۱ الگوریتم Needleman-wunsch (۱۹۷۰) – برنامه‌سازی پویا

فرض کنید X_i، i کاراکتر اول X را نظر بگیرید. Y_j را نیز به‌طور مشابه تعریف می‌کنیم. حال می‌خواهیم میزان شباهت X_i و Y_j را به دست آوریم. برای این کار فرض کنید، بهترین امتیاز هم‌ردیفی X_i و Y_j را $V(i,j)$ در نظر بگیرید.

در هم‌ترازی بهینه برای X_i و Y_j سه حالت زیر می باشد:

۱. آخرین بازها با یکدیگر هم‌تراز شده‌اند.
۲. آخرین باز X_i با یک indel هم‌تراز گردیده است.
۳. آخرین باز Y_j با یک indel هم‌تراز گردیده است.

به خاطر جمع پذیر بودن Scores مابقی هم‌ترازها بایستی بهینه باشند.

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

شرایط پایه‌ای:

$$V(0,0) = 0$$

$$V(i,0) = \sum_{k=1}^{i} \sigma(x_k, -); 1 \le i \le n$$

$$V(0,j) = \sum_{k=1}^{j} \sigma(x_j, -); 1 \le j \le m$$

به طور مثال تابع امتیاز زیر را در نظر بگیرید.

$$\sigma(a,b) = \begin{cases} +2, & a,b \in \Sigma \text{ and } a = b \\ -1, & o.w \end{cases}$$

حال می‌خواهیم بهترین هم‌ترازی دو رشته‌ی ACGTA و ATTA را به دست آوریم. طبق فرمول‌های داده شده، جدول زیر را پر می‌نماییم.

	_	A	C	G	T	A
_	0	-1	-2	-3	-4	-5
A	-1	2	1	0	-1	-2
T	-2	1	1	0	2	1
T	-3	0	0	0	2	1
A	-4	-1	-1	-1	1	4

نگارنده: بهنام حاتمی ورزنه
ایمیل: Behnam.hatami@gmail.com

که با توجه به نحوه‌ی پر شدن جدول دو هم‌ردیفی بهینه‌ی زیر وجود دارد.

A	C	G	T	A
A	-	T	T	A

A	C	G	T	A
A	T	-	T	A

۱-۱ پیچیدگی زمانی

پیچیدگی زمانی این الگوریتم برابر $O(mn)$ است. می‌توان با استفاده از الگوریتم four-Russions آن را بهبود بخشید.

۱-۲ پیچیدگی حافظه

اگر تنها مقدار هم‌ترازی بهینه مدنظر باشد، با حافظه‌ی $O(\min\{m,n\})$ می‌توان جواب را به دست آورد. زیرا تنها نیاز به نگهداری دو سطر یا دو ستون آخر است. اما اگر بخواهیم خود هم‌ترازی را نیز به‌عنوان خروجی بدهیم، $O(mn)$ حافظه برای نگهداری کل جدول نیاز است و دلیل آن نیز نیاز به نگهداری کل اشاره‌گرها در جدول است. می‌توان با الگوریتم Hirschberg آن را کاهش داد و مسئله را با زمان اجرای $O(mn)$ و حافظه ی $O(\min\{m,n\})$ حل نمود.

۲ الگوریتم Hirschberg (۱۹۷۷) – هم‌ترازی با حافظه‌ی خطی

همان‌طور که می‌دانیم، در پاره‌ای از موارد می‌توان زمان زیاد را تحمل کرد ولی میزان حافظه‌ی مصرفی زیاد را نمی‌توان.

مثال: ژنوم انسان 3×10^9 باز دارد و اگر بخواهیم یک رشته‌ی ۱۰۰ تایی را با این رشته هم‌ردیف نماییم، نیاز به حافظه‌ی 3×10^{11} خواهد بود، که در حقیقت ۳۰۰ گیگابایت حافظه است!!

سؤال: آیا می‌توان نیاز به نگهداری همه‌ی اشاره‌گرها را با افزایش میزان محاسبات کاهش داد.

