۱. **الف)** جهش در یک ژن می‌تواند باعث تغییر در عملکرد یا ساختار پروتئینی شود که آن ژن کد می‌کند. در سلول‌های عضلانی، پروتئین‌هایی خاص نقش‌های کلیدی در بازسازی و ترمیم دارند، از جمله پروتئین‌هایی که در چرخه سلولی، تمایز سلولی، و فرآیندهای التهابی نقش دارند. اگر جهش باعث تغییر یا از کار افتادن این پروتئین‌ها شود، می‌تواند روند بازسازی عضلات را مختل کند. چندین حالت ممکن است اتفاق بیفتد:

کاهش تولید یا تغییر ساختار پروتئین‌های کلیدی: اگر جهش به نحوی باشد که مانع تولید صحیح پروتئین‌ها یا منجر به تولید پروتئین‌هایی با ساختار نادرست شود، این پروتئین‌ها نمی‌توانند عملکرد طبیعی خود را در چرخه سلولی و ترمیم عضلات ایفا کنند. به عنوان مثال، پروتئین‌هایی که در فرایند تقسیم سلولی و تمایز سلول‌های بنیادی به سلول‌های عضلانی نقش دارند، در صورت اختلال، باعث کاهش تعداد سلول‌های جدید عضلانی می‌شوند.

فعالیت غیر طبیعی در سیگنالینگ سلولی: پروتئین‌ها همچنین در مسیرهای سیگنالینگ که به سلول‌ها اطلاع می‌دهند که باید تقسیم یا ترمیم شوند، نقش دارند. جهش در ژن ممکن است باعث شود که این مسیرهای سیگنالینگ به درستی فعال نشوند و یا سیگنال‌های اشتباهی ارسال کنند. این موضوع می‌تواند باعث از کار افتادن فرآیند بازسازی شود.

ایجاد پروتئین‌های سمی یا غیرعملکردی: گاهی جهش‌ها منجر به تولید پروتئین‌هایی می‌شوند که نه تنها عملکرد طبیعی ندارند، بلکه ممکن است برای سلول‌ها نیز سمی باشند. تجمع این پروتئین‌ها می‌تواند باعث مرگ سلولی یا جلوگیری از فرایند ترمیم شود.

نقص در فرایندهای ضروری دیگر مرتبط با بازسازی: برخی از پروتئین‌های تولید شده در عضلات، علاوه بر تقسیم سلولی، در کاهش التهاب و تحریک ترمیم نقش دارند. جهش در ژن می‌تواند منجر به نقص در این پروتئین‌ها و در نتیجه عدم تنظیم صحیح التهاب پس از آسیب عضلانی شود.

بنابراین، جهش در ژنی که مسئولیت تولید پروتئین‌های ضروری در عضلات را دارد، می‌تواند بازسازی عضلات را مختل کند و موجب کاهش یا توقف روند ترمیم عضلات پس از آسیب شود.

ب) در این وضعیت، چون ژن جهش‌یافته در کروموزوم X قرار دارد و بیماری به صورت وابسته به جنسیت منتقل می‌شود، وضعیت هر یک از فرزندان (پسر و دختر) به وجود یا عدم وجود نسخه سالمی از ژن در کروموزوم X دیگر بستگی دارد.

پسر: پسرها دارای یک کروموزوم X و یک کروموزوم Y هستند، و بنابراین اگر ژن جهش‌یافته را از یکی از والدین (معمولاً مادر) به ارث ببرند، این تنها نسخه از ژن خواهد بود که دارند. در نتیجه، پسر بیماری

را به‌طور کامل بروز خواهد داد، زیرا نسخه سالمی از ژن در کروموزوم Y وجود ندارد که بتواند نقص ژن جهش‌یافته در کروموزوم X را جبران کند.

دختر: دخترها دو کروموزوم X دارند، و اگر ژن جهش‌یافته را از یکی از والدین به ارث ببرند، معمولاً روی یک کروموزوم X این جهش را دارند و روی کروموزوم X دیگر نسخه سالمی از ژن را. در این حالت، به دلیل وجود نسخه سالم، دختر به‌طور کامل بیمار نمی‌شود و معمولاً ناقل بیماری خواهد بود؛ یعنی ممکن است برخی نشانه‌های خفیف را بروز دهد، اما به شدت پسر مبتلا نخواهد شد، زیرا نسخه سالم ژن تا حدی عملکرد را حفظ می‌کند.

