گزارش پروژه دوم درس طراحی الگوریتم‌ها (James Bond)

امیرعلی صادقی فرشی (۹۹۱۲۸۳۴)

دیتاست مورد استفاده: برای این پروژه از لغت‌نامه‌ی انگلیسی آکسفورد که در این لینک موجود می‌باشد استفاده شده است. به‌طور دقیق‌تر، از فایل Oxford 3000 Word List.txt استفاده شده است. در این لغت‌نامه بعضی کلمات به‌صورت تکراری ظاهر شده‌اند؛ مثلاً close. در فایل ذکر شده مثلاً برای close، دو سطر close 1 و close 2 وجود دارد اما در فایل دیگر (Oxford 3000 Word List No Spaces.txt) به‌صورت close_1 و close_2 وجود دارد. شیوه‌ی اول برای کار پردازش راحت‌تر است چون با استفاده از تابع `split()` در پایتون می‌توان محتویات داخل فایل را چه با استفاده از کاراکتر New Line و چه با استفاده از فاصله جدا کرد و بنابراین به‌راحتی می‌توانیم close را نیز به دست آوریم. دو پیش‌پردازش بر روی این دیتاست انجام شد که به شرح زیر است:

۱. تمام حروف کلمات به حروف کوچک تبدیل شدند. هدف از این کار این است که اگر به‌عنوان مثال کلمه‌ی I به معنی «من» در جمله موجود باشد و چون جمله‌ی ورودی تماماً با حروف بزرگ است، ما ناچاریم همه‌ی حروف را ابتدا کوچک کنیم و بنابراین برای مطابقت با لغت‌نامه، تمام حروف کلمات در لغت‌نامه نیز باید کوچک باشند.
۲. بعضی افعال ساده و اساسی زبان انگلیسی در این دیتاست موجود نبودند که به آن اضافه شدند. این کلمات عبارتند از:
is - did - am - are - was - were - didnt - werent - wasnt - arent - isnt
لازم به ذکر است افعال منفی مخفف، بدون کاراکتر ' اضافه شده‌اند چون در جمله‌ی ورودی نیز همه‌ی علائم ورودی حذف شده‌اند.

توابع مورد استفاده: برای جداسازی جمله و نمایش جمله‌ی جداشده از دو تابع parse و print_parsing استفاده شده است. تابع parse با گرفتن جمله، در صورتی که قابل parse نباشد این موضوع را به کاربر اطلاع می‌دهد. اگر قابل parse باشد، ماتریس حاوی نقاط شکست (در ادامه توضیح داده می‌شود) را به تابع print_parsing می‌دهد. تابع print_parsing تمام parsingهای ممکن را برای جمله‌ی داده‌شده چاپ می‌کند.

الگوریتم مورد استفاده: برای جداسازی جمله از روشی مبتنی بر برنامه‌ریزی پویا استفاده شده است. بدین‌منظور دو ماتریس valid و break_points که هر دو دارای اندازه‌ی $(length + 1) \times (length + 1)$ هستند که در آن $length$ طول جمله است.

ماتریس $valid$ ابتدا با مقادیر صفر مقداردهی می‌شود. خانه‌ی $valid(i,j)$ اگر مکان i تا j جمله:

۱- قابل parse نباشد، مقدار صفر خواهد داشت.

۲- قابل parse به یک کلمه‌ی معتبر و یک بخش قابل parse (یا تهی) باشد، مقدار یک خواهد داشت.

۳- قابل parse به دو بخش قابل parse (که اولین بخش یک کلمه‌ی معتبر نیست) باشد، مقدار دو خواهد داشت.

ماتریس valid به‌صورت بالامثلثی تکمیل خواهد شد. ترتیب پر شدن خانه‌های آن به‌ترتیب از چپ به راست و سپس پایین به بالا است. به‌عنوان مثال: $valid(0,1)$ <- $valid(1,2)$ <- $valid(0,2)$ <- $valid(2,3)$ <- $valid(1,3)$ <- $valid(0,3)$ <- ...

