به نام خدا

**پروژه ژنتیک**

**محمدرضا شش پری**

**8101198417**

## هدف انجام پروژه

آشنایی با الگوریتم ژنتیک و حل مساله رمزگشایی با استفاده از آن

## تعریف مساله

همانطور که می دانیم، در الگوریتم رمزگشایی ما با استفاده از یک کلید، می توانیم متنی را رمزگذاری یا متنی رمزشده را رمزگشایی کنیم.

حال در این پروژه به ما یک متن رمزگذاری شده به همراه یک متن که شامل تمام کلمات مجاز است، داده شده و ما باید با استفاده از آن متن یک دیکشنری ساخته و با استفاده از الگوریتم ژنتیک، ابتدا کلید رمز را یافته و در آخر متن را رمزگشایی کنیم.همچنین طول کلید رمز بصورت پیشفرض برابر با 14 می باشد ولی می توان آن را تغییر داد.

همچنین لازم به ذکر است که دراین پروژه تنها کاراکتر های شامل حروف 26 گانه انگلیسی رمزگذاری می شوند و بقیه کاراکتر ها به همان صورت ترجمه می شوند.

## خواندن ورودی

در ابتدا فایل های متنی global_text و encoded_text را خوانده و در دو رشته ذخیره می کنیم.

حال از آنجایی که باید کلمات را global_text استخراج کنیم و همچنین این فایل شامل کاراکترهای ناخواسته است، با استفاده از regex ، globalText را به کلماتی که از حروف متوالی انگلیسی تشکیل شده اند، می شکنیم و حروف بدست آمده را در یک دیکشنری با کلید طول رشته ورودی و مقادیر تمام کلمات با آن طول ذخیره می کنیم.

توجه : برای خواندن فایل ها از utf-8-sig استفاده کردیم تا یک سری کراکتر خوانده شوند و در روند خواندن ورودی رمزگشایی اختلال بوجود نیاورند.

## خواندن ورودی

همچنین از فایل encodedText نیز یک کپی شامل کلماتی که فقط حاوی حروف انگلیسی هستند گرفتیم تا در مراحل بعدی و برای رمزگشایی ،از آن استفاده کنیم.

## مدلسازی و تعریف ها

ژن و کروموزوم: یک ژن را یکی از حروف انگلیسی(lowercase) در نظر گرفته و کروموزوم را رشته ای از ژن ها به طول key تعریف کردیم.

توجه:در تمامی فرآیند های حل مساله به جز مرحله آخر (چاپ رمز) تمام کاراکتر ها را lowercase درنظر می گیریم.

جمعیت اولیه: لیستی به طول 100 کروموزوم که ژن های کروموزوم آن بصورت رندم انتخاب شده اند.

تابع fitness : این تابع یک کروموزوم را به عنوان ورودی گرفته و خروجی آن ، **تعداد** کلماتی از متن رمگذاری ورودی است که بعد از رمزگشایی، در دیکشنری وجود دارند. این مقدار عددی بین 0 تا تعداد کلمات متن ورودی رمزگذاری شده است و هر چه مقدار آن بالاتر باشد به معنای سازگاری بیشتر است.

تابع crossover : این تابع دو کروموزوم(به همراه مقادیر سازگاری شان) را گرفته و تعداد 3 ژن رندم از آن دو کروموزوم را انتخاب کرده و با هم جابجا میکند ودر صورت افزایش مقدار فیتنس، دو کروموزوم جدید و در غیراینصورت دوکروموزوم ورودی را خروجی می دهد.

همچنین تعداد آن 3 ژن میتواند کم و زیاد (بوسیله متغیر NUM_CROSS_INDEX ) شود.

تابع mutation : این تابع یک کروموزوم ورودی گرفته و برای یک ژن رندم از آن ، بهترین حالت را انتخاب می کند. بهترین حالت یعنی یک کاراکتر برای آن ژن انتخاب می کند که مقدار fitness آن حداکثر شود و در آخر کروموزوم ساخته شده (که می تواند همان کروموزوم ورودی باشد) را خروجی می دهد.