۲. **الف)** جهش در ژن کدکننده DNA پلی‌مراز که باعث کاهش دقت این آنزیم در جایگزینی نوکلئوتیدهای صحیح می‌شود، می‌تواند به افزایش خطاهای تکثیری منجر شود. در فرآیند تکثیر DNA، DNA پلی‌مراز مسئول قرار دادن نوکلئوتیدهای مکمل در برابر هر نوکلئوتید از رشته قالب است. کاهش دقت این آنزیم به این معناست که احتمال وقوع جهش‌های نقطه‌ای (مانند جایگزینی یک نوکلئوتید با نوکلئوتید دیگر) در طول توالی جدید DNA افزایش می‌یابد.

وقوع این جهش‌ها می‌تواند بر توالی ژنی تأثیر بگذارد و در نتیجه عملکرد پروتئین‌هایی که از روی این ژن‌ها ترجمه می‌شوند را نیز دچار اختلال کند. بسته به نوع و محل جهش، تغییرات می‌توانند پیامدهای متفاوتی داشته باشند:

<u>تغییرات خاموش (Silent Mutations)</u>: در برخی موارد، جهش ممکن است منجر به تغییر کدون شود، اما چون بعضی از کدون‌ها کد کننده همان اسید آمینه هستند، پروتئین تولیدی تغییر نمی‌کند. این تغییرات معمولاً اثری بر عملکرد پروتئین ندارند.

<u>تغییرات معنادار (Missense Mutations)</u>: گاهی جهش موجب تغییر در کدون شده و اسید آمینه‌ای متفاوت در پروتئین جایگزین می‌شود. این تغییرات می‌توانند منجر به تغییر ساختار یا عملکرد پروتئین شوند. به عنوان مثال، ممکن است پروتئین دیگر قادر به انجام وظیفه‌ی اصلی خود نباشد یا در عملکرد آن اختلال ایجاد شود.

<u>تغییرات بی‌معنا (Nonsense Mutations)</u>: اگر جهش به تولید یک کدون توقف (Stop codon) منجر شود، ممکن است زودتر از موعد، ترجمه متوقف گردد. این امر منجر به تشکیل یک پروتئین ناقص و احتمالاً غیرفعال می‌شود که در نتیجه، عملکرد اصلی خود را از دست می‌دهد.

تغییرات قاب خوانشی (Frameshift Mutations): اگر یک یا چند نوکلئوتید حذف یا اضافه شود، تغییر قاب خوانشی ایجاد می‌شود که در نتیجه، تمامی کدون‌ها پس از محل جهش تغییر خواهند کرد. این امر می‌تواند منجر به تولید پروتئینی با ساختار و عملکرد کاملاً متفاوت و احتمالاً غیرفعال شود.
در مجموع، افزایش خطاهای تکثیری به دلیل جهش در DNA پلی‌مراز می‌تواند منجر به تجمع جهش‌های مضر در سلول‌ها شود که در طولانی‌مدت ممکن است عملکرد طبیعی پروتئین‌ها و فرآیندهای سلولی را به خطر بیندازد و حتی به بیماری‌ها و مشکلات ژنتیکی منجر شود.

