

## به نام خدا

علی غفاری ثابت - ۹۸۲۳۹۰۳

### توضیح کد:

- ۱- ابتدا فایل لغات را خوانده و کلمات را در یک دیکشنری ذخیره می کنیم.
- ۲- سپس رشته ورودی را از کاربر دریافت می کنیم و رشته و طول آن را به ترتیب در متغیر های `string` و `n` ذخیره می کنیم.
- ۳- یک آرایه به نام `path` و به طول `n+1` که تمام خانه هایش در ابتدا برابر منفی یک است را ایجاد می کنیم. این آرایه به شکل برنامه نویسی پویا پر می شود. خانه `i` ام این آرایه برابر `z` است که `string[i:z]` یک کلمه صحیح و `string[j:]` یک رشته قابل تبدیل به تعدادی کلمه صحیح است. خانه `n` ام این آرایه برابر `n` است زیرا بازه ای که ایجاد می شود خارج از محدوده رشته است و فقط می تواند یک رشته خالی باشد. (اندیس آرایه از ۰ شروع می شود و `string[i:z]` به معنی خانه `i` ام تا خانه `z` ام رشته است که خانه `i` ام جزو خود بازه و خانه `z` ام خارج از بازه است. `string[j:]` یعنی بازه ای از رشته از خانه `z` ام تا آخر رشته.) بنابراین برای خانه `i` ام یک حلقه برای `z` های بزرگتر از `i` نیاز داریم.
- ۴- در نهایت از خانه `i=0` شروع می کنیم و تا زمانی که `i` مخالف `n` است `string[i, path[i]]` را به همراه `i` فاصله در انتهای آن چاپ می کنیم و `i` را برابر `path[i]` قرار می دهیم.

### پیچیدگی زمانی:

در این الگوریتم از دو حلقه تو در تو که در حلقه بیرونی `i` از `n` تا ۰ یکی یکی کم می شود و حلقه درونی از `n` تا `i+1` یکی یکی کم می شود استفاده شده که پیچیدگی زمانی آن برابر  $\theta(n^2)$  است و همچنین برای چاپ نتیجه متغیر `i` از ۰ تا `n` زیاد می شود که در بدترین حالت (یکی یکی اضافه شدن) آن حلقه `n` بار تکرار می شود که پیچیدگی زمانی آن برابر  $\theta(n)$  است. در نتیجه پیچیدگی زمانی کل برنامه برابر  $\theta(n)$  است. (`n` طول رشته ورودی است.)

- اگر پیچیدگی زمانی عمل قطعه بندی رشته (`string[i:z]`) را  $\theta(n)$  در نظر بگیریم پیچیدگی زمانی حلقه های تو در تو به دلیل اینکه داخل حلقه درونی یک عمل قطعه بندی رشته قرار دارد برابر  $\theta(n^3)$  و در نتیجه پیچیدگی زمانی کل برنامه برابر  $\theta(n^3)$  خواهد بود.

### پیچیدگی حافظه:

به جز طول رشته از یک آرایه `path` به طول `n+1` استفاده کردیم. بنابراین پیچیدگی زمانی برابر  $\theta(n)$  است. (`n` طول رشته ورودی است.)