ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ.

ДНК – Дезоксірибунуклеїнова кислота

тДНК – Фрагмент ДНК Ti-плазмиды или Ri-плазмиды

РНК – Рибонуклеїнова кислота

СНП – Секвенування нового покоління

ПЛР – Експериментальний метод молекулярної біології, спосіб значного збільшення малих концентрацій бажаних фрагментів ДНК в біологічному матеріалі (пробі).

ЗІЗ – Засоби індивідуального захисту

AWK — мова для обробки файлів тексту.

ВСТУП

Актуальність

В наші дні для пошуку по геномам тисяч організмів, що складаються з мільярдів пар нуклеотидів використовуються комп'ютерні програми. Даний програмний продукт може однозначно зіставити (вирівняти) схожі послідовності ДНК в геномах різних видів; часто такі послідовності несуть схожі функції, а відмінності виникають в результаті дрібних мутацій, таких як заміни окремих нуклеотидів, вставки нуклеотидів, і їх «випадання» (делеції).Також актуальність даного алгоритму полягає в тому що великі послідовності вирівнюються швидше ніж у використовуємих нині алгоритмів вирівнювання (Нідлмана-Вунша, Сміта-Уотермана).

Новизна

Існуючі алгоритми глобального та локального вирівнювання(Нідлмана-Вунша, Сміта-Уотермана) використовують фіксований «штраф» за початок делеції та за її продовження, що при отриманні результатів дає трохи хибні результати. Але цього можна запобігти застосувавши алгоритм вирівнювання з афінною штрафною функцією. Це надасть змогу отримати більш точні результати. Головна відмінність такого алгоритму від звичайних це штрафування початку делеції та штрафування її продовження але з меншою вагою, що є доречним з при оцінюванні результатів та надає більш тоний результат.

Мета роботи

Метою дипломної роботи є створення додатку призначенного для вирівнювання біологічних послідовностей методом динамічного програмування з афінним штрафом.
Для досягнення мети були поставлені наступні задачі:

1. Зібрати та проаналізувати інформацію про існуючі методи вирівнювання
2. Реалізувати алгоритм вирівнювання біологічних послідовностей
3. Провести аналіз отриманих вихідних даних
4. Провести аналіз небезпек та шкідливих факторів при роботі з програмним продуктом в аудиторії, запропонувати заходи щодо зменшення їх впливу

РОЗДІЛ 1 БІОІНФОРМАТИКА

Вступ

Під біоінформатикою зазвичай розуміють використання комп'ютерів для вирішення біологічних задач. В даний час це майже виключно завдання молекулярної біології. Причина цього в тому, що за останні 20-25 років накопичено воістину колосальний експериментальний матеріал саме про будову і функціонування біологічних молекул (білків і нуклеїнових кислот), як приклад досить привести геном людини. Цей матеріал вимагає розвинених комп'ютерних методів для свого аналізу. Тому біоінформатика як на нашому факультеті, так і в переважній більшості світових наукових центрів розуміється як синонім обчислювальної молекулярної біології.

Біоінформатика - не є "чистою" наукою в повному розумінні слова. Швидше за все - це інструмент для аналізу, систематизації та отримання знань про живу матерію.

Біоінформатика - використовується для:

1 - зберігання і маніпулювання даними про структуру нуклеїнових кислот і білків - від первинної до третинної

2 - конструювання праймерів

3 - передбачення функцій продукту певного гена

4 - моделювання та передбачення вторинної та третинної структури білків на основі знання первинної структури.

1.1 Цілі біоінформатики

Уявімо собі геном невеликої бактерії - це безперервний рядок довжиною в 1-10 мільйонів символів, і далеко не вся ДНК кодує білки.

Перший тип біоінформатичного завдання - це завдання пошуку в нуклеотидних послідовностях особливих ділянок, ділянок, що кодують білки, ділянок, що кодують РНК (наприклад, тРНК), ділянок зв'язування з регуляторними білками і ін. І це не завжди прості завдання, наприклад, гени еукаріотичних організмів складаються з чергувань «Осмислених» і «безглуздих» фрагментів (екзонів і інтронів), і відстань між «осмисленими» фрагментами може досягати тисяч нуклеотидів. Нехай ген знайдений. Що він кодує? Навіщо він потрібен? Якщо мова йде про ділянку ДНК, що кодує білок, то за допомогою вельми простої операції - трансляції з використанням відомого генетичного коду можна отримати. амінокислотні (білкові) послідовності. З відомих на сьогодні 4 273 512 білків близько 94% послідовностей - це саме такі гіпотетичні трансляти, і більше про них нічого не відомо. Швидкість надходження інформації з автоматичних секвенаторів перевищує швидкість нашого розуміння її сенсу.

Але біологічні об'єкти - це об'єкти, що виникли в процесі еволюції. Порівняльно-еволюційний підхід - один з найпотужніших підходів в біології. Наприклад, функція білка з одного організму добре експериментально вивчена, в іншому організмі знайшли білок зі схожою амінокислотною послідовністю. Можна, можливо припустити, що другий (невідомий) білок виконує ту ж або схожу функцію. І тут відразу виникає кілька питань. По-перше, що значить схожа послідовність? Як порівнювати послідовності? При якому ступені подібності послідовностей можна припускати, що білки виконують подібні функції?

Порівняння послідовностей (вирівнювання) є найважливішим завданням біоінформатики. Важко знайти сучасного біолога, ні разу не використав програми Blastp і ClustalX, поява цих програм - вже великий успіх біоінформатики. Але сучасні біоінформатики незадоволені і постійно вдосконалюють методи вирівнювань. Можна привести багато прикладів того, як порівняльно-еволюційний підхід у поєднанні з біоінформатіческімі методами породжує нове біологічне знання.

Генетичні тексти - тексти з великою часткою шуму, порівнюючи родинні послідовності, в ряді випадків вдається відфільтрувати шум і виявити сигнал, наприклад, коротку послідовність нуклеотидів, здатну зв'язуватися з білком - регулятором, або амінокислотні залишки в ферменті, що відповідають за зв'язування субстрату. Щоб бути впевненими в результаті, біоінформатики використовують теорію ймовірності та математичну статистику.

1.2 Секвенування

Секвенування біополімерів (білків і нуклеїнових кислот - ДНК і РНК) - визначення їх амінокислотної або нуклеотидної послідовності. В результаті секвенування отримують формальний опис первинної структури лінійної макромолекули у вигляді послідовності мономерів в текстовому вигляді. В наш час використовуються методи секвенування нового покоління(СНП).

СНП дозволяє «прочитати» одночасно відразу кілька ділянок геному, що є головною відмінністю від попередніх методів секвенування. СНП здійснюється з допомогою повторюваних циклів подовження ланцюга, індукованого полимеразой, або багаторазового лігування

олігонуклеотидів. В ході СНП можуть генеруватися до сотень Мегабаз і гігабаз нуклеотидних послідовностей за один робочий цикл.

Всі основні принципи роботи технологій СНП базуються на секвенування ДНК-чипів, використовуючи інтерактивні циклічні ферментативні реакції з подальшим збором отриманої інформації у вигляді ілюстрацій. Отримані дані використовуються для відновлення нуклеотидноїпослідовності або, як для технології SOLiD, дінуклеотідних «квітів». Незважаючи на різні методи отримання копій (ампліфікація) ділянок геному і на технічну різницю диференціації різних нуклеотидів в прочитаними послідовності, загальна схема роботи для всіх секвенаторов одна.

Перший етап секвенування - створення бібліотеки випадкових послідовностей ДНК, які можна буде зшити з загальнодоступними адаптерні послідовностями.

Другий етап - створення амплікон за допомогою ПЛР, які будуть використані як зразки.

Третій етап - визначення первинної структури всіх фрагментів.

Існують методи «Класичного секвенування»: Існує два основних метода секвенування: метод Максама-Гілберта (заснований на хімічному розщепленні ДНК з однієї основи) і метод Сангера (дідезоксі-метод). Метод Сангера більш надійний і простий у виконанні, на практиці його використовують частіше але у сучасних секвенаторах використовують вдосконалені запатентовані методи - СНП.

Найросповсюдженішими методами СНП є: Illumina-SOLEXA, SOLiD.

1.2.1 Метод Маскама і Гілберта (хімічний)

Один з методів заснований на хімічній деградації ДНК. Він був запропонований в 1976 році Максамом і Гілбертом і названий їх ім'ям. Суть методу зводиться до наступного: один з кінців фрагмента ДНК мітять за допомогою ізотопу фосфору 32Р. Останнім часом замість радіоактивної вводять флюоресцирующую мітку. Її можна «чіпляти» і до нуклеотидам, причому для кожного типу нуклеотидів підбирати різне забарвлення. Препарат міченої ДНК ділять на чотири порції і кожну з них обробляють реагентом, специфічно руйнує одне або два з чотирьох підстав, причому умови реакції підбирають таким чином, щоб на кожну молекулу ДНК доводилося лише кілька пошкоджень.

Руйнування йде в 2 етапи. На першому етапі відбувається модифікація азотистого підстави і подальше вищепленію його. На другому етапі проводять гідроліз ДНК в місцях вищепленію підстав. Пуринові основи модифікуються диметилсульфатом. Аденіновую залишки метилирующей по третьому атома азоту, гуанінових - по положенню N7. Якщо таку модифікацію обробити 0,1 М HCl при 0 ° С, то вищепляются метіладенін. При подальшій інкубації в лужному середовищі (0,1 М NaOH) при температурі + 90оС відбувається руйнування цукрово-фосфатного зв'язку в місцях вищепленію підстав. Обробка пошкоджених молекул Піперидин призводить до гідролізу ДНК по залишкам метілгуаніна. Піримідинові підстави модифікуються гидразином. У солі середовищі модифікується і цитозин, і тимін, в присутності 2 М NaCl модифікується тільки цитозин. При подальшій обробці Піперидин відбувається розщеплення ДНК по точкам

модифікації. Можна використовувати і інші реакції хімічної модифікації підстав і розщеплення по ним молекул ДНК.