ب) اگر جهشی در ژن کدکننده DNA هلیکاز رخ دهد و این آنزیم قادر نباشد با سرعت مناسب دو رشته DNA را باز کند، فرآیند تکثیر DNA با اختلالاتی مواجه خواهد شد. DNA هلیکاز نقش بسیار حیاتی در باز کردن دو رشته DNA و جدا کردن آن‌ها دارد تا بتواند امکان دسترسی آن‌ها برای فعالیت DNA پلی‌مراز فراهم کند. حال اگر هلیکاز به درستی یا با سرعت مناسب این کار را انجام ندهد، مشکلات زیر ممکن است ایجاد شوند:
توقف یا کاهش سرعت تکثیر: DNA پلی‌مراز برای ادامه تکثیر نیاز دارد که دو رشته DNA باز شده و به حالت تک رشته‌ای درآیند. در صورتی که DNA هلیکاز نتواند این دو رشته را به موقع و با سرعت کافی باز کند، DNA پلی‌مراز به علت عدم دسترسی به رشته قالب، نمی‌تواند به طور پیوسته عمل کند. این وضعیت ممکن است منجر به توقف یا کاهش سرعت کل فرآیند تکثیر DNA شود.
ایجاد ناپایداری ساختاری: باز نشدن مناسب دو رشته DNA می‌تواند موجب ایجاد تنش و ناپایداری ساختاری در DNA شود. در محل‌هایی که رشته‌ها به خوبی باز نمی‌شوند، احتمال ایجاد حلقه‌ها یا پیچیدگی‌های اضافی در ساختار DNA بیشتر می‌شود، که می‌تواند روند تکثیر را دشوارتر و ناپایدارتر کند.
تشکیل قطعات ناکامل (Okazaki Fragments): در رشته تأخیرکننده (lagging strand)، تکثیر به صورت قطعات اوکازاکی انجام می‌شود و نیاز است که DNA به طور منظم باز شود تا این قطعات بتوانند به هم متصل شوند و یک رشته‌ی پیوسته ایجاد کنند. با کاهش کارایی DNA هلیکاز، رشته تأخیرکننده به درستی باز نمی‌شود و این امر می‌تواند باعث شود که قطعات اوکازاکی به درستی به هم متصل نشوند یا ناقص بمانند، و در نهایت رشته DNA کامل و سالمی تشکیل نگردد.
افزایش احتمال خطاهای تکثیری: باز نشدن صحیح دو رشته می‌تواند باعث خطاهای بیشتری در اتصال نوکلئوتیدها شود. DNA پلی‌مراز برای اضافه کردن نوکلئوتیدهای مکمل نیاز به یک رشته قالب مشخص دارد، و هرگونه نقص در باز شدن دو رشته باعث می‌شود که این فرآیند با دقت کمتری انجام گیرد و خطاهایی در توالی ایجاد شود.

در نتیجه، جهش در DNA هلیکاز و کاهش کارایی آن، نه تنها فرآیند تکثیر را کند می‌کند بلکه به تولید DNAهای ناکامل، ناقص یا دارای خطا منجر می‌شود. این نواقص در ساختار DNA ممکن است بر عملکرد ژن‌های جدید تأثیر بگذارد و زمینه‌ساز مشکلات ژنتیکی یا بیماری‌ها شود.

۳. چالش‌ها:

شرایط پایدارسازی در دماهای بالا: چون آنزیم‌های این باکتری‌ها در دماهای بالا فعال‌اند، خالص‌سازی آن‌ها در دماهای پایین‌تر ممکن است موجب از دست رفتن فعالیت یا تغییر ساختار آنزیمی شود. بنابراین، باید شرایط آزمایشگاه را به گونه‌ای تنظیم کرد که آنزیم‌ها در دمایی نزدیک به دمای بهینه عملکردشان استخراج و خالص‌سازی شوند. در صورت نیاز به کار در دماهای پایین‌تر، اضافه کردن تثبیت‌کننده‌هایی مانند گلیسرول، یون‌های فلزی، یا پروتئین‌های کمکی برای پایدارسازی آنزیم‌ها مفید است.

پایداری و فعالیت آنزیم در محیط‌های مختلف شیمیایی: در مراحل مختلف خالص‌سازی، از مواد شیمیایی خاصی مانند شوینده‌ها و نمک‌ها استفاده می‌شود که ممکن است روی پایداری و فعالیت آنزیم‌ها اثر منفی بگذارند. برای کاهش این خطر، لازم است شرایط خالص‌سازی مانند pH و غلظت مواد شیمیایی بهینه‌سازی شود.

مقاومت آنزیم به دناتوره شدن در طی مراحل خالص‌سازی: با وجود مقاوم بودن این آنزیم‌ها به حرارت، ممکن است در حین فرایندهای خالص‌سازی (مثلاً در معرض فشار یا محیط‌های با اسیدیته بالا) دچار دناتوره شدن شوند. به همین دلیل، استفاده از بافرهای مناسب و حفظ محیط کشت در طی مراحل جداسازی ضروری است.

