**Devansh P. Gupta**

**01609278**

**CIS 455 – Homework #2**

Problem 2-1. (20 points) Jones & Pevzner, Problem 4.1, page 119.

1. Write an algorithm in pseudo code that given a set X, calculates the multi set ΔX.

2. Implement the above algorithm in Mat lab, provide your source code and testing results to submit.

Mat lab is available in Dion 303 and on the Virtual Computing Lab: http://www.umassd.edu/cits/vcl

Solution: For example, if X={0, 2, 4, 7, 10}, then ΔX={2, 2, 3, 3, 4, 5, 6, 7, 8, 10}, which are the ten pair wise distances between these points.

Algorithm:

Input: A multi set X containing n numbers

Output: Set ΔX, which has pair wise distance between these numbers

Steps:

1 a<--A multi set element, consisting of elements

2 b<--Empty set containing only zeros

3 count<--1

4 for i <-- 0 to every every set of a

5 loop

6 for j<--i+1 to every set of a

7 loop

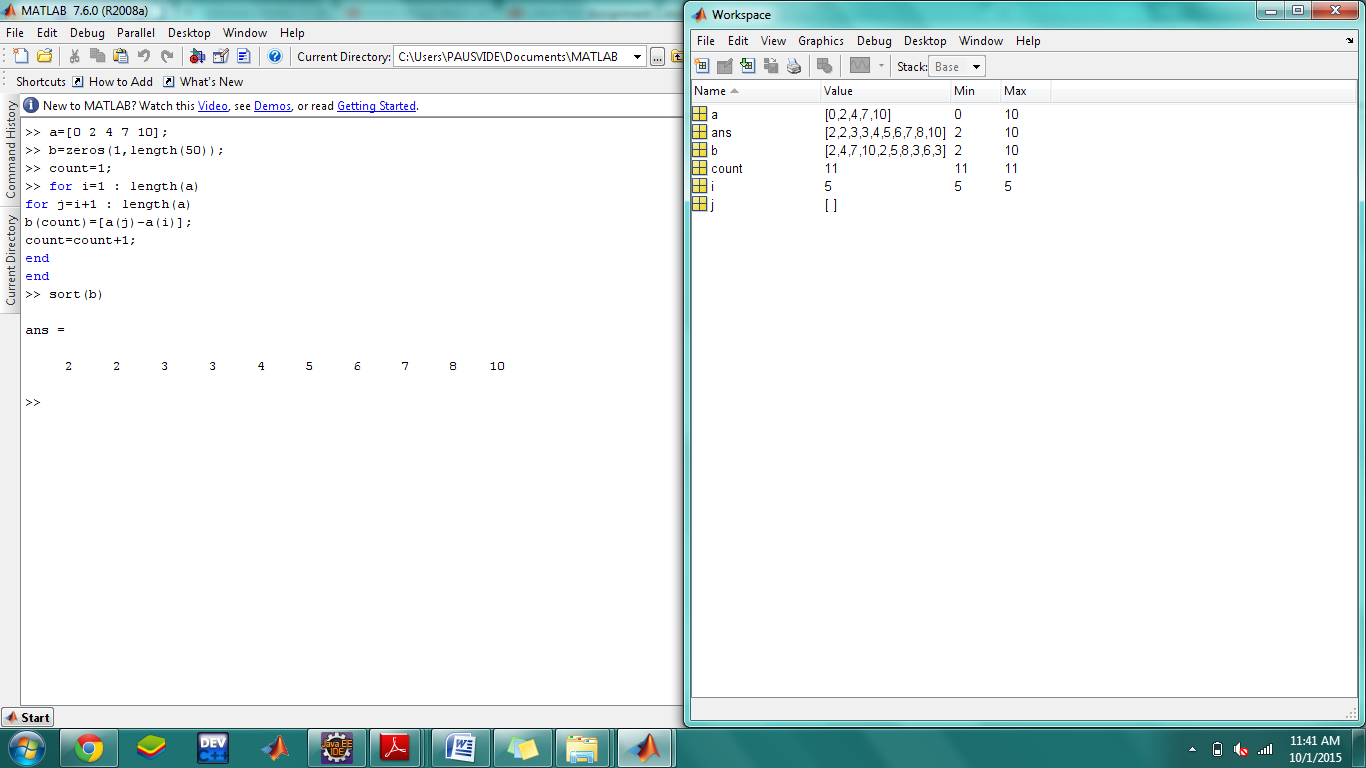
8 b(count) <-- Subtracting Next element to Previous element and inserting in b

9 count<-- count+1

10 end loop

11 end loop

12 Output "Sorted b"



Running time: Total computations:2n-1

Problem 2-2. (20 points) Jones & Pevzner, Problem 4.2, page 119.

