دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف

ژنومیک محاسباتی

مطهری و فروغمند
پاییز ۱۴۰۰

هم‌ترازی توالی

جلسه دوم

نگارنده: مهدی کافی

۱ مروری بر مباحث گذشته

این جلسه، جلسه اولی است که در کلاس درس داده می‌شود و جلسه قبلی مقدمه و آشنایی با درس بود.

۲ رشته‌ها

در ژنومیک محاسباتی، منظورمان از توالی، رشته‌های زیستی هستند که در ادامه با برخی از این رشته‌ها بیشتر آشنا خواهیم‌شد.

۱.۲ رشته‌های DNA

رشته‌های DNA از واحدهای ساختاری شیمیایی به نام نوکلئوتید[۱] ساخته شده‌اند. این واحدها از سه بخش گروه فسفات، گروه قند و یکی از چهار نوع باز نیتروژنی ساخته شده‌اند. برای ایجاد یک رشته DNA زنجیره‌ای از نوکلئوتیدها در کنار یکدیگر قرار گرفته و رشته DNA را تشکیل می‌دهند. چهار نوع باز نیتروژنی که در نوکلئوتیدها دیده می‌شوند عبارتند از آدنین[۲]، تایمین[۳]، گوانین[۴] و سایتوزین[۵] که به ترتیب با حروف A, T, G و C نمایش داده می‌شوند. منظور از توالی ژنومی، یک رشته از الفبای این چهار نوکلئوتید است که این رشته توسط روش‌هایی استخراج و به عنوان ورودی

[۱] Nucleotide
[۲] Adenine
[۳] Thymine
[۴] Guanine
[۵] Cytosine

به ما داده می‌شود. ژنوم انسان حدودا از ۳ میلیارد جفت بازی[۶] تشکیل شده‌است.

۲.۲ رشته‌های پروتئینی

رشته‌های پروتئینی از اسیدهای آمینه[۷] تشکیل شده‌اند و با توجه به اینکه بیست نوع اسید آمینه داریم، رشته‌های پروتئینی از الفبایی بیست حرفی ساخته می‌شوند. این رشته‌ها طولی کمتر از ۲۰۰۰ اسید آمینه دارند.

۳ هم‌ترازی

در ابتدا می‌خواهیم بررسی کنیم که چرا هم‌ترازی دارای اهمیت ویژه‌ای است. در زیست شناسی می‌دانیم که اگر دو رشته DNA به یکدیگر شبیه باشند، آن دو فعالیت یکسانی خواهندداشت. محققان بیوانفورماتیک در بسیاری از مواقع شباهت بین توالی‌های زیستی را مورد مطالعه قرار می‌دهند تا ساختار و یا فعالیت آن‌ها را به دست آورند. در زیر چند کاربرد این مقایسه آورده شده‌است[Sun09].

- پیشبینی عملکرد زیستی یک ژن (یا یک RNA و یا یک پروتئین): اگر توالی یک ژن به ژن دیگری با عملکرد شناخته‌شده نزدیک باشد، می‌توانیم تخمین بزنیم که این ژن هم عملکردی مشابه ژن دیگر دارد.
- پیدا کردن فاصله تکاملی: گونه‌ها به وسیله تغییرات در ژنومشان تکامل می‌یابند. با محاسبه شباهت ژنوم‌ها می‌توانیم به فاصله تکاملی گونه‌ها پی ببریم.
- کمک به ساخت ژنوم: تکنولوژی‌های فعلی تنها می‌توانند بخش‌های کوچکی از رشته‌های DNA را بخوانند. پروژه ژنوم انسان[۸] بر پایه اشتراک‌های تعداد زیادی قطعه کوچک DNA ژنوم کامل انسان را بازسازی کرد. این اطلاعات اشتراکی با مقایسه رشته‌ها به دست آمدند.
- پیدا کردن نواحی مشترک در دو ژنوم: اگر دو رشته نسبتا بلند در دو ژنوم شبیه و یا حتی یکسان باشند احتمالا نشان دهنده آن است که دو ناحیه، دو ژن هملوگ[۹] (ژن‌هایی که از یک جد یکسان تکامل یافته‌اند و کارایی یکسانی دارند) هستند.
- پیدا کردن نواحی تکراری در ژنوم: می‌دانیم که در ژنوم انسان زیررشته‌های یکسان بسیاری وجود دارند که به آن‌ها تکرار[۱۰] گفته می‌شود. روش‌های مقایسه توالی می‌توانند به پیدا کردن آن‌ها کمک کنند.