## روند کلی برنامه

در ابتدا یک جمعیت اولیه می سازیم و تا زمانی که به جواب مساله نرسیدیم (کروموزومی که فیتنس آن به اندازه تعداد کلمات متن ورودی باشد) ، این جمعیت را بصورت نزولی و برحسب مقدار فیتنس سورت می کنیم.

سپس 20 درصد کروموزوم اول را بصورت مستقیم و به اندازه 80 درصد تعداد جمعیت را بصورت crossover دو کروموزوم رندم از جمعیت می سازیم.

بعد از ساخته شدن جمعیت جدید، روی 10 درصد رندم از این جمعیت mutation را انجام می دهیم و خروجی حاصل را بعنوان جمعیت جدید در حلقه بعدی استفاده می کنیم.

## خروجی تست کیس


این خروجی بطور میانگین در زمان 30 ثانیه بدست می آید.

## سوالات

1.جمعیت اولیه کم باعث پوشش توزیع قسمت کمی از حالت ها شده و موجب کند شدن روند رسیدن به پاسخ می شود.جمعیت اولیه زیاد هم باعث طول کشیدن روند پردازش داده ها در هر مرحله و کاهش اثر جهش های ژنتیکی می شود.

2. خیلی بستگی به جمعیتی که اضافه می شود دارد. در کد من، سبب کاهش سرعت بخاطر افزایش داده ها و افزایش زمان پردازش داده ها و همچنین سبب افزایش دقت بخاط افزایش تنوع کروموزوم های برتر شد.همچنین تعداد نسل هایی که موجب رسیدن به جواب میشد نیز کمتر شد.

3.crossover برای ترکیب دو کروموزوم و mutation برای ارتقای یک کروموزوم است.ما اگر از crossover استفاده نکنیم، تنوع جمعیتی مان کم می شود و تشابه بین نسل هایمان خیلی زیاد می شود که این می تواند احتمال دیرتر رسیدن به پاسخ مساله را بیشتر کند.همچنین اگر از mutation استفاده نکنیم،احتمال این که به جواب درست نزدیک شویم ولی بطور دقیق به آن نرسیم خیلی زیاد می شود و همچنین می تواند سبب ارتقای زیاد نیافتن هر نسل نسبت به قبلی اش شود.

4.در کد من، mutation دقت را بیشتر بالا می برد، چون همه حالت ها برای یک ژن را بررسی می کند و بهترین از لحاظ افزایش بیشتر fitness را انتخاب می کند ولی خب زمان خیلی بیشتری نسبت به crossover باید برای انجام آن صرف شود.

5. در اعمال crossover ، دو کروموزوم را بصورت رندم انتخاب کردیم و از آنجایی که داخل crossover نیز از رندم برای انتخاب ایندکس ها استفاده می شود، احتمال تغییر نکردن جمعیت خیلی پایین می آید.

6.بنظرم برای جمعیت های کم ، بهتر بود از mutation استفاده شود. چون دقت را بالا میبرد و از آنجایی که تعداد کم است، تاثیر زمان خیلی به چشم نمی آید و این که احتمال جهش ژن ها(بخاطر تعداد کم) بیشتر است.ولی برای جمعیت های زیاد crossover بهتر است ، چون هم در زمان صرفه جویی می شود و هم این که بخاطراحتمال تولید نسل های مشابه کمتر می شود و این احتمال رسیدن به جواب را بیشتر می کند.

7. تعریف دیکشنری اولیه را با توجه به ورودی سوال تغییر دهیم (کلید دیکشنری طول کلمات یا حرف اول کلمات یا ترکیب هر دو باشد.)

بجای انتخاب رندم برای mutation ،فقط نفرات برتر هر نسل را mutation کنیم و این که در هر نسل(نسبت به نسل قبل) جمعیت را کمتر و سازگارترکنیم.

تابع crossover را فقط روی کروموزوم های برتر(و نه بصورت رندم) صدا بزنیم.