Consider partial digest L = {1,1,1,2,2,3,3,3,4,4,5,5,6,6,6,9,9,10,11,12,15} .

Solve the Partial Digest problem for L (i.e., find all X such that ΔX = L).

Make sure you find every X such that ΔX = L.

Hint: You can solve this problem by doing either one of the following:

(1) Manually follow the steps of PARTIALDIGEST(L) on page 90.

or

(2) Write a program that implements ANOTHERBRUTEFORCEPDP on page 88 and apply it to L.

Solution:

1. Remove 15 from L and insert it into X. We know this must be the length of the DNA sequence because it is the largest fragment.

L = {1,1,1,2,2,3,3,3,4,4,5,5,6,6,6,9,9,10,11,12,15}

X = { 0, 15}

Take 12 from L and make y = 3 or 12. But since the two cases are symmetric, we can assume y = 12. We find that the distances from y=12 to other elements in X are D(y, X) = {12,3}, so we remove {3,12} from L and add 12 to X.

L = {1,1,1,2,2,3,3,4,4,5,5,6,6,6,9,9,10,11}

X = { 0, 12,15}

==============================================================

Take 11 from L and make y =4 or 11. But since the two cases are symmetric, we can assume y = 11. We find that the distances from y=11 to other elements in X are D(y, X) = {11,1,4}, so we remove {11,1,4} from L and add 11 to X.

L = {1,1,2,2,3,3,4,5,5,6,6,6,9,9,10}

X = { 0, 11,12, 15}

=============================================================

Take 10 from L and make y = 5 or 10. But since the two cases are symmetric, we can assume y = 10. We find that the distances from y=10 to other elements in X are D(y, X) = {10,1,2,5}, so we remove {1,2,5,10} from L and add 10 to X.

L = {1,2,3,3,4,5,6,6,6,9,9}

X = { 0, 10, 11, 12, 15}

==============================================================

Take 9 from L and make y = 6 or 9. But since the two cases are symmetric, we can assume y = 9.

We find that the distances from y=6 to other elements in X are D(y, X) = {9,1,2,3,6}, so we remove {9,1,2,3,6} from L and add 6 to X.

L = {3,4,5,6,6,9}

X = { 0, 9, 10, 11, 12, 15}

===============================================================

Take 9 from L and make y = 6 or 9. But since the two cases are symmetric, we can assume y = 6. We find that the distances from y=6 to other elements in X are D(y, X) = {6,3,4,5,6,9}, so we remove {9,3,4,5,6,9} from L and add 6 to X.

L = {}

X = { 0, 6, 9, 10, 11, 12, 15}

==============================================================

L is now empty, so we have a solution, which is X.

0 6 9 10 11 12 15

=============================================================

1. To find other solutions, we backtrack.

L = {1,2,3,3,4,5,6,6,6,9,9}

X = { 0, 10, 11, 12, 15}

This time we will explore y = 6. D(y, X) = {6,4,5,6,9}, so we remove {6,4,5,6,9} from L and add 6 to X.

L={1,2,3,3,6,9}

x={0,6,10,11,12,15}

===============================================================

We find that the distances from y=9 to other elements in X are D(y, X) = {9,3,1,2,3,6}, so we remove {9,3,1,2,3,6} from L and add 9 to X.

L = {}

X = { 0, 6, 9, 10, 11, 12, 15}

L is now empty, so we have a solution, which is X.

Which is same as above

===============================================================

1. To find other solutions, we backtrack.

Take 10 from L and make y =5 or 10. But since the two cases are symmetric, we can assume y = 5. We find that the distances from y=5 to other elements in X are D(y, X) = {5,6,7,10} which is not a subset of L, so we can't explore this branch.