В результаті виходить набір мічених фрагментів, довжини яких визначаються відстанню від зруйнованого підстави до кінця молекули. Фрагменти, що утворилися в усіх чотирьох реакціях, піддають електрофорезу в чотирьох сусідніх доріжках; потім проводять радіоавтографія, і ті фрагменти, які містять радіоактивну мітку, залишають "відбитки" на рентгенівській плівці. Відповідно до положення відбитків можна визначити, на якій відстані від міченого кінця знаходилося зруйноване підставу, а знаючи це підстава - його положення. Так набір смуг на рентгенівській плівці визначає нуклеотидну послідовність ДНК. Аналогічно спостерігають флюоресцентні фарбування. Якщо для кожного з чотирьох нуклеотидів був підібраний свій колір флюоресцентной мітки, то при електрофорезі їх наносять на 1 доріжку. Тоді розташування нуклеотидів відзначено штрихами різного кольору, а процедуру зчитування легко автоматизувати.

1.2.2 Метод Сенгера (ферментативний)

Інший метод, розроблений Сенгером і носить його ім'я, заснований не на хімічному, а на ферментативном підході. Сенгер використовував ДНК-полімерази I. У клітці цей фермент бере участь в процесі реплікації, заповнюючи прогалини між знову синтезованими фрагментами ДНК (фрагментами Окадзакі). Для роботи ферменту в пробірці потрібні попередники ДНК - дезоксірібонуклеотідтріфосфати (dNTP), а також одноцепочечная матриця, на якій повинен бути невеликий дволанцюжкові ділянку - запал, з якого починається синтез (рис.1.1). Були також

синтезовані модифіковані дідезоксірібонуклеотіди, в яких дезоксирибоза 3'-ОН відсутня, для кожного з чотирьох основ ДНК. ДНК-полімераза включає ці попередники в ДНК. Однак, включившись в ДНК, модифіковане підставу не може утворити фосфодіефірную зв'язок з наступним дезоксірібонуклеотідов. В результаті зростання (елонгація) даної ланцюга зупиняється (терминирующего) в тому місці, де в ДНК включився дідезоксірібонуклеотід (ddNTP). Тому їх називають терминаторами елонгації.

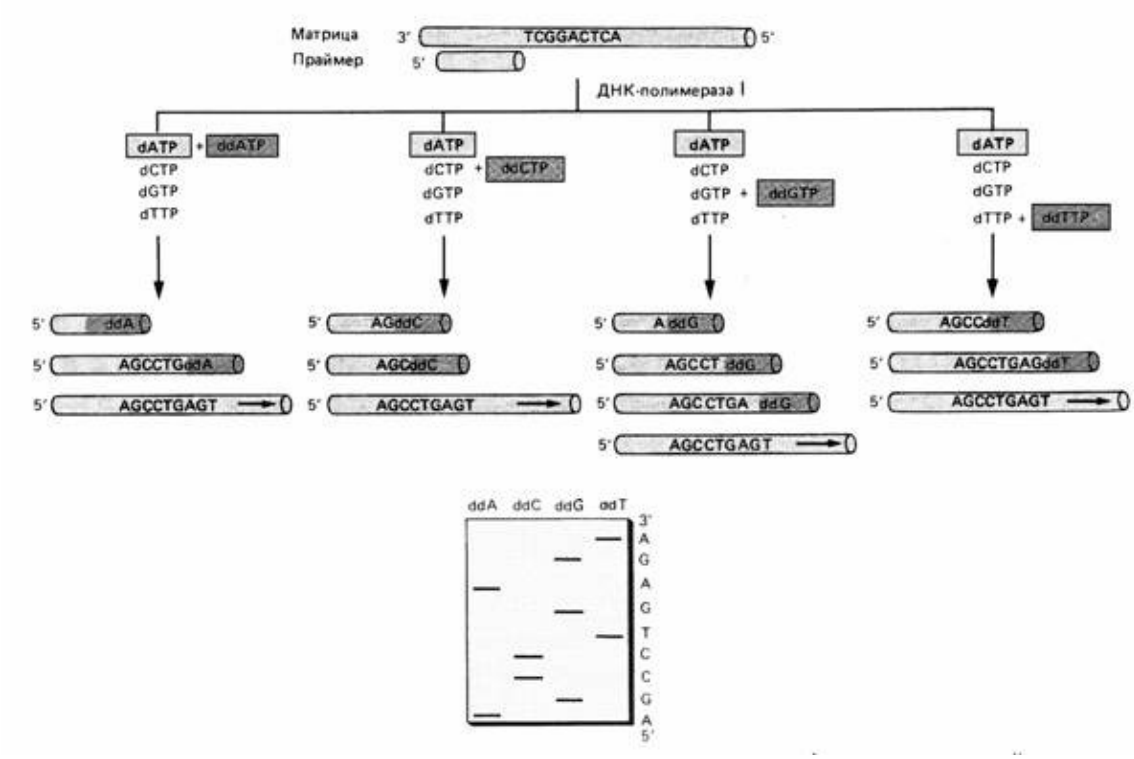

Рисунок 1.1. Ферментативний метод секвенування ДНК

Реакційна суміш по Сенгер складається з ланцюга ДНК, нуклеотидную послідовність якої треба визначити, короткого фрагмента "міченої" ДНК, комплементарної кінцевого відрізку цього ланцюга (запал), одного з чотирьох ddNTP і відповідного dNTP в строго певному співвідношенні (щоб вони конкурували), а також інших трьох dNTP.

Готують чотири суміші, кожна з яких містить один з чотирьох ddNTP. У кожній з пробірок утворюється набір мічених фрагментів різної довжини. Довжина їх залежить від того, в якому місці в ланцюг включений дефектний нуклеотид. Отримані мічені фрагменти ДНК розділяють в поліакриламідному гелі (з точністю до одного нуклеотиду), проводять радіоавтографія і по картині розподілу фрагментів в чотирьох пробах встановлюють нуклеотидную послідовність ДНК (рис. 1.1).

В даний час визначення точної нуклеотидної послідовності будь-якого сегмента ДНК помірної довжини - цілком здійсненне завдання. Уже визначено послідовність кількох сотень генів про- та еукаріот. Знаючи послідовність гена і генетичний код, легко визначити амінокислотнихпослідовність кодованого їм білка. Раніше для визначення структури білка доводилося робити ретельний і досить трудомісткий аналіз виділеного і очищеного білка. Зараз часто буває простіше визначити структуру білка через нуклеотидную послідовність, ніж за допомогою прямого секвенування. Якщо секвенування білка займає місяці і навіть роки, то ДНК вдається секвенувати за кілька тижнів.

Визначення послідовності ДНК призвело також до того, що були виявлені області, які не кодують білки, але беруть участь у регуляції експресії генів і реплікації ДНК. У 1996 році був секвенирован геном дріжджів, в 1998 р - геном арабидопсиса, у 2000 році - геном людини, проте в даному випадку мова йде тільки про встановлення послідовності нуклеотидів, так як генетична структура і функції окремих ділянок геному ще не ідентифіковані, це більш складна задача.

Відразу слідом за розробкою швидких методів секвенування з'явилися настільки ж швидкі і прості методи синтезу порівняно довгих олігонуклеотидів з певною, заздалегідь заданою послідовністю. Тепер за

три-чотири дні можна синтезувати послідовність з 12 - 20 нуклеотидів. Автоматизація цієї процедури ще більш полегшує і прискорює синтез. З'явилися прилади - ДНК-синтезатори, які виконують цю роботу за кілька годин.

1.2.3 Illumina / Solexa

Метод Illumina / Solexa - метод секвенування нового покоління, розроблений компанією Solexa.

В основі методу лежить принцип секвенування шляхом синтезу:

Одноланцюгові фрагменти ДНК закріплюються на твердій основі. ДНК-залежна ДНК полімераза синтезує комплементарную ланцюг. Вбудовування кожного нового нуклеотиду реєструється за допомогою камери. У методі Solexa використовуються 3'модифіковані нуклеотиди з приєднаними флуоресцентними мітками різних кольорів. Модифікація нуклеотидів не дозволяє ДНК-полімерази приєднати більше одного нуклеотиду. Флюоресценція ініціюється коротким імпульсом лазера і тип приєднаного нуклеотиду визначається за кольором флюоресцентной мітки. Модифікація нуклеотиду блокується (полімераза тепер може рухатися далі) і цикл повторюється знову. В результаті вдається визначити послідовність ДНК довжиною до 250 нуклеотидів. Таку послідовність ДНК називають прочитанням або Рідом - (калька з англійського).

Перший секвенатор Genome Analyzer 1G був представлений компанією Solexa в 2006 році. Довжина прочитання становила 30-35 нуклеотиду, можна було отримати близько 1 Gb інформації. HiSeq 2000 (2011 рік) здатний секвенувати 6 людських геномів за 11 днів. Довжина

прочитання становить 100 нуклеотидів, можна отримати 600 Gb інформації.

1.2.4 SOLiD

SOLiD (англ. Sequencing by Oligonucleotide Ligation and Detection) - технологія нового покоління секвенування ДНК, що розвивається компанією Life Technologies, комерційно доступна з 2006 року. SOLiD дозволяє секвенувати разом сотні мільйонів і навіть мільярди коротких послідовностей.

SOLiD використовує метод лігування на відміну від інших платформ, таких як Roche-454 піросеквенірованія (метод з'явився в 2005 році, в 2009 році вже створював мільйони послідовностей довжиною 200-400 пар основ) і система Solexa (зараз належить Illumina) (метод з'явився в 2006 році, в 2009 році вже створював мільйони послідовностей довжиною 50-100 пар основ), які використовують секвенування за допомогою синтезу.

