**qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

|  |
| --- |
| Computational Biology  Assignment: 1  10/4/2012  Ajinkya Potdar 108738610  Siddhesh Shirsat 1088805949  Sugandha Rudra108721036 |
|  |

Computational Biology

Assignment – 1

1. Entrez is a cross database search. It retries an answer to a query across all different databases and journals such as Protein Sequence Database, PubMed,etc.It is a retrieval system designed for searching several linked databases.
   1. On searching for Amyloid Precursor Protein, the search result returns various references to the query string. The references for the genome sequence are found in Protein Sequence Database. It returns multiple results corresponding to the different organisms it’s found in.

The result differs in the organisms it’s found in, journals it’s referenced in and its sequence. For Instance, the top most search corresponds to the organism: Homo Sapiens (Humans). It then lists out the different journals that the query exists in. The protein sequence for amyloid precursor protein found in humans is

mklltglvfc slvlgvssrs ffsflgeafd gardmwrays dmreanyigs dkyfhargny

daakrgpggv waaeaisdar eniqrffghg aedsladqaa newgrsgkdp nhfrpaglpe

ky

* 1. Banana Genome Sequence was sequenced recently. It can be found in genome sequence database. It is found in ‘Musa acuminata’. It consists of various banana and plantain varieties. Different citations can be found in other databases such as Pubmed, Books, Site search, etc.
  2. The number of organisms represented in Entrez Genome from each main domain of life archaea, bacteria, and eukaryota **[source:http://www.ncbi.nlm.nih.gov/Sitemap/Summary/statistics.html].**The number of genomes completely sequenced can be categorized into three different categories:
* Archaea: 230 Organisms
* Bacteria:3419 Organisms
* Eukaryotes:1752 Organisms

In total, the number of organisms whose genomes are sequenced is 5401 organisms.

Mammals ‘Mammalia’ :186 Organisms

Plants ‘Embryophata’ :625 Organisms

Fungi :376 Organisms

Bacteria :3419 Organisms

Viruses :3027 organisms

1. PubMed is a database of all biological/medical literature provided by NCBI. It is a free database accessing life sciences and biomedical topics. It comprises of more than 22 million citations. It contains abstracts from PubMed Central and publisher websites. There are various PubMed tools such as PubMed Mobile, Citation and Batch Matcher, Clinical Queries, Topical Specific Queries.
2. *Amyotrophic Lateral Sclerosis(ALS)* is the most common form of neuron disease. It is typically characterized by adult-onset degeneration of upper and lower motor neurons, and is usually fatal within a few years of onset. Diagnosis occurs most often between the ages of 40 and 60. No clear causes have been found and just one medication, riluzole (only currently FDA-approved drug), extends survival. Researchers have identified some of the cellular processes that occur after disease onset, including mitochondrial dysfunction, protein aggregation, oxidative stress, excitotoxicity, inflammation, and apoptosis. Mitochondrial disease may be a primary event in neurodegeneration or occur secondary to other cellular processes, and may itself contribute to oxidative stress, excitotoxicity, and apoptosis. About 90% of ALS cases are sporadic and the remaining 10% of ALS cases are familial (FALS). In about 20% of FALS cases, the cause can be attributed to a mutation in the Cu2+/Zn2+ superoxide dismutase 1 (SOD1), a ubiquitously-expressed free-radical defense enzyme. The mutations cause misfolding of this normally stable homodimeric protein.

Individuals with ALS may benefit from care by a multidisciplinary team that includes a neurologist, specially trained nurses, pulmonologist, speech therapist, physical therapist, occupational therapist, respiratory therapist, nutritionist, psychologist, social worker, and genetics professional. Oral secretions in those with bulbar symptoms can be reduced with tricylic antidepressants and other anticholinergic agents. Pseudobulbar affect can be managed with antidepressants. Swallowing difficulties can be alleviated by thickening liquids and pureeing solid food and, eventually, by use of a gastrostomy tube to help maintain caloric intake and hydration. Medications such as baclofen and benzodiazepines can help relieve spasticity and muscle cramps. Alphabet boards and computer-assisted devices can aid communication. Other assistive devices such as walkers, wheelchairs, bathroom modifications, hospital beds, and Hoyer lifts can aid in activities of daily life. Ventilatory assistance may include BIPAP and/or mechanical ventilation. Hospice care in terminal stages is beneficial.

1. *Isistius brasiliens(*Cookiecutter Shark*)*

TaxonomyID:862898  
Genbankcommon name: cookie cutter shark  
Inherited blast name: sharks and rays  
Rank: species

Only the taxonomical classification of the *Isistius brasiliens* is defined. Other information provided id the nucleotide sequence and the partial recombinational activating protein.A phylogenetic study of 590 and 52 sequence of cookie cutter shark is present in the population study data sets.

1. There are 26 papers of Prof. Skiena mentioned on PubMed.

Following are the numbers of papers not found in the corresponding databases.

<http://www.informatik.uni-trier.de/>ley/db = 12

Google Scholar = 0

the ACM Digital Library = 22

the Web of Knowledge database = 4

3.