==============================================================

1. To find other solutions, we backtrack.

L = {1,1,2,2,3,3,4,5,5,6,6,6,9,9,10}

X = { 0, 11,12, 15}

Take 10 from L and make y = 5 or 10. But since the two cases are symmetric, we can assume y = 5. We find that the distances from y=5 to other elements in X are D(y, X) = {5,6,7,10}, so we remove {5,6,7,10} from L and add 5 to X.

But 7 is not a subset of L, So we can't explore any further

==============================================================

1. To find other solutions, we backtrack.

L = {1,1,1,2,2,3,3,4,4,5,5,6,6,6,9,9,10,11}

X = { 0, 12,15}

Take 11 from L and make y =4 or 11. But since the two cases are symmetric, we can assume y = 4. We find that the distances from y=4 to other elements in X are D(y, X) = {4,8,11}, so we remove {4,8,11} from L and add 4 to X.

But 8 is not a subset of L, So we can't explore any further

==============================================================

1. L = {1,1,1,2,2,3,3,3,4,4,5,5,6,6,6,9,9,10,11,12,15}

X = { 0, 15}

Take 12 from L and make y = 3 or 12. But since the two cases are symmetric, we can assume y = 3. We find that the distances from y=3 to other elements in X are D(y, X) = {3,12}, so we remove {3,12} from L and add 3 to X.

L = {1,1,1,2,2,3,3,4,4,5,5,6,6,6,9,9,10,11}

X = { 0, 3, 15}

==============================================================

Take 11 from L and make y =4 or 11. But since the two cases are symmetric, we can assume y = 11. We find that the distances from y=11 to other elements in X are D(y, X) = {11,1,4}, so we remove {11,1,4} from L and add 11 to X.

L = {1,1,2,2,3,3,4,5,5,6,6,6,9,9,10}

X = { 0,3, 11, 15}

=============================================================

Take 10 from L and make y = 5 or 10. But since the two cases are symmetric, we can assume y = 10. We find that the distances from y=10 to other elements in X are D(y, X) = {10,1,2,5}, so we remove {1,2,5,10} from L and add 10 to X.

L = {1,2,3,3,4,5,6,6,6,9,9}

X = { 0, 3, 10, 11, 15}

===============================================================

Take 9 from L and make y = 6 or 9. But since the two cases are symmetric, we can assume y = 9.

We find that the distances from y=6 to other elements in X are D(y, X) = {9,1,2,3,6}, so we remove {9,1,2,3,6} from L and add 6 to X.

L = {3,4,5,6,6,9}

X = { 0, 3, 9, 10, 11, 15}

==============================================================

Take 9 from L and make y = 6 or 9. But since the two cases are symmetric, we can assume y = 6. We find that the distances from y=6 to other elements in X are D(y, X) = {6,3,4,5,6,9}, so we remove {9,3,4,5,6,9} from L and add 6 to X.

L = {}

X = { 0, 3, 6, 9, 10, 11, 15}

===============================================================

L is now empty, so we have a solution, which is X.

0 3 6 9 10 11 15

Problem 2-3. (10 points) Jones & Pevzner, Problem 4.5, page 119.

Prove that the sets U ‍⊕ V = {u + v : u ∈ U, v∈ V}and U ⊖ V = {u − v : u∈ U, v∈ V} are homo metric for any two sets U and V.

Solution:

If all elements of U ‍⊕ V and U ⊖ V are same then it is homo metric. If The partial digests of these two sets is the same multi set then it is said to be homo metric.

Let U and V be two sets of numbers. One can verify that the multi sets

U ‍⊕ V = {u + v : u ∈ U, v ∈ V }

and

U ⊖ V = {u − v : u ∈ U, v ∈ V }

are homo metric (a problem at the end of this chapter). The “nontrivial” nine point example above came from U = {6, 8, 10} and V = {−6, 1, 4}.

Indeed U ⊕ V ={0,2,4,7,9,10,11,12,14} while U ⊖ V ={2,4,5,6,7,9,12,14,16} as illustrated below:

|  |  |  |  |
| --- | --- | --- | --- |
| U ⊕V | 6 | 8 | 10 |
| -6 | 0 | 2 | 4 |
| 1 | 7 | 9 | 11 |
| 4 | 10 | 12 | 14 |

|  |  |  |  |
| --- | --- | --- | --- |
| U⊖ V | 6 | 8 | 10 |
| -6 | 12 | 14 | 16 |
| 1 | 5 | 7 | 9 |
| 4 | 2 | 4 | 6 |

Since partial digest of sets U ‍⊕ V and U ⊖ V are not same so it is not homemetric,but it can be true for some cases and not all.