Ці всі методи знизили вартість секвенування в 2006 році з $ 0.01 / основу, приблизно, $ 0.0001 / основу і збільшили потужність з 1 млн основ / секвенатор / день (2004 рік) до більш 5 млн основ / секвенатор / день (2009 рік). Існують більш 30 публікацій, що описують використання методу в нуклеосомної позиціонуванні, транскрипционном профайлінга або ланцюг-специфічному RNA-Seq, транскрипционном профайлінга в одиничної клітині і пересеквенірованіі людського генома.

1.3Вирівнювання послідовностей

1.3.1. Цілі вирівнювання послідовностей

Першим етапом філогенетичного аналізу є ідентифікація вставок та делецій, що мали своє місце в еволюційній історії аналізованої групи послідовностей . Цю процедуру називають вирівнювання послідовностей, яка має на меті виявити гомологічні позиції аналізованих послідовностей, встановлення найбільш вірогідного, тобто, того, що вимагає найменшого числа еволюційних подій, сценарію еволюції аналізованої групи

1.3.2 Принципи вирівнювання послідовностей

Вирівнювання послідовностей: дві послідовності вирівняні, якщо вони розташовані один відносно іншого таким чином, щоб максимально представити їх подібність.

В процесі еволюції послідовності зазнають множинні мутаційні події втрати і придбання протяжних шматків послідовності. Тому при порівнянні декількох гомологічних послідовностей виникає задача встановлення відповідності один одному окремих протяжних ділянок послідовностей. У цьому випадку прийнято говорити про вирівнюванні послідовностей. У найпростішому випадку вирівнюються дві послідовності (парне вирівнювання), в більш складних випадках вирівнюється цілий набір послідовностей (множинне вирівнювання). Як правило множинне вирівнювання здійснюється на основі результатів парного

Розглянемо гомологічні нуклеотидні послідовності 1 та 2 (рис. 1.2).

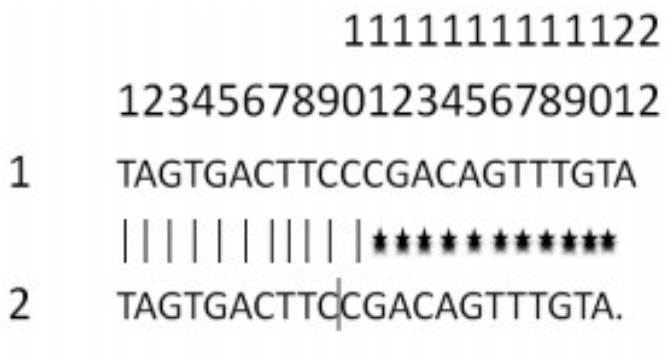

Рисунок 1.2- Гомологічні послідовності, перший варіант

Якщо розглянути гомологічні нуклеотидні послідовності 1 і 2, наведені на рисунку 1.1. видно, що перші десять нуклеотидів у цих

послідовностей однакові (тут і далі відмічені вертикальними лініями), а наступні нуклеотиди є відмінними (тут і далі відмічені зірочками). Крім того, послідовність 1 довша ніж послідовність 2 на один нуклеотид. Можна припустити що після дивергенції цих послідовностей від їх спільного предка відбулося як мінімум десять нуклеотидних замін і делеція останнього нуклеотида в послідовності 2, тобто не менше одинадцяти мутацій. Однак слід відмітити, що нуклеотиди 11-20 у послідовності 2 присутні втому ж порядку і у послідовності 1, однак знаходяться в позиціях 12-21 (виділені курсивом). Таким чином, найбільш вірогідним сценарієм еволюції цих послідовностей було не одинадцять, а одна еволюційна подія: делеція нуклеотида Т в позиції 11 послідовності 2. В такому випадку послідовності 1 і 2 можуть бути вирівняні наступним чином (Рис. 1.3):

Рисунок 1.3 - Гомологічні послідовності, другий варіант

Варто зазначити, що в типовому випадку однаково вірогідним буде і сценарій, коли в послідовності 1 після позиції 10 відбулася вставка нуклеотиду Т. Як правило, під час аналізу генетичних послідовностей невідомо, чи мала місце вставка у цій позиціх в одній послідовності або ж делеція - у відповідній позиції іншої послідовності. Тому говорять про наявність прогалини в цій позиції послідовностей. В англомовній

літературі використовується також термін індел. Процедуру вирівнювання варто розуміти як встановлення гомологічних ділянок генетичних послідовностей: після проведення вирівнювання передбачається, що кожна позиція послідовності 1 є гомологічною відповідній позиції послідовності 2. Базовим принципом вирівнювання послідовностей є встановлення найбільш вірогідного сценарію їх еволюції, тобто, такого сценарію, який вимагав би найменшого числа еволюційних подій: мінімального числа замін при мінімальному числі прогалин. Таким чином, метою вирівнювання є максимізація числа однакових нуклеотидів або амінокислот у відповідних позиціях послідовностей при мінімізації числа необхідних для цього прогалин.

1.4 Стійкість відносно мутацій та інші зміни котрі враховуються під час вирівнювання та їх значення

Геном може змінюватися в результаті мутацій, а також, в разі статевого розмноження, за рахунок кросинговеру, при якому може відбуватися обмін генами междуразнимі геномами. Мутації, в свою чергу, діляться на локальні, які викликають заміну невеликої подстроки геномної послідовності на іншу невелику подстроку, і нелокальних, які зводяться до розрізання послідовності на частини, подвоєння і видалення деяких з них, заміну деяких на комплементарні і з'єднанню того, що вийшло , в деякому порядку. Кроссинговер теж підходить під цей шаблон, але з тією відмінністю, що може відбуватися обмін частинами між послідовностями.

Локальні мутації можна розділити на три види:

1. Заміна - найбільш часта мутація. При такій мутації один нуклеотид замінюється на інший.

2. Вставка - поява в послідовності нового нуклеотиду.

3. Видалення - зникнення з послідовності одного нуклеотиду.

Ступінь відмінності послідовностей по відношенню до таких зміною називаються ється редакційним відстанню. Хоча пошук редакційного відстані в загальному випадку - непросте завдання для довгих рядків, в даному методі не потрібно шукати редакційного відстані в загальному випадку. Справа в тому, що якщо редакційне відстань між пов'язаним фрагментом і відповідної підрядком досить велике, то таке розбиття свідомо неоптимально: можна зробити цей фрагмент вільним і таким чином зменшити сумарну ступінь відмінності.

Більш того, навіть якщо фрагмент містить трохи змін щодо його

довжини, але всі ці зміни розташовані недалеко один від одного, можна розбити цей фрагмент на три фрагменти, вирізавши з нього шматок, який містить велику частина змін, і замінивши його вільним фрагментом. В результаті, у багатьох випадках можна не брати до уваги (або не продовжувати вважати) редакційне відстань між фрагментами, оскільки і так зрозуміло, що оптимальне розбиття не містить такої фрагмент.

Крім того, використання редакційного відстані не дає локальним мутацій істотно змінювати ступінь відмінності: якщо мутація потрапила у вільний фрагмент, то ступінь відмінності може лише злегка змінитися за рахунок зміни довжини фрагмента, а якщо мутація потрапила в пов'язаний фрагмент, то, знову ж таки, відповідну зміну редакційного відстані не буде великим.

На відміну від локальних мутацій, які позначаються на редакційному відстані, нелокальних мутації безпосередньо позначаються на самому розбитті.

Наприклад, якщо піддати послідовність дуплікації (вставити в неї допию її підрядка), то до відповідного розбиття можна теж вставити копію його підрядка (при цьому крайні фрагменти можуть виявитися обрізаними). При цьому внесок в ступінь відмінності від доданих фрагментів не перевищує внесок від вихідних фрагментів. Крім того, певний внесок у ступінь відмінності відбувається через те, що той фрагмент, в якому знаходиться місце вставки, виявляється розділений на два. Інші нелокальних мутації враховуються аналогічно. При обліку інверсій використовується те, що підрядка, відповідні зв'язаних фрагментів, можна брати як з вихідної послідовності, так і з комплементарної до неї.

1.5 Алгоритм Нідлмана-Вунша та Сміта-Уотермана, глобальне та локальне вирівнювання

Алгоритм Нідлмана-Вунша був опублікований в 1970 році і дозволяє визначити ступінь подібності послідовностей, а також знаходити глобальне вирівнювання, знаходити, який саме символ з однієї послідовності відповідає деякого символу з іншої послідовності. Для своєї роботи алгоритм використовує матрицю подібності, яка вказує, наскільки схожими вважати різні нуклеотиди. Для різних нуклеотидів використовуються негативні елементи. Тому послідовності повинні містити деяку частку співпадаючих нуклеотидів, для того щоб бути

визнання гомологічнимі. Використання матриці дозволяє надавати різну вагу різних замін нуклеотідов. Наприклад, оскільки транзиції більш вірогідні, ніж трансверсії, логічно вважати послідовності, що відрізняються заміною пурину на пурин або піримідину на піримідин , більш схожими, ніж ті, які відрізняються заміною пурину на піримідин або навпаки. Взагалі, матриця дозволяє приписати будь-яку вагу будь-яким замін. Зазвичай використовується симетрична матриця, однак застосування несиметричною матриці дозволяє розрізняти заміни в одну і в іншу сторону.