۱.۳ مسئله فاصله ویرایشی

در اینجا می‌خواهیم که تعریف دقیق‌تری از شباهت ارائه دهیم و به این منظور از یک مسئله در علوم کامپیوتر استفاده می‌کنیم. این مسئله به این صورت تعریف می‌شود که ورودی آن دو رشته S و T است. این دو رشته می‌توانند بیانگر توالی‌های زیستی مثل DNA باشند. هدف این است که با کمترین تعداد عملیات مجاز یکی از رشته‌ها به طور مثال رشته S را به رشته T تبدیل کنیم. عملیات مجاز نیز حذف کردن یک حرف، اضافه کردن یک حرف و تغییر یک حرف به حرف دیگر هستند.

به طور مثال با داشتن دو رشته $S = \text{actgacct}$ و $T = \text{accgt}$ می‌توانیم تعداد بهینه عملیات برای تبدیل رشته S به رشته T را به صورت زیر و با استفاده از روش برنامه نویسی پویا به دست آوریم.

		۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۱		_	a	c	t	g	a	c	c	t
۲	_	۰	۱	۲	۳	۴	۵	۶	۷	۸
۳	a	۱	۰	۱	۲	۳	۴	۵	۶	۷
۴	c	۲	۱	۰	۱	۲	۳	۴	۵	۶
۵	c	۳	۲	۱	۱	۲	۳	۳	۴	۵
۶	g	۴	۳	۲	۲	۱	۲	۳	۴	۵
۷	t	۵	۴	۳	۲	۲	۲	۳	۴	۴

[۶] Base pair
[۷] Amino acid
[۸] Human genome project
[۹] Homologous
[۱۰] Repeat

$$D(i,j) = \begin{cases} D(i-۱, j-۱) & i = j \\ \min[D(i, j-۱), D(i-۱, j), D(i-۱, j-۱)] + ۱ & i \neq j \end{cases}$$

پس از ساخت جدول با روش برنامه نویسی پویا، حداقل تعداد عملیات لازم برای تبدیل رشته S به رشته T در پایین‌ترین و سمت راست‌ترین خانه جدول قرار دارد. در این مسئله دیده می‌شود که برای تبدیل بهینه رشته S به رشته T به چهار عملیات مجاز نیاز داریم. حال اگر بخواهیم عملیات را به دست آوریم کافیست که از خانه‌ای که امتیاز تبدیل بهینه در آن است شروع کنیم و با توجه به قواعد گفته‌شده در فرمول بالا، به سمت چپ و بالا حرکت کنیم تا به بالاترین و چپ‌ترین خانه جدول برسیم. به طور مثال در اینجا با شروع از خانه $D(۷, ۱۰)$ و با توجه به اینکه حروف مربوط به این ستون و ردیف یکسان هستند متوجه می‌شویم که در اینجا نیاز به عملیاتی نیست و به خانه $D(۶, ۹)$ میرویم در اینجا می‌توانیم به دو خانه $D(۶, ۸)$ و $D(۵, ۸)$ برویم به طور مثال خانه $D(۶, ۸)$ را انتخاب می‌کنیم و به معنای حذف کردن حرف c از رشته S است سپس به خانه $D(۶, ۷)$ میرویم و حرف c را از رشته S حذف می‌کنیم و سپس به خانه $D(۶, ۶)$ رفته و حرف a را از رشته S حذف می‌کنیم. این بار به خانه $D(۵, ۵)$ رفته و این بدان معناست که حرف g از رشته S در این رشته باقی می‌ماند. در اینجا به خانه $D(۴, ۴)$ و حرف t از رشته S را به حرف c تبدیل می‌کنیم. سپس به ترتیب به خانه‌های $D(۳, ۳)$ و $D(۲, ۲)$ میرویم و حروف c و a از رشته S را تغییر نمی‌دهیم. پس از این مراحل رشته S با چهار عملیات به رشته T تبدیل می‌شود. همانطور که دیدیم می‌توانستیم از مسیرهای متفاوتی به خانه $D(۲, ۲)$ برسیم و این بدان معناست که راه حل مسئله فاصله ویرایشی یکتا نیست.

با روش گفته‌شده دیدیم که نیاز داریم دو بار جدول را پیمایش کنیم. یک بار برای پر کردن آن و یک بار به منظور مشخص کردن عملیات لازم. می‌توانستیم با صرف حافظه در هنگام پر کردن جدول، عملیات لازم را در جدولی دیگر نگه داریم و زمان اجرای الگوریتم را کاهش دهیم. در هر دو صورت، اگر طول یکی از رشته‌ها m و طول دیگری n باشد زمان اجرا و حافظه مصرفی از مرتبه $O(mn)$ خواهندبود.