Problem 2-4. (18 points) Jones & Pevzner, Problem 4.9, page 121 (read the DDP description on pages 119-121).

1. Come up with a brute force algorithm for the DDP and show pseudo code of your algorithm.

2. Suggest a branch-and-bound approach to improve its performance.

Solution:

1) Input:

A – fragment lengths from the digest with enzyme A.

B – fragment lengths from the digest with enzyme B.

X – fragment lengths from the digest with both A and B.

Output: A – location of the cuts in the restriction map for the enzyme A.

B – location of the cuts in the restriction map for the enzyme B.

Steps:

1 count<--0;

2 for i <-- 0 to n\*2

3 diff[count] <-- Subtracting every fragment with every other fragment of other enzyme

4 Incrementing count

Code:

#include<stdio.h>

#include<stdlib.h>

int compare (const void \*elem1, const void \*elem2){

return \*(int\*)elem1 - \*(int\*)elem2;

}

int main()

{

int A[50],B[50],temp[50],diff[50];

int i,j,n,count=0;

printf("\nPlease enter the nos of elements in A & B :");

scanf("%d",&n);

printf("\nPlease enter elements for A :");

for(i=0;i<n;i++)

{

scanf("%d",&A[i]);

}

printf("\nPlease enter elements for B :");

for(i=0;i<n;i++)

{

scanf("%d",&B[i]);

}

for(i=0;i<n;i++)

{

temp[i]=A[i];

}

for(i=n;i<n\*2;i++)

{

temp[i]=B[count];

count++;

}

qsort(temp,n\*2,sizeof(int),compare);

printf("\nPrinting elements in Temp:");

for(i=0;i<n\*2;i++)

{

printf("\t%d",temp[i]);

}

count=0;

for(i=0;i<n\*2;i++)

{

diff[count]=temp[i+1]-temp[i];

count++;

}

printf("\nPrinting elements for A-B:");

printf("%d",temp[0]);

for(i=0;i<n\*2-1;i++)

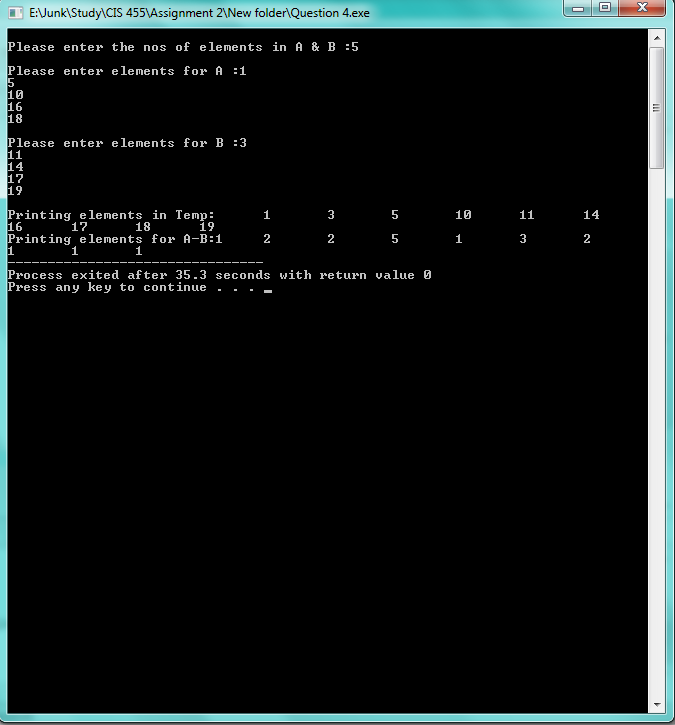
{

printf("\t%d",diff[i]);

}

return 0;

}

Output:

2) By using Branch & Bound Strategy we can solve Double Digest Problem.