استفاده از روش‌های خاص برای جداسازی پروتئین‌های مقاوم به حرارت: برخی از روش‌های کروماتوگرافی مانند کروماتوگرافی حرارتی یا فاز معکوس (reverse phase) برای خالص‌سازی پروتئین‌های مقاوم به حرارت مؤثرند. در این روش‌ها می‌توان پروتئین‌ها را با تنظیم دما و محیط مناسب به خوبی جدا کرد.

راهکارها:

استفاده از افزودنی‌های تثبیت‌کننده: افزودنی‌هایی مثل یون‌های فلزی، گلیسرول، یا ساکاروز می‌توانند پایداری آنزیم را در شرایط متغیر بهبود دهند.

کنترل شرایط دما و pH: باید شرایط آزمایشگاهی تا حد امکان به شرایط طبیعی باکتری‌ها نزدیک باشد تا از فعالیت و پایداری آنزیم‌ها در مراحل مختلف اطمینان حاصل شود.

تست‌های فعالیت آنزیمی در مراحل مختلف خالص‌سازی: در هر مرحله از فرآیند خالص‌سازی، انجام آزمایش فعالیت آنزیمی به تعیین حفظ فعالیت کمک می‌کند و از دست دادن احتمالی فعالیت آنزیم را مشخص می‌سازد.

۴. الف) برای محاسبه‌ی فاصله همینگ بین دو توالی DNA، باید تعداد مکان‌هایی که حروف دو رشته با هم متفاوت هستند را بشماریم.

حالا، به صورت جفتی از ابتدای توالی‌ها، حروف را با هم مقایسه می‌کنیم:

موقعیت	۱	۲	۳	۴	۵	۶	۷
x	A	G	C	T	G	A	C
y	A	G	C	A	G	T	C
تفاوت؟	-	-	-	✓	-	✓	-

در اینجا، تفاوت‌ها در موقعیت‌های ۴ و ۶ وجود دارند. بنابراین فاصله همینگ برابر ۲ است.

ب) برای محاسبه‌ی Edit Distance بین دو توالی DNA، باید حداقل تعداد عملیات لازم (شامل جایگزینی، حذف، یا اضافه کردن کاراکترها) را برای تبدیل یک توالی به دیگری بیابیم. برای این کار، از روش برنامه‌ریزی پویا استفاده می‌کنیم. در هر موقعیت، ما تصمیم می‌گیریم که آیا باید یک عملیات حذف، اضافه یا جایگزینی انجام دهیم تا حداقل تغییرات ممکن را به دست آوریم. کد زیر عملیات محاسبه‌ی این فاصله را برای دو توالی داده‌شده نشان می‌دهد:

```python
# Define the sequences
x = "AGCTGAC"
y = "AGCAGTC"

# Initialize a matrix for dynamic programming
len_x = len(x)
len_y = len(y)

# Create a (len_x + 1) x (len_y + 1) matrix for storing edit distances
dp = [[0] * (len_y + 1) for _ in range(len_x + 1)]

# Fill the matrix
for i in range(len_x + 1):
    for j in range(len_y + 1):
        # Base cases: if one of the strings is empty
        if i == 0:
            dp[i][j] = j  # Cost of all insertions
        elif j == 0:
            dp[i][j] = i  # Cost of all deletions
        # If characters are the same, no additional cost
        elif x[i - 1] == y[j - 1]:
            dp[i][j] = dp[i - 1][j - 1]
        # If characters are different, consider the costs of insertion, deletion, and substitution
        else:
            dp[i][j] = 1 + min(dp[i][j - 1],    # Insertion
                               dp[i - 1][j],    # Deletion
                               dp[i - 1][j - 1]) # Substitution

# The edit distance is in the bottom-right corner of the matrix
edit_distance = dp[len_x][len_y]
edit_distance
```

```
2
```

برای توالی‌های X و Y می‌توان به صورت زیر این فاصله را محاسبه کرد:

- جایگزینی T در موقعیت ۴ با A
- جایگزینی A در موقعیت ۶ با T

بنابراین فاصله ادیت برابر ۲ است.

