به نام خدا

گزارش پروژه سوم (باند، جیمز باند!) غزل پوراسفندیار بروجنی (9820453)

قرار است یک رشته حاوی حروف بزرگ انگلیسی را دریافت کرده و آن را به کلمات انگلیسی معنی دار، تفکیک کنیم.

برای این کار، متدهای زیر را نوشته ایم.

```
public static HashMap<String , Boolean> readDictionaryFromFile (){...}

public static boolean isValid(String word){...}

public static void tracingTree(String str, String result) {...}

public static void sentenceBreak(String str){...}
```

- در متد readDictionaryFromFile لغات مورد پذیرش در زبان انگلیسی را از فایل متنی با فرمت txt خوانده و در یک HashMap ذخیره می کنیم. فایل مربوط به لغات معنی دار حاوی حدود 4000 کلمه انگلیسی است که در بین آنها حروف مجزا هم، وجود دارد و اگر هر حرف الفبا به تنهایی یک کلمه با معنا باشد هر رشته ای به حروف تتشکیل دهنده ای تقسیم می شود و در نتیجه همه رشته ها با معنی تلقی می شوند در حالی که ممکن است رشته ای بی معنا مثل "ABCDEFG" بوده باشد.به همین دلیل حروف الفبای انگلیسی به جز "a" و "I" که در انگلیسی دارای معنی هستند را با value = false ذخیره کردیم.
- متد isValid یک کلمه را به عنوان ورودی دریافت می کند و اگر در دیکشنری وجود داشت و مقدار value آن هم برابر true بود، معتبر بودن این کلمه را به صورت Boolean بر می گرداند.
- در متد tracingTree الگوریتم شکستن جمله به کلمات به صورت بازگشتی انجام می شود. الگوریتم از پیشوند یک حرفه، دو حرفه، سه حرفه و ... شروع می کند و باقی مانده رشته را مجددا چک می کند. اگر پیشوند امید بخش نبود (دارای معنی نباشد یا با انتخاب آن، ما بقی رشته بدون معنا شود) این شاخه از درخت دیگر پیمایش نشده (پیمایش این شاخه با عبارت return متوقف می شود) و به سراغ پیشوند بعدی می رویم تا زمانی که همه شاخه های حاوی گره امید بخش پیمایش شوند و رشته به جملات دارای کلمات بامفهوم تبدیل شود.
- متد sentenceBreak متدی است که مرتب کردن فرمت خروجی و فراخوانی متد اصلی tracingTree را به عهده دارد.

مثالی از ورودی برنامه :

```
sentenceBreak( str: "ILOVEICECREAMANDMANGO");
sentenceBreak( str: "IAMBATMAN");
sentenceBreak( str: "CALLSECURITYATMIAMIAIRPORTBECAUSEITHINKABOMBISABOUTTOGOOFF");
sentenceBreak( str: "ABORTTHEPLANMEETATTHEDARKCABIN");
sentenceBreak( str: "LADIESANDGENTLEMENPLEASEWEARYOURMASKSEVERYWHERE");
sentenceBreak( str: "MYGRANDMOTHERALWAYSREMINDEDMETODRIVECAREFULLYANYTIMEIAMONMYWAYTOHOME");
```

مثالی از خروجی برنامه :

```
_____________________________________
i love ice cream and man go
i love ice cream and mango
i love icecream and man go
i love icecream and mango

_____________________________________
i am bat man
i am batman

_____________________________________
call security at miami air port be cause i think a bomb is about to go off
call security at miami air port because i think a bomb is about to go off
call security at miami airport be cause i think a bomb is about to go off
call security at miami airport because i think a bomb is about to go off

_____________________________________
abort the plan meet at the dark cabin

_____________________________________
ladies and gentle men please wear your masks every where
ladies and gentle men please wear your masks everywhere
ladies and gentlemen please wear your masks every where
ladies and gentlemen please wear your masks everywhere

_____________________________________
my grand mother always reminded me to drive care fully any time i am on my way to home
my grand mother always reminded me to drive care fully anytime i am on my way to home
my grand mother always reminded me to drive carefully any time i am on my way to home
my grand mother always reminded me to drive carefully anytime i am on my way to home
my grandmother always reminded me to drive care fully any time i am on my way to home
my grandmother always reminded me to drive care fully anytime i am on my way to home
my grandmother always reminded me to drive carefully any time i am on my way to home
my grandmother always reminded me to drive carefully anytime i am on my way to home
```

محاسبه پیچیدگی زمانی:

فرض کنیم عبارت ورودی MANGO باشد:

کل حالاتی که ممکن است به وجود بیاید حالتی است که بین هر دو حرف یک اسپیس داشته باشیم یا نداشته باشیم. یعنی تعداد کل حالات برابر می شود با : 2^4 = 16

اما حالاتی که چک می شوند به این صورت هستند:

(علامت ✗ نشانگر غیر امید بخش بودن آن مرحله است)

M + ANGO ✗

MA + NGO ✗

MAN + GO

MAN + G + O

MAN + GO ✓

MANG + O ✗

MANGO ✓

پس با این روش بسیاری از گره های درخت پیمایش نمی شوند اما در بدترین حالت باید همه درخت را پیمایش کنیم. پس برای رشته ای به طول n، به تعداد 2^{n-1} حالت را باید بررسی کنیم. پس پیچیدگی زمانی این برنامه از مرتبه $O(2^n)$ خواهد بود زیرا :

$O(2^{n-1}) = O(\frac{2^n}{2}) = O(2^n)$