Branch & bound algorithm can be used to create a tree structure and then can be used to find an optimal solution in less time. It checks for all possible nodes in the string and splits up into several branches. By having this approach it can reduce complexity.

Problem 2-5. (12 points) Jones & Pevzner, Problem 4.12, page 121.Given a long text string T and one shorter pattern string s, find the first occurrence of s in T (if any).

1. Show pseudocode of your algorithm.

2. What is the complexity of your algorithm?

Solution:

Input:

str1<--Enter String 1

str2<--Enter subset of string 2

Output:

Find location of subset string 2 in main string 1

Step:

1 str1<--Enter string 1

2 str2<--Enter substring 2

3 n1<--length of string 1

4 n2<--length of string 2

5 j<--0

6 for i<--0 to n1-n2

7 for k<--0 to n1

8 if str1[i+k] equals to str2[k])

9 Increment count

10 if count equals to n2

11 loop

12 loc <-- i

13 return loc

14 end loop

15 else

16 count <-- 0

Code:

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

int stringmatch(char str1[50],char str2[50],int n1,int n2);

int main()

{

int n1,n2,loc;

char str1[50];

char str2[50];

int i;

printf("\nPlease enter elements in STR1:");

scanf("%s",&str1[i]);

n1=strlen(str1);

printf("\nPlease enter elements in STR2:");

scanf("%s",&str2[i]);

n2=strlen(str2);

loc=stringmatch(str1,str2,n1,n2);

printf("\nString matches at %d location",loc+1);

return 0;

}

int stringmatch(char str1[50],char str2[50],int n1,int n2)

{

int count=0;

int i,j=0,k=0,loc=0;

for(i=0;i<n1-n2;i++)

{

for(k=0;k<n2;k++)

{

if(str1[i+k]==str2[k])

{

count++;

if(count==n2)

{

loc=i;

return loc;

}

}

else

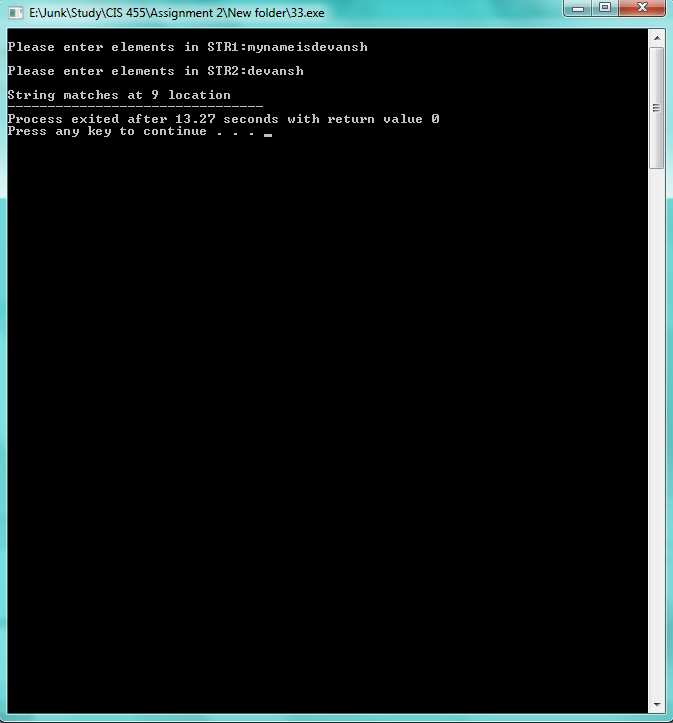
count=0;

}

}

}

Output:



Nos of comparisons: n2\*(n1-n2+1)

Complexity:O(n1\*n2)

For Random Case: Ө(n1+n2)

Problem 2-6. (10 points) Jones & Pevzner, Problem 4.16, page 122.

Problem 4.16

The following algorithm is a cousin of one of the motif finding algorithms we have

considered in this chapter. Identify which algorithmis a cousin of YETANOTHERMOTIFSEARCH

and find the similarities and differences between these two algorithms.

