یاداشت‌های درس مبانی بیوانفورماتیک

مدرس: ابوالفضل مطهری

آرایه‌ی پسوندی و تبدیل Burrows-Wheeler

گردآورنده: محمد امین ادریسی

دی ماه ۱۳۹۳

۱ آرایه‌ی پسوندی

تعریف ۱ *آرایه‌ی پسوندی[۱] برای رشته‌ی S آرایه‌ای است که عضو iام آن برابر است با محل شروع iامین پسوند (پس از مرتب‌سازی الفبایی[۲]) از این رشته.*

برای تولید این آرایه، ابتدا همه‌ی پسوندها را برای یک رشته، نوشته و سپس آنها را براساس حروف الفبا مرتب می‌کنیم. حال با همان ترتیب اندیس‌های محل شروع این پسوندها را در یک آرایه ذخیره می‌کنیم. آرایه‌ی حاصل، آرایه‌ی پسوندی مورد نظر است.

مثال ۱ *بدست آوردن آرایه‌ی پسوندی برای رشته‌ی $S = banana\$$:*
ابتدا تمام پسوندهای ممکن را بدست می‌آوریم. در جدول زیر پسوندها قبل از مرتب‌سازی نمایش داده شده‌اند.

[۱] Suffix Array
[۲] lexicographical order

index	suffix
7	$
6	a$
5	na$
4	ana$
3	nana$
2	anana$
1	banana$

حال باید پسوندها را بر اساس ترتیب الفبایی مرتب کنیم. باید توجه داشت که در این مرتب‌سازی اگر حرف a ارزش بیشتری نسبت به حرف b دارد، سمبل انتهایی رشته، از همه‌ی حروف دارای ارزش بیشتری است. بعد از مرتب‌سازی می‌توان اعضای آرایه‌ی پسوندی را به صورت زیر مشخص کرد:

i	$A[i]$	suffix
1	7	$
2	6	a$
3	4	ana$
4	2	anana$
5	1	banana$
6	5	na$
7	3	nana$

از روی این جدول آرایه‌ی پسوندی برای این رشته به صورت $A = [۷ ۶ ۴ ۲ ۱ ۵ ۳]$ *بدست می‌آید.*

۲ تبدیل Burrows-Wheeler

این تبدیل یکی از ابزارهای فشرده‌سازی متن در کامپیوتر به حساب می‌آید؛ در واقع بعد از تبدیل یک رشته با این روش، آن را با استفاده از الگوریتم $\text{run} - \text{length} - \text{encoding}$ فشرده می‌کنند. برای بدست آوردن تبدیل Burrows-Wheeler مربوط به یک رشته، ابتدا تمامی گردش‌های[۳] رشته را بدست آورده و آنها را طبق ترتیب الفبایی مرتب می‌کنیم. در اینجا نیز مانند قبل، بیشترین ارزش برای آخرین سمبل رشته است. اگر آخرین ستون را برای این جدول جدا کرده و به عنوان یک آرایه در نظر بگیریم، این آرایه، همان تبدیل BW مربوط به این رشته‌ی مورد نظر است.

مثال ۲ *بدست آوردن تبدیل BW برای رشته‌ی* $S=banana$

[۳] Rotation

$\$$	b	a	n	a	n	a
a	$\$$	b	a	n	a	n
a	n	a	$\$$	b	a	n
a	n	a	n	a	$\$$	b
b	a	n	a	n	a	$\$$
n	a	$\$$	b	a	n	a
n	a	n	a	$\$$	b	a

در جدول فوق ستون آخر که همان تبدیل مورد نظر است با رنگ دیگری نمایش داده شده است.

میان این تبدیل و آرایه‌ی پسوندی رابطه‌ای وجود دارد که اگر رشته‌ی حاصل از تبدیل را BWT و آرایه‌ی پسوندی را A بنامیم و رشته‌ی اصلی نیز S باشد،این رابطه به صورت زیر است:

$$BWT[i] = \begin{cases} S[A[i]-1] & \text{if } A[i] \neq 1 \\ \$ & \text{if } A[i] = 1 \end{cases}$$

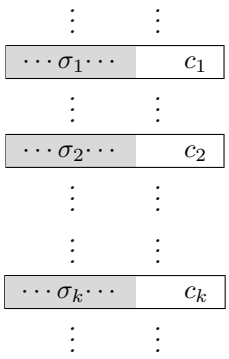
نکته ۱ *اگر فرض کنیم که حرف c در ستون اول k تکرار داشته باشد و آنها را به ترتیب به صورت $c_1, c_2, \cdots, c_k$ بنامیم، ترتیب حضور آنها در ستون آخر نیز به همین ترتیب است.با توجه به شکل زیر اگر ادامه‌ی رشته پس از c_k را σ_k بنامیم، داریم:*

$$\begin{array}{cc} \vdots & \vdots \\ c_1 & \cdots\sigma_1\cdots \\ c_2 & \cdots\sigma_2\cdots \\ \vdots & \vdots \\ c_{k-2} & \cdots\sigma_{k-2}\cdots \\ c_{k-1} & \cdots\sigma_{k-1}\cdots \\ c_k & \cdots\sigma_k\cdots \\ \vdots & \vdots \end{array}$$

