لیست به جست و جوی مقدار خواسته شده می پردازد.

در روش binary search ما آرایه را تقسیم می کنیم و در نصف آرایه به شکل فراکتالی جست و جو انجام می دهیم.