جواب: بله، به‌طور مثال می‌توان سطر به سطر $V(i,j)$ را محاسبه نمود تا به سطر نهایی برسیم و تنها اشاره‌گرهای سطر آخر ذخیره نماییم. حال می‌توان با استفاده از اشاره‌گرها، عقب‌گرد کرد و به سطر قبلی برسیم. حال به یک مسئله‌ی کوچک‌تر رسیدیم که می‌توان به‌صورت بازگشتی آن را نیز حل کرد. میزان حافظه و زمان اجرای الگوریتم برابر است با:

معیار	میزان
حافظه	$O(min\{m,n\})$
زمان اجرا	$O(nm \times min\{m,n\})$

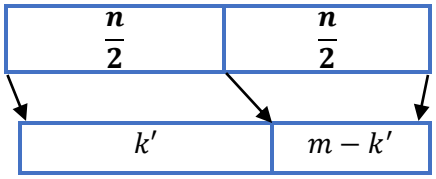
سؤال: آیا می‌توان زمان اجرای الگوریتم بالا را بهبود بخشید؟

تعریف: برای هر رشته‌ی α رشته α^r را به‌عنوان معکوس رشته معرفی می‌نماییم.

تعریف: برای دو رشته تعریف می‌نماییم، $V^r(i,j)$ به‌عنوان بهترین شباهت میان $X^r(1 \dots i)$ و $Y^r(1 \dots j)$ تعریف می‌کنیم.

لم: $V(n,m) = \max_{\{0\le k\le m\}} V\left(\frac{n}{2},k\right) + V^r\left(\frac{n}{2},m-k\right)$

طرف اول: برای هر $k' \in \{0,\dots,m\}$ می‌توانیم $Y(1\dots k')$ و $X\left(1\dots\frac{n}{2}\right)$ را هم‌ردیف نموده و سپس $X\left(\frac{n}{2}+1,\dots,n\right)$ را بها $Y(k'+1,\dots,m)$ هم‌ردیف نمود.



طبق تعریف امتیاز هم‌ردیفی اول $V\left(\frac{n}{2},k'\right)$ و هم‌ردیفی دوم $V^r\left(\frac{n}{2},m-k'\right)$ می‌باشند. بنابراین داریم:

$$V(n,m) \ge \max_{\{0\le k\le m\}} V\left(\frac{n}{2},k\right) + V^r\left(\frac{n}{2},m-k\right)$$

طرف دوم: بهترین هم‌ردیفی X و Y را در نظر بگیرید و k^* را آخرین حرفی از رشته‌ی Y در نظر بگیرید که با حرف $X\frac{n}{2}$ یا قبل‌تر از آن هم‌ردیف شده است در نظر بگیرید. بنابراین $X[1\dots\frac{n}{2}]$ با $Y[1\dots k^*]$ هم‌ردیف گردیده است و $X[\frac{n}{2}+1\dots n]$ و $Y[K^*+1\dots m]$ هم‌ردیف گردیده است. بنابراین داریم:

$$V(n,m) \le V\left(\frac{n}{2},k^*\right) + V^r\left(\frac{n}{2},m-k^*\right)$$

$$\le \max_{\{0\le k\le m\}}\left[V\left(\frac{n}{2},k\right) + V^r\left(\frac{n}{2},m-k\right)\right]$$

بنابراین بهترین هم‌ردیفی باید دقیقاً از $\left(\frac{n}{2},k^*\right)$ عبور نماید.

تعریف: L را مسیر بهینه‌ی هم‌ردیفی در نظر بگیرید که از خانه‌ی $(0,0)$ شروع گردیده و در خانه‌ی (n,m) پایان یابد.

تعریف: $L_{\{\frac{n}{2}\}}$ را یک زیر مسیر از L در نظر بگیرید که با آخرین حلقه‌ی L در ردیف $\frac{n}{2}-1$ آغاز گردیده و با اولین حلقه L در ردیف $\frac{n}{2}+1$ پایان می پذیرد.

لم: k^* ردیف $\frac{n}{2}$ در $O(nm)$ زمان و $O(m)$ فضا قابل محاسبه است. همچنین $L_{\{\frac{n}{2}\}}$ در همین محدودیت زمانی قابل محاسبه است.

اثبات: با اجرای الگوریتم برنامه‌سازی پویا برای X و Y و ادامه دادن تا ردیف $\frac{n}{2}$ می‌توانیم اشاره‌گرهای ردیف آخر را محاسبه نماییم. در این حالت $V(\frac{n}{2},k)$ برای تمام kها محاسبه می‌گردد و میزان حافظه‌ی $O(m)$ می‌باشد. به همین ترتیب X^r و Y^r را هم ردیف کرده و تا ردیف $\frac{n}{2}$ پیش می‌رویم. در این حالت $O(m)$ حافظه نیاز بوده و تمام مقادیر $V^r\left(\frac{n}{2},m-k\right)$ را برای تمام kها بدست می‌آوریم. حال $V(\frac{n}{2},k)$ و $V^r(\frac{n}{2},m-k)$ را با هم جمع نموده و مقدار ماکزیمم را پیدا می‌نماییم که همان K^* است. این عمل هم $O(m)$ زمان دارد.

