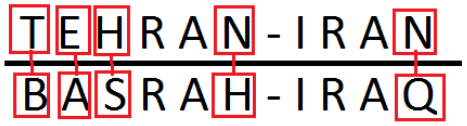
درس طراحی الگوریتم
دانشکده ریاضی. دانشگاه صنعتی خواجه نصیرالدین طوسی. پاییز ۱۴۰۱
کتاب مرجع: طراحی الگوریتم. جان کلاینبرگ و اوا تاردش

۱ مسئله همترازسازی توالی ها sequence alignment

یک توالی دنباله ای از عناصر است.

$$a_1, a_2, a_3, \ldots, a_n$$

برای مثال رشته یک توالی از حروف است. یک رشته دی ان ای، یک توالی از پروتئینهای مختلف است که هر کدام با یک کد مشخص نشان داده میشود.
در برخی از زمینه های کاربردی مثل بیوانفورماتیک، میخواهیم میزان تشابه یا تفاوت توالی های مختلف را بدست آوریم. برای این کار نیاز به یک معیار تشابه یا فاصله بین توالی ها داریم.
یک معیار ساده برای فاصله توالی های با طول یکسان فاصله همینگ Hamming است. فاصله همینگ دو توالی X و Y که طول یکسان دارند، برابر با تعداد حروف متناظر در توالیهای X و Y است که با هم متفاوت هستند. اینجا حروفی که اندیس یکسان دارند با هم مقایسه میشوند.



فاصله همینگ = 5

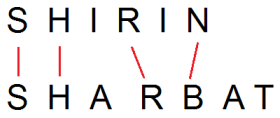
فاصله همینگ در برخی از کاربردها (برای توالی های با طول یکسان) مناسب است اما یک مشکل اصلی دارد. اگر جایی در توالی یک شیفت اتفاق بیافتد، برای مثال یک کاراکتر جدید در رشته درج شود یا حذف شود، ممکن است باعث ایجاد یک توالی با فاصله بسیار زیاد شود. به نمونه زیر دقت کنید. با وجود اینکه دو توالی زیر تا حد زیادی مشابه هستند، فاصله همینگ آنها بالاست.

1 2 3 4 5 6 7

7 1 2 3 4 5 6

برای رفع این ایراد از معیارهای دیگر برای فاصله توالیها استفاده میشود که خیلی حساس به درج و حذف کاراکترها نباشند. یک معیار متفاوت از همترازسازی alignment کاراکترهای دو رشته استفاده میکند. در فاصله همینگ، تنها زوج حروفی که در اندیس یکسان هستند با هم مقایسه میشوند. میتوانیم این محدودیت را برداریم و دو حرف که موقعیت (اندیس) متفاوت در دو رشته دارند با هم مقایسه شوند. علاوه بر این، اجازه

میدهیم که یک حرف در همترازسازی اصلا استفاده نشود و در عوض جریمه ای برای آن میپردازیم. البته اینجا یک شرط اساسی وجود دارد. اگر دو کاراکتر $X[i]$ و $Y[j]$ همتراز شوند، دیگر نمیتوانیم دو کاراکترهای $X[i']$ و $Y[j']$ را همتراز کنیم اگر $i' > i$ و $j' < j$. به عبارت دیگر همترازسازی ضربدری را اجازی نمیدهیم. به شکل زیر دقت کنید.



یک همترازسازی معتبر

یک همترازسازی نامعتبر

با توجه به توصیفات بالا، یک مسئله بهینه سازی تعریف میشود. دو توالی X با طول n و توالی Y با طول m داده شده است. میخواهیم یک همترازسازی معتبر با کمترین هزینه بین X و Y پیدا کنیم. هزینه این همترازسازی برابر با فاصله بین X و Y است.

هزینه همترازسازی معتبر M بین X و Y بصورت زیر تعریف میشود.