**Source code for finding Shortest Common Superstring**:

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class ShortestCommonSuperString {

public static void main(String args[]) throws IOException {

String sequence = "ACGTTCGAAAACGTTGCAC";

int m = 5;

int l = 5;

List<String> listFragments = generateFragments(sequence, m, l);

System.out.println("sequence = " + sequence);

printFragments(listFragments);

//System.out.println(getMaxOverlap("CCCCCCE", "CCEE"));

System.out.println("Shortest Common SuperString = " + getShortestCommonSuperString(listFragments));

}

public static List<String> generateFragments(String str, int m, int l) {

Set<String> fragments = new HashSet<String>();

while(fragments.size() < m) {

int r = (int)(Math.random()\*(str.length()-l+1));

fragments.add(str.substring(r, r+l));

}

List<String> result = new ArrayList<String>();

result.addAll(fragments);

return result;

}

public static void printFragments (List<String> fragments) {

for (String fragment: fragments) {

System.out.println("fragmment = " + fragment);

}

}

public static String getMaxOverlap(String str1, String str2) {

String overLap = "";

for (int i=1; i<=str2.length(); i++) {

String str2subString = str2.substring(0, i);

if (i>str1.length()) {

break;

}

String str1subString = str1.substring(str1.length()-i);

if (str2subString.equals(str1subString)) {

overLap = str2subString;

}

}

return overLap;

}

public static String getShortestCommonSuperString(List<String> fragments) {

while (fragments.size() > 1) { // loop until the list contains only one string

String overLap = "";

String overLapString1 = "";

String overLapString2 = "";

for (String fragment1: fragments) {

for (String fragment2: fragments) {

if (fragment1.equals(fragment2)) {

continue;

}

String overLapTemp = getMaxOverlap(fragment1, fragment2);

if (overLapTemp.length() > overLap.length()) {

overLap = overLapTemp;

overLapString1 = fragment1;

overLapString2 = fragment2;

}

}

}

if (overLap.length() > 0) {

fragments.remove(overLapString1);

fragments.remove(overLapString2);

fragments.add(overLapString1+overLapString2.substring(overLap.length()));

} else {

String noOverLapString = "";

for(String f: fragments) {

noOverLapString += f;

}

fragments.clear();

fragments.add(noOverLapString);

}

}

return fragments.size() == 1 ? fragments.get(0) : "";

}

public double getAccuracy() {

double accuracy = 0;

return accuracy;

}

}

a. How large a project does it take (as a function of n and m) before reconstruction times

starts to be a problem?

With a sampling rate of 5% i.e. m = n/20 and l = 10, where n is the length of the input sequence.

Reconstruction time starts to be a problem when n=10,000. Time required for reconstruction is 46 seconds.

b. What coverage do you need before the reconstructed superstring tends to be correct (the

same as the input is generated from)?

|  |  |  |  |
| --- | --- | --- | --- |
| Length of input sequence (n) | Number of fragments (m) | Length of each fragment (l) | Coverage  (m\*l/n) |
| 100 | 43 | 10 | 4.3 |
| 200 | 81 | 10 | 4.05 |
| 300 | 136 | 10 | 4.5 |
| 400 | 161 | 10 | 4.02 |
| 500 | 233 | 10 | 4.6 |

The mean coverage needed before the reconstructed superstring tends to be correct is 4.29

c. How is the accuracy of reconstruction affected by coverage and the fragment length?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Length of input sequence (n) | Number of fragments (m) | Length of each fragment (l) | Coverage  (m\*l/n) | Accuracy |
| 100 | 40 | 5 | 2 | 10.63 |
| 100 | 40 | 7 | 2.8 | 29.82 |
| 100 | 40 | 9 | 3.6 | 92.43 |
| 100 | 40 | 11 | 4.4 | 94.27 |
| 100 | 40 | 13 | 5.2 | 96.15 |
| 100 | 40 | 16 | 6.4 | 100 |

Thus we can see as the coverage increases accuracy increases.

d.

English sequence used:

I like playing football and scoring goals. Passing is the essence of the game. I support Arsenal !!!

DNA sequence used:

atattaggtttttacctacccaggaaaagccaaccaacctcgatctcttgtagatctgttctctaaacgaactttaaaatctgtgtagct gtcgctcggctgcatgcctagtgcacctac

It is a Corona Virus.

