تمرین ۶ درس طراحی الگوریتم

سید صالح اعتمادی
امیر خاکپور
مهسا سادات رضوی

دانشگاه علم و صنعت ۹۷-۹۸

لطفا به نکات زیر توجه کنید:

- مهلت ارسال این تمرین شنبه ۲۴ فروردین ساعت ۱۱:۵۹ ب.ظ است.
- این تمرین شامل سوال های برنامه نویسی است، بنابراین توجه کنید که حتماً موارد خواسته شده را رعایت کنید. .
- نام شاخه، پوشه و پول ریکوست همگی دقیقا "A۶" باشد.
- در صورتی که به اطلاعات بیشتری نیاز دارید می توانید با آیدی تلگرام sargdsra@ یا mhsarzvi@ در ارتباط باشید.
- اگر در حل تمرین شماره ی ۶ مشکلی داشتید،لطفا به sargdsra@ یا mhsarzvi@ مراجعه کنید.

موفق باشید.

توضیحات کلی تمرین

تمرین این هفته ی شما،۴ سوال دارد که باید به همه ی این سوال ها پاسخ دهید. برای حل این سری از تمرین ها مراحل زیر را انجام دهید:

۱. ابتدا مانند تمرین های قبل، یک پروژه به نام A۶ بسازید.

۲. کلاس هر سوال را به پروژه ی خود اضافه کنید و در قسمت مربوطه کد خود را بنویسید. هر کلاس شامل دو متد اصلی است:

متد اول: تابع solve است که شما باید الگوریتم خود را برای حل سوال در این متد پیاده سازی کنید.

متد دوم: تابع process است که مانند تمرین های قبلی در TestCommon پیاده سازی شده است. بنابراین با خیال راحت سوال را حل کنید و نگران تابع process نباشید! زیرا تمامی پیاده سازی ها برای شما انجام شده است و نیازی نیست که شما کدی برای آن بزنید.

۳. اگر برای حل سوالی نیاز به تابع های کمکی دارید؛ می توانید در کلاس مربوط به همان سوال تابع تان را اضافه کنید.

اکنون که پیاده سازی شما به پایان رسیده است، نوبت به تست برنامه می رسد. مراحل زیر را انجام دهید.

۱. یک UnitTest برای پروژه ی خود بسازید.

۲. فولدر TestData که در ضمیمه همین فایل قرار دارد را به پروژه ی تست خود اضافه کنید.

۳. فایل GradedTests.cs را به پروژه ی تستی که ساخته اید اضافه کنید. توجه کنید که مانند تمرین های قبل، لازم نیست که برای هر سوال TestMethod بنویسید. تمامی آنچه که برای تست هر سوالتان نیاز دارید از قبل در این فایل برای شما پیاده سازی شده است.

دقت کنید که TestCommon تغییر یافته است. بنابراین شما باید نسخه ی جدید آن را با دستورgit Pull دریافت کنید .

۱ بدست آوردن تبدیل Burrows Wheeler برای یک رشته

تبدیل Burrows Wheeler یک متن، نمادهای متن را به گونه ای تغییر می‌دهد تا براحتی قابل فشرده سازی شود. همچنین، این تبدیل برگشت پذیر نیز هست. یعنی از روی تبدیل BW یک متن میتوانیم متن اصلی را بدست آوریم. کاربرد این تبدبل علاوه بر فشرده سازی متن، در حل کردن الگوریتم تطبیق چندگانه الگوها و ... است.

بدست آوردن تبدیل Burrows Wheeler برای یک رشته بصورت مقابل تعریف می‌شود: اول تمام چرخش‌های متناوب ممکن از متن را تشکیل دهید. یک چرخش متناوب از جداکردن پسوندی از رشته از انتهای رشته و اضافه کردن آن به ابتدای رشته بدست می آید. سپس این رشته ها را به ترتیب الفبایی مرتب کنید تا یک ماتریس M با سایز ITEXTI × ITEXTI تشکیل شود. BWT برای رشته ورودی برابر با ستون آخر ماتریس M است.