- اگر $(i,j) \in M$ یعنی $X[i]$ و $Y[j]$ با همتراز شده اند و $X[i] \neq Y[j]$ آنگاه جریمه β اعمال میشود. اگر $X[i] = Y[j]$ آنگاه جریمه ای اعمال نمی شود.
- اگر حرف $X[i]$ با هیچ حرفی از Y همتراز نشود، جریمه δ اعمال میشود. به همین ترتیب، اگر حرف $Y[j]$ با هیچ حرف X همتراز نشود، جریمه δ اعمال میشود.
- در نهایت هزینه همترازسازی M برابر با مجموع جریمه های اعمال شده است.

برای مثال، در شکل زیر دو همترازسازی متفاوت بین دو رشته نشان داده شده است.

M1

M2

داریم

$$cost(M1) = 5\delta + \beta, \qquad cost(M2) = \delta + 3\beta$$

فاصله بین دو توالی X و Y برابر با هزینه بهترین همترازسازی معتبر بین دو توالی است.

$distance(X,Y) = \{cost(M) \mid$ وقتی M یک همترازسازی معتبر بین X و Y است $\}$

۱.۱ برنامه نویسی پویا برای پیدا کردن یک همترازسازی بهینه

با استفاده از تفکر بازگشتی، میتوانیم یک راه حل الگوریتمی بر اساس تکنیک برنامه نویسی پویا برای پیدا کردن همترازسازی بهینه طراحی کنیم. فرض کنید M همترازسازی بهینه بین X و Y باشد.

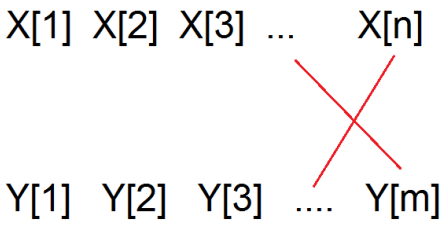
میتوانیم ادعا کنیم از سه حالت زیر حداقل یکی اتفاق می افتد. دقت کنید حالات دوم و سوم ممکن است هر دو اتفاق بیافتند.

۱. $(n,m) \in M$
به عبارت دیگر کاراکترهای $X[n]$ و $Y[m]$ با هم همتراز شده اند.

۲. $X[n]$ همتراز نشده است.

۳. $Y[m]$ همتراز نشده است.

توجه کنید اگر $X[n]$ و $Y[m]$ هر دو با کاراکترهای دیگر همتراز شده باشند آنگاه یک حالت ضربدری ایجاد میشود و لذا M معتبر نخواهد بود.



تعریف: مقدار $OPT(i,j)$ برابر با هزینه همترازسازی بهینه بین دو توالی $X[1]\dots X[i]$ و $Y[1]\dots Y[j]$ است.

با توجه به توصیفات و تعریف بالا داریم

$$OPT(n,m) = \min\{OPT(n-1,m)+\delta, OPT(n,m-1)+\delta, OPT(n-1,m-1)+cost(X[n],Y[m])\}$$

در اینجا داریم

$$cost(X[n],Y[m]) = \begin{cases} \beta & X[n] \neq Y[m] \\ 0 & X[n] = Y[m] \end{cases}$$

بطور کلی برای هر i,j میتوانیم بنویسیم

$$OPT(i,j) = \min\{OPT(i-1,j)+\delta, OPT(i,j-1)+\delta, OPT(i-1,j-1)+cost(X[i],Y[j])\}$$

عبارت بازگشتی بالا نشان میدهد حل زیر مسئله $OPT(i,j)$ با استفاده از حل زیرمسائل کوچکتر $OPT(i-1,j)$ و $OPT(i,j-1)$ و $OPT(i-1,j-1)$ قابل انجام است.

جدول OPT را میتوانیم سطر به سطر از پایین به بالا یا ستون به ستون از چپ به راست پر کنیم. این کار در زمان $O(nm)$ قابل انجام است. مانند موارد قبل که از تکنیک برنامه سازی پویا بررسی کردیم، جواب بهینه M از خود جدول OPT قابل استخراج است.