	А	Г	Т	Ц
А	10	−1	−4	−3
Г	−1	7	−3	−5
Т	−4	−3	8	0
Ц	−3	−5	0	9

Рисунок 1.4 Приклад матриці подібності

Тут А, Г, Т і Ц позначають, відповідно, аденін, гуанін, тимін і цитозин, а числа в матриці вказують ступінь подібності між двома нуклеотидами. Алгоритм Нідлман-Вунша спосіб зіставити символи двох послідовностей так, що сума значень подібності для відповідних символів максимально. Крім того, алгоритм може враховувати вставки і видалення. При цьому вважається, що символ в одному рядку, якому не відповідає ніякої символ з іншої рядки, має деякий рівень подібності, який є параметром алгоритму (наприклад, -5). Для роботи алгоритм використовує матрицю Fi, j, де і і j змінюються від нуля до довжини, відповідно, першої та другої рядків. Спочатку алгоритм инициализирует

Fi, 0 і F0, j рівними, відповідно, di та dj для всіх i і j. Потім він обчислює значення в матриці за такою формулою (1):

$$F_{ij} = max \begin{cases} F_{i-1,\,j-1} + S_{A_{i-1,}B_{j-1}} \\ F_{i-1,\,j} + d \\ F_{i,\,j-1} + d \end{cases} \qquad (1)$$

Тут A_i і B_j - відповідно i-й і j-й символи в першій і другій рядках, S -матриця подібності, а d - ступінь подібності символу, який нічому не відповідає. Іншими словами, це рівень подібності, що додається при вставці або видаленні. Після того, як матриця порахована, значення правого нижнього її елемента і буде ступенем схожості двох послідовностей. Для того щоб зробити вирівнювання, необхідно відстежити, яким шляхом з'явилося це значення, і перейти до попередньому. Наприклад, якщо $F_{i-1,\,j-1} + S_{A_{i-1,}B_{j-1}}$, то елемент (i, j) з'явився з елемента (i-1, j-1), і т. д. Елементи у верхньому рядку відбулися з елементів лівіше себе, елементи з лівого стовпчика - з елементів вище себе. Таким чином, якщо переходити від елемента в правому нижньому кутку до попередньому, від нього до попередній, і т. д., то в підсумку шлях приведе до лівого верхнього елементу, у якого немає попереднього. Тепер, перехід виду (i, j) → (i - 1, j - 1) означає, що (i - 1) -му символу в першому рядку відповідає (j - 1) -й символ у другому рядку. Перехід виду (i, j) → (i-1, j) означає, що (i-1) -го символу першої рядки нічого не відповідає, а перехід (i, j) → (i, j - 1) - що (j - 1) -му символу другого рядка нічого не відповідає. Хоча цей алгоритм завжди знаходить оптимальне вирівнювання, його істотним недоліком є великий час роботи і велике споживання пам'яті.

Алгоритм Сміта-Вотермана аналогічний алгоритму Нідлман-Вунша, але при цьому вирішує завдання локального вирівнювання: знаходить

подстроки першого і другого рядків, які мають максимальним схожістю, а також вирівнює їх. Алгоритм був опублікований в 1981 році. Як і алгоритм Нідлман-Вунша, алгоритм Сміта-Вотермана використовує матрицю подібності. Це дозволяє враховувати різні заміни з різною вагою. Також він дозволяє враховувати різні додавання і видалення по-різному в залежності від того, який саме нуклеотид був доданий або видалений. Реалізація алгоритму Сміта-Вотермана схожа на реалізацію алгоритму Нидлман-Вунша: береться матриця $F_{i,j}$ такого ж розміру, однак, елементи F_i , 0 і F_0 , j ініціалізуются нулями. Перерахунок відбувається за такою формулою (2):

$$F_{i,j} = max \begin{cases} F_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ F_{i-1,j} + D_{A_{i-1}} \\ F_{i-1,j} + D_{B_{j-1}} \\ 0 \end{cases} \quad (2)$$

Тут A_i і B_j - символи з першої і другої рядків, S - матриця подібності для пар символів, I - вектор вартостей додавання (взятих з протилежного знаком), а D - вектор вартостей видалення. Тепер максимальний ступінь подібності буде максимальним елементом матриці. Якщо переходити від цього елемента по ланцюжку попередніх, то шлях закінчиться в якомусь нульовому елементі - не обов'язково лівому верхньому. Індекси цих двох елементів рівні індексам початків і кінців підстрок: перші індекси - в першому рядку, другі- в другому. Шлях інтерпретується так само, як і в алгоритмі Нідлмана-Вунша. Як і алгоритм Нідлмана-Вунша, алгоритм Сміта-Вотермана завжди знаходить оптимальне рішення, та й час роботи і займана пам'ять у нього такі ж. Це робить ці алгоритми неприйнятними для роботи з великою кількістю генетичного матеріалу.

Висновки до розділу 1

Під час опису відмінностей між цими двома алгоритмами, підходами глобального та локального вирівнювання загалом, можливо використовувати фізичне поняття фазового переходу.

Залежно від обраних начальних умов, системи штрафів, комп'ютерна програма вирівнювання будет виходити або із того, що вирівнювані послідовності є повністю перекритими, та прагне вирівняти послідовності по всій їх довжині, або із того, що ці послідовності перекриваються не повністю та мати на меті знайти перекриті ділянки. Алгоритми Нідлмана – Вунша та Сміта – Уотермена направлені на пошук математично оптимального вирівнювання при заданих параметрах. Вище були розглянуті випадки, коли за будь-яке неспівпадання амінокислот (нуклеотидів) в алгоритмі Нідлмана – Вунша назначається оцінка 0, а в алгоритмі Сміта – Уотермена – однаковий штраф.

Однак, виходячи із експериментальних спостережень та міркувань щодо того, що вірогідність різних заміщень (замін) неоднакова, можливо призначати і різний штраф за різноманітне неспівпадіння амінокислот (нуклеотидів). Відомо, що амінокислоти певного класу частіше заміщуються амінокислотами того ж класу. Відповідно до заміщень у межах одного класу можливо призначати менший штраф.

Таким чином, для будь-якої амінокислоти можна оцінити вірогідність її заміщення будь-якою іншою амінокислотою та назначати конкретному заміщенню більший чи менший штраф залежно від вірогідності його заміщення. Ці вірогідності можна уявити у вигляді таблиці, матриці штрафів, які відображають вірогідності заміщень (та збереження) амінокислот.

Виходячи із різних міркувань та експериментальних даних, є запропонований ряд таких матриць. Найчастіше використовуються PAM та BLOSUM при вирівнюванні послідовної матриці.

РОЗДІЛ 2 Аналітична частина

Вступ

Задача динамічного програмування є актуальною в біоінформатиці за рахунок розбиття одної складної дії на більш менші рекурсивні піддії, що доє змогу скоротити час обчислення алгоритму. У якості мови програмування було обрано python за рахунок його поширення в біоінформатиці а також низький поріг входу до мови програмування а також за його мультиплатформенність.

2.1 Динамічне програмування та огляд алгоритму вирівнювання з афінним штрафом

Динамічне програмування в теорії управління і теорії обчислювальних систем - спосіб вирішення складних завдань шляхом розбиття їх на більш прості підзадачі. Він застосовується до завдань з оптимальною підструктури , що виглядає як набір перекриваючих підзадач, складність яких трохи менше вихідної. В цьому випадку час обчислень, в порівнянні з «наївними» методами, можна значно скоротити.

Ключова ідея в динамічному програмуванні досить проста. Як правило, щоб вирішити поставлене завдання, потрібно вирішити окремі частини завдання (підзадачі), після чого об'єднати рішення підзадач в одне загальне рішення. Часто багато хто з цих підзадач однакові. Підхід динамічного програмування полягає в тому, щоб вирішити кожну

підзадачу тільки один раз, скоротивши тим самим кількість обчислень. Це особливо корисно у випадках, коли число повторюваних підзадач експоненціально велике.

Метод динамічного програмування зверху - це просте запам'ятовування результатів вирішення тих підзадач, які можуть повторно зустрітися надалі. Динамічне програмування знизу включає в себе переформулювання складного завдання у вигляді рекурсивної послідовності простіших підзадач.

Для парного вирівнювання біологічних послідовностей використовуємо афінну функцію $y(g)=-d-(g-1)e$, де d та e відповідає значенню штрафу за відкриття та продовження пропуску. При використанні афінної функції штрафів кожна комірка $F_{(i,j)}$ матриці динамічного програмування повинна містити три значення. Перше $-M_{(i-1,j)}$ відповідає найкращій вазі вирівнювання на даному етапіза умови, що символ x_i вирівняний з F_i . $F_{x(i,j)}$ відповідає найкращій вазі за умови, що символу x_i відповідає пропуск. Третє значення $I_{y(i,j)}$ - найбільша вага за умови, що символу y_i відповідає пропуск. Після ініціалізації, рекурсивно заповнюємо трьохрівневу матрицю динамічного програмування. Для кожного з трьох можливих станів знаходиться найкраща вага вирівнювання відповідно до співвідношень (3) (4) (5):

$$I_{x(i,j)}=max\begin{Bmatrix} M_{(i-1,j)}-d \\ I_{x(i-1,j)}-e \end{Bmatrix} \quad (3)$$

$$I_{y(i,j)}=max\begin{Bmatrix} M_{(i-1,j)}-d \\ I_{y(i-1,j)}-e \end{Bmatrix} \quad (4)$$

$$M_{(i,j)}=max\begin{Bmatrix} M_{(i-1,j-1)}+s_{(x_i y_i)} \\ I_{x(i-1,j-1)}-s_{(x_i y_i)} \\ I_{y(i-1,j-1)}-s_{(x_i y_i)} \end{Bmatrix} \quad (5)$$

Значення $s_{(x_i y_i)}$, що збільшують вагу вирівнювання у випадку співпадіння або заміни символів, отримають з матриць замін амінокислот Pam або Blosum. Процедуру зворотного проходу починаємо з комірки,що має максимальну вагу та закінчуємо в лівому верхньому куті де $I_{x(i,j)}$, $I_{y(i,j)}$ або $M_{(i,j)}$ приймає нульове значення.

Систему співвідношень (3) (4) (5) можливо витончено зобразити за допомогою діаграми, зображеної на (рис. 2.1).

