جلسه‌ی پیش درباره‌ی روش‌هایی برای کاهش پیچیدگی مکانی الگوریتم global alignment بحث شد (Hirshberg).

همچنین درباره‌ی راه‌های سرعت بخشیدن به الگوریتم روشی گفته شد (Four Russians).

حالت خاصی از مساله‌ی global alignment، مساله‌ی edit distance است، که با داشتن دو رشته، می‌خواهیم تعداد جاهایی که باید ویرایش شود تا رشته‌ی دوم از روی رشته‌ی اول بدست آید، پیدا کنیم. منظور از ویرایش یکی از عمل‌های insert، delete و یا تغییر است. در ادامه یکی از مسائل edit distance به نام k-difference global alignment را بیان می‌کنیم.

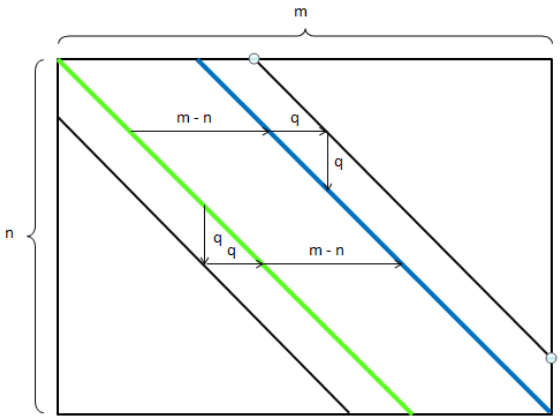
۱ K-difference global alignment

تعریف مساله: برای دو رشته‌ی x و y داده شده و مقدار ثابت k، بهترین هم‌ردیفی را پیدا کنید که حداکثر در مجموع k جابجایی و indel داشته باشیم. به عبارت دیگر:

$$d_{edit}(x,y) \leq k \qquad (۱)$$

به دنبال روشی هستیم که با فرض وجود شرط بالا، مساله را سریع‌تر حل کنیم.

این مساله را می‌توانیم با dynamic programming حل کنیم. ولی نکته این است که حرکت‌هایی که ضرب دری انجام می‌شود، بعضی وقت‌ها امتیاز ۱- می‌دهد. ولی حرکت‌هایی که ستونی یا سطری هستند، مطمئنا ۱- می‌دهد. بنابراین وقتی بخواهیم شرط (۱) را برقرار کنیم، در شکل زیر در پر کردن جدول، نباید از قسمتی که هاشور خورده بیرون بزنیم. زیرا در غیر این صورت شرط (۱) نقض می‌شود و نشان‌دهنده‌ی این است که مساله به ازای این k جواب ندارد و نیازی به ادامه دادن نیست. سوال این است که حاشیه را چطور تعیین کنیم. به عبارت دیگر در شکل (۱)، مقدار q چه باشد که اگر بیش‌تر از این مقدار از خط آبی دور شویم و سپس دوباره به آن برگردیم، به معنی نقض شرط (۱) است (به طور مشابه برای خط سبز استدلال می‌شود). با دقت کردن به شکل (۱) می‌توان مقدار q را از رابطه‌ی زیر بدست آورد. در این شکل m و n طول دو رشته هستند که فرض شده $m > n$ است. توجه شود که در هم‌ردیف شدن دو رشته، فاصله‌ی $m-n$ بین آن‌ها حتما وجود دارد.



شکل ۱

$$2q + (m-n) = k$$

حال فرض کنید k را از ابتدا نداریم. فقط گفته شده edit distance بین دو رشته x و y را بدست آورید. چه راه حلی پیشنهاد می کنید؟ روش به این صورت است که از $k = 1$ شروع می‌کنیم و جدول را پر می کنیم. اگر توانستیم جدولی بدست آوریم که مساله حل است، توجه شود که الگوریتم زمانی fail می‌کند که هاشور مجاز را رد کنیم. در این صورت دیگر ادامه نمی‌دهیم. یعنی مسیرهای غیر مجاز را kill کرده‌ایم. همچنین اگر فاصله‌ی نهایی که الگوریتم بدست می دهد بزرگتر از

نگارنده: فاطمه فرحناک غازانی
ایمیل: f.farahnak.g@gmail.com

k باشد، مساله به جواب نرسیده و fail شده است. در مرحله بعد k را افزایش می دهد مثلا k را مساوی ۲ قرار می دهیم. اگر مساله حل نشد $k = 4$، اگر نشد $k = 8$ و الی آخر. جوابی که بدست می آید، جواب بهینه برای global alignment است. در این حالت میزان محاسباتی که انجام می شود، برابر است با:

اگر k^* فاصله‌ی واقعی باشد:

$$O((\sum_{l=0}^{L} 2^l)\, m) = O(k^*\ m) \qquad k^* = 2^L$$

حال می‌خواهیم دو مساله را مطرح کنیم که دیگر هدف در آن‌ها، بدست آوردن global alignment نیست. این حالت را وقتی استفاده می کنیم که مثلا α-globin ها را گرفته ایم. و مطمئنیم که واقعا هومولوگ[۱] هستند و فقط می‌خواهیم جاهایی را که خراب شده اند، پیدا کنیم. ولی گاهی نیاز داریم کل ژنوم انسان و موش را بررسی کنیم و قسمت‌های شبیه به هم آن‌ها را تشخیص دهیم. این مساله local alignment نامیده می‌شود.