Hamming Distance فقط تفاوت‌های یک‌به‌یک را در مکان‌های مشابه توالی‌ها در نظر می‌گیرد و صرفاً تعداد موقعیت‌های متفاوت را می‌شمارد. به همین دلیل، این فاصله تنها زمانی قابل محاسبه است که دو توالی طول یکسانی داشته باشند. Edit Distance اطلاعات بیشتری ارائه می‌دهد زیرا به ما امکان می‌دهد تغییرات دقیق‌تری شامل حذف‌ها، اضافه‌ها و جایگزینی‌ها را در نظر بگیریم. این معیار می‌تواند به بررسی‌های عمیق‌تری از تغییرات ژنتیکی، مانند حذف‌ها یا درج‌های ژنومی، کمک کند و الگوهای پیچیده‌تر تغییرات ژنتیکی را نیز آشکار کند.

۵ • برای هم‌ترازی دو توالی داده‌شده با استفاده از سیستم امتیازدهی داده‌شده، ابتدا ماتریس امتیازدهی را ایجاد می‌کنیم و سپس با استفاده از محاسبات دینامیکی، بهترین هم‌ترازی را با توجه به امتیازهای مربوط به جریمه‌ها پیدا می‌کنیم. ابتدا ماتریس را ایجاد می‌کنیم و سپس به بهترین ترتیب، توالی‌ها را هم‌تراز می‌کنیم. هم‌ترازی بهینه برای دو توالی به صورت زیر است:

موقعیت	۱	۲	۳	۴
x	G	C	A	C
y	G	C	-	C
امتیاز	۲	۲	-۵	۲

در این هم‌ترازی، یک شکاف (gap) در توالی y در موقعیت سوم قرار گرفته است تا بیشترین هم‌خوانی بین دو توالی به دست آید. امتیاز نهایی هم‌ترازی برای دو توالی برابر ۱ است. کد محاسبه امتیاز بهینه هم‌ترازی دو توالی به صورت زیر است:

```python
import numpy as np

# Sequences
x = "GCAC"
y = "GCC"

# Scoring system
match_score = 2
mismatch_score = -2
gap_open_penalty = -5
gap_extend_penalty = -1

# Initialize scoring matrices
len_x = len(x) + 1
len_y = len(y) + 1

# Initialize matrices for scoring: main (alignment) matrix, gap in x, gap in y
score_matrix = np.zeros((len_x, len_y))
gap_x_matrix = np.full((len_x, len_y), -np.inf)
gap_y_matrix = np.full((len_x, len_y), -np.inf)

# Initialize first row and column for gap penalties
for i in range(1, len_x):
    score_matrix[i, 0] = gap_open_penalty + (i - 1) * gap_extend_penalty
    gap_x_matrix[i, 0] = gap_open_penalty + (i - 1) * gap_extend_penalty

for j in range(1, len_y):
    score_matrix[0, j] = gap_open_penalty + (j - 1) * gap_extend_penalty
    gap_y_matrix[0, j] = gap_open_penalty + (j - 1) * gap_extend_penalty
```

```python
# Fill matrices
for i in range(1, len_x):
    for j in range(1, len_y):
        # Calculate match/mismatch score
        if x[i - 1] == y[j - 1]:
            match_mismatch_score = match_score
        else:
            match_mismatch_score = mismatch_score

        # Update main score matrix with match/mismatch and diagonal values
        diag_score = score_matrix[i - 1, j - 1] + match_mismatch_score
        score_matrix[i, j] = max(diag_score, gap_x_matrix[i - 1, j - 1], gap_y_matrix[i - 1, j - 1])

        # Update gap matrices with gap penalties
        gap_x_matrix[i, j] = max(score_matrix[i - 1, j] + gap_open_penalty,
                                 gap_x_matrix[i - 1, j] + gap_extend_penalty)

        gap_y_matrix[i, j] = max(score_matrix[i, j - 1] + gap_open_penalty,
                                 gap_y_matrix[i, j - 1] + gap_extend_penalty)

        # Take the maximum score among score_matrix, gap_x_matrix, and gap_y_matrix
        score_matrix[i, j] = max(score_matrix[i, j], gap_x_matrix[i, j], gap_y_matrix[i, j])

# Final alignment score
alignment_score = score_matrix[len(x), len(y)]
alignment_score
```

```
1.0
```