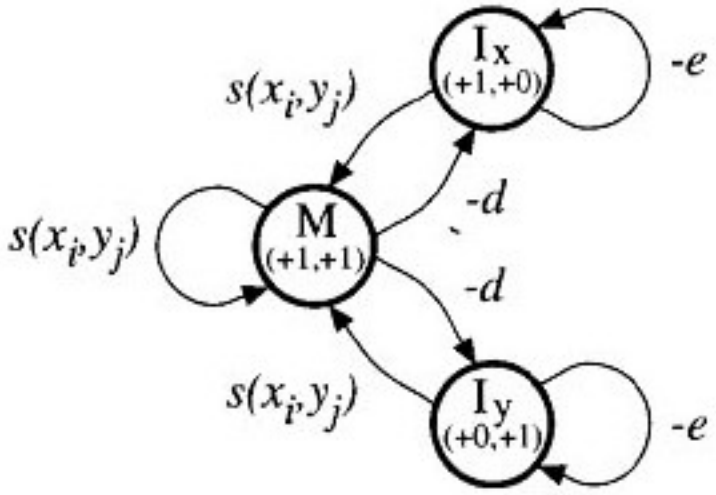

Рисунок 2.1 – діаграма взаємозв'язків між трьома станами, використовуваних для вирывнювання з аффіною штрафною функцією

Еволюція біологічних послідовностей відбувається під впливом точкових мутацій. Враховуючи гіпотезу молекулярного годинника, кількість мутацій, що відбувається в біологічних послідовностях, пропорційна часу, за який вони відбуваються. Експериментально було показано, що найбільш вірогідним є еволюційний процес, при якому

втрачається цілий фрагмент амінокислотної або нуклеотидної послідовності. Така особливість накладає суттєві обмеження на застосування алгоритмів глобального та локального вирівнювання, що в своїй основі використовують фіксовані штрафи за відкриття та продовження пропуску. Враховуючи особливості еволюції біологічних послідовностей, для задач парного вирівнювання необхідно використовувати афінну функцію штрафів, що максимально штрафує відкриття пропуску та мінімально - продовження пропуску.

2.2 Мова програмування Python

Python проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм, ніж shell. З іншого боку, вона краще за C обробляє помилки, і, будучи мовою дуже високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на C потребує значних витрат часу.

Завдяки більш загальним типам даних, Python застосовують до більш широкого кола задач, ніж Awk і навіть Perl, у той же час багато речей на мові Python робляться настільки ж просто.

Python дозволяє розбивати програми на модулі, що потім можуть бути використані в інших програмах. Python поставляється з великою бібліотекою стандартних модулів, які можна використовувати як основу для нових програм або як приклади при вивченні мови. Стандартні модулі надають засоби для роботи з файлами, системними викликами,

мережними з'єднаннями і навіть інтерфейсами до різних графічних бібліотек.

Python - інтерпретована мова, що дозволяє заощадити значну кількість часу, що зазвичай витрачається на компіляцію. Інтерпретатор можна використовувати інтерактивно, що дозволяє експериментувати з можливостями мови, писати шаблони програм або тестувати функції при розробці “знизу-вверх”. Він також зручний як настільний калькулятор. Python дозволяє писати дуже компактні й зручні для читання програми. Програми, написані мовою Python, звичайно значно коротші еквівалента на C або C++ з декількох причин:

- типи даних високого рівня дозволять Вам виразити складні операції однією інструкцією;
- групування інструкцій виконується за допомогою відступів замість фігурних дужок;
- немає необхідності в оголошенні змінних;

Python розширювана мова: знання C дозволяє додавати нові функції, що вбудовуються, або модулі для виконання критичних операцій з максимальною швидкістю або написання інтерфейсу до комерційних бібліотек, доступним тільки у двійковій формі. Інтерпретатор мови Python може бути вбудований у програму, написану на C, і використовувати його як розширення або командну мову для цієї програми. Python використовується в даний час десятками тисяч програмістів в усьому світі, і число людей, що використовують його, швидко зростає, подвоюється і потроюється щороку. Python приваблює користувачів з ряду

причин. Він використовується для розробки програм і дозволяє провести розробку набагато швидше, ніж традиційні мови типу C, C++ або Java . Ця мова працює однаково добре на Windows, UNIX, Macintosh, і OS/2, може використовуватися, для легкої розробки як малих додатків чи сценаріїв, так і для розгортання великих програм. Python пропонує доступ до могутнього і легкого у використанні комплекту 29 інструментальних засобів графічного інтерфейсу користувача. Традиційні машинні мови типу C і Pascal мають ряд характеристик, наприклад, сувора типізація, базові типи, складні (і звичайно довгі) цикли, і потреба у великих кількостях кодів для виконання відносно малих задач. Java досить новий, але розділяє більшість характеристик, включених у цей перелік. Програмісти, знайомі з традиційними мовами погодяться, що відсутність суворої типізації полегшує роботу з Python.

2.3 Python у порівнянні з іншими мовами

Відмінностей Python від інших мов доволі багато, перерахуємо основні з них:

- Керування пам'яттю - цілком автоматичне — не потрібно хвилюватися щодо розподілу або звільнення пам'яті. Немає загрози “небезпечного посилання”. Java - єдина мова, що пропонує таку концепцію.

- Типи зв'язані з об'єктами, а не зі змінними. Це означає, що змінній може бути призначене значення будь-якого типу, і що (наприклад) масив може містити об'єкти різних типів. Традиційні мови не надають такої можливості.

- Операції звичайно виконуються в більш високому рівні абстракції. Це частково результат того, як написана мова, і частково результат розширеної стандартної бібліотеки кодів, що поставляється разом з Python.

Ці та інші особливості Python роблять розгортання додатків надзвичайно швидким. Продуктивність створеного додатку залежить від його особливостей. Звичайно, для чисельного алгоритму, що виконує звичайну арифметику цілого числа в циклі 'for', неважливо, на якій мові він написаний. Але для “середнього” додатка, збільшення продуктивності може бути просто дивовижним. Один недолік Python, у порівнянні з найбільш традиційними мовами, полягає в тому, що це - не цілком компільована мова; замість цього, вона частково транслює програму до внутрішньої форми байт-коду, і цей байт-код виконується інтерпретатором Python. Однак, у перспективі – сучасні комп'ютери мають так багато невикористовуваного обчислювального потенціалу, що для 90% додатків швидкодія зв'язана з вибором мови. Java теж компілюється в байт-код, але в даний час працює повільніше ніж Python у більшості випадків. Крім того, дуже просто об'єднати Python з модулями, написаними на C або C++, які можна використовувати, щоб збільшити швидкість роботи програм в критичних ділянках.

2.4 PYTHON у біоінформатиці та Biopython

Популярність Python пов'язана з низьким порогом входження. Почати писати на Python досить легко, але при цьому мова дає можливість реалізовувати складний функціонал. Існує безліч біоінформатичних бібліотек, таких як Biopython, що дозволяє не винаходити велосипед.

Що таке Biopython?

Проект Biopython це міжнародна асоціація розробників вільно доступних Python інструментів для обчислювальної молекулярної біології. Сайт « www.biopython.org » надає online ресурси модулів, скриптів і посилань для розробників, заснованого на Python програмного забезпечення, для існуючих наукових досліджень.

В основному ми просто любимо програмувати на Python і хочемо зробити це настільки легким, наскільки можливо, для використання Python в біоінформатиці, для створення високоякісних, що допускають повторне використання модулів і сценаріїв.

Основний реліз Biopython має більщую функціональність, що включає:

Здатні аналізуватися файли біоінформатики за допомогою Python, що містять придатні для використання структури даних, включають підтримку наступних форматів:

Blast output (протокол блокової асинхронноїпередачі) - як і від автономного потоку, так і від WWW Blast

- Clustalw
- FASTA
- GenBank
- PubMed і Medline
- ExPASy файли, такі як Enzyme і Prosite
- SCOP, що включають 'dom' і 'lin' файли
- UniGene
- SwissProt

Файли підтримуваних форматів можуть бути оброблені запис, за записом або індексовані і доступні через інтерфейс Dictionary (словника).

Правила для взаємодії з популярними сервісами біоніформатікі, такими як:

- NCBI - Blast, Entrez і PubMed сервісами
- ExPASy - Swiss-Prot і Prosite входами, так само як Prosite пошук

Інтерфейс до поширених програм біоінформатики, таким як:

- Автономної Blast від NCBI
- Clustalw програма вирівнювання

- EMBOSS інструментам командного рядка

Стандартний клас послідовність, який має справу з послідовностями, ідентифікаторами на послідовностях, і функціями послідовностей.

Інструменти для виконання поширених операцій з послідовностями, такими як трансляція, транскрипція і обчислення ваги (weight calculations).

Правила для виконання класифікації даних використовуючи k Nearest Neighbors, Naive Bayes або Support Vector Machines.

Правила для взаємодії з вирівнюванням, що включають стандартні способи створення і використання матриць заміни.

Правила полегшують поділ паралельних завдань на окремі процеси.

Засновані на графічному інтерфейсі програми для виконання основних дій з послідовностями, трансляцій, Blast і т.д.

Велику документацію і допомогу до використовуваних модулів, включаючи цей файл, On-line документацію, адреси Web-сайтів і поштових розсилок.

2.5 Розробка програмного продукту

Найменування ПП

Найменування ПП:

"Вирівнювання Біологічних послідовностей методом динамічного програмування з афінною штрафною функцією."

2.5.1 Призначення і область застосування

ПП призначений для:

- вирівнювання біологічних послідовностей.
- застосування у навчальному процесі.

2.5.2 Вимоги до функціональних характеристик

ПП повинен забезпечувати можливість виконання перерахованих нижче функцій:

1) Зчитувати, порівнювати декілька біологічних послідовностей.
2) Видавати вирівняні біологічні послідовності.
3) Демонструвати візуально вирівнювання біологічних послідовностей.
4) працювати на найбільш відомих операційних системах(Windows, Mac OS, Linux).