تعریف مساله: دو رشته x و y داده شده است. می‌خواهیم قسمت‌های شبیه به هم آن‌ها را پیدا کنیم. در واقع روش ساده به این صورت است که همه‌ی جفت زیر رشته‌ها را از دو رشته در نظر گرفته و بینشان global alignment انجام می‌دهیم.

تعداد زیر رشته‌ها در هر رشته برابر $O(m^2)$ و $O(n^2)$ است. با توجه به این که برای هر جفت هزینه‌ی محاسباتی $O(mn)$ است، هزینه‌ی محاسباتی کل برابر $O(m^3n^3)$ است.

به دنبال راه حل بهینه‌ای هستیم. آیا روشی با $O(mn)$ قابل ارائه است؟

البته توجه شود که اگر روشی با $O(mn)$ وجود داشته باشد، باز هم برای پیدا کردن هومولوگ های ژنوم انسان و موش که هر کدام طول 10^9 دارند قابل بدست آوردن نیست.

نکته ای که باید دقت شود این است که در امتیازدهی دو تا ماتریس داشتیم:

۱) ماتریسی که امتیاز هم ردیفی دو رشته را نشان می داد (با فرض این که دو رشته هومولوگ باشند).
۲) مدل دیگر از hypothesis testing بدست می‌آمد و می‌خواستیم به این سوال پاسخ دهیم که آیا دو رشته از یک جا آمده اند یا نه؟ امتیازدهی باید به گونه‌ای می‌بود که در انتها با مقایسه کردن با یک مقدار آستانه برای مثال اگر بزرگتر از صفر بود، می گفتیم از یک جا آمدند و برعکس.

حال می‌خواهیم مقداردهی به گونه‌ای باشد که اگر امتیازشان کمتر از صفر بود بگوییم از یک جا نیامده‌اند. بنابراین به دنبال آن‌هایی هستیم که امتیازشان اولا بزرگتر از صفر باشد و همچنین بیشترین امتیاز را داشته باشند. تحت این شرایط مسأله را حل می کنیم.

ابتدا یک مسأله تعریف می‌کنیم:

فرض کنید دو رشته ی $x[1 \dots i]$ و $y[1 \dots j]$ را در نظر می گیریم. می خواهیم suffix هایی از هر کدام را پیدا کنیم که global alignment شان بیشترین امتیاز را داشته و بزرگتر از صفر هم باشد (زیرا در غیر این صورت با هم رابطه‌ای ندارند). این بیشترین امتیاز را $v(i,j)$ می نامیم.

رویکردمان برای حل آن بازگشتی است. یعنی برای $v(i+1, j+1)$ داریم:

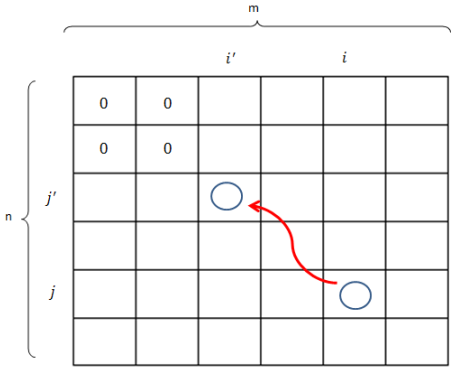
---

[۱] Homolog

$$v(i+1,j+1) = max \begin{cases} v(i,j) + \sigma(x_i,\ y_j) \\ v(i,j+1) + \sigma(x_{i+1}, -) \\ v(i+1,j) + \sigma(-, y_{j+1}) \\ 0 \end{cases}$$

با داشتن ماتریس مربوط به v، مسأله‌ی local alignment چگونه حل می‌شود و ارتباطشان چیست؟

به این صورت عمل می‌کنیم که در ماتریس v، بیشترین مقدار آن را بدست می‌آوریم سپس تا جایی به عقب می‌رویم که به صفر برسیم. یعنی در شکل زیر داریم:



شکل ۲

$$x[i' \dots i]$$
$$\downarrow$$
$$y[j' \dots j]$$

هم‌ردیفی بالا، بهترین هم‌ردیفی محلی است و امتیازش $v(i,j)$ است.