حال با استفاده از اشاره‌گرها بازگشت می‌نماییم. با استفاده از اشاره‌گرهای اول به اولین خانه به نام K_1 در $\frac{n}{2}-1$ دسترسی می‌نماییم. با استفاده از اشاره‌گرهای دوم می‌توانیم به خانه‌ی K_2 در ردیف $\frac{n}{2}+1$ پی ببریم. این دو مسیر با یکدیگر زیر مسیر $L_{\{\frac{n}{2}\}}$ را می‌سازند.

درمجموع با زمان اجرای $O(nm)$ و حافظه‌ی $O(m)$ برای یافتن K_1 و K_2 و K^* لازم است. برای آنالیز زمانی فرض می‌نماییم که مقدار محاسبه برای پر کردن یک جدول برنامه‌نویسی پویا برابر است با cnm. (برای یک مقدار ثابت c) مثلاً برای محاسبه‌ی قسمت اول $\frac{cnm}{2}$ محاسبه و برای قسمت دوم(مسیر معکوس) $\frac{cnm}{2}$ لازم است و بنابراین سر جمع cnm زمان نیاز دارد.

پس از مرحله‌ی اول مسئله به دو زیر مسئله زیر تبدیل می‌گردد:

$$\begin{pmatrix} X\left[1 \dots \frac{n}{2}-1\right] \\ Y[1 \dots K_1] \end{pmatrix} and \begin{pmatrix} X\left[\frac{n}{2}+1 \dots n\right] \\ Y[K_2 \dots m] \end{pmatrix}$$

$$\frac{cn}{2}k^* + \frac{cn}{2}(m-k^*) = \frac{cnm}{2}$$

$$\frac{cn}{4}k^{**} + \frac{cn}{4}(k^*-k^{**}) + \cdots = \frac{cnm}{4}$$

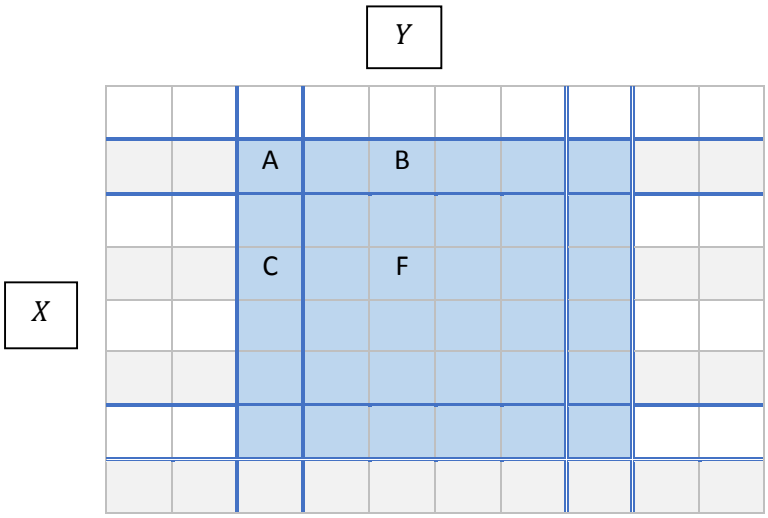
$$\sum_{i=1}^{\log n} \frac{cnm}{2^{i-1}} \leq 2cnm$$

زمان کل:

$$O(cnm)$$

۳ الگوریتم Four – Russians

برای سادگی فرض می‌نماییم که X و Y دارای طول یکسان n می‌باشند.



مطابق شکل بالا، E یک بخش رشته‌ی Y و D یک بخش رشته‌ی X است که مقابل قسمت رنگ‌شده آمده است. بدیهی است که V در ناحیه‌ی F فقط تابعی از مقدار V در A و B و C و مقدار رشته در E و D است.

تعریف: block function به تابع میان (A,B,C,D,E) به F را تابع بلوکی می‌نماییم. بدیهی است که مقدار سطر آخر و ستون آخر نیز تنها به (A,B,C,D,E) رابطه دارند به تابع میان این دو تابع بلوکی محدودشده اطلاق می‌گردد.

روش:

۱- ابتدا جدول برنامه‌ریزی پویا، را با بلوک‌های t پوشش می‌دهیم. که در آن، سطر نهایی هر بلوکی با سطر ابتدایی بلوک بعدی مشترک است. با فرض آن‌که $n = k(t-1)$ به‌طورکلی k^2، t^2-بلوک داریم.