YETANOTHERMOTIFSEARCH(DNA, t, n, l)

s<-- (1, 1, . . . , 1)

bestMotif FIND(s, 1, t, n, l)

return bestMotif

FIND(s, currentSeq, t, n, l)

i<--currentSeq

bestScore<--0

for j<--1 to n − l + 1

s(i)<--j

bestPossibleScore <-- Score(s, i) + (t − i) · l

if bestPossibleScore > bestScore

if currentSeq ≠ t

s<--FIND(s, currentSeq + 1, t, n, l)

if Score(s) > bestScore

bestScore<--Score(s)

bestMotif<--s

return bestMotif

Solution: The above given algorithm resembles to Branch And Bound Motif Search.

YETANOTHERMOTIFSEARCH uses recursive method and BRANCHANDBOUNDMOTIFSEARCH uses iterative method. Recursive method causes more memory then iterative method, so iterative method is much more efficient then recursive method.

Since BRANCHANDBOUNDMOTIFSEARCH uses iterative method it can be considered to more efficient then YETANOTHERMOTIFSEARCH.

YETANOTHERMOTIFSEARCH uses more memory because of extensive use of stack. But same is not the case with BRANCHANDBOUNDMOTIFSEARCH. BRANCHANDBOUNDMOTIFSEARCH algorithm uses BYPASS and NEXTVERTEX function to calculate the best motif and the optimal solution.

Problem 2-7. (10 points) Jones & Pevzner, Problem 4.17, page 123.

Derive a tighter bound for the branch-and-bound strategy for the Median String problem.

Hint: Split an l-mer w into two parts, u and v. Use TotalDistance(u,DNA) +TotalDistance(v,DNA) to bound TotalDistance(w,DNA).

Solution:

BranchAndBoundMedianStringSearch(DNA,t,n,l )

1. s <-- (1,1,…,1)
2. bestDistance ← ∞
3. i <-- 1
4. while i > 0
5. if i < l
6. prefix <-- Nucleotide string corresponding to (s1, s2, …, si)
7. optimisticDistance <-- TotalDistance(prefix,DNA)
8. if optimisticDistance > bestDistance
9. (s, i ) <-- Bypass(s,i, l, 4)
10. else
11. (s, i ) <-- NextVertex(s, i, l, 4)
12. else
13. word <-- Nucleotide string corresponding to (s1, s2, …, sl)
14. if TotalDistance(s,DNA) < bestDistance
15. bestDistance <-- TotalDistance(word, DNA)
16. bestWord <-- word
17. (s,i) <-- NextVertex(s,i,l, 4)
18. return bestWord

Total Distance For (U)

1. for(i=1;i<=4;i++)
2. if i < l
3. prefix <-- Nucleotide string corresponding to s1,s2,.....,si
4. optimistic distance <-- TotalDistance(prefix,DNA)
5. if optimisticDistance > bestDistance
6. (s, i ) <-- Bypass(s,i, l, 4)
7. else
8. (s, i ) <-- NextVertex(s, i, l, 4)
9. else
10. word <-- Nucleotide string corresponding to (s1, s2, …, sl)
11. if TotalDistance(s,DNA) < bestDistance
12. bestDistance <-- TotalDistance(word, DNA)
13. bestWord <-- word
14. (s,i) <-- NextVertex(s,i,l, 4)
15. return bestWord

Total Distance For (V)

1. for(i=5;i<=8;i++)
2. if i < l
3. prefix <-- Nucleotide string corresponding to s1,s2,....,si
4. optimistic distance <-- TotalDistance(prefix,DNA)
5. if optimisticDistance > bestDistance
6. (s, i ) <-- Bypass(s,i, l, 4)
7. else
8. (s, i ) <-- NextVertex(s, i, l, 4)
9. else
10. word <-- Nucleotide string corresponding to (s1, s2, …, sl)
11. if TotalDistance(s,DNA) < bestDistance
12. bestDistance <-- TotalDistance(word, DNA)
13. bestWord <-- word
14. (s,i) <-- NextVertex(s,i,l, 4)
15. return bestWord

Add the TotalDistance(U) + TotalDistance(V)

Adding U and V

Refrences:

[1] Dr. Firas Khatib, "Double Digest Mapping" Lecture\_9-17-2015,(Power point presentation).

[2] Neil C. Jones And Pavel A. Pevzner "An Introduction To Bioinformatics Algorithm ".

[3] Dr. Firas Khatib, "DNA Mapping and Brute Force Algorithms" Lecture\_9-15-2015, (Power point presentation).