۲.۳ مسئله هم‌ترازی ساده

پاسخ مسئله فاصله ویرایشی را می‌توان به فاصله بین دو رشته تعبیر کرد به طوریکه اگر دو رشته یکسان به عنوان ورودی به آن بدهیم، پاسخ صفر را خواهد برگرداند. می‌توان مسئله دیگری تحت عنوان مسئله هم‌ترازی ساده عنوان کرد که میزان شباهت دو رشته را مشخص می‌کند. ورودی این مسئله نیز دو رشته S و T است که می‌توانند رشته‌های زیستی باشند. خروجی مسئله یه جدول است که سطر اول آن یکی از رشته‌ها به طور مثال رشته S است که تعدادی «ـ» به آن اضافه شده‌است و سطر دیگر رشته دیگر است که به آن هم تعدادی «ـ» اضافه شده‌است. هدف مسئله نیز آن است که جدول خروجی بیشترین تعداد ستون مساوی را داشته باشد.

۳.۳ الگوریتم Needleman-Wunsch

این الگوریتم یک روش حل مسئله هم‌ترازی ساده است که بر پایه برنامه نویسی پویا است. ورودی آن دو رشته S و T و یک ماتریس شباهت حروف $\delta : (\Sigma \cup \{-\})^۲ \to R$ هستند. در این الگوریتم $V(i,j)$ بیشترین امتیاز هم‌ترازی بین $S[۱..i]$ و $T[۱..j]$ است. این الگوریتم از رابطه بازگشتی زیر برای پر کردن جدول برنامه نویسی پویا استفاده می‌کند.

$$V(i,j) = \max \begin{cases} V(i-۱, j-۱) + \delta(S[i], T[j]) & \text{match/mismatch} \\ V(i-۱, j) + \delta(S[i], -) & \text{delete} \\ V(i, j-۱) + \delta(-, T[j]) & \text{insert} \end{cases}$$

برای حالت پایه نیز از روابط زیر استفاده می‌کند.

$$\begin{cases} V(۰,۰) = ۰ \\ V(۰,j) = V(۰, j-۱) + \delta(-, T[j]) & \text{times j Insert} \\ V(i,۰) = V(i-۱, ۰) + \delta(S[i], -) & \text{times i Delete} \end{cases}$$

به طور مثال با داشتن دو رشته $\mathrm{S} = \mathrm{acaatcc}$ و $\mathrm{T} = \mathrm{agcatgc}$ و ماتریس شباهت δ به صورت زیر، می‌توانیم هم‌ترازی را توسط الگوریتم Needleman-Wunsch انجام دهیم.

δ					
	_	A	C	G	T
_		_۱	_۱	_۱	_۱
A	_۱	۲	_۱	_۱	_۱
C	_۱	_۱	۲	_۱	_۱
G	_۱	_۱	_۱	۲	_۱
T	_۱	_۱	_۱	_۱	۲

	۱	۲	۳	۴	۵	۶	۷	۸	۹
۱		_	A	G	C	A	T	G	C
۲	_	۰	_۱	_۲	_۳	_۴	_۵	_۶	_۷
۳	A	_۱	۲	۱	۰	_۱	_۲	_۳	_۴
۴	C	_۲	۱	۱	۳	۲	۱	۰	_۱
۵	A	_۳	۰	۰	۲	۵	۴	۳	۲
۶	A	_۴	_۱	_۱	۱	۴	۴	۳	۲
۷	T	_۵	_۲	_۲	۰	۳	۶	۵	۴
۸	C	_۶	_۳	_۳	۰	۲	۵	۵	۷
۹	C	_۷	_۴	_۴	_۱	۱	۴	۴	۷

در این الگوریتم نیاز است که به هنگام پر کردن جدول امتیاز هم‌ترازی، مسیری که هر خانه از جدول از کدام خانه قبل از خودش محاسبه شده‌است را در جدولی دیگر نگهداری کنیم و هنگامیکه جدول امتیاز پر شد از پایین‌ترین و راست‌ترین خانه جدول مسیرها شروع کنیم و به سمت خانه اول جدول حرکت کنیم و عملیات لازم را بر روی رشته‌ها اعمال کنیم تا هم‌ترازی را به دست آوریم. به عنوان نمونه هم‌ترازی دو رشته S و T فوق به صورت زیر خواهدبود که امتیاز ۷ دارد.

S	A	_	C	A	A	T	C	C
T	A	G	C	A	_	T	G	C

اگر دو رشته ورودی طول‌های m و n داشته‌باشند، زمان اجرا و حافظه مصرفی این الگوریتم هر دو از مرتبه $O(mn)$ خواهندبود زیرا که پر کردن جدول به زمانی از مرتبه گفته‌شده و نگهداری مسیر هم‌ترازی نیز به حافظه‌ای از مرتبه فوق نیاز دارند. البته می‌توانیم با روش‌هایی، میزان حافظه مورد نیاز را تا حدی کاهش دهیم.