(obtained from <http://www.ncbi.nlm.nih.gov/nuccore/FW503166.1>)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Length of input sequence (n) | Number of fragments (m) | Length of each fragment (l) | Coverage  (m\*l/n) | Accuracy for English Text | Accuracy for DNA sequence |
| 100 | 40 | 5 | 2 | 17.72 | 10.59 |
| 100 | 40 | 7 | 2.8 | 63.24 | 37.10 |
| 100 | 40 | 9 | 3.6 | 94.19 | 92.35 |
| 100 | 40 | 11 | 4.4 | 98.05 | 92.41 |
| 100 | 40 | 13 | 5.2 | 100 | 92.35 |

1. Algorithm to find longest suffix of A that exactly matches the prefix of B

* Build the suffix tree of A. This can be done in O(n) time where n is length of the string.
* We can search for string B in the suffix tree. Save the current index of B that is present in the suffix tree. Let's see it is i. It will take O(k) time where k is length of string B.
* When the character in B is not present in the tree, we stop the search and return I.
* Now we can print B[0...i] to get the longest prefix of B that matches A.

e.g. Let’s say

A= RAFAADD

B = AADDRERE

Then we can build suffix tree of A and search for string B in this tree and while traversing save current index of B (let’s say it is i). When we come to 5th character in the string i.e. R, we see that it is not present in the tree and we stop the search.

We return 4 which was the saved index earlier. So the Longest common suffix of A that exactly matches the prefix of B is AADD.

This Algorithm takes O(n+k) time.

5. Finding Biological Palindrome:

* A palindrome reads the same forwards and backwards.
* Biological Palindrome is a sequence which is equal to its reverse complement.
* Such palindromes bind/fold to create secondary structures in sequences, which often have biological significance.

e.g. GAATTC is a biological palindrome.

* Longest Common substring calculation:

Longest common substring for two strings s1 and s2 can be found out by , building a (basic) suffix tree for the string s1$s2#, where `$' is a special terminator for s1 and `#' is a special terminator for s2. The longest common substring is indicated by the deepest fork node that has both `...$...' and `...#...'(no$) beneath it.

* Using Longest Common substring algorithm to find Biological Palindrome

Let us say we have a sequence having biological palindrome.

* To find biological palindrome, we find the longest common substring where

s1= sequence string containing biological palindrome.

s2 = reverse complement of string s1

* We can modify longest common substring problem to get least common Ancestor from suffix tree.
* We can find the longest sub-palindrome in linear time by asking the ‘length’ of the LCA of S[i] and S[i + n + 1] for each 1 ≤ i ≤ n.

Example: Build a suffix tree S = s1#s2$

ACATTAAT#ATTAATGT$

In the corresponding suffix tree it can be seen that the least common ancestor has maximum length 6 and it is ATTAAT.

So the maximum biological palindrome is ATTAAT in the above string.

6.

a. Greedy heuristic (find the largest palindrome) doesn't always work.

Counter Example: ABBABABA

Partitioning using Greedy Heuristic: A-B-BABAB-A = 4

Optimal Partitioning: ABBA-B-ABA = 3

b. Repeatedly finding the largest left-most remaining palindrome doesn't always work.

Counter Example: AMMADAM

Partitioning using greedy heuristic: AMMA-D-A-M = 4

Optimal Partitioning: A-M-MADAM = 3

c.

We can solve this problem using Dynamic programming.

Let us say we have string str and we are finding the optimal number of partitions.

Let i be the starting index and j is the ending index within string str where we are finding minimum number of partitions. The initial value of i is 0 and that of j is n-1. Then we see following possible conditions:

1. minPartitions(str, i, j) = str[i] if i == j. // When string is of length 1.

2. minPartitions(str, i, j) = str, if str is a palindrome.

3. // If none of the above conditions are true, then minPartitions (str, i, j) can be calculated recursively using the following formula.

minPartitions (str, i, j) = minPartitions(str, i, k)) V minPartitions(str, k+1, j)

where k is such that, min { len(minPartitions(str, i, k)) + len(minPartitions(str, k+1, j)) }

where k varies from i to j-1.

This algorithm stores the solutions to sub-problems in two arrays Pal[][] and Cost[][], and reuses the calculated values.

We can consider following steps while solving this problem.

**Step 1**:

Initialize Pal[][]. Pal[i][j] is 1 if str[i....j] is palindrome.

We first initialize all elements within Pal with 0.

As all the elements with length 1 are palindromes we initialize diagonal of matrix Pal with 1.

We can calculate all the palindromes by following algorithm.

K =1

for var= k to n

{

i=j= var

while S[i] == S[j]

{

Pal[i][j] =1

i--

j++

}

}

K =1

for var= k to n

{

i = var

j= var+1

while S[i] == S[j]

{

Pal[i][j] =1

i--

j++

}

}

**Step 2**:

Now we build the cost array C where cost represents number of cuts needed in the substring.

for (L=2; L<=n; L++)

{

// For substring of length L, set different possible starting indexes

for (i=0; i<n-L+1; i++)

{

j = i+L-1; // Set ending index

// IF str[i..j] is palindrome, then C[i][j] is 0

if (P[i][j] == true)

C[i][j] = 0;

else

{

// Make a cut at every possible location starting from i to j,

// and get the minimum cost cut.

C[i][j] = INT\_MAX;

for (k=i; k<=j-1; k++)

C[i][j] = min (C[i][j], C[i][k] + C[k+1][j]+1);

}

}

}

// Return the min cut value for complete string. i.e., str[0..n-1]

return C[0][n-1];

The above algorithm takes O(n^3) time.

Based on this Cost array we can select the cuts to be made in the string.