برای پر کردن هر کدام از خانه‌های آن، در صورتی که آن زیررشته از جمله خودش یک کلمه باشد، مقدار یک به خود خواهد گرفت. در غیر این‌صورت، با نقاط میانی مختلف بین i و j امتحان می‌کنیم. اگر به‌ازای یک نقطه‌ی میانی، دو بخش با مقدار valid غیرصفر پیدا شد، مطابق حالت ۲ و ۳ که در بالا ذکر شد یکی از مقادیر ۱ یا ۲ را به خانه‌ی کنونی اختصاص خواهیم داد. در نهایت اگر خانه‌ی $valid(0, length)$ مقداری غیرصفر داشته باشد، جمله قابل parse است در غیر این‌صورت با دیکشنری داده‌شده نمی‌توان جمله‌ی کنونی را parse کرد.

همزمان با پر کردن ماتریس valid، ماتریس break_points نیز پر می‌شود. این ماتریس حاوی نقاط شکست است. مثلاً برای عبارت iam نقطه‌ی شکست برابر ۱ است که در نتیجه‌ی آن می‌توان این عبارت را به i am تجزیه کرد. ممکن است یک عبارت از نقاط مختلفی امکان شکست داشته باشد، به همین جهت، ماتریس break_points برای هر کدام از خانه‌هایش، یک لیست حاوی نقاط شکست ممکن دارد.

برای پر کردن این ماتریس، اگر زیررشته i تا j جمله خودش یک کلمه باشد، در خانه‌ی متناظر آن در break_points عدد ۱- را اضافه می‌کنیم. همچنین اگر حالت ۲ در ماتریس valid اتفاق بیفتد، نقطه‌ی میانی‌ای که باعث آن شده است را در خانه‌ی متناظر آن در break_points اضافه می‌کنیم.

در صورتی که یک جمله قابل parse باشد، همان‌طور که گفته شد تابع print_parsings وظیفه‌ی چاپ همه‌ی parsingهای ممکن را دارد. این تابع از خانه‌ی $break\_points(0, length)$ شروع می‌کند و به‌صورت بازگشتی تا جایی که به انتهای جمله برسد پیش می‌رود و به‌ازای هر بار که به انتهای جمله برسد، جمله‌ی جداشده را چاپ خواهد کرد.

پیچیدگی زمانی: ماتریس‌های ذکر شده هر دو $O(n^2)$ خانه دارند که $n = length$. لذا برای پر کردن هر دوی آن‌ها در مجموع زمان $O(n^2)$ سپری خواهد شد.

شبیه‌سازی: برای شبیه‌سازی تعدادی جمله‌ی ورودی و خروجی متناظر با آن را در زیر مشاهده می‌کنیم. اگر کلمه‌ای در لغت‌نامه موجود نباشد، همان‌طور که گفته شد، این موضوع به اطلاع کاربر می‌رسد. همچنین اگر جمله‌ای چند parsing داشته باشد، همه‌ی آن‌ها چاپ می‌شوند:

```
52      sentence = "AWAY"
53      parse(sentence)
54      print('')
55
56      sentence = "IHOPETHISISOURLASTPROJECTINTHISCOURSE"
57      parse(sentence)
58      print('')
59
60      sentence = 'WEAREVERYTIREDRIGHTNOW'
61      parse(sentence)
62      print('')
63
64      sentence = 'SLEEPISINTHEDICTIONARY'
65      parse(sentence)
66      print('')
67
68      sentence = 'SLEEPYISNOTINTHEDICTIONARY' # Sleepy is not in the dictionary
69      parse(sentence)
70      print('')
```

```
a way
away

i hope this i sour last project in this course
i hope this is our last project in this course

we are very tired right now
wear every tired right now

sleep is in the dictionary

This sentence cannot be parsed!
```