۶. **الف)** ایده اصلی MUSCLE ایجاد روشی برای هم‌تراز کردن چندتایی توالی‌های پروتئینی است که سرعت و دقت بالاتری نسبت به روش‌های قبلی ارائه می‌دهد. الگوریتم MUSCLE شامل سه مرحله‌ی اصلی است:

مرحله پیش‌نویس اولیه: ابتدا فاصله بین توالی‌ها با استفاده از "k-mer" محاسبه می‌شود که روشی سریع برای اندازه‌گیری شباهت توالی‌ها بدون نیاز به هم‌ترازی اولیه است. سپس یک درخت راهنما با روش UPGMA ساخته می‌شود تا ساختار اولیه‌ی هم‌ترازی را مشخص کند.

مرحله بهبود یافته: پس از ایجاد هم‌ترازی اولیه، فاصله بین توالی‌ها دوباره محاسبه می‌شود. سپس یک درخت جدید با دقت بالاتر ساخته شده و هم‌ترازی‌ها بر اساس این درخت بازسازی می‌شوند.

مرحله‌ی بهینه‌سازی نهایی: در این مرحله، با استفاده از یک روش بهینه‌سازی تکراری، خطاهای احتمالی در هم‌ترازی کاهش یافته و دقت نهایی بهبود می‌یابد. هم‌ترازی‌های قبلی مورد بازبینی قرار می‌گیرند و تغییرات لازم برای بهبود دقت اعمال می‌شوند.

ب) در این مقاله، واژه "hits" به توالی‌های پروتئینی اشاره دارد که از جستجوی پایگاه داده PSI-BLAST بازیابی شده‌اند و با توالی اصلی شباهت بالایی دارند (مثلا مقدار e-value کمتر از ۰.۰۱ است که نشان‌دهنده شباهت قابل‌توجه بین توالی اصلی و توالی‌های بازیابی‌شده است). سپس این توالی‌های مرتبط یا hits به عنوان داده‌های ورودی برای هم‌ترازسازی استفاده می‌شوند تا دقت و کارایی هم‌ترازسازی بهبود یابد.

ج) MUSCLE دارای سه مرحله اصلی است که هرکدام به بهبود دقت و کارایی همترازی چندتایی توالی‌ها کمک می‌کند:

مرحله پیش‌نویس اولیه: در این مرحله، یک هم‌ترازی اولیه با سرعت بالا تولید می‌شود. ابتدا فاصله بین توالی‌ها از طریق شمارش k-merها (زیرتوالی‌های کوتاه از توالی‌ها) محاسبه می‌شود. سپس، با استفاده از روش خوشه‌بندی UPGMA، یک درخت راهنما ساخته می‌شود که نمایانگر رابطه توالی‌ها است. هم‌ترازی اولیه به ترتیب شاخه‌های درخت ایجاد می‌شود و در هر گره از درخت، پروفایل توالی‌های فرزندان به‌صورت تکی یا چندتایی هم‌تراز می‌شوند. این مرحله به‌دلیل استفاده از k-mers سریع است و یک هم‌ترازی ابتدایی تولید کرده که در مراحل بعدی استفاده می‌شود.

مرحله پیش‌رونده بهبودیافته: در این مرحله، هم‌ترازی اولیه بازبینی می‌شود تا دقت آن افزایش یابد. فاصله‌ها این بار با استفاده از روش دقیق‌تر کیمورا (Kimura) محاسبه می‌شوند که خطای محاسبه

فاصله‌های توالی‌ها را کاهش می‌دهد. سپس، یک درخت راهنما جدید ساخته شده و هم‌ترازی‌ها بر اساس این درخت بهینه‌سازی‌شده بازسازی می‌شوند.

مرحله نهایی بهینه‌سازی: در این مرحله، با استفاده از یک روش بهینه‌سازی تکراری، همترازی موجود بهینه‌سازی می‌شود. در هر گام، یک شاخه از درخت حذف می‌شود و هم‌ترازی توالی‌های هر زیرشاخه به‌صورت مستقل بررسی و اصلاح می‌شود. اگر هم‌ترازی جدید و بهتری حاصل شود ، آن را جایگزین هم‌ترازی قبلی می‌کند. این مرحله به کاهش خطاهای احتمالی در هم‌ترازی و افزایش دقت نهایی کمک می‌کند، زیرا به‌طور پیوسته هم‌ترازی‌های نهایی اصلاح می‌شوند تا به نتیجه‌ای با دقت بالاتر برسند.