در این سوال شما باید الگوریتمی بنویسید که BWT برای یک رشته ورودی که با حروف A T G C ساخته شده است و با نماد $ پایان می‌یابد را برگرداند. سایز رشته ورودی بین ۱ تا ۱۰۰۰ کاراکتر است.

ـ برای دیباگ و تست کردن جواب های خود میتوانید از این لینک استفاده کنید.

نمونه ۱

ورودی:

```
AA$
```

خروجی:

```
AA$
```

$$M(Text) = \begin{bmatrix} \$ & A & A \\ A & \$ & A \\ A & A & \$ \end{bmatrix}$$

نمونه ۲

ورودی:

```
ACACACAC$
```

خروجی:

```
CCCC$AAAA
```

$$M(Text) = \begin{bmatrix} \$ & A & C & A & C & A & C & A & C \\ A & C & \$ & A & C & A & C & A & C \\ A & C & A & C & \$ & A & C & A & C \\ A & C & A & C & A & C & \$ & A & C \\ A & C & A & C & A & C & A & C & \$ \\ C & \$ & A & C & A & C & A & C & A \\ C & A & C & \$ & A & C & A & C & A \\ C & A & C & A & C & \$ & A & C & A \\ C & A & C & A & C & A & C & \$ & A \end{bmatrix}$$

نمونه ۳

ورودی:

```
AGACATA$
```

خروجی:

```
ATG$CAAA
```

$$M(Text) = \begin{bmatrix} \$ & A & G & A & C & A & T & A \\ A & \$ & A & G & A & C & A & T \\ A & C & A & T & A & \$ & A & G \\ A & G & A & C & A & T & A & \$ \\ A & T & A & \$ & A & G & A & C \\ C & A & T & A & \$ & A & G & A \\ G & A & C & A & T & A & \$ & A \\ T & A & \$ & A & G & A & C & A \end{bmatrix}$$

۲ بدست آوردن رشته از روی BWT

در سوال قبل با چگونه بدست آوردن BWT برای یک رشته آشنا شدید و دیدیم این عمل برای فشرده‌سازی متون انجام می‌شود. برای تکمیل فرآیند فشرده‌سازی لازم است تا بتوانیم برعکس این فرآیند را نیز انجام دهیم. یعنی بتوانیم از روی BWT یک رشته، رشته اصلی را بدست آوریم. در این سوال باید الگوریتمی بنویسید که رشته اصلی را از روی BWT رشته بدست آورد. ورودی الگوریتم یک رشته شامل یک نماد $ است که با حروف G T C A ساخته شده است و سایز رشته ورودی بین ۱ تا ۱۰۰۰۰۰۰ کاراکتر است. خروجی الگوریتم آن رشته اصلی است که تبدیل BW آن در ورودی داده شده است.

نمونه ۱

ورودی:

```
AC$A
```

خروجی:

```
ACA$
```

$$M(Text) = \begin{bmatrix} \$ & A & C & A \\ A & \$ & A & C \\ A & C & A & \$ \\ C & A & \$ & A \end{bmatrix}$$

نمونه ۲

ورودی:

```
AGGGAA$
```

خروجی:

GAGAGA$

$$M(Text) = \begin{bmatrix} \$ & G & A & G & A & G & A \\ A & \$ & G & A & G & A & G \\ A & G & A & \$ & G & A & G \\ A & G & A & G & A & \$ & G \\ G & A & \$ & G & A & G & A \\ G & A & G & A & \$ & G & A \\ G & A & G & A & G & A & \$ \end{bmatrix}$$

۳ تطبیق الگوها با استفاده از رشته‌های فشرده شده

یکی دیگر از قابلیت‌های جالب BWT ، این است که به ما این قابلیت را می‌دهد که بدون de compress کردن رشته بتوانیم مسئله Pattern Matching را حل کنیم. بدین ترتیب حافظه کمتری مصرف می‌شود.
الگوریتم BWMatching تعداد تکرار شدن الگوها را در متن می شمارد. اطلاعاتی که این الگوریتم در اختیار دارد تنها ستون اول ماتریس، ستون آخر (یعنی BWT) و یک mapping از ابتدا به انتها است.