از آنجایی که تکرارهای این حرف پشت‌سرهم به صورت الفبایی مرتب‌شده‌اند، پس می‌توان گفت که از نظر ارزش دنباله‌ها به صورت زیرند:

$$\sigma_1 \geq \sigma_2 \geq \cdots \sigma_k$$

از طرفی در کل جدول مرتب شده، تمام چرخش‌های رشته‌ی اصلی وجود دارند، به ازای چرخش $c_1\sigma_1$، $\sigma_1 c_1$ نیز در این جدول وجود دارد. پس با توجه به رابطه‌ی فوق برای σ_iها، $\sigma_1 c_1$ بالاتر از $\sigma_2 c_2$ در جدول اصلی قرار می‌گیرد؛ یعنی شکل زیر به دست می‌آید:



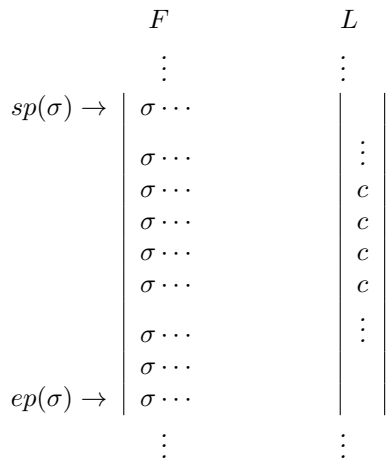
پس ترتیب c_iها در ستون آخر نیز به هم نمی‌خورد.

از جدول BW می‌توان برای exact pattern matching استفاده کرد اما پیش از بیان آن نکته‌ی زیر را مطرح می‌کنیم:

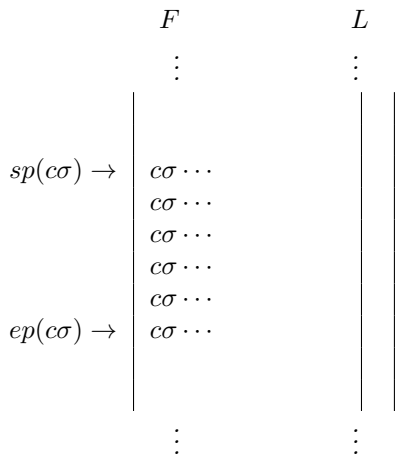
نکته ۲ *اگر $sp(\sigma)$ و $ep(\sigma)$ به ترتیب نقاط آغازین و پایانی ظهور رشته σ در جدول BW باشند و $M[c]$ نیز مکان اولین ظهور حرف c در ستون اول جدول BW باشد، آنگاه برای هر حرف c خواهیم داشت:*

$$sp(c\sigma) = M[c] + occ(c, sp(\sigma) - 1)$$
$$ep(c\sigma) = M[c] + occ(c, ep(\sigma)) - 1$$

که در اینجا تابع $occ(c, x)$ فراوانی تجمعی مربوط به تکرارهای حرف c در ستون آخر جدول است از آغاز تا اندیسی که به عنوان ورودی می‌گیرد یعنی x.
اگر ستون اول را با حرف F و ستون آخر را با L نشان دهیم داریم:



با استفاده از $sp(c\sigma)$ و $ep(c\sigma)$ می‌توان مکان‌های ظهور رشته‌ی $c\sigma$ را در جدول BW یافت.



با استفاده از نکته‌ی فوق می‌توان به وجود یا عدم وجود یک زیررشته در یک رشته که جدول BW را برای آن ایجاد کرده‌ایم، پی برد. در واقع حکم زیر را می‌توان نتیجه گرفت:

نتیجه ۱ *برای یک رشته مانند S که جدول BW آن را ایجاد کرده‌ایم و رشته‌ی σ:*

$$\sigma \text{ یک زیررشته‌ی } S \text{ است} \iff sp(\sigma) \leq ep(\sigma)$$

مثال ۳ *آیا زیررشته‌ی ana در رشته‌ی $banana\$$ وجود دارد؟*
برای پاسخ به این سؤال با استفاده از روش بالا، از انتهای زیررشته آغاز کرده و اولین کاراکتر را در جدول BW مربوط به $banana\$$ پیدا می‌کنیم و در صورت وجود، هر بار حرف بعدی را از آخر مانند حرف c در روش گفته شده به رشته‌ای که تاکنون پیدا کرده‌ایم، می‌افزاییم.