د) روش k-mer در MUSCLE به تسریع محاسبه شباهت توالی‌ها کمک می‌کند، زیرا به‌جای هم‌تراز کردن کامل توالی‌ها، از شمارش توالی‌های کوتاه با طول ثابت k استفاده می‌شود. این روش، مراحل به این صورت است:

- از هر توالی، زیرتوالی‌هایی با طول مشخص k استخراج و شمارش می‌شوند. هر زیرتوالی یک k-mer است تعداد هر زیر توالی در هر توالی ثبت می‌شود.
- برای هر دو توالی، فراوانی k-merهای مشترک محاسبه می‌شود. این فراوانی‌ها با هم مقایسه شده و فاصله k-mer بین توالی‌ها با توجه به تعداد k-merهای مشترک تعیین می‌شود. هرچه تعداد k-merهای مشترک بیشتر باشد، شباهت بین دو توالی بیشتر در نظر گرفته می‌شود.
- از آنجا که فاصله k-mer نیازی به هم‌ترازی کامل اولیه ندارد، محاسبه شباهت بسیار سریع انجام می‌شود. این روش برخلاف روش‌های هم‌ترازسازی کلاسیک که به زمان و محاسبات پیچیده نیاز دارند، یک تخمین تقریبی اما سریع از شباهت را ارائه می‌دهد.

ه) مرحله بهینه‌سازی، به دلیل امکان اصلاح خطاهای احتمالی در هم‌ترازی‌های قبلی اهمیت ویژه‌ای دارد و دقت نهایی را به حداکثر می‌رساند. ابتدا یک شاخه از درخت راهنما حذف می‌شود تا توالی‌ها به دو زیرشاخه تقسیم شوند. هر زیرشاخه به عنوان یک گروه مستقل از توالی‌ها در نظر گرفته می‌شود. سپس پروفایل هر زیرشاخه (که نشان‌دهنده هم‌ترازی توالی‌های درون آن زیرشاخه است) با پروفایل زیرشاخه دیگر هم‌تراز می‌شود. بعد از انجام هم‌ترازی مجدد، مجموع امتیازات جفتی (SP score) محاسبه می‌شود. اگر امتیاز جدید نشان‌دهنده بهبود هم‌ترازی باشد، این هم‌ترازی جدید پذیرفته و جایگزین همترازی قبلی می‌شود؛ در غیر این صورت، هم‌ترازی قبلی حفظ می‌شود. این مرحله به الگوریتم اجازه می‌دهد خطاهای هر مرحله هم‌ترازی اصلاح شوند و ستون‌های هم‌ترازی را بر اساس شباهت با دقت بالاتری بازبینی شود.

و) MUSCLE به دلیل طراحی کارآمد و ویژگی‌های خاصی که دارد، برای مجموعه داده‌های بزرگ با تعداد زیادی از توالی‌ها بسیار مناسب است. برخی از این ویژگی‌ها عبارتند از:

- محاسبه سریع فاصله توالی‌ها با استفاده از k-mer
- بهینه‌سازی تکراری MUSCLE که باعث افزایش دقت در این روش است.
- پیاده‌سازی کارآمد با زمان‌بندی مناسب: الگوریتم MUSCLE برای هم‌ترازی داده‌های بزرگ بهینه‌سازی شده است، به طوری که می‌تواند هم‌ترازی هزاران توالی بلند را در مدت زمان کوتاهی انجام دهد. به عنوان مثال، MUSCLE قادر است ۵۰۰۰ توالی با طول میانگین ۳۵۰ را در حدود ۷ دقیقه هم‌تراز کند که این ویژگی آن را برای پروژه‌های عظیم بیوانفورماتیکی مناسب می‌کند.
- اگر منابع محدود باشد، با استفاده از مدیریت بهینه حافظه این الگوریتم کارآیی خوبی دارد و مصرف حافظه بهینه‌ای برای داده‌های بزرگ فراهم می‌کند.

((پایان))