2.5.3Вимоги до надійності

Надійне (стійке) функціонування ПП має бути забезпечене

виконанням Замовником сукупності організаційно-технічних заходів,перелік яких наведено нижче:

а) організацією безперебійного живлення технічних засобів;

б) відсутністю сторонніх або шкідливих програм, що можуть привести до непрацездатності даної програми.

2.5.4 Час відновлення після відмови

Час відновлення після відмови, викликаного збоєм електроживлення технічних засобів (іншими зовнішніми чинниками), не фатальним збоєм (не крахом) операційної системи, не повинно перевищувати 10-ти хвилин за умови дотримання умов експлуатації технічних і програмних засобів.

Час відновлення після відмови, викликаного несправністю технічних засобів, фатальним збоєм (крахом) операційної системи, не повинно перевищувати часу, необхідного на усунення несправностей технічних засобів і переустановлення програмних засобів.

Відмови ПП унаслідок некоректних дій користувача при взаємодії з ПП через GUI-інтерфейс неприпустимі. У ПП необхідна реалізація засобів для захисту від некоректного вводу характеристик пацієнтів.

2.5.5 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що пред'являються

до технічних засобів в частині умов їх експлуатації.

2.5.6 Вимоги до кваліфікації та чисельності персоналу

Для роботи з ПП достатньо однієї людини з середнім рівнем комп’ютерної грамотності.

2.5.7 Вимоги до інформаційної та програмної сумісності

- **Вимоги до інформаційних структур і методів розв’язання**

  Додаткові вимоги не пред’являються

- **Вимоги до вихідних кодів та мов програмування**

  Додаткові вимоги не пред’являються

- **Вимоги до програмних засобів, які використовують ПП**

  До складу технічних засобів повинен входити будь-який ПК з розповсюдженою ОС типу: Windows, Linux, MacOS та ін.
Також, має бути встановлений додаток-бібліотека python.

2.5.8 Вимоги до захисту інформації та ПП

Вимоги до захисту інформації та ПП не пред'являються.

2.5.9 Спеціальні вимоги

ПП повинен забезпечувати роботу одного користувача за допомогою консольного інтерфейсу

2.5.10 Вимоги до інформаційних структур і методів розв'язання

Програма має можливість завантажувати файл у FASTA форматі де зберігається цілісна структура біологічної послідовності, користувач має змогу завантажити її через інтерфейс.

2.5.11 Вимоги до вихідних кодів та мов програмування

ПП має бути розроблено за допомогою python для забезпечення максимальної мультиплатформеності.

2.5.12 Стадії розробки

Розробка повинна бути проведена в сім стадій:

1) системний аналіз та аналіз вимог;

2) ознайомлення з літературою з розділу біонформатика;

3) Аналіз та формування макету обраного алгоритму

а) розбиття алгоритму на послідовні кроки;

б) створення макету алгоритму;

4) тестування алгоритму та його відпрацювання:

а)аналіз кожного кроку алгоритму;

б) аналіз та порівняння вихідних даних.

5) Створення програми:

а) створення логіки програми

б) розробка інтерфейсу та внутрішніх процесів

6) тестування і налагодження роботи ПП;

7) розробка документації;

2.5.13 Етапи розробки

На стадії системного аналізу та аналізу вимог повинно бути сформоване технічне завдання, діаграма Ганта та мережевий графік, які узгоджені та затверджені Заказником.

На стадії проектування повинні бути виконані перераховані нижче етапи робіт:

1) розробка Ієрархічної структури та розрахунок нев'язки;

2) розробка ERD, SADT, DFD діаграм;

3) розробка блок-схем основної частини ПП та усіх її допоміжних функцій;

4) розробка USE-CASE діаграми за допомогою мови UML.

На стадії кодування повинні бути виконані перераховані нижче етапи робіт:

1) реалізація інтерфейсу ПП в pycharm;

2) розібраний алгоритм вирівнювання біологічних послідовностей

На стадії тестування і налагодження роботи ПП проводиться випробування системи на наявність помилок, достовірність роботи всіх підсистем і системи в цілому.

На стадії розробки документації повинні бути виконані перераховані нижче етапи робіт:

1) оформлення документації, яка перерахована на етапі системного аналізу та аналізу вимог;

2) оформлення результатів етапів кодування та тестування;

3) оформлення посібника користувача.

На стадії оформлення звіту проводиться підведення підсумків та опис результатів усіх виконаних робіт

Висновки до розділу 2

У цьому розділі розглянуто етапи створення програмного продукту, обґрунтовано використання обраної мови програмування та наведено використання цієї мови у біоінорматиці, а також використання інших мов програмування у біоінформатиці.

РОЗДІЛ 3 ОХОРОНА ПРАЦІ

Вступ

Система вирівнювання біологічних послідовностей. Робота виконувалась в комп'ютерному класі. Проектується система вирівнювання біологічних послідовностей для навчальних цілей та використання її у науково-дослідницьких роботах.

3.1 Характеристика приміщення

Комп'ютерний клас знаходиться на 5 поверсі п'ятиповерхової будівлі (каб. 5-07). Приміщення сухе, не запилене, підлога вкрита лінолеумом та не проводить електричний струм. План комп'ютерного класу зображено на рис. 3.1. Параметри та основні елементи комп'ютерного класу відображені в таблиці 3.1.

Таблиця 3.1 Характеристики та перелік обладнання комп'ютерного класу

Назва	**Характеристика**	**номер**	**Кіль-ть**
Параметри кабінету, м	3,69*5,77*3,52*5,89*3,1	-	-
Працюючі місця	8 чол.	-	-
Площа, $м^2$	42	-	-
Об'єм, $м^3$	130	-	-

Продовження таблиці 3.1

Штучне освітлення	лампи ЛБ-40	-	2
Природна вентиляція	Вентиляція загальна	-	-
Підлога	Вкрита лінолеумом	-	-
Стіни	Вкрито жовтою водоемульсійною фарбою	-	-
Стеля	Вкрито білою водоемульсійною фарбою	-	-
Компьютер	Монітор: 17" Philips 17S4LSB/62 Системний блок: Acer Aspire XC-605 Intel Pentium Dual Core G3240 (2.8 ГГц) / RAM 4 ГБ / HDD 500 ГБ / Intel HD Graphics	1	8
Кондиціонер	Toshiba RAS-M22SKV-E 6кВт	2	1
Вікно	123*100*12	3	2
Радіатор опалення водяний	Kraft Bimetal 500/77	4	2
Двері	94*230*12	5	1
Стіл комп'ютерний	80*75*160	6	8
Стілець	67*94*60	7	8

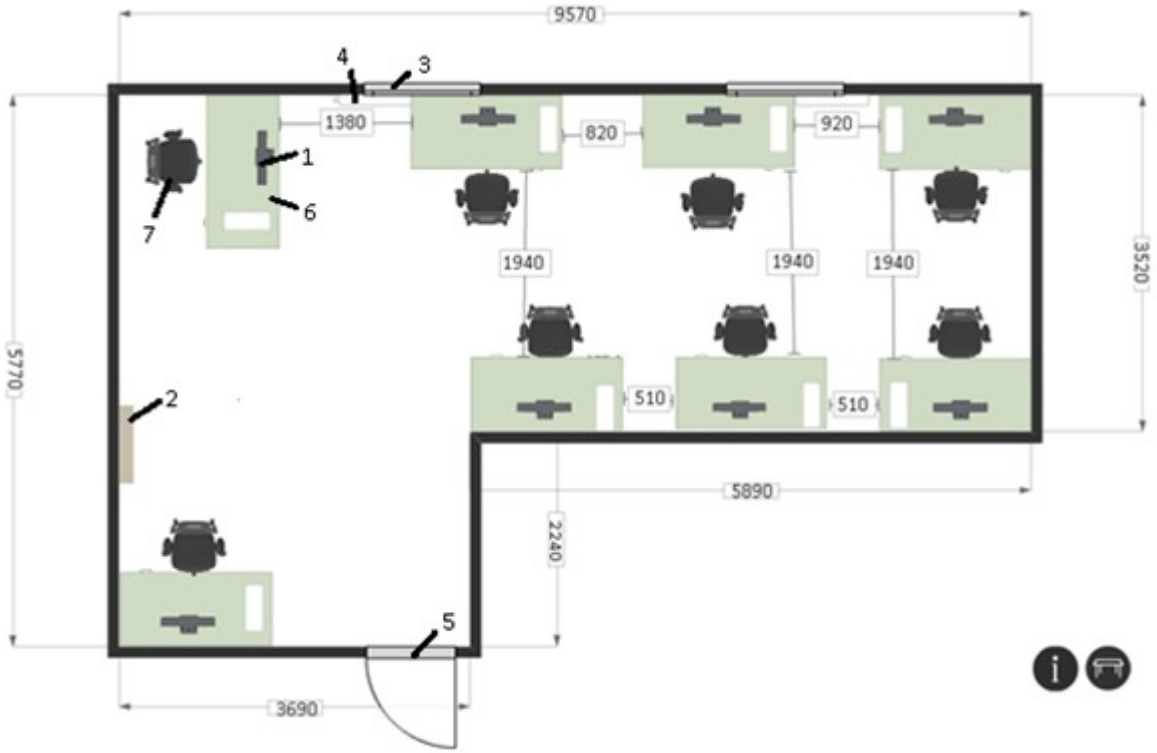

Рисунок 3.1 - План комп'ютерного класу

Таблиця 3.2 Нормативні та фактичні значення параметрів

Параметр	Нормативне значення	Фактичне значення
Відстань між робочими місцями	≥ 1.5 м	< 1,5 м
Відстань від монітору до стіни	≥ 1 м	≥ 0.13 м
Площа на одну людину	≥6 м^2	≥ 6 м^2

Так як фактичні значення параметрів відхиляються від норми приведені заходи нормалізації характеристик приміщення

Таблиця 3.3 - Заходи нормалізації характеристик приміщення

Вид заходу	Спосіб уникнення небезпеки
Технічні заходи	Не передбачено
Організаційні заходи	Необхідно збільшити відстань між робочими місцями на 1м. Збільшити відстань від стіни до монітору на 0,9 м.

