**ندا جمالی**

**9631813**

**طراحی الگوریتم**

**گزارش پروژه**

در ابتدا همه ی کلمات که در یه فایلی هستند رو که خط به خط جدا شده اند ، خط به خط این کلمات را می‌خوانیم و داخل یک مجموعه ای که هر خط اون فایل میشه یک عضو اون مجموعه ما میریزیم.

فقط حواسمان باید باشد که از stripe استفاده میکنیم که اگر فاصله ای یا / سر و ته کلمات باشد آنها حذف بشوند.

برای چه داخل مجموعه میریزیم؟ برای اینکه داخل مجموعه ریختن در واقع سرعت lock up اش بیشتر است در واقع چون حشو میشه در مجموعه سرعت O(1) در حالی که توی لیست O(n) است.

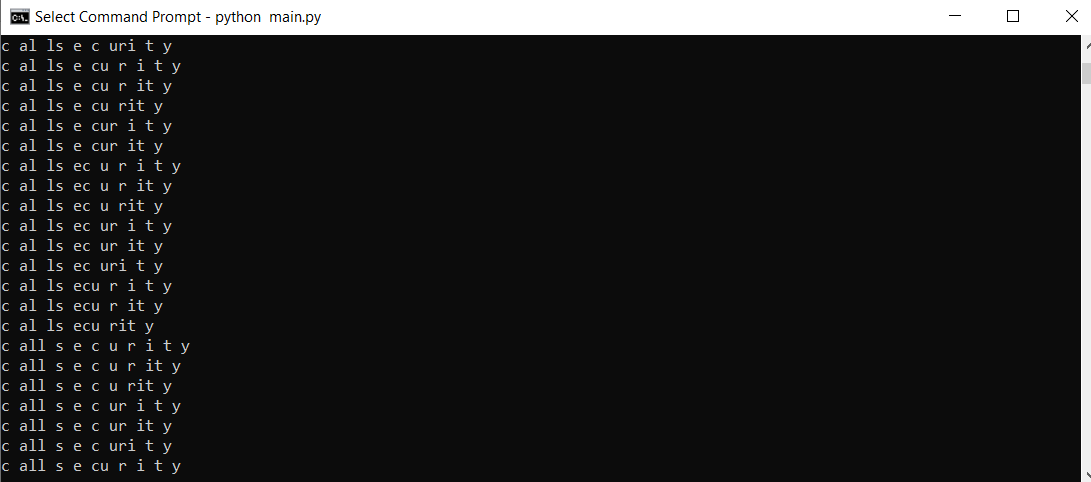
تابع دیکودر هم بازگشتی کار میکند اولین کارمان این است که هر ورودی که بهش میدهیم رو lower میکند یعنی حروف بزرگ باشد حروف کوچیک میکند بعد میایم دونه به دونه از رشته ی اصلیمون برمی‌داریم میچسبونیم به یک prefixes ای که اولش تُهی است. یکی یکی کاراکتر هارو برمی‌داریم به آن میچسبانیم.

بعد هر دفعه این prefixes شامل کلمه ی درست باشد دوباره همین تابع رو به ازای خودش صدا میزنیم به ازای اون رشته ی باقی مونده به جز اون prefixes چون اون یا کلمه ی درست بود وآن را میزاریم کنار به عنوان جواب و دوباره همین تابع رو به ازای رشته ی باقی مونده صدا میزنیم. ولی آن روند پیدا کردن prefixes توی تابع قبلی همچنان ادامه دارد چون ممکن است بریم جلو تر و آن کلمه کوچیک است یه تیکه ای از کلمه بزرگتر باشد مثلا در همین مثالی که خودش داده بود Airport هم Air یه کلمه است هم Airport .

پس وقتی یه بار Air رو پیدا کردیم به لیستمون اضافه میکنیم و دوباره تابع رو صدا میزنیم تا Airport رو هم اضافه کنیم و دوباره تابع رو صدا میزنیم تا ادامه ی کلمات هم پیدا بشوند و در آخر هم به ازای هر لیستی گرفتیم اگر تا اخر رفتیم و آخرین prefixes که داشتیم جز یه کلمه درستی بود اون prefixes رو هم چاپ میکنیم و کل اون نتیجه ای هم که به ازای اون دنباله تابع ها صدا زده شده آن را هم چاپ میکنیم .

**(اجرای برنامه)**

**ورودی:callsecurity**



**ورودی:callme**