۴.۳ برابری مسئله فاصله ویرایشی و مسئله هم‌ترازی ساده

در این بخش می‌خواهیم نشان دهیم که مسئله‌های فاصله ویرایشی و هم‌ترازی ساده به نحوی معادل یکدیگر هستند. به این منظور قضیه‌ای را به این صورت تعریف می‌کنیم که اگر امتیاز هم‌ترازی بهینه a^* باشد و هزینه روش ویرایش بهینه e^* باشد، خواهیم داشت $e^* = |T_a| - a^*$ که در آن T_a طول یکی از رشته‌ها پس هم‌ترازی (طول رشته به علاوه «ـ»های اضافه) است. برای اثبات این قضیه به این صورت عمل می‌کنیم که قضیه را به دو بخش تقسیم می‌کنیم و سعی می‌کنیم که در ابتدا اثبات کنیم به ازای هر هم‌ترازی یک روش ویرایش خواهیم داشت به طوریکه $e = |T_a| - a$ و سپس اثبات می‌کنیم که برای هر روش ویرایش می‌توانیم یک هم‌ترازی ارائه دهیم به گونه‌ای که $a \geq |T_a| - e$.

برای بخش اول کافیست فرض کنیم که یک هم‌ترازی داریم و می‌خواهیم رشته بالایی در جدول هم‌ترازی را به رشته پایینی تبدیل کنیم. از سمت چپ جدول شروع به حرکت می‌کنیم. به ازای هر match از حرف رشته بالایی بدون تغییر عبور می‌کنیم. به ازای هر insertion کافیست که حرف از رشته پایینی را به رشته بالایی اضافه کنیم. به ازای هر deletion کافیست که حرف از رشته بالایی را حذف کنیم و در نهایت به ازای هر mismatch کافیست که حرف از رشته بالایی را به حرف از رشته پایینی تغییر دهیم. با روش ارائه شده، با داشتن هر هم‌ترازی می‌توانیم یک ویرایش داشته‌باشیم که تنها به تعداد indel و mismatch ها نیاز به عملیات مجاز از مسئله فاصله ویرایشی دارد. بنابراین بخش اول اثبات شد.

برای بخش دوم در ابتدا فرض می‌کنیم که روش ویرایش یک روش بهینه است. در اینصورت می‌توانیم از خروجی ویرایش برای تولید جدول هم‌ترازی استفاده کنیم. برای این منظور به این صورت عمل می‌کنیم که از سمت چپ رشته‌ای به رشته دیگر تبدیل شده‌است شروع به حرکت می‌کنیم. به ازای حروفی که تغییر نکرده‌اند در جدول هم‌ترازی match قرار می‌دهیم و حروف را در ستونی یکسان می‌نویسیم. به ازای حروفی که از رشته حذف شده‌اند کافیست که در جدول هم‌ترازی در ستون مربوطه حرف از رشته را بنویسیم و برای رشته دیگر در هم‌ترازی «ـ» قرار دهیم. برای حروفی که به رشته اضافه شده‌اند نیز دقیقا برعکس حالت قبل عمل می‌کنیم به اینصورت که حرف را برای رشته دیگری در جدول هم‌ترازی مینویسیم و برای رشته مورد نظر «ـ» قرار می‌دهیم و در نهایت برای حروفی که تغییر کرده‌اند کافیست که در جدول هم‌ترازی mismatch قرار دهیم. با این روش از روی روش ویرایش بهینه یک هم‌ترازی به دست می‌آید که امتیاز آن برابر با $|T_a| - e^*$ است. حال فرض می‌کنیم که روش ویرایش بهینه نیست به طور مثال به هنگام تغییر یک حرف به حرف دیگر، ابتدا حرف را به یک حرف میانی و سپس به حرف مورد نظر تبدیل می‌کند در این صورت میزان فاصله

ویرایشی بیشتر شده ولی در نهایت خروجی این روش ویرایشی بهینه خواهدبود و می‌توانیم با روش گفته‌شده جدول هم‌ترازی را بسازیم با این تفاوت که به دلیل زیاد شدن فاصله ویرایشی (e) و ثابت ماندن اندازه رشته پس از ساخت هم‌ترازی، امتیاز هم‌ترازی به صورت $a \geq |T_a| - e$ می‌شود.

مراجع

[Sun09] Wing-Kin Sung. *Algorithms in bioinformatics: A practical introduction.* CRC Press, 2009.