در این سوال شما باید الگوریتمی بنویسید که بتواند الگوها را در یک متن فشرده شده پیدا کند.
در خط اول فایل ورودی یک رشته که فرم BWT متن است وجود دارد. این رشته با نمادهای G C T A ساخته شده است و طول آن بین ۱ تا ۱۰۰۰ حرف است و در آن نماد $ نیز وجود دارد. در خط بعدی یک عدد صحیح بین ۱ تا ۵۰۰۰ وجود دارد که نشان دهنده ی تعداد الگوهاست. در n خط بعدی الگوهایی که باید در رشته ورودی یافت شوند وجود دارد.
شما باید در فایل خروجی لیستی از اعداد صحیح که هر یک نشان دهنده ی تعداد تکرار الگوی متناظرش در متن است را برگردانید.

نمونه ۱

ورودی:

```
AGGGAA$
1
GA
```

خروجی:

```
3
```

در این مثال متن $GAGAGA بوده که الگوی GA سه بار در آن تکرار شده است.

نمونه ۲

ورودی:

```
ATT$AA
2
ATA
A
```

خروجی:

```
2 3
```

در این مثال متن $ATATA بوده که شامل ۲ الگوی ATA و سه الگوی A است.

نمونه ۳

ورودی:

```
AT$TCTATG
2
TCT
TATG
```

خروجی:

```
0 0
```

در این مثال متن ATCGTTTA بوده که شامل هیچ یک از الگوهای داده شده در ورودی نیست.

۴ تشکیل Suffix Array برای یک رشته

در تمرین قبل با Suffix Tree ها آشنا شدیم و دیدیم در عمل حافظه زیادی اشغال می‌کنند. آرایه پیشوندی یک راه حل جایگزین برای درخت پیشوندی است که از نظر حافظه بهینه عمل می‌کند. بعنوان مثال برای ذخیره درخت پسوندی ژنوم انسان نیاز به ۶۰ گیگ حافظه داریم درحالی آرایه پسوندی تنها ۱۲ گیگ حافظه اشغال می‌کند. برای ساختن Suffix Array یک رشته به این صورت عمل می‌کنیم که تمامی پسوندهای رشته ورودی را به ترتیب الفبایی (با فرض $ بعنوان اولین نماد الفبا) مرتب می‌کنیم. آرایه پسوندی، لیستی از ایندکس اولین نماد از این پسوندهای مرتب شده در رشته ورودی است. در این سوال باید آرایه پسوندی برای یک رشته ورودی ساخته شده با حروف G T C A که با نماد $ تمام می شود را تشکیل دهید.

ـ برای دیباگ و تست کردن جواب های خود میتوانید از این لینک استفاده کنید.

نمونه ۱

ورودی:

```
GAC$
```

خروجی:

```
3 1 2 0
```

Sorted suffixes:

```
3   $
1   AC$
2   C$
0   GAC$
```

نمونه ۲

ورودی:

```
GAGAGAGA$
```

خروجی:

```
8 7 5 3 1 6 4 2 0
```

Sorted suffixes:

8	$
7	A$
5	AGA$
3	AGAGA$
1	AGAGAGA$
6	GA$
4	GAGA$
2	GAGAGA$
0	GAGAGAGA$

نمونه ۳

ورودی:

```
AACGATAGCGGTAGA$
```

خروجی:

```
15 14 0 1 12 6 4 2 8 13 3 7 9 10 11 5
```

Sorted suffixes:

15	$
14	A$
0	AACGATAGCGGTAGA$
1	ACGATAGCGGTAGA$
12	AGA$
6	AGCGGTAGA$
4	ATAGCGGTAGA$
2	CGATAGCGGTAGA$
8	CGGTAGA$
13	GA$
3	GATAGCGGTAGA$
7	GCGGTAGA$
9	GGTAGA$
10	GTAGA$
11	TAGA$
5	TAGCGGTAGA$