آیا همه‌ی هم‌ردیفی‌هایی که امتیازشان بزرگتر از صفر است، بدست می‌آوریم؟

خیر، زیرا بین همه‌ی امتیازها ماکزیمم می‌گیریم و فقط بهترین‌ها را نگه داشته و بقیه را از دست می‌دهیم.

توجه شود که تشخیص مکان‌های هومولوگ بودن دو ژنوم ۱۱نسان و موش از مرتبه‌ی 10^{18} خواهد بود که قابل اعمال نیست.

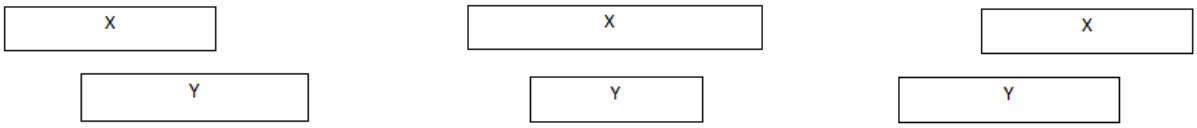
سؤال: آیا می توان سرعت را افزایش داد؟

خودتان فکر کنید!

مسأله‌ی دیگر که مطرح است، مسأله‌ی End space free global alignment است.

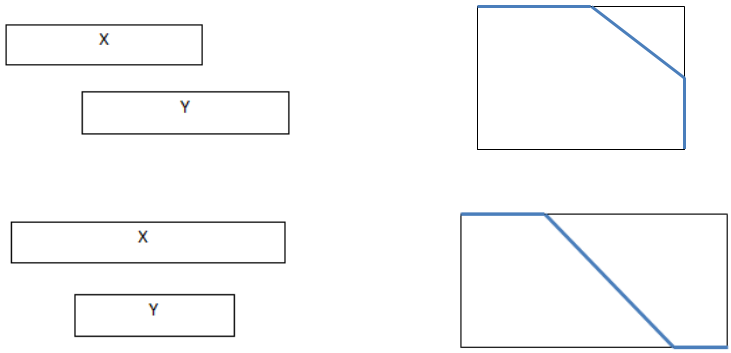
۲ End space free global alignment

در مسأله‌ی DNA sequencing که بعدا بحث خواهیم کرد، ما از یک DNA دو تا fragment را خوانده‌ایم و برایمان مهم است که با هم هم‌پوشانی دارند یا نه. اگر هم‌پوشانی داشته باشند، به هم می‌چسبانیم. یعنی مثلا شکل‌هایی مثل زیر:

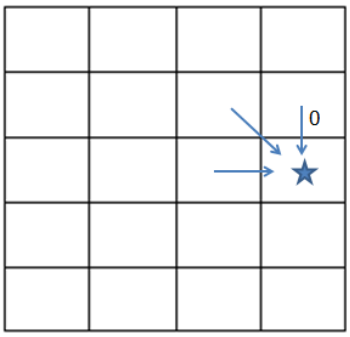


در این حالت برای GAP های ابتدایی و انتهایی امتیاز صفر در نظر گرفته و منفی نمی‌گذاریم. فقط برایمان مهم است که آیا این دو رشته باهم رابطه دارند یا نه و اگر دارند از کدام نوع است.

برای مثال برای دو شکل زیر، جدول معادل هر کدام نشان داده شده است:



در شکل زیر، برای پر کردن جدول در ابتدا سطر و ستون اول را مقدار صفر مقداردهی اولیه می‌کنیم. در سطر و ستون آخر نیز، مثلا برای پر کردن خانه‌ی ★ از سه طریق باید عمل شود. ولی برای حالتی که از خانه‌ی بالایی‌اش استفاده می‌شود، امتیاز هم‌ردیف کردن را صفر در نظر می‌گیریم (به جای منفی). در انتها در سطر آخر و ستون آخر خانه‌ای را انتخاب می‌کنیم که بیشترین امتیاز را در خود دارد و زودتر پدیدار شده و از آن‌جا به عقب بر می‌گردیم.



شکل ۳

$$v(i^*,m) = \arg\max_{1\le j\le m} v(i,m)$$

$$v(n,j^*) = \arg\max_{1\le j\le n} v(n,j)$$

۳ بحث‌هایی درباره‌ی GAP

در این قسمت می‌خواهیم بحثی درباره‌ی GAP داشته باشیم. تا این‌جا وقتی یک GAP اتفاق می‌افتاد مثل جابجایی با آن رفتار کرده و جریمه‌ی آن را لحاظ می‌کردیم. ولی واقعیت این است که مثلا سه تا از حرف‌های ACGT باهم Drop می‌شوند و درست نیست ۳ تا جریمه‌ی مستقل از هم در نظر بگیریم.