F						L
$\$$	b	a	n	a	n	a
a	$\$$	b	a	n	a	n
a	n	a	$\$$	b	a	n
a	n	a	n	a	$\$$	b
b	a	n	a	n	a	$\$$
n	a	$\$$	b	a	n	a
n	a	n	a	$\$$	b	a

ابتدا داریم $c = a$ که طبق جدول مرتب‌شده:

$$sp(a) = 2 \quad ep(a) = 4$$

سپس $c = n$ است که برای $sp(na)$ با استفاده از جدول فوق، داریم:

$$\begin{aligned} sp(na) &= M[n] + occ(n, sp(a) - 1) \\ &= 6 + occ(n, 2 - 1) \\ &= 6 + occ(n, 1) \\ &= 6 + 0 \\ &= 6 \end{aligned}$$

و برای $ep(na)$ نیز داریم:

$$\begin{aligned} ep(na) &= M[n] + occ(n, ep(a)) - 1 \\ &= 6 + occ(n, 4) - 1 \\ &= 6 + 2 - 1 \\ &= 7 \end{aligned}$$

حال آخرین حرف a است که قرار می‌دهیم، $c = a$:

$$\begin{aligned} sp(ana) &= M[a] + occ(a, sp(na) - 1) \\ &= 2 + occ(a, 6 - 1) \\ &= 2 + occ(a, 5) \\ &= 2 + 1 \\ &= 3 \end{aligned}$$

$$\begin{aligned} ep(ana) &= M[a] + occ(a, ep(na)) - 1 \\ &= 2 + occ(a, 7) - 1 \\ &= 2 + 3 - 1 \\ &= 4 \end{aligned}$$

پس برای این زیررشته داریم که :

$$sp(ana) \leq ep(ana)$$

و در نتیجه ana یک زیررشته‌ی $banana\$$ است.

۱.۲ بازسازی رشته‌ی اصلی از روی ستون L

برای بازسازی رشته‌ی اصلی از روی ستون آخر جدول BW کافی آرایه‌ی C را به صورت زیر تعریف کنیم:

$C[j]$ = تعداد تکرارهایی که کاراکتر jام از ستون L قبل از خود داشته است

حال با استفاده از آرایه‌ی M که قبلاً تعریف کردیم و این آرایه، می‌توان از روی ستون L رشته‌ی اصلی را بازسازی کرد به این شکل که ابتدا تعریف می‌کنیم:

$$V[j] = M[L[j]] + C[j]$$

سپس برای بار اول این تابع را محاسبه می‌کنیم؛ عدد بدست آمده را با استفاده از $M[V[j]]$ به کاراکتر تبدیل کرده و آن را در قالب یک رشته به نام R نام‌گذاری می‌کنیم. حال تا رسیدن به کاراکتر $ مراحل زیر را تکرار می‌کنیم:

۱. j جدید را برابر با $V[j]$ قبلی قرار می‌دهیم.

۲. $V[j]$ جدید را براساس آن محاسبه می‌کنیم.

۳. مقدار بدست آمده را با استفاده از M به کاراکتر تبدیل کرده و آن را از چپ به رشته‌ی R اضافه می‌کنیم.

۴. اگر آخرین کاراکتر بدست آمده $ نبود، به مرحله اول می‌رویم و وگرنه از حلقه خارج می‌شویم.

با اتمام این عمل، رشته‌ی اصلی از انتها به ابتدا بدست می‌آید.

۲.۲ کاهش حجم ذخیره‌سازی

در رابطه‌ی گفته شده برای $V[j]$ می‌توان به جای C از occ نیز استفاده کرد که در نتیجه رابطه‌ی $V[j]$ به صورت زیر در می‌آید:

$$V[j] = M[L[j]] + occ(L[j], j) - ۱$$

حال برای ذخیره‌ی occ می‌توان در ستون L فقط در گام‌های kتایی، occ را محاسبه و ذخیره کرد. در این صورت وقتی می‌خواهیم برای یک خانه ی نامعلوم از ستون L مقدار occ مربوطه را پیدا کنیم، به نزدیک‌ترین خانه‌ی معلوم رفته و با استفاده از تعداد تکرارهایی که در مسیر دیده شده‌اند، occ آن را می‌یابیم.
از آنجایی که با جدول BW می‌توان تکرارهای یک زیررشته را در رشته‌ی اصلی پیدا کرد، کافی است از روی آرایه‌ی پسوندی مکان‌های تکرار را بیابیم، اما به جای ذخیره‌ی همه‌ی این آرایه می‌توان مانند occ بخشی از آن را ذخیره کرد که این کار نیز حجم را بسیار کاهش می‌دهد.