3.2 Оцінка небезпечних і шкідливих виробничих факторів

Таблиця 3.4 - Небезпечні та шкідливі виробничі фактори

Види небезпеки	Джерела небезпеки
Фізичні	Освітлення, мікроклімат, пожежонебезпека, електробезпека
Психофізіологічні	Монотонність праці

3.2.1 Фізичні джерела небезпек

3.2.1.1 Мікроклімат

Таблиця 3.5 - Параметри мікроклімату

Період року	Температура повітря, ^{0}C		Відносна вологість, %		Швидкість руху повітря, м/с	
	Норм. значення	Реал. значення	Норм. значення	Реал. значення	Норм. значення	Реал. значення
Холодний	22-24	20-23	60-40	60	0.1	0.1
Теплий	23-25	25-27	60-40	45	0.1	0.1

У комп’ютерному класі показники мікроклімату не виходять за межі норми, виконано заходи і засоби для нормалізації параметрів мікроклімату.

Таблиця 3.6 - Заходи нормалізації мікроклімату

Вид заходу		Спосіб уникнення небезпеки
Технічні	В обладнанні	Термопаста для комп’ютера Titan TTGG30030.

		Застосовуються вентилятори (DeepCool Beta 200ST)) у комп’ютерах
	В приміщенні	Налагодити кондиціонер(-2 ^{0}C) Опалення 2 біметалічних радіатори
Організаційні		Вологе прибирання приміщення.

3.2.1.2 Шум

Джерела шуму: комп’ютери, зовнішній шум. Шум у кабінеті є постійним. Кількість працюючих не повинна перевищувати 8 чоловік. Джерела та наслідки шуму вказані в таблиці 3.7. Заходи подолання шуму представлені в таблиці3.9.

Таблиця 3.7 – Джерела та наслідки шуму

Джерело шуму	**Наслідки**
Кондиціонер	Зниження рівня працездатності; Втомлюваність і загальна раздратованість; Віброакустичний дискомфорт.
ПК	
Внутрішній шум	
Зовнішній шум від транспорту	

Встановлені в аудиторії вікна пропускають шум з вулиці.

Таблиця 3.8 – Порівняння нормованого і фактичного шумів

Джерело шуму	Рівень звуку Li, дБА	Час впливу звуку t, год	Кількість n, шт	Нормований рівень шуму, дБА (ДСН 3.3.6.037-99)	Фактичний рівень шуму, дБА
Кондиціонер	30	8	1	<50	87,13
ПК	25	8	8		
Зовнішній шум (вулиця та внутрішній шум)	35	8	1		

Таблиця 3.9 – Заходи подолання шуму

Вид заходу		**Засоби подолання небезпеки**
Технічні заходи	У технологічному обладнанні	Раз на пів року проводити огляд техніки(змащування кулерів,миття кондиціонеру, нанесення термопасти).
	У приміщенні	Встановити металопластикові вікна. (20%) Обшити стіни плитами на основі скловолокна ISOVER (20%) Встановити перегородки з ДСП та шумоізоляційною тканиною для зменшення шуму в робочому місці (60%)
Організаційні заходи		Режим праці і відпочинку, дотримування правил технічної експлуатації.

3.2.1.3 Електробезпека

За ступенем небезпеки ураженням електричним струмом дане приміщення відноситься до приміщень без підвищеної небезпеки. Кількість комп’ютерів на одній лінії не перевищує чотири. В приміщенні все обладнання підключено до мережі змінного струму напругою 220 В,

та інші характеристики вказані в таблиці 3.11. Джерела та наслідки електронебезпеки вказані у таблиці 3.10 та 3.12.

Таблиця 3.10 Джерела ураження струмом

Джерела	**Небезпеки**	
	Для людини	Для техніки
ПК	стресовий стан, смерть	Вихід з ладу
Кондиціонер		
Розетки		

Таблиця 3.11 Характеристика електромережі

Напруга живлення від мережі	220 В
Частота	50 Гц
Сила струму	від 0,1 до 1А

Таблиця 3.12 Заходи захисту від дії електричного струму

Вид захисту		Засоби подолання небезпек:
Технічні	В обладнанні	Ізоляція обладнання відповідно нормам; підключення устаткування відповідно вимогам ПБЕ і ПУЕ;
	В приміщенні	Щитки ЕЩР-0-6 та блоки розеток Р-06М; прихована проводка; підлога вкрита антистатичним лінолеумом.
Організаційні		Встановлення знаків безпеки (кнопка ручного вмикання, евакуаційний вихід, вогнегасник, заборона палити); проведення інструктажу з правил техніки безпеки.
ЗІЗ		Не передбачені

3.2.1.4 Пожежонебезпека

Таблиця 3.13 Джерела небезпеки

Джерело	Фактор	Наслідок
ПК	Коротке замикання	Виникнення пожежі. Загроза життю.
Розетки	Перенавантаження, коротке замикання	

Для запобігання пожежі у приміщенні вжито наступні заходи протипожежної безпеки зазначені в таблиці 3.13

Таблиця 3.14 Заходи нормалізації пожежної безпеки

Вид заходу		Спосіб уникнення небезпеки
Технічні	В обладнанні	Підключення відповідно норм ПБЕ.
	В приміщенні	Установка пожежних датчиків
Організаційні		Проведення інструктажів з ТБ.

3.2.1.5 Електромагнітне випромінювання

У робочому кабінеті присутнє незначне електромагнітне випромінювання. Джерела та наслідки небезпеки вказані у таблиці 3.15. Заходи подолання небезпеки вказані в таблиці 3.16.

Таблиця 3.15 Джерела та наслідки електромагнітного випромінювання

Джерело	Наслідок
Електромагнітне випромінювання	Незначна негативна дія на організм працюючих
Електромагнітне випромінювання від ноутбука.	

Таблиця 3.16 Заходи безпеки

Вид заходу	**Засоби подолання небезпеки**
Технічні заходи	Використання «сплячого режиму»
Організаційні заходи	Робити перерви в роботі
ЗІЗ	Непередбачені

Висновки до розділу

У цій частині дипломної роботи були розглянуті умови роботи та техніка безпеки в комп'ютерному класі. Створені умови мають забезпечувати комфортну роботу. На підставі вивченої літератури з даної теми, були визначені можливі небезпечні та шкідливі умови мікроклімату, освітлення, рівня шуму, електромагнітного випромінювання, електро та пожежної безпеки. Дотримання умов, що визначають оптимальну організацію аудиторії, дозволить зберегти вискоку працездатність протягом учбового дня.

ЗАГАЛЬНІ ВИСНОВКИ

В результаті роботи було створено програмний продук для вирівнювання біологічних послідовностей методом динамічного програмування з афінним штрафом. Проведений аналіз існуючих алгоритмів вирівнювання. В дипломній роботі було виконано поставлені задачі, а саме:

1. Зібрано та проаналізовано інформацію про існуючі методи вирівнювання
2. Реалізувано алгоритм вирівнювання біологічних послідовностей
3. Проведено аналіз отриманих вихідних даних
4. Проведено аналіз небезпек та шкідливих факторів при роботі з програмним продуктом в аудиторії, запропонувано заходи, щодо зменшення їх впливу

ДОДАТОК А

Фрагмент коду алгоритму

```
#=============================================================
# Printing and utility functions
#=============================================================

Infinity = float('inf')


def make_matrix(sizex, sizey):
    """Creates a sizex by sizey matrix filled with zeros."""
    return [[0]*sizey for i in xrange(sizex)]


def print_matrix(x, y, A):
    """Print the matrix with the (0,0) entry in the top left
    corner. Will label the rows by the sequence and add in the
    0-row if appropriate."""


    if len(x) == len(A):
        print "%5s" % (" "),
    else:
        print "%5s %5s" % (" ","*"),
        y = "*" + y


    for c in x:
        print "%5s" % (c),
    print

    for j in xrange(len(A[0])):
        print "%5s" % (y[j]),
        for i in xrange(len(A)):
            print "%5.0f" % (A[i][j]),
        print
```

```
def is_complement(a, b):
     assert len(a) == len(b) == 1
   return (a.upper(), b.upper()) in [
      ("A", "T"), ("T", "A"),
      ("C", "G"), ("G", "C"),
      ("A", "U"), ("U", "A")
   ]
#=============================================================
# Alignment Parameters
#=============================================================

class ScoreParam:
     def __init__(self, match, mismatch, gap, gap_start=0):
       self.gap_start = gap_start
       self.gap = gap
       self.match = match
       self.mismatch = mismatch

   def matchchar(self, a,b):
            assert len(a) == len(b) == 1
       if a==b:
          return self.match
       else:
          return self.mismatch

   def __str__(self):
       return "match = %d; mismatch = %d; gap_start = %d; gap_extend =
%d" % (
             self.match, self.mismatch, self.gap_start, self.gap
       )
#=============================================================
# Sequence Alignment
#=============================================================

def local_align(x, y, score=ScoreParam(10, -5, -7)):

   # create a zero-filled matrix
   A = make_matrix(len(x) + 1, len(y) + 1)
```

```
        best = 0
        optloc = (0,0)

            for i in xrange(1, len(x)+1):
            for j in xrange(1, len(y)+1):

                # the local alignment recurrance rule:
                A[i][j] = max(
                  A[i][j-1] + score.gap,
                  A[i-1][j] + score.gap,
                  A[i-1][j-1] + score.matchchar(x[i-1], y[j-1]),
                  0
                )

                # track the cell with the largest score
                if A[i][j] >= best:
                    best = A[i][j]
                    optloc = (i,j)

        print "Scoring:", str(score)
        print "A matrix ="
        print_matrix(x, y, A)
        print "Optimal Score =", best
        print "Max location in matrix =", optloc
        # return the opt score and the best location
        return best, optloc


    def affine_align(x, y, score=ScoreParam(10, -2, -7, -15)):
        """Global alignment with affine penalties. We assume we are
maximizing."""
        M = make_matrix(len(x) + 1, len(y) + 1)
        X = make_matrix(len(x) + 1, len(y) + 1)
        Y = make_matrix(len(x) + 1, len(y) + 1)

        for i in xrange(1, len(x)+1):
            M[i][0] = -Infinity
            X[i][0] = -Infinity
            Y[i][0] = score.gap_start + i * score.gap
```

```
for i in xrange(1, len(y)+1):
   M[0][i] = -Infinity
   X[0][i] = score.gap_start + i * score.gap
   Y[0][i] = -Infinity

for i in xrange(1, len(x)+1):
   for j in xrange(1, len(y)+1):

      M[i][j] = score.matchchar(x[i-1], y[j-1]) + max(
            M[i-1][j-1],
            X[i-1][j-1],
            Y[i-1][j-1]
      )

      X[i][j] = max(
            score.gap_start + score.gap + M[i][j-1],
            score.gap + X[i][j-1],
            score.gap_start + score.gap + Y[i][j-1]
      )

      Y[i][j] = max(
            score.gap_start + score.gap + M[i-1][j],
            score.gap_start + score.gap + X[i-1][j],
            score.gap + Y[i-1][j]
      )

opt = max(M[len(x)][len(y)], X[len(x)][len(y)], Y[len(x)][len(y)])
print "x = %s & y = %s" % (x,y)
print "Scoring:", str(score)
print "M matrix ="
print_matrix(x,y,M)
print "X matrix ="
print_matrix(x,y,X)
print "Y matrix ="
print_matrix(x,y,Y)
print "Optimal =", opt
return opt
```