ممکن است یک جا GAP بزرگ داشته باشیم که جریمه‌ی آن متفاوت محاسبه می‌شود. مثلا یک امتیاز GAP opening داریم یک امتیاز GAP extension که هم‌زمان باید در نظر گرفته شود.

X_1	X_2	X_3

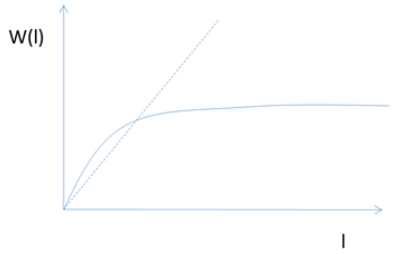
Y_1	–	Y_3

اگر داشته باشیم:

$$S = \lg\frac{P(x,y)}{P(x)P(y)} = \lg\frac{P(x1,y1)}{P(x1)P(y1)} - \underbrace{GAP - penalty}_{W(l)} + \lg\frac{P(x3,y3)}{P(x3)P(y3)}$$

در حالتی که فرض iid در نظر گرفته شود (تک تک indel ها مستقل از هم رخ داده اند) w(l) به صورت خطی است. ولی در واقعیت منحنی مشابه شکل زیر داریم. یعنی اینکه ۲۰ تا GAP اتفاق بیافتد یا ۳۰ تا خیلی با هم فرق ندارد.

یعنی برای جریمه‌ی GAP ها یک تابعی از طول GAP را داریم.



حال فرض کنید دو رشته‌ی x و y داده شده و هدف پیدا کردن بهترین هم ردیفی است.

$$V(i,j) = Max\{G(i,j), F(i,j), E(i,j)\}$$

که هر کدام از روابط زیر بدست می‌آید:

x_1	...	x_i
y_1	...	y_j

$$G(i,j) = V(i-1, j-1) + \delta(x_i, y_j)$$

x_1	...	x_i	–
y_1	...	y_k	... y_j

$$E(i,j) = \max_{0 \le k \le j-1}\{G(i,k) - w(j-k)\}$$

x_1	...	x_l	...	x_i
y_1	...	y_j	–	

$$F(i,j) = \max_{0 \le l \le i-1}\{G(l,j) - w(i-l)\}$$

برای مثال در محاسبه ی $E(i,j)$ بین امتیاز مربوط به هم‌ردیف شدن حرف‌های رشته اول با تعداد GAP های مختلف با طول 0 تا $j-1$ با بیشترین امتیاز را بدست می آورد.

به نظر می رسد شکل کشیده شده در بالا برای $W(l)$ مفهوما درست نیست زیرا GAP های مستقل، $W(l)$ بیشتری دارند و در نهایت طبق روابط، مورد علاقه‌تر هستند، در حالتی که هدفمان برعکس بود.

هزینه ی محاسبای این الگوریتم برابر $O(mn(m+n))$ است.

۱-۳ Affine GAP

یک تابع $W(l)$ برای GAP، affine GAP نام دارد که به صورت زیر است:

$$W(l) = w_g + l\, w_s$$

که یک رابطه ی خطی با l دارد، که w_g جریمه برای GAP opening و w_s جریمه برای GAP extention است. در این حالت می توان هزینه ی محاسبات را نسبت به حالت کلی تر کاهش داد و به $O(mn)$ رساند.

توجه شود که روابط مربوط به $E(i,j)$ و $F(i,j)$ را می‌توان به صورت زیر تغییر داد:

$$E(i,j) = max(E(i,j-1), v(i,j-1) - w_g) - w_s$$

$$F(i,j) = max(F(i-1,j), v(i-1,j) - w_g) - w_s$$

برای درک بهتر روابط، تعاریف هریک را مجددا بیان می‌کنیم:

$G(i,j)$: بهترین امتیاز برای وقتی است که x_i و y_j زیر هم قرار گرفته باشند.

$F(i,j)$: بهترین امتیاز برای وقتی است که زیر x_i حتما یک GAP است.

$E(i,j)$: بهترین امتیاز برای وقتی است که بالای y_j حتما یک GAP است.

بنابراین در حالتی که offline GAP فرض شد، برای مثال در رابطه‌ی جدید برای $E(i,j)$ اولا بالای y_j که حتما GAP است. سپس بین دو حالت زیر بیشترین امتیاز را بدست می‌آوریم:

۱) بالای y_{j-1} نیز GAP است. که امتیاز برابر $E(i,j-1) - w_s$ می شود.
۲) $v(i,j-1) - w_s - w_g$ که هم امتیاز GAP extention و هم GAP opening در نظر گرفته می‌شود.