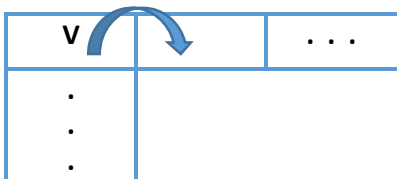
۲- می‌توانیم مقدار اولیه سطر و ستون صفر را در آوریم.

۳- می‌توانیم سطر و ستون بعدی را توسط تابع بلوکی محدودشده به دست آوریم و آن‌قدر ادامه دهیم تا در سلول (n, n) مقدار نهایی را بیابیم.

فرض نماییم که تابع بلوکی محدودشده را داشته باشیم. در این صورت برای یک ورودی خاص $O(t)$ لازم است که خروجی آن را بیابیم. در جمع $\Theta\left(\frac{n^2}{t^2}\right)$ بلوک وجود دارد. بنابراین کل زمان موردنیاز برابر است با $\Theta\left(\frac{n^2}{t}\right)$. حال اگر t را عددی در مرتبه $\log n$ قرار دهیم داریم، زمان موردنیاز، برابر است با $\Theta\left(\frac{n^2}{\log n}\right)$.

اگر از مدل Unit-Cost Ram استفاده کنیم، به دست آوردن خروجی در زمان ثابت قابل محاسبه است و درنتیجه زمان اجرای کل الگوریتم برابر است با $\Theta\left(\frac{n^2}{(\log n)^2}\right)$.

حال بایستی زمان محاسبات اولیه را دربیاوریم. زمان لازم برای هر نمونه $\Theta(t^2)$ است. اما در مورد تعداد نمونه‌ها هیچ تقریبی نداریم. طبق شکل زیر، اگر V را داشته باشیم، خانه‌ی بعدی حداکثر چه میزان اختلاف می‌تواند داشته باشد؟



فرض کنید هر خانه با دانستن خانه‌ی مجاور خود، حداکثر s حالت مختلف داشته باشد. آنگاه تعداد شرایط موجود برابر است با $s^{2(t-1)}$ و برای ورودی Σ^{2t} در نتیجه تعداد کل ورودی برابر است با $\Sigma^{2t}S^{2(t-1)}$ بنابراین کل زمان می شود

$$O(s^{2t}\Sigma^{2t}t^2)$$

و با داشتن $t = \frac{log_{s\Sigma}}{2}$ زمان محاسبه برابر است با $O[n(\log n)^2]$

میزان حافظه‌ی موردنیاز:

۱. حافظه‌ی حالت نگهداری مسیر $O(s^{2t}\Sigma^{2t}t)$

۲. حافظه‌ی نگهداری کل خانه‌های جدول $O(s^{2t}\Sigma^{2t}t^2)$

در چه صورت مقدار offset محدود می‌باشد؟

مثال:

$$\sigma(a,b) = \begin{cases} 2, & a,b \in \Sigma \; and \; a = b \\ -1, & o.w \end{cases}$$

که در آن گزینه‌ها حداکثر یکی کمتر و حداکثر ۳ تا بیش‌تر از مجاور خود هستند. که در کل ۵ گزینه را به ما می‌دهد.

مثال: Edit distance مطابق فرمول زیر تعریف می‌شود:

$$\sigma(a,b) = \begin{cases} 0, & a,b \in \Sigma \; and \; a = b \\ -1, & o.w \end{cases}$$

و برای حالتی که بخواهیم جریمه را کمینه کنیم از رابطه‌ی زیر استفاده می‌کنیم:

$$\sigma(a,b) = \begin{cases} 0, & a,b \in \Sigma \; and \; a = b \\ 1, & o.w \end{cases}$$

لم: در هر سطر و ستون و یا قطر جدول برنامه‌ریزی پویا، دو خانه مجاور دارای اختلاف حداکثر ۱ هستند.

اثبات:

طبق رابطه‌ای که برای پر کردن جدول استفاده می‌کنیم، داریم:

$$D(i,j) \leq D(i-1,j) + 1$$

حال می‌خواهیم کران پایین را حساب کنیم. اگر X_i با Y_l هم ردیف شده باشد، داریم:

$$D(i,j) \geq D(i-1,l-1) + (j-l)$$

و از طرفی داریم:

$$D(i-1,j) \leq \; D(i-1,l-1) + (j-l+1)$$

و در نتیجه داریم:

$$D(i-1,j) \leq \; D(i,j) + 1$$

و در نتیجه داریم:

$$D(i-1,j) - 1 \leq D(i,j) \leq D(i-1,j) + 1$$