ДОДАТОК Б

Інструкція користувача

Для запуску програмного продукту потрібно на комп'ютер на котрому планується використовувати продукт встановити Python 2.6.1 за посиланням https://www.python.org/download/releases/2.6.1/

Після чого в командному рядку запустити "Python" Ви отримаете ">>>" щоб запустити програмни й продукт треба вказати наступне ">>>from align423 import *" Потім потрібно запустити функцію affine_align та ввести дві послідовності після чого є опціонально можливість ввести параметри штрафної функції.Приклад показано на рисунку Б.1 .

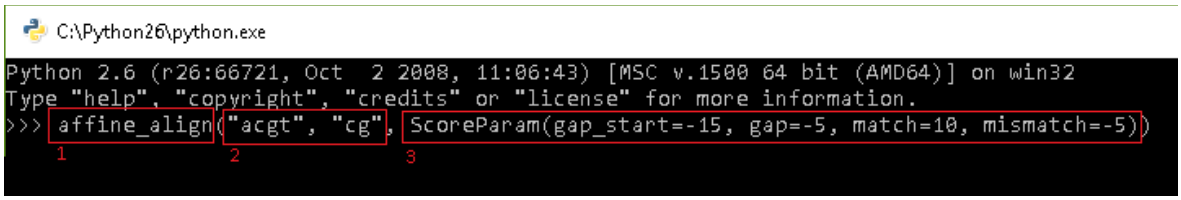

Рисунок Б.1 – Вхідні параметри

Синтаксис вводу послідовності та параметри штрафної функції.

1- запуск функції вирівнювання
2- перша та друга послідовності для вирівнювання
3- Параметри штрафної функції

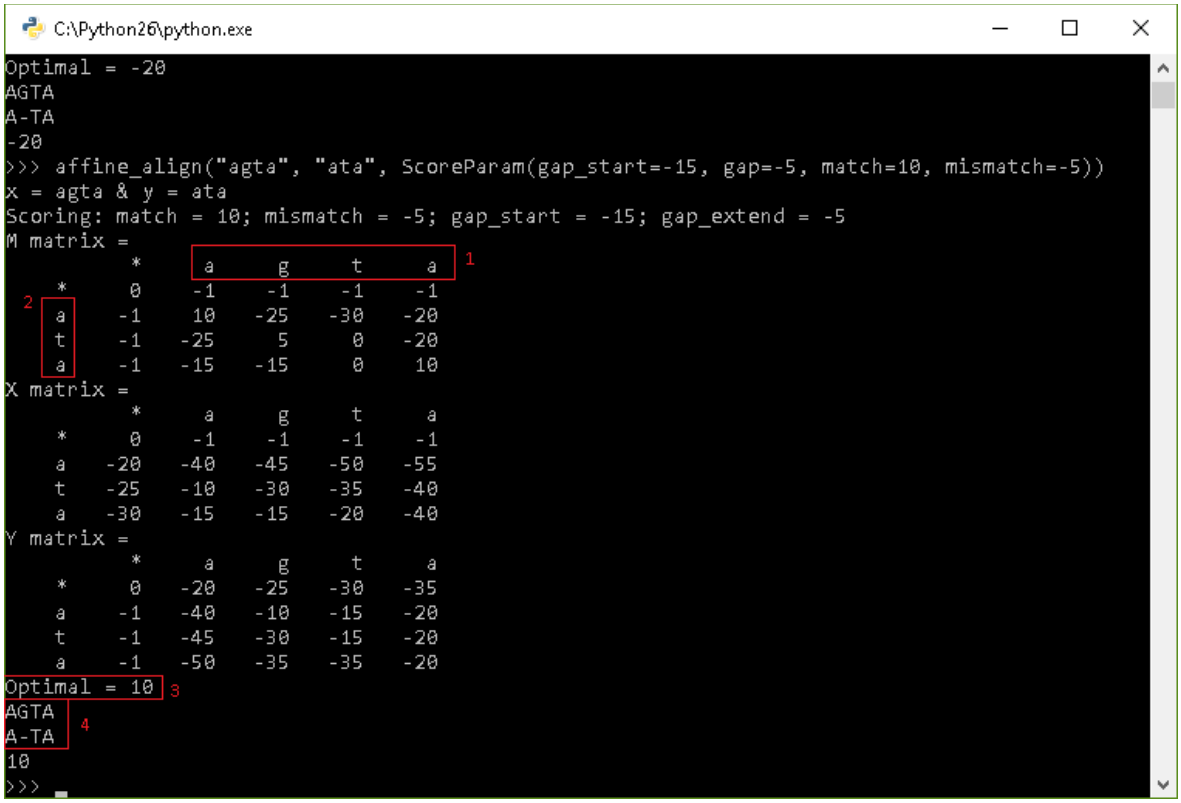

Рисунок Б.2 – Вихіні дані

Приклад вихідних данних.

1- Послідовність X
2- ПослідовністьY
3- Оптимальна матриця
4- Результат вирівнювання

ПЕРЕЛІК ПОСИЛАНЬ

1. Gap Penalties: [Електронний ресурс]. – Режим доступу : https://www.cs.umd.edu/class/fall2010/cmsc423/lectures
2. Федор Царев, Андрей Лушников. Динамическое програмирование: [Електронний ресурс]. – Режим доступу : http://www.slideshare.net/fedor.tsarev/04-1009043
3. Что такое phyton?: [Електронний ресурс]. – Режим доступу: http://www.plug.org.ua/documentation/about-python
4. RichardDurbin, Sean R. Eddy, AndersKrogh, GraemeMitchison.Biologicalsequenceanalysis .- Cambridge university press.- 2008. - 356p.
5. Sequence alignment and substitution matrices.[Електронний ресурс]. – Режим доступу: http://www.proteinstructures.com/Sequence/Sequence/sequence-alignment.html
6. Biophython [Електронний ресурс]. – Режим доступу: http://biopython.org/
7. Smith-Waterman Algorithm - Local Alignment of Sequences. [Електронний ресурс]. – Режим доступу: http://vlab.amrita.edu/?sub=3&brch=274&sim=1433&cnt=1
8. Kinser J. Python For Bioinformatics (Series in Biomedical Informatics) / Jason Kinser. – Ontario, 2009. – (Malloy, inc.)-101p.
9. Kinser J. Python For Bioinformatics (Series in Biomedical Informatics) / Jason Kinser. – Ontario, 2009. – (Malloy, inc.)-174p.
10. Kinser J. Python For Bioinformatics (Series in Biomedical Informatics) / Jason Kinser. – Ontario, 2009. – (Malloy, inc.)-345p.
11. Динамічне програмування.[Електронний ресурс]. – Режим доступу: http://goo.gl/pECb5S

12. Охорона праці. [Електронний ресурс]. – Режим доступу: http://www.diagram.com.ua/info/ohrana/ohrana-truda37.shtml
13. Butler, John M (2001). Forensic DNA Typing. Elsevier. c. 14–15.
14. An Introduction to Bioinformatics Algorithms. [Електронний ресурс]. – Режим доступу: https://goo.gl/ebckAD
15. Молекулярная биология. Структура и биосинтез нуклеиновых кислот. — М.: Высшая школа, 1990.
16. Smith T. F., Waterman M. S. Identification of common molecular subsequences // Journal of Molecular Biology. — 1981. — Vol. 147, no. 1. — Pp. 195 – 197.
17. The seq Object. Електронний ресурс]. – Режим доступу: http://biopython.org/wiki/Seq
18. Li WH. Molecular Evolution. Sunderland: Sinauer Associates; 1997.
19. Needlman S, Wunsch C. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 1970; 48:443-453.
20. Вирівнювання послідовностей ДНК (Aligment) [Електронний ресурс]. – Режим доступу: http://www.phylogenetics.ru/lang/ru/2011/06/nucleotid-alignment/
21. list of sequence alignment software. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/List_of